UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO
LICENCIATURA EN ACTUARÍA
FACULTAD DE CIENCIAS

Análisis, diseño e implementación de VULCANO: Sistema de Gestión de Tesorería Automatizado en la empresa del sector financiero ISBIT SA DE CV

TRABAJO PROFESIONAL

QUE PARA OBTENER EL GRADO ACADÉMICO DE:
Licenciado en **Actuaría**

PRESENTA:
Sebastian Acosta Checa

DIRECTOR DE TRABAJO PROFESIONAL:
Dr. Humberto Andrés Carrillo Calvet

Ciudad Universitaria, CDMX. Junio, 2022

ÍNDICE

TABLA DE ILUSTRACIONES

1. Introducción

De acuerdo con el articulo 30 de la Ley Para Regular a las Instituciones de Tecnología Financiera "*Un Activo Virtual es la representación de valor registrada electrónicamente y utilizada entre el público como medio de pago para todo tipo de actos jurídicos y cuya transferencia únicamente puede llevarse a cabo a través de medios electrónicos. En ningún*

caso se entenderá como activo virtual la moneda de curso legal en territorio nacional, las divisas ni cualquier otro activo denominado en moneda de curso legal o en divisas" (SECRETARIA DE HACIENDA Y CREDITO PUBLICO. 9 marzo 2018, LRITF, Diario Oficial de la Federación).

ISBIT SA de CV es una empresa mexicana que fue fundada en el año 2013 por Sebastian Acosta Checa. En el 2016, la institución desarrolló una plataforma de intercambio, custodia y pagos con Activos Virtuales. ISBIT ofrece sus servicios mediante múltiples portales y marcas comerciales:

- http://isbit.mx
- https://www.isbit.co/
- https://isbit.exchange/
- https://global.isbit.exchange/trading/btcusd
- https://aion.com.mx
- https://app.aion.com.mx/trading/btcusd
- https://stage.isbit.exchange/trading/ethusd

El principal activo virtual que maneja la empresa es el Bitcoin, el cual a principios del 2019 su capitalización era de 64 billones de dólares y ésta ha ido incrementando a una velocidad considerable, de tal manera que en julio del 2021 su capitalización alcanzó los 725 billones de dólares, lo cual indica que ha aumentado más de 1,132% su valor en un periodo de menos de 31 meses.

Una consecuencia de que la capitalización del BTC aumentara, es que también aumentó la demanda de usuarios en la empresa y por ende el volumen y el número de transacciones acrecentó, al grado de que era muy difícil llevar el control manual de las operaciones por lo que fue necesario buscar una solución que automatizara los procesos de tesorería de la empresa, tales como: cobranza, dispersiones, auditoría y cambios de divisas.

Existen softwares y APIs comercialmente disponibles que la empresa consideró como potenciales soluciones, ente los cuales destacan STP, Integrity Software, Openpay y Conekta. Estos sistemas permiten automatizar algunas o todas las operaciones de la tesorería, conciliaciones bancarias, acceso a la banca electrónica, pagos y cobranzas, entre otras funcionalidades, sin embargo; no se adecuaron completamente a las necesidades de la empresa. La solución óptima fue invertir en investigación y desarrollo para crear una

herramienta que cumpliera con las necesidades propias de la institución, así inició el proyecto que denominamos “VULCANO''.

VULCANO es una herramienta que contribuye en aumentar la seguridad de la operación de ISBIT, así como minimizar los riesgos de liquidez, operativos y cibernéticos de la tesorería de la institución.

Desde un principio se tomó la consideración técnica de que VULCANO y el resto de los sistemas informáticos de ISBIT deberían estar débilmente acoplados: Permitir que las máquinas y los usuarios de un sistema distribuido sean independientes entre sí en lo fundamental, pero que interactúen en cierto grado para coordinar su trabajo y para transmitir datos entre ellas.

Uno de los objetivos del diseño del sistema es que múltiples aplicaciones, departamentos y áreas pueden intercambiar información relacionada con la tesorería. En el caso particular de ISBIT, los sistemas que interactúan con la tesorería son:

A) Plataforma digital para intercambio de activos virtuales (Exchange),
B) Sistema Gestion Empresarial, Herramienta de contabilidad y facturación electrónica (ERP)
C) Sistema automatizado para prevención de lavado de dinero (Artemisa/METIS).
D) Robot especializado para operaciones de Arbitraje Financiero y Formación de Mercado (CAERUS)

Las tecnologías que se utilizaron para el desarrollo e implementación de VULCANO son:

a) NodeJS para la implementación de las funcionalidades del servidor backend
b) angular, html y css para la implementación de la capa visual
c) Nginx fue empleado para servir la aplicación web y API sobre el protocolo HTTP así como lograr escalabilidad al proporcionar un balanceador de carga y proxy inverso
d) Docker fue empleado para la creación de contenedores e imágenes como parte de la arquitectura orientada a microservicios
e) Kubernetes fue utilizado para gestionar y orquestar las imágenes creadas en formato Docker
f) AMQP (por sus siglas en ingles: Advanced Message Queuing Protocol) se empleó para lograr una transmisión confiable de mensajes al almacenarlos en una pila de memoria, llevando a cabo una comunicación asíncrona robusta a intermitencias en la disponibilidad de microservicios que forman parte de la arquitectura.
g) MongoDb fue elegido como el principal almacen de datos.
h) Microsoft Azure fue elegida la Nube para hospedar toda la infraestructura de la aplicación incluyendo, pero no limitado a recursos de redes, computo, bases de datos, almacenamiento y dispositivos de seguridad.

La mayor inquietud de la empresa al momento de empezar con este proyecto fue cumplir con los objetivos de hacer más eficaz el funcionamiento de la tesoreria de Moneda Nacional y Divisas. Con VULCANO se busca aumentar la seguridad de tesorería de la empresa y la velocidad de las operaciones, maximizar la transparencia, así como acrecentar la redundancia y facilitar la instalación, mantenimiento y evolución futura del sistema.

VULCANO podría implementarse en otras organizaciones de diferentes industrias que requieren automatizar procesos ligados con la tesorería. La herramienta VULCANO, por ejemplo, podría ser usada por una empresa aseguradora para pagar de manera automatizada reclamaciones aprobadas por un proceso de inteligencia artificial sin requerir intervención humana; también podría ser usado por una empresa de préstamos por internet para conciliar y aplicar de forma automatizada mediante VULCANO los pagos para amortizar los créditos enviados por sus clientes, sin necesidad de pedirle a los mismos que envíen comprobantes de pago por correo.

A lo largo de este escrito, se pretende explicar a fondo cuáles fueron las bases y las herramientas usadas para lograr resolver la problemática planteada en la Tesoreria de la empresa ISBIT SA DE CV, así como la metodología y los recursos empleados para dicho fin, tales como: microservicios, contenedores, aplicaciones, librerías, mitigación de riesgos de ciberseguridad, diseño de interfaces y algoritmos.

También se presentan los resultados y las conclusiones derivados de la experiencia obtenida.

2. Objetivos

A partir de un exhaustivo Análisis de la tesorería (fortalezas, oportunidades, debilidades y riesgos) de ISBIT SA DE CV, se plantearon cuatro objetivos para el nuevo sistema de gestión de tesorería que se deberían lograr para el buen funcionamiento de la empresa ISBIT. Estos cuatro objetivos son la base del criterio de éxito para evaluar el software VULCANO.

I. Aumentar Seguridad de Tesorería de la Empresa

Un requerimiento clave de la tesorería de la empresa ISBIT SA DE CV para operar de forma óptima es usar la automatización de procesos para mitigación de riesgo. El riesgo en el manejo de un tesoreria corporativa puede tomar muchas formas en los siguientes rubros:

<u>Riesgo Operativo</u>

- Actuación con dolo, neglicencia o impericia de funcionarios de la empresa
- Errores operativos causados por humanos, tal como el envio duplicado en una

dispersión de pagos por error.

- Abuso de confianza, el cual abarca la manipulación de procesos por funcionarios corruptos dentro de la organización para beneficio propio abusando de sus credenciales o nivel de acceso.

Riesgo Regulatorio y de Cumplimiento

- Realizar operaciones con personas sancionadas o con personas en listas negras publicadas por autoridades con jurisdicción sobre nuestras operaciones.
- Realizar operaciones que violen ordenamientos que pongan a la empresa en riesgo de ser sancionada. ISBIT por ejemplo se tiene que someter a lo establecido en la LFPIORPI para evitar sanciones de la autoridad.

Riesgo Cibernetico

- Riesgos de ataques cibernéticos, tales abusos al sistema para realizar retiros de dinero no autorizados por clientes.
- Robo de información patrimonial/financiera sensible de clientes
- Robo de secretos como contraseñas que podrían ser utilizados, por ejemplo,
- para aplicar un deposito inexistente aumentando el saldo en la cuenta de un cliente de manera incorrecta.

Riesgo de Mercado

- La imposibilidad de cerrar una operación cambiara de cobertura (“Hedge” en ingles) con divisas o activos virtuales causada por un retrado al transferir fondos de una institución financiera a otra es un riesgo que podemos limitar automatizando el proceso con un robot.

Por ejemplo, para que ISBIT pueda mitigar su Riesgo Regulatorio y de Cumplimiento, es importante que VULCANO sea capaz de recabar la información sobre la identidad (nombre, registro federal de contribuyente, código postal) asociada a las cuentas bancarias desde las que recibe pagos o a las que dispersa pagos.

Alineado con lo anterior para controlar el Riesgo de Cumplimiento, ISBIT tiene como política interna solo autorizar depósitos y retiros de moneda FIAT en la plataforma de intercambio de activos virtuales en los casos en los que la identidad del titular del contrato en la plataforma ISBIT coincida con la identidad del titular de la cuenta bancaria utilizada como origen o destino de fondos. Es decir, Si existe una persona de nombre Alicia, titular de una cuenta/contrato en ISBIT (https://isbit.co), VULCANO debe de bloquear/devolver depósitos dirigidos a la cuenta ISBIT de “Alicia”, desde cuentas bancarias distintas, a nombre de “Beto” o cualquier persona diferente a la titular, de manera automatica. De igual manera, VULCANO debe bloquear retiros de los fondos custodiados ISBIT a nombre de “Alicia”, a cuentas en bancos a nombre de persona de nombre diferente “Catalina” o cualquier otra persona diferente de la titular, de manea automatica.

II. Aumentar Velocidad de las Operaciones

La implementación del sistema VULCANO debe contribuir a agilizar los siguientes procesos:

- Aumentar la velocidad de conciliación de pagos recibidos. El tiempo que tarda en llegar una orden de pago es muy importante para los clientes: entre más rápido sea la acreditación del pago, más eficiente será el servicio. Lo que buscamos principalmente es que el tiempo de espera sea mejor que en el escenario que involucra a cualquier intermediario (tal como las entidades conocidas en la industria como “agregador de pagos” o “pasarela de pagos”).
- Aumentar la velocidad de dispersiones y envio de ordenes de pago (giros bancarios, nacionales, así como internacionales).
- Aumentar la velocidad y agailidad del proceso de auditar la tesoreria para Cumplimiento Fiscal
- Aumentar la velocidad y agilidad del procesod e auditar la tesoreria para Cumplimiento en materila de Prevencion de Lavado de Dinero, de tal forma que sea viable realizar auditorias cada 24 horas de manera automatizada sin causar fatiga a las áreas relacionadas con PLD.

La automatización de procesos no es un fin en si mismo, sino un medio para lograr que las actividades que se tienen que realizar de manera repetitiva y monótona ocurran de manera confiable y mas rápido que si fueran realizadas por un operador humano.

III. Aumentar la Transparencia

El equipo de cumplimiento de ISBIT necesita transparencia en tiempo real sobre eventos que impactan la tesoreria para realizar sus funciones de auditoría y así cumplir con las disposiciones oficiales en materia de PLD.

De acuerdo con el Art. 17, Fracción XVI de la LFPIORPI, se entenderá como Actividad Vulnerable y, por lo tanto, objeto de identificación *“El ofrecimiento habitual y profesional de intercambio de activos virtuales por parte de sujetos distintos a las Entidades Financieras, que se lleven a cabo a través de plataformas electrónicas, digitales o similares, que administren u operen, facilitando o realizando operaciones de compra o venta de dichos activos propiedad de sus clientes o bien, proveen medios para custodiar, almacenar, o transferir activos virtuales distintos a los reconocidos por el Banco de México en términos de la Ley para Regular las Instituciones de Tecnología Financiera”.*

A partir del 03 de febrero del 2020, todos los Sujetos Obligados operando con Activos Virtuales debieron realizar el trámite de alta y registro como Actividad Vulnerable, para efectos de la Ley Federal de Prevención e Identificación de Operaciones con Recursos de Procedencia Ilícita (LFPIORPI) ante el Servicio de Administración Tributaria, mediante el SPPLD (Sistema del Portal de Prevención de Lavado de Dinero: https://sppld.sat.gob.mx/pld/interiores/sppld.html) y adicionalmente notificar a la SAT por conducto de la oficiala de partes de la AGAFF (Administración General Auditoria Fiscal Federal) mediante un oficio.

Tratándose de persona moral, a partir de dicha fecha, también se debe designar a un representante encargado de cumplimiento, en términos del artículo 20 de la mencionada Ley (LFPIORPI).

En conformidad con el penultimo párrafo del articulo 17 de la LFPIORPI, a partir de las operaciones realizadas el 02 de abril del 2020, se deben presentar Avisos a más tardar el 17 del mes siguiente en el que se realizó el acto u operación a la Unidad de Inteligencia Financiera (UIF) por conducto del SAT, cuando el monto de las operaciones que realice el cliente superen el umbral: acumulación de operaciones igual o mayor a 645 Unidades de Medida y Actualizacion en un periodo de 6 meses, desde el ultimo corte al proceso de acumulación.

La herramienta VULCANO, debe facilitar al equipo encargado de cumplimiento de ISBIT SA DE CV, el fácil acceso a la información de las transacciones realizadas mediante base de datos indexada y clasificarlas para la generación automatizada de reportes en tiempo y forma.

VULCANO debe exponer un API a METIS, el Sistema Automatizado utilizado para monitorear transacciones y generar alertas relacionadas con LD y FT.

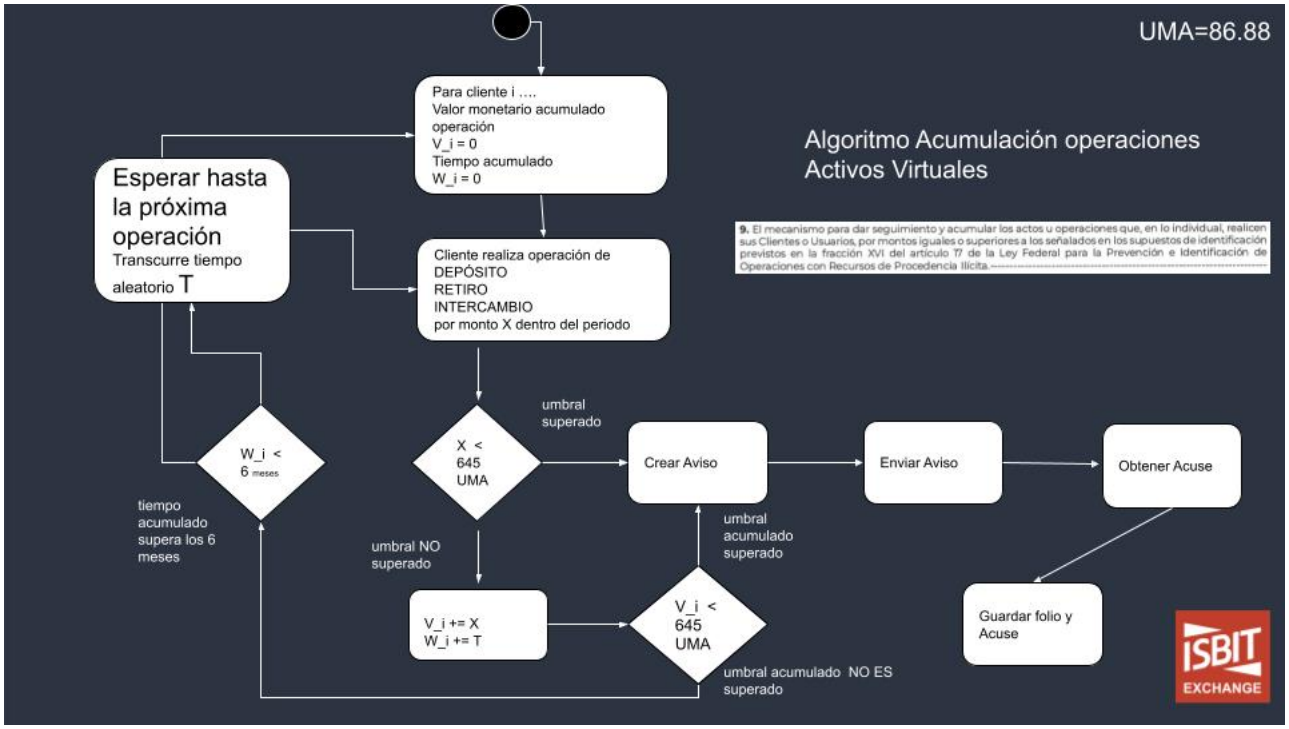


Ilustración Algoritmo Acumulación operaciones Activos Virtuales

IV. Aumentar redundancia, tolerancia a fallos para garantizar disponibilidad

Leyva, J., & Soto, G. (1 Junio 2018) nos relatan como durante el mes de abril de 2018, varias instituciones bancarias del sistema financiero mexicano fueron vulneradas por ciberataques donde los delincuentes extrajeron aproximadamente 3 millones de pesos de KUSPIT y BANJERCITO, 145 millones de pesos de Banorte y 150 de Inbursa. Durante varios días las instituciones afectadas, suspendieron operaciones o las realizan con lentitud al operar en modalidad de contingencia.

La crisis demostró la importancia para cualquier empresa de tener canales transaccionales con múltiples bancos simultáneamente para no suspender operaciones en ninguna circunstancia.

Debido a la experiencia de intermitencias y retrasos en los bancos, se tomó la decisión de incorporar en los objetivos de diseño de VULCANO la capacidad de operar con múltiples bancos simultáneamente.

V. Aumentar interoperabilidad, facilidad de instalación, mantenimiento y evolución futura del sistema

Un análisis de la oferta de servicios de Banca Transaccional de diferentes instituciones financieras mexicanas demuestra como el camino a la integración vía H2H presenta grandes diferencias entre una institución y las demás.

Un objetivo importante en el diseño de VULCANO es proporcionar a las aplicaciones cliente, una API que, de forma transparente, traduzca los mensajes a los protocolos y formatos requeridos por cada institución.

VI. Aumentar la desintermediación de los procesos de cobro y pago

Al quitar a los agregadores de pago y permitir a las empresas que usan VULCANO operar directamente con los bancos, se pueden reducir las comisiones relacionadas a la cobranza. En el siguiente esquema se muestra la comparación de hacer pagos mediante un agregador de pagos vs VULCANO.

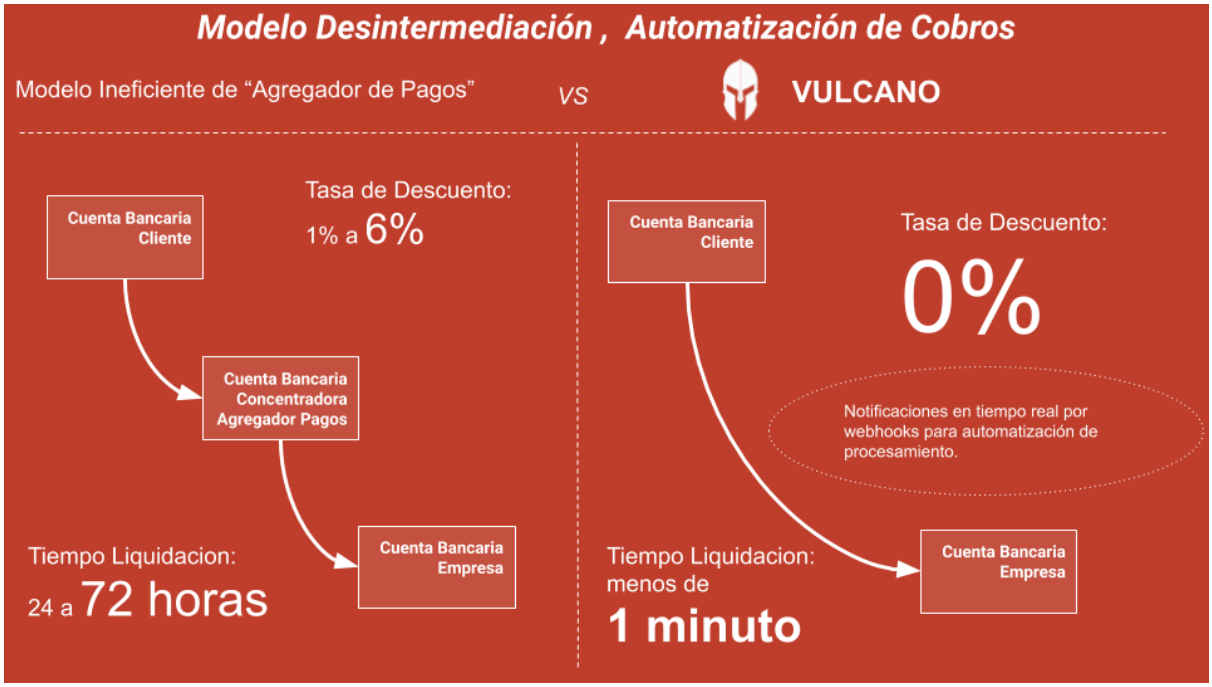


Ilustración 1 Modelo Desentirmediación, Automatización de cobros

Podemos ver que las principales diferencias serían el tiempo de liquidación y el porcentaje de tasa por el servicio.

3. Desarrollo del trabajo

Gestión de proyecto

Es importante que en un principio se tome en cuenta el organigrama de la empresa ISBIT, con la finalidad de que se visualice el papel que toman en el proyecto VULCANO las áreas correspondientes.

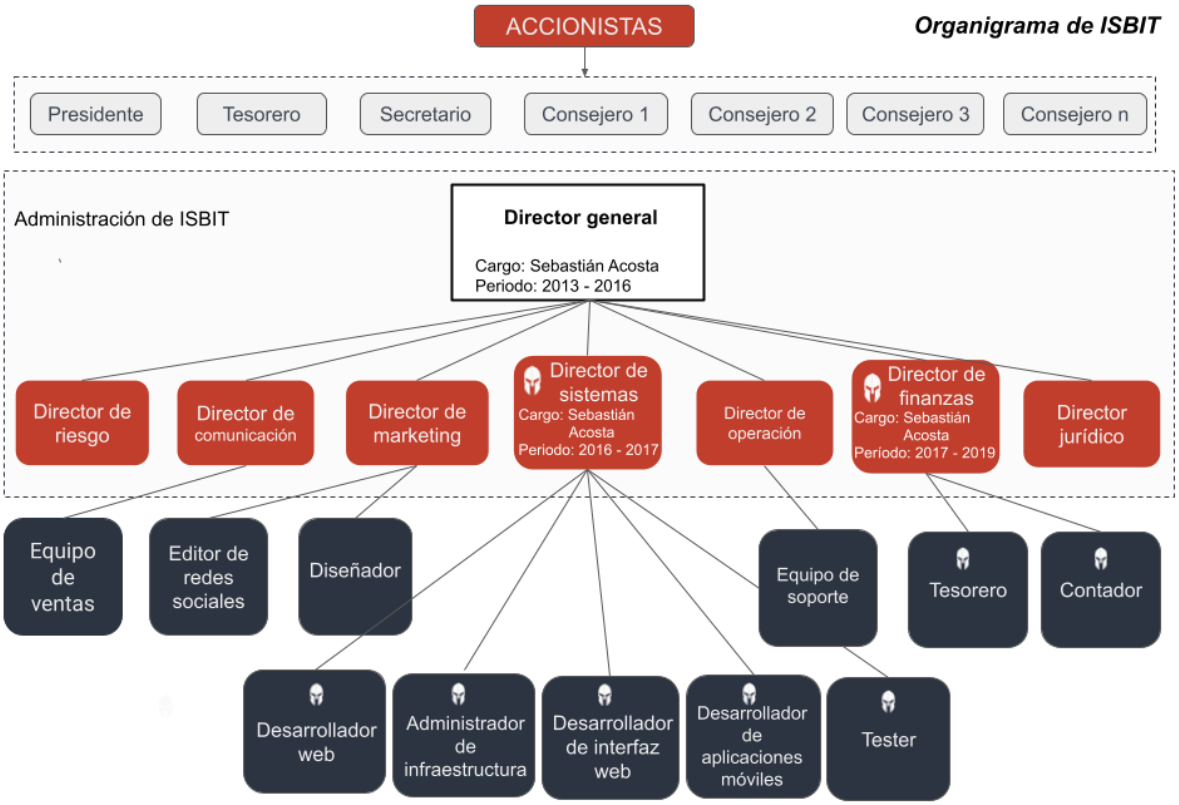


Ilustración 2 Organigrama de ISBIT

Una vez ubicada la participación de VULCANO en la empresa, explicaré la gestión del proyecto.

La metodología usada para la administración del proyecto “VULCANO” fue **Scrum**, la cual consiste en un marco teórico para desarrollar y mantener productos complejos especialmente el desarrollo de software. De acuerdo con los creadores de esta metodología (Schwaber, K., & Sutherland, J., 2010) la gestión de un proyecto Scrum se centra en definir cuáles son las características que debe tener el producto a construir (qué construir, qué no y en qué orden) y en vencer cualquier obstáculo que pudiera entorpecer la tarea del equipo de desarrollo. Los proyectos avanzan a través de una serie de iteraciones llamadas sprints.

Los elementos del marco Scrum son los siguientes:

Equipo Scrum: un equipo Scrum regularmente tiene entre cinco y nueve personas, aunque puede ser utilizado fácilmente por equipos de una persona; este equipo no incluye ninguno de los roles tradicionales de ingeniería de software, como programador, diseñador, probador o arquitecto. Todos en el proyecto trabajan colectivamente para completar lo solicitado de un sprint.

Pila de productos: es una lista de funciones priorizadas que contiene todas las funciones o cambios deseados en el producto. La cartera de productos es una lista de características deseadas, el backlog del sprint es una lista de tareas que se deben completar en un sprint.

Reunión de planificación de Sprint: al comienzo de cada Sprint, se lleva a cabo una reunión de planificación de Sprint, durante la cual el propietario del producto presenta los elementos principales de la cartera de pedidos del producto al equipo. El equipo de Scrum selecciona el trabajo que pueden completar durante el próximo sprint. Luego, ese trabajo se mueve de la lista de trabajos pendientes del producto a una lista de tareas pendientes, que es la lista de tareas necesarias para completar los elementos de la lista de productos pendientes que el equipo se ha comprometido a completar en el sprint.

Scrum diario: cada día durante el sprint, se lleva a cabo una breve reunión que ayuda a establecer el contexto para el trabajo de cada día y ayuda al equipo a mantenerse encaminado. Todos los miembros del equipo deben asistir al scrum diario.

Reunión de revisión de sprint: al final de cada sprint, el equipo demuestra la funcionalidad completa en una reunión de revisión de sprint, durante la cual, el equipo muestra lo que lograron durante el sprint.

Sprint retrospectiva: al final de cada sprint, el equipo lleva a cabo una retrospectiva, que es una reunión durante la cual todo el equipo reflexiona sobre las mejoras y los cambios que en conjunto desean para mejorar su funcionamiento

Los roles de esta metodología son los siguientes:

Product Owner (o Propietario del producto): se asegura de que el equipo Scrum trabaje de forma adecuada desde la perspectiva del negocio.

Durante una etapa del proyecto asumí el Rol SCRUM de **"Product Owner":** me aseguré de que el equipo trabajara de forma adecuada y definí la visión del proyecto, fue mi responsabilidad que las tareas acordadas en cada sprint se llevaran a cabo. Cada sprint (iteración) consistió en tener como resultado una nueva versión del software **VULCANO** totalmente operativo con nuevas funciones y pruebas unitarias, pruebas funcionales y pruebas de integración.

Scrum Master (o Facilitador): Es el responsable del cumplimiento de las reglas del marco scrum. Se asegura que estas son entendidas por la organización y de que se realiza el trabajo conforme a ellas. Elimina los obstáculos que impiden que se desarrolle el objetivo del *sprint*. Asesora y da la formación necesaria al propietario del producto y al equipo de desarrolladores.

Desarrollador: Cada uno de los profesionales que realizan la entrega del incremento de producto generado en cada sprint (denominado incremento). Es recomendable un pequeño equipo de 3 a 9 personas con las habilidades transversales necesarias para realizar el trabajo (análisis, diseño, desarrollo, pruebas, documentación, etc).

Los desarrolladores del proyecto VULCANO éramos cinco personas; fue indispensable enfocarnos en el análisis, diseño, desarrollo, pruebas y documentación. Era responsabilidad de cada uno de nosotros realizar entregas del desarrollo en el tiempo acordado.

Auxiliares, Stakeholders (Clientes, Proveedores, Vendedores, etc): son aquellos que no tienen un rol formal y no se involucran frecuentemente en el "proceso Scrum", sin embargo, deben ser tomados en cuenta. Los stakeholders son las personas que hacen posible el proyecto y para quienes el proyecto producirá el beneficio acordado que justifica su desarrollo. Solo participan directamente durante las revisiones del sprint.

Durante ciertas etapas del proyecto vulcano también tuve que asumir el rol de **Auxiliar:** como papel auxiliar, apoyé como stakeholder y aporte conocimiento sobre los requerimientos de las Areas de Tesorería y Cumplimiento de la empresa.

Sistemas que interactúan con la tesorería de ISBIT; Exchange, ERM, METIS, CAERUS.

Recordemos que VULCANO es un sistema débilmente acoplado por lo que interactúa de cierta manera con cuatro sistemas que ayudan a la empresa a realizar diferentes tareas relacionadas. Estos sistemas son los siguientes:

1. **EXCHANGE:** Es una aplicación en la nube que los clientes de ISBIT pueden usar para comprar y vender Activos Virtuales. EXCHANGE requiere procesar en tiempo real depósitos de dinero FIAT (ejemplo pesos mexicanos, dólares) de clientes que constituyen el colateral con el que van a comprar alguna cantidad de Activos Virtuales. EXCHANGE también requiere enviar en tiempo real órdenes de pago por diferentes canales tales como SPEI, SPID, SWIFT, TEF y transferencia mismo banco para liquidar obligaciones con clientes a los que hay que entregar los recursos obtenidos por ellos al vender los Activos Virtuales de los que son titulares la plataforma ISBIT.

Ilustración 3 Aplicación EXCHANGE

2. **ERP** (software contable y de generación de reportes financieros y fiscales): requiere obtener información de las operaciones en la tesorería para ligarlas a pólizas contables, así como comprobantes fiscales digitales por internet (CFDI).

3. **METIS** (software orientado a las tareas de prevención de lavado de dinero, prevención de financiamiento del terrorismo, prevención de fraude y robo de identidad, así como Conocimiento de Cliente, auditoría y cumplimiento regulatorio). METIS cruza la información transaccional de la tesorería con diferentes fuentes tales como listas negras, bases de datos de domicilios y documentos de identidad para detectar y prevenir LD y FT en la plataforma digital de ISBIT. METIS es el software que con información proporcionada por vulcano es responsable del envio de Avisos a la unidad de Inteligencia Financiera. METIS

puede solicitar a VULCANO enviar un pago de una cantidad insignificante (un centavo) a la cuenta bancaria que un usuario tiene registrada en ISBIT para el retiro de sus fondos e inmediatamente después obtener el Comprobante Electrónico de Pago CEP (ver en **Anexo A** ejemplo de cómo funciona el CEP) de Banxico donde están contenidos datos como nombre verdadero y RFC asociados a la cuenta bancaria. Si los datos en el CEP no coinciden con los datos reportados por el cliente al registrarse en ISBIT se podría tratar de Robo de Identidad o de una transacción con terceros no permitida, ya que ISBIT tiene la política de solo permitir transacciones donde la identidad de el titular de la cuenta en el EXCHANGE y la identidad asociada a la cuenta en el banco, coinciden para prevenir lavado de dinero. Es decir, mediante VULCANO, METIS puede verificar la identidad bancaria de un cliente de ISBIT.

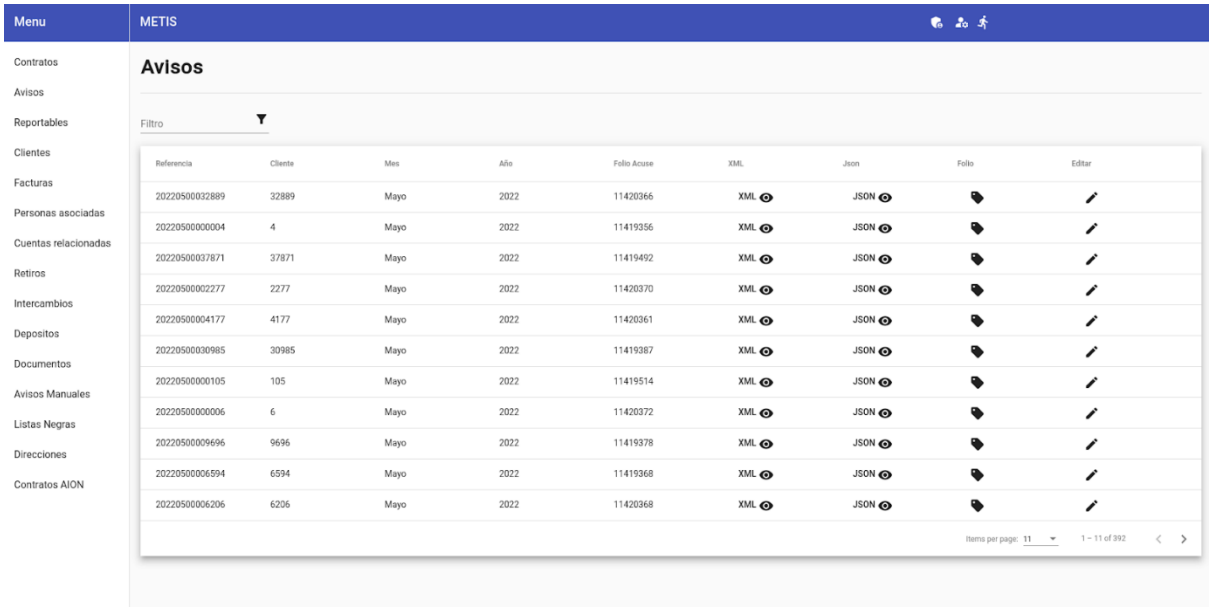


Ilustración 4 METIS, Sistema Automatizado PLD y reportes UIF

4. **CAERUS** denomina al "Robot" que utilizamos en la empresa para realizar ejecutar una combinación de estrategias de Formación de Mercado ("Market Making") y Abritraje Financiero con el fin de garantizar que exista liquidiez en el Libro de Ordenes de la plataforma ISBIT. El componente de "Market Making" esta basado en las Ideas presentadas por Avellaneda, M., & Stoikov, S. (2006). El componente de Arbitraje Financiero de CAERUS coloca ordenes contrarias (coberturas) en mercados que están en diferentes plataformas e inclusive diferentes países. Periódicamente es necesario balancear el portafolio cuando se acumula una moneda nacional o Activo Virtual en algún mercado en particular. Aquí es donde VULCANO tiene la función de recibir instrucciones de CAERUS para automatizar el envio de transferencias internacionales y domesticas necesarias para recuperar el equilibrio del portafolio mediante un balanceo oportuno.

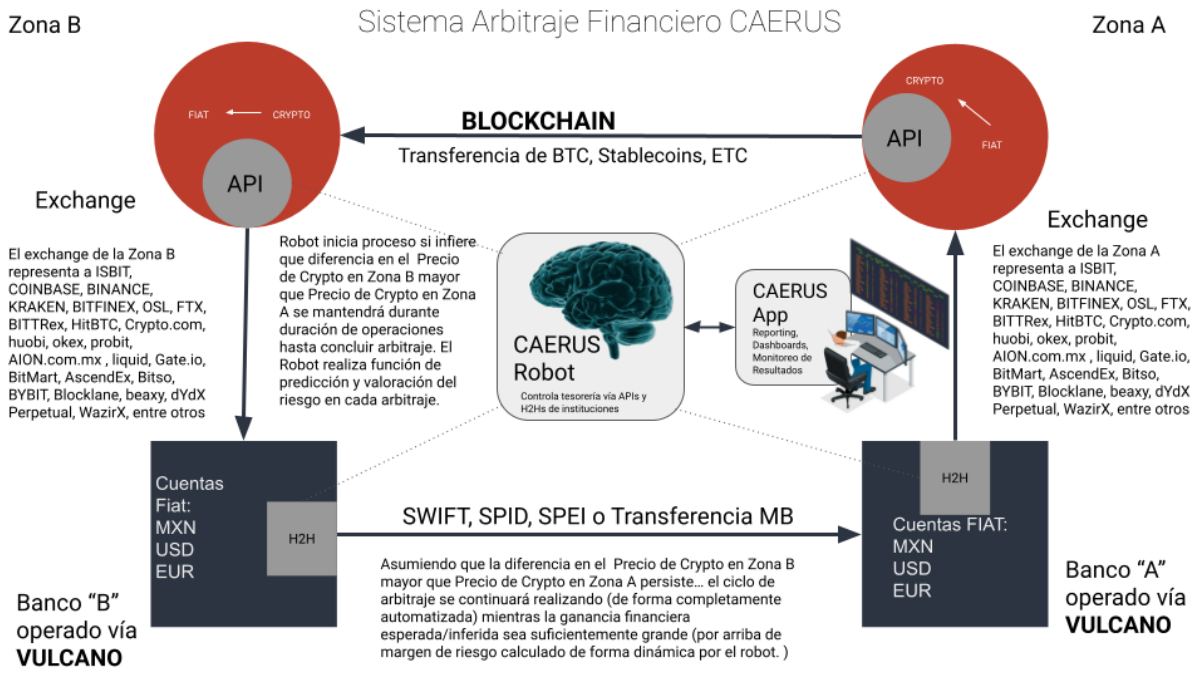


Ilustración 5 CAERUS, Robot Automatización Arbitraje Financiero

Definición de casos de uso y el dominio del problema

El producto que algunos bancos ofrecen para ayudar a las empresas con necesidad de automatizar las operaciones de su tesoreria es comúnmente conocido en la industria bajo el nombre "H2H" o "Host to Host". La razón por la que este servicio de Administración de Efectivo es conocido bajo ese nombre se debe a que este tipo de enlaces requieren establecer una red VPN entre una maquina de la empresa y una maquina del banco. La arquitectura H2H mas común es aquella en la que la empresa y el banco intercambian archivos encriptados por PKI (Public Key Infraestructure) mediante el protocolo **SFTP** (Secure File Transfer Protocol).

Entre las ventajas de implementar un sistema H2H destacan:

- *Integra la operación de la tesorería coportativa a una conexión directa entre El Banco y la empresa (ERP)*
- *Procesa grandes volúmenes de transacciones*
- *Realizar conciliaciones sin intervención manual.*
- *Girar instrucciones de pago en directo desde de ERP (o apps como METIS, CAERUS).*
- *Recibir notificaciones de pago vía email para el pagador y beneficiario.*
- *Obténer reportes en Swift, BAI2, Santander, etc.*
- *Realizar reintentos automáticos en operaciones mismo Banco.*
- *Acceder a mensajería SWIFT (MT940 y 942).*
- *Monitor en línea la gestión de operaciones y accesos.*

Un sistema HOST TO HOST (H2H) implementado correctamente resuelve las ineficiencias y riesgos asociados al manejo de tesorería y sus diferentes operaciones (cobranza, dispersiones, envío de pagos, auditoria) de empresas asociadas a diversos sectores que

requieren automatización de procesos incluyendo, pero no limitandose a las empresas FinTech, e-commerce, sector asegurador, financieras entre otras.

La gran mayoría de las empresas carecen de una conexión “Host to Host” directa con la banca. El no contar un H2H no permite que el departamento de tesorería realice de forma automatizada un alto volumen de transacciones. Los requisitos de infraestructura y capital humano para implementar una conexión “Host to Host” con Instituciones financieras representan una importante barrera de entrada. A nuestra empresa le tomó un periodo no menor a doce meses la implementación de H2H por cada banco. La industria demanda una herramienta capaz reducir el costo de tiempo (horas hombre) y recursos monetarios para la implementación de un H2H. Una herramienta que facilite la interconexión con la banca podría catalizar la innovación financiera en las empresas.

Algunas empresas utilizan intermediarios (agregadores, pasarelas de pago) que no sólo limitan la agilidad de las transacciones, sino que cobran comisiones altas por cada transacción procesada. VULCANO, no es un agregador de pagos o pasarela de pagos. VULCANO es una solución completa que abarca la infraestructura de software y hardware dedicada necesaria para la conexión directa entre la empresa y diferentes bancos, en un ambiente multi-canal. El proyecto VULCANO busca combinar un modelo de desintermediación con la tolerancia de fallos obtenida mediante redunancia de canales transaccionales.

Se plantea “Vulcano” como un software que permite a las empresas operar sus cuentas bancarias mediante un sistema integrado de manejo de tesorería que implementa enlaces seguros “Host to Host” entre los equipos de computo de la tesorería de las empresas y múltiples bancos. Vulcano, permite al grupo empresarial administrar con un robot, sus múltiples cuentas bancarias en distinitas instituciones y recibir notificaciones en tiempo real de pagos y cobros, además, permite la disponibilidad inmediata de los fondos en las cuentas bancarias de la empresa ya que en ningún momento son retenidos o retrasados por los intermediarios (tales como agregadores o procesadores de pagos). Se busca desarrollar un modelo de desintermediación seguro, rápido y de bajo costo en su implementación y mantenimiento.

La herramienta VULCANO debe ser capaz de abarcar los siguientes casos de uso de ISBIT SA DE CV:

- Enviar pagos SPEI, SWIFT, SPID, NOMINA, TEF, MISMO BANCO
- Automatizar pagos de servicios
- Facilitar que el robot, ERP y aplicaciones de la empresa como EXCHANGE, METIS o CAERUS, se puedan comunicar con VULCANO mediante una Interfaz de Programacion de Aplicaciones (API) simple
- VULCANO debe transmitir de forma segura a los servidores de cada banco las instrucciones/ordenes mediante los protocolos “Legacy” correspondientes,

para controlar todos los aspectos de la tesorería.
- Obtener notificaciones en tiempo real de movimientos en las múltiples cuentas del grupo empresarial del que forma parte ISBIT SA DE CV

Análisis del Problema

Vulcano permite automatizar las tareas relacionadas con la tesorería en diferentes áreas de la empresa (ISBIT SA DE CV) para reducir los problemas y riesgos introducidos por la realización de tareas de manera manual por operadores humanos.

La empresa tiene un sitio de internet (EXCHANGE) que a su vez utiliza VULCANO para recibir y procesar pagos de forma automatizada de los clientes que realizan operaciones de compraventa de Activos Virtuales desde la comodidad de sus computadoras o celulares inteligentes.

Por ejemplo, una vez que VULCANO detecta (por medio de un numero de referencia) que una ficha de deposito está pagada, envía notificaciones que desencadenan de manera automática los siguientes procesos:

- La acreditación (conciliación) del pago de tal manera que el usuario ve reflajado su saldo actualizado en la interfaz de la plataforma digital
- La generación de asientos contables de la operación en contabilidad (ERP)
- Generación de la factura CFDI (Comprobante Fiscal Digital por Internet) correspondiente a la operación
- Envio de correo al cliente notificando de la acreditación (o rechazo) del deposito
- La información de la operación también debe ser enviada a METIS en tiempo real. En caso de superación de umbral acumulado en periodo correspondiente, METIS procede de manera automatizada a la generación y envio de aviso establecido para la Actividad Vulnerable en formato XML por conducto del SPPLD (Sistema del Portal de Internet para Prevención de Lavado De Dinero) designado por la UIF (Unida de Inteligencia Financiera). Es importante enviar avisos en tiempo y forma ya que la suma de las multas por envíos extemporáneos y omisiones son onerosas.

De este modo el riesgo de manejo de procesos de Tesoreria críticos por humanos que pueden cometer errores por negligencia, impericia o actuar con dolo o mala fe es mitigado.

Se busca implementar **VULCANO** para auditar operaciones y automatizar procesos de cumplimiento y prevención de lavado de dinero. Las aplicaciones ligadas a estas áreas de la empresa pueden ingresar a Vulcano con permiso de lectura, pero no de escritura

mediante control RBAC (“Role Based Access Control”) de Vulcano para adquirir esta información en tiempo real. En el anexo B hay más información acerca del funcionamiento RBAC de VULCANO.

Análisis de la solución del problema

El caso de uso de envio de instrucciones de pago deberá ser implementado por VULCANO a través de una serie de procesos débilmente acomplados:

1. Vincular al robot encargado de envio de instrucciones de pago mediante **API JSON REST** de cara a la tesorería de la empresa.
2. Validar instrucción de pago enviado por el robot de la tesorería de la empresa.
3. Guardar la información de la transacción en la base de datos MongoDB.
4. Generar layout con formato de CECOBAN con la información extraída de la base de datos.
5. Encriptar layout con formato de CECOBAN utilizando llaves privadas que serán resguardadas en Azure Key vault (Hardware Security Module -HSM- en la nube de microsoft).
6. Establecer conexión privada entre el servidor de la empresa y del banco a través de Vulcano. El robot enviará un layout encriptado mediante el túnel creado entre el cortafuegos de vulcano (la empresa) y el cortafuegos del servidor del banco. El layout va destinado a un servidor Secure File Transfer Protocol (SFTP) dentro de la infraestructura del banco.
7. Recuperar archivo de salida del servidor SFTP del banco y desencriptado del archivo
8. Parsear archivo para extraer resultado de procesamiento.
9. Actualizar base de Datos con el resultado del procesamiento de layout (rechazado, error, éxito, en cola)
10. Envio de notificaciones con cambio en estatus de pago enviado

Para obtener información de la transacción Vulcano realiza los siguientes pasos inversos en el proceso para la obtención del comprobante bancario (de forma transparente o “invisible” para el usuario).

1.- El robot deberá obtener y descargar el “layout” con resultados del procesamiento devuelto por el banco con el estatus de las transacciones.
2.-El archivo de salida del banco es recuperado mediante el túnel VPN (establecido con ayuda del cortafuegos de la empresa): descargado del servidor SFTP del banco y guardado en AFS (Azure File Storage).
3.-Desencriptado de archivo de respuesta del banco mediante las claves privadas almacenadas en AKV (Azure Key vault).
4.-Enviar notificaciones con resultado de las transacciones a todos los sistemas suscritos (vía webhooks).
5.-Recuperar comprobantes de cada transacción en estados de cuenta (CEP de Banxico en

formato XML y PDF en caso de transacciones tipo SPEI) y almacenarlos junto con representación JSON en la base de datos para consultas y búsquedas futuras.

Ahora platiquemos como el sistema VULCANO debe ayudar a gestionar riesgos en el manejo de la Tesoreria. A continuación, se describe cómo funciona el proceso propuesto para detectar y prevenir casos de Robo de Identidad, Fraude y Lavado de Dinero.

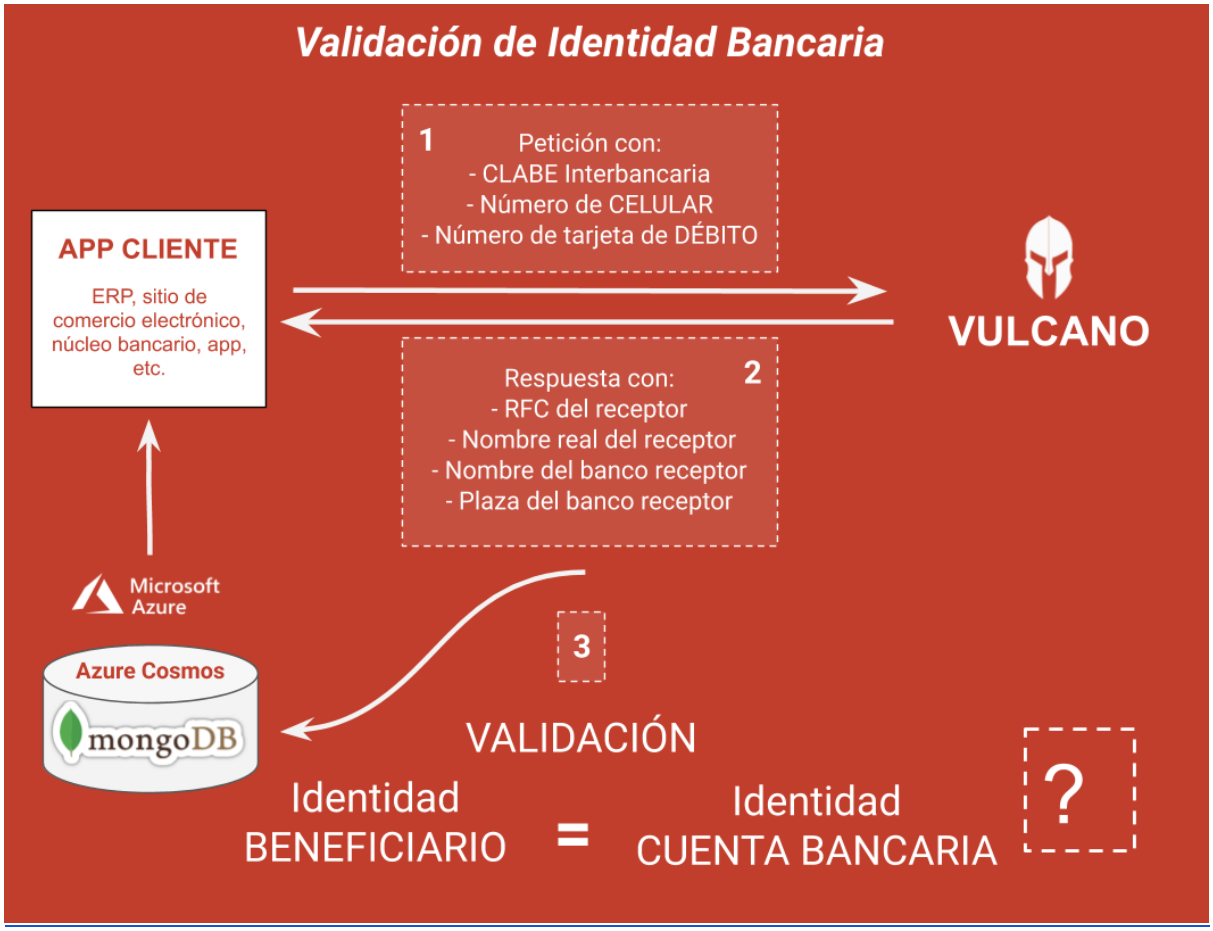


Ilustración 6 Validación de Identidad Bancaria

VULCANO obtiene la información de Identidad asociada a una cuenta bancaria dada (RFC/Curp, Nombre del Titular de la Cuenta, Nombre del banco receptor) realizando el proceso descrito en el diagrama **"Validacion de Identidad Bancaria"** anterior. El proceso consiste en enviar una transferencia SPEI (generalmente una cantidad pequeña, un centavo) a la cuenta bancaria que necesitamos identificar. Posteriormente se recaba el CEP (Comprobante Electronico de Pago) de dicha transacción. Finalmente, se parsea la información contenida en los CEP del Banco de México y se almacenan los datos de identidad asociados a la cuenta contenidos en dicho documento y se guardan en base de datos. El proceso de obtener CEPs, descargarlos, guardarlos, procesarlos y analizarlos es

tedioso de realizar de manera manual, por lo que VULCANO debe ser capaz de automatizar estas tareas como parte de la solución.

El reto para obtener los CEP de banco de México es la ausencia de una API estandarizada. Por tal motivo es necesario crear un robot conocido en la industria como "Araña" o *web scraper (como definido en Mitchell R., 2015)* para para llenar los formularios web del portal público de banxico y recuperar los datos de interés, actuando como Agente Usuario en el contexto de un *Navegador sin cabeza* (en ingles "headless browser"). Los navegadores utilizados por el algoritmo de VULCANO para recuperar CEPs de Banxico fueron PhantomJS y CasperJS.

Nuestro Agente Usuario es un programa de Javascript que es afectado por una latencia significativa al enlazarse al portal de Banxico atraves de internet. La necesidad de volver a intentar la petición a Banxico en caso de intermitencia implica, por tanto que es necesario ejecutar en **paralelo** y **simultáneamente** varios procesos de Agente Usuario para obtener un número grande de CEPs en un tiempo razonable. La arquitectura de procesamiento paralelo propuesta es ilustrada en el Diagrama siguiente:

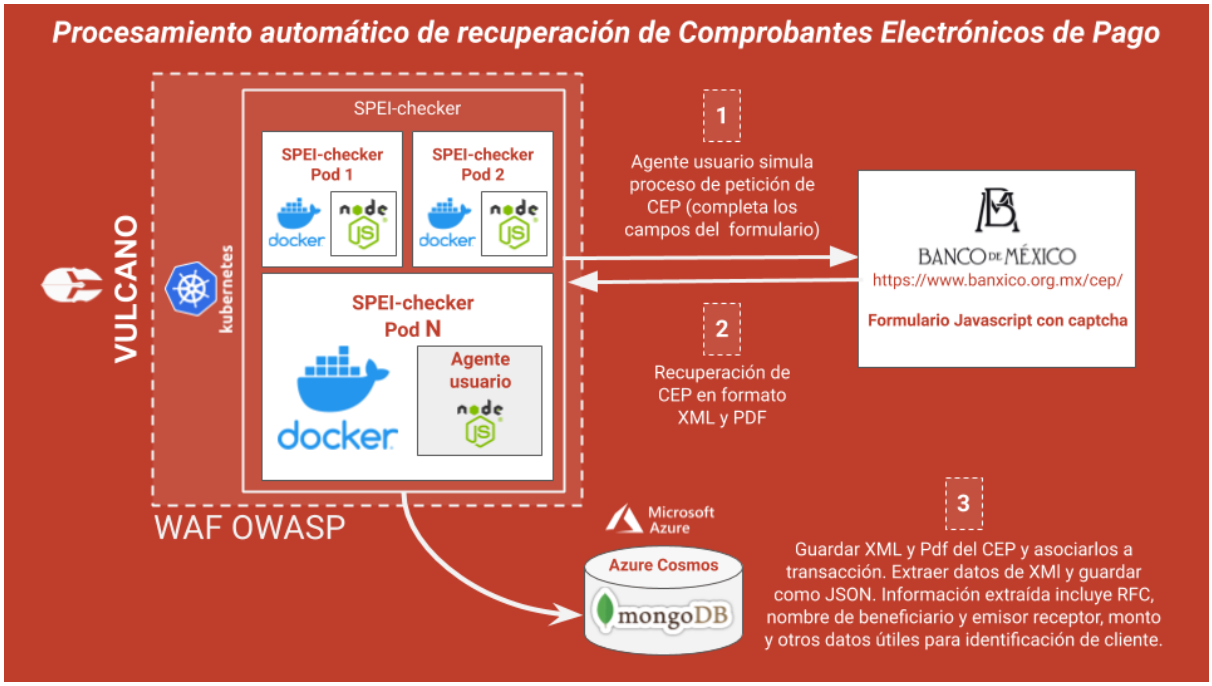


Ilustración 7 Procesamiento paralelo de recuperación de Comprobantes Electronicos de Pago

En el diagrama anterior hay muchos contenedores de docker que corren el proceso "Web Scraper" basado en NODE JS y en el navegador sin cabeza ***PhantomJS***. Internamente nos referimos al algoritmo de recuperación de CEPs de transacciones enviadas al SPEI como ***"SPEI-checker"*** en la ilustración de arriba. De manera análoga, al algoritmo de recuperación de CEPs de transacciones con dólares enviados al SPID, lo nombramos ***"SPID-checker"***. Hay varios retos relacionados con la implementación de ***SPEI-checker***. El proceso de

obtención de un CEPs individual es tardado ya que implica muchas peticiones web a un sitio con alta latencia (la pagina web de BANXICO). Para mitigar este problema el *clúster* de kubernetes crea 1,2, …, *N* instancias del algorimo como es mostrado en el diagrama de arriba. El numero de contenedores, *N*, debe ser variable y controlado de manera dinámica por el Orquestador dependiendo de la carga de trabajo.

Otro reto importante en la implementación del algoritmo ***SPEI-checker***, esta relacionado con el hecho de que los CEPs normalmente no están disponibles de manera inmediata después de enviar un SPEI. El tiempo *T* que toma un CEP en estar disponible en el portal de Banxico es totalmente aleatorio y desproporcionadamente mas lento que el tiempo que toma la liquidación. La liquidación, de hecho, es bastante rápida en la mayoría de los casos:

> *"El índice de disponibilidad del SPEI durante su horario de operación para 2015 fue de 99.98 por ciento. Por su parte, el tiempo promedio para liquidar un pago fue de 1.9 segundos" (Banco de Mexico, marzo 2016)*

La distribución del tiempo de espera *T* para que el CEP de una transacción este disponible en el portal de BANXICO podría ser modelado de manera adecuada con una distribución de probabilidad continua y de cola pesada con dominio en los reales positvos.

En base a experiencia empírica de trabajar con miles de CEPs, la empresa ISBIT ha recabado datos que demuestran que existe una alta varianza ***var(T)*** en el tiempo que toma un CEP en estar disponible en el sistema a partir del envio de una transacción al SPEI. Usualmente toma pocos segundos, pero en algunos casos puede llegar a tomar varias horas o inclusive varios días. Esto se debe a ha que el CEP esta disponible hasta que el banco receptor notifica a Banxico, lo cual a su vez depende de la congestión en los sistemas del banco receptor que pudiera retrasar la notificación de confirmación a enviada a BANXICO, retrasos por transacciones impactadas por monitoreo de PLD/FT del banco receptor, entre otros factores. ISBIT, conforme ha su experiencia empírica propia, ha observado cientos de casos en los que a pesar de que los fondos de una transacción enviada o recibida son acreditados en tiempo record de pocos segundos, el CEP no esta disponible sino hasta dentro de varias horas o varios días. Inclusive podemos hablar de un tiempo de obtención de CEP "infinito", para efectos de modelado matemático de la distribución, en los casos en que el CEP nunca es generado ya que el banco receptor no confirma la transacción por cualquier motivo (Cuando el estado es Cancelado, Rechazado, En proceso de devolución, Devuelto, No Liquidado).

La siguiente tabla, construida con base en la información publicada por Banxico, indica los estados de una transacción en el SPEI o SPID:

Estados de pago en transacciones realizadas por SPEI® y SPID®

Nota importante: La acreditación del pago en la cuenta del beneficiario y el envío de la información para generar el CEP correspondiente, es responsabilidad del banco beneficiario de la orden de transferencia

Estado de Pago	Descripcion del Estado de Pago	¿Absorbente?
En proceso	El SPEI® o SPID® ha recibido del banco ordenante la instrucción de pago, pero no la ha liquidado.	transitorio
Liquidado	El pago ha sido liquidado en el SPEI® o SPID® y éste ha enviado la notificación correspondiente al banco del beneficiario. En caso de que los recursos no estuvieran disponibles para el beneficiario éste podría presentar al banco receptor la impresión del estado del pago, con el fin de solicitar que se reconozca el monto en la cuenta beneficiaria.	Estado Transitorio
Cancelado	El pago fue recibido por el SPEI® o SPID®, sin embargo, fue cancelado por el banco que recibió la instrucción de pago de su cliente, previo a que fuera liquidado por el SPE	Estado

	SPID®. El banco emisor deberá reintegrar el importe del pago al titular de la cuenta ordenante. En caso de que esto no ocurra, el titular de la cuenta ordenante podrá presentar al banco emisor, la impresión del estado del pago para que le solicite que efectúe la reintegración del monto.	Absorbente
Rechazado	El pago fue rechazado por el SPEI® o SPID® debido a errores en la información que recibió de la institución emisora para procesar el pago o a un problema grave de segur El banco emisor deberá reintegrar el importe del pago al titular de la cuenta ordenante. En caso de que esto no ocurra, el titular de la cuenta ordenante podrá presentar al banco emisor, la impresión del estado del pago para que le solicite que efectúe la reintegra del monto.	Estado Absorbente
En proceso de devolución	El pago fue liquidado y posteriormente devuelto por el banco del beneficiario. La devolución se encuentra pendiente de liquidación.	Estado transitorio
Devuelto	El pago fue devuelto por el banco del beneficiario; dicha devolución fue liquidada por el SPEI® o SPID® e informada al banco que originalmente envió la orden de pago. Este último ya está en posibilidades de depositarlo en la cuenta del cliente que instruyó el pago. En caso de que el banco que originalmente envió la orden no realice el depósito, el titular de la cuenta ordenante podrá presentar al banco emisor, la impresión del estad pago para que le solicite que efectúe el depósito del monto.	Estado Absorbente
No Liquidado	El pago se recibió en el SPEI® o SPID® pero no pudo ser liquidado durante la jornada operativa respectiva y al cierre del sistema se eliminó. El banco del ordenante deberá reintegrar el monto del pago al cliente que le instruyó el mismo. En caso de que el banco ordenante no reintegre el monto del pago a su cliente, el titular de la cuenta ordenante podrá presentar al banco emisor, la impresión del estad pago para que le solicite que efectúe la reintegración del monto.	Estado Absorbente
No encontrado	El SPEI® o SPID® no ha recibido una orden de pago que cumpla con el criterio de búsqueda especificado.	Estado Transitorio
Cep generado	Existe un XML y PDF firmado por banco receptos. En este punto la transacción es irreversible, es decir final. Es en el único estado en que podemos estar 100% que los fo han sido acreditados en la cuenta bancaria del benefiario.	Estado Absorvente

Ilustración 8 Estados de pago en transacciones realizadas en el SPEI o SPID de Banxico

Por las razones anteriores pensamos que la distribución idónea para caracterizar la variable aleatoria tiempo de espera en este caso es una de "cola pesada" con soporte exclusivamente en los números positivos. Una distribución de "cola pesada" es tal que la función generadora de momentos es infinita, es decir

$$\int_{-\infty}^{\infty} e^{tx} F(x) = \infty \; para \; toda \; t > 0.$$

Se propone que los tiempos de espera están dados por la distribución **Log-Cauchy** la siguiente función de densidad:

$$f(x;\mu,\sigma) = \frac{1}{x\pi}\left[\frac{\sigma}{(\ln(x)-\mu)^2+\sigma^2}\right] \text{ para } \; x > 0$$

Durante el periodo comprendido desde enero hasta diciembre de 2018 la empresa realizó el registro de los tiempos de espera para recuperación de **243** CEPs iniciando la cuenta regresiva a partir del momento de la propagación de la transacción por el banco *emisor*. De la muestra de **243 CEPs** se tuvieron 5 devoluciones por distintas causas de error y 16 CEPs cuyo estatus en el SPEI fue "liquidada" o "en proceso de liquidación" pero nunca se obtuvo confirmación y generación del CEP a pesar del éxito del pago. Banxico no mantiene por tiempo indefinido los CEPs disponibles para ser descargados. Por lo que en la muestra nos limitamos a intentar recuperar CEPs hasta por un máximo de *H*=90 días. Entonces podemos pensar que se tuvieron 16+5=21 tiempos de espera "infinitos".

En base a los datos $(t_{1,}t_{2,},\ldots,t_{n=243,})$ empíricos recabados por la empresa se calcularon estimadores para los parámetros de ubicación y escala de la distribución. Este fue el resultado para el parámetro de ubicación:

$$\hat{\mu} = \begin{cases} \ln\left(t_{(n+1)/2}\right) & si \; n \; impar \\ \dfrac{\ln\left(t_{(n+1)/2}\right)+\ln\left(t_{\frac{n+1}{2}+1}\right)}{2} & si \; n \; par \end{cases} = 1.9797 \text{ segundos}$$

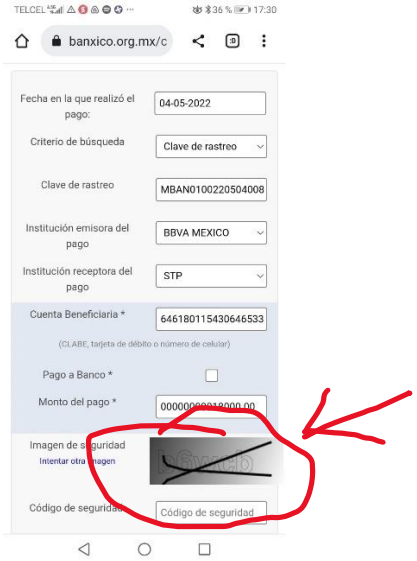
De lo anterior podemos concluir que la mediana de la distribución es $e^{1.9797} = 7.2405 \ldots$ segundos. Lo cual si concide con la experiencia de la empresa ya que la mitad de los CEPs se recuperaron en menos de 7.2405 segundos.

Por otro lado, el resultado para el parámetro de escala fue el siguiente:

$$\hat{\sigma} = media(|\ln(T_i) - \hat{\mu}|) = 22{,}767.84 \text{ segundos} = 6.3244 \text{ horas}$$

La consecuencia practica de la alta variabilidad en el tiempo de espera tiene la implicación que si queremos obtener el CEP en el menor tiempo posible, es necesario realizar múltiples intentos periódicamente hasta obtener una respuesta dedinitiva o

superar el *timeout.* El sitio de Banxico tiene un sistema para prevenir ataques de denegación de servicios, que despues de un numero (indeterminado) de intentos no exitosos para recuperar un CEP, solicita al usuario la solución de un "***captcha"*** como se puede apreciar en la siguiente imagen:



En los casos en que el formulario requiera de un captcha para ser enviado, nuestro algoritmo/robot (agente usuario ***phantomjs***) debe ser capaz de reconocer el texto contenido en una imagen distorsiondada, aleatoria y siempre diferente como en el ejemplo anterior. En el diagrama siguiente se muestra como este caso debe ser manejado por el algoritmo ***SPEI-checker*** .

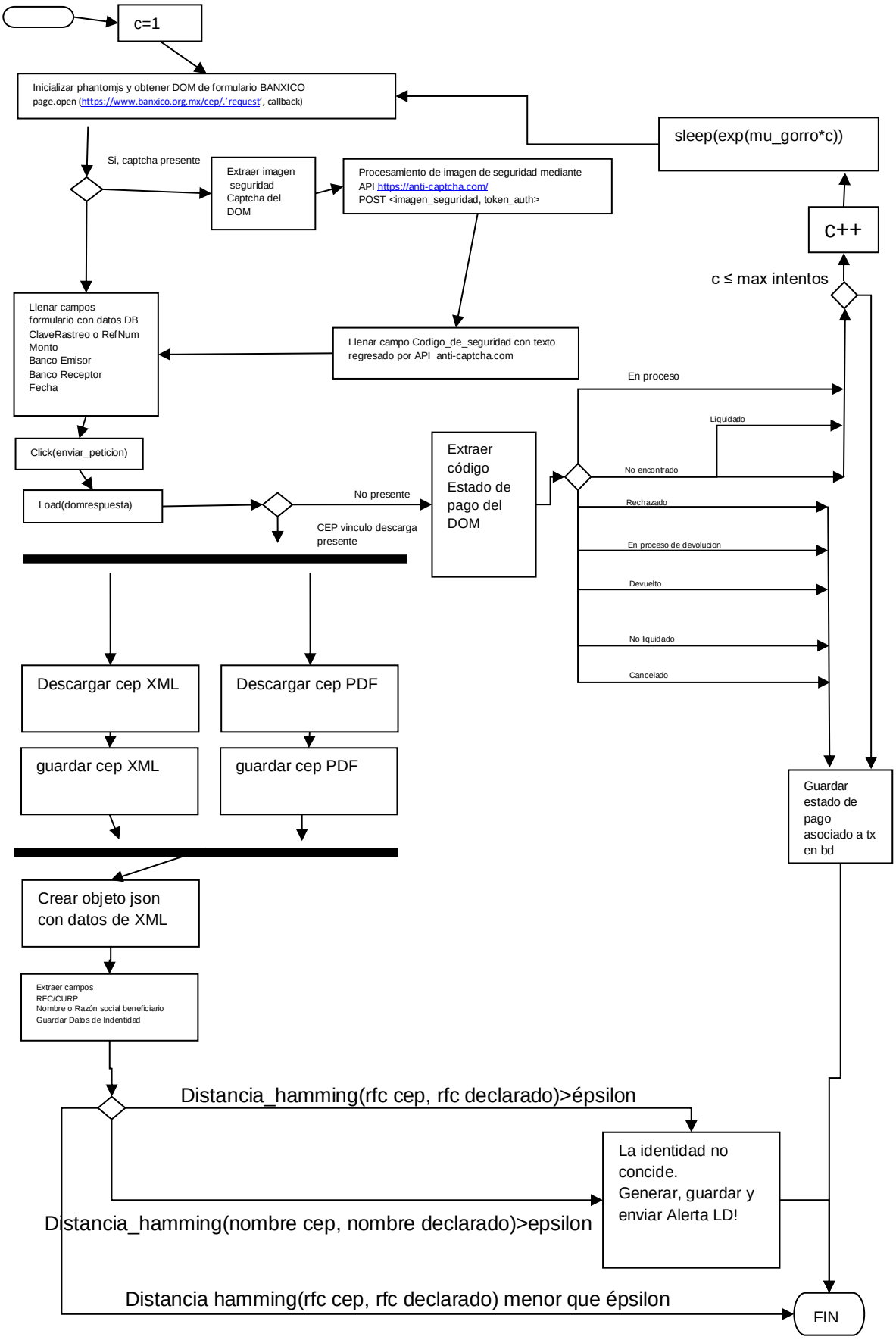


Ilustración 9 Algoritmo SPEI-checker (SPID-checker) obtención automatizada de CEPs

Notar que en el diagrama anterior ***mu_gorro*** representa el estimador del parámetro de ubicación $\hat{\mu}$ de la distribución. El algoritmo trata de recuperar el CEP de una transacción, realizando una cantidad de intentos máxima, y aumentando del tiempo entre cada intento de forma exponencial. Los casos en los que nunca se va obtener un CEP ya que el estado del pago es "abosorvente" deben ser manejados por la lógica de negocios.

Es importante destacar que se requiere desarrollo también de un algoritmo SPID-checker con exactamente la misma lógica de negocio que SPEI-checker para recuperar comprobantes de transacciones en dólares.

Modelado de base de datos no relacional (MongoDB)

MongoDB es una base de datos orientada a documentos. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos tienen formato JSON lo que hace muy fácil trabajar con ellos desde aplicaciones en lenguajes de programación como ***javascript*** o ***typescript***. Para auentar la eficiencia y minimiza el uso de espacio el motor de MongoDB internamente almacena una representación binaria de los documentos JSON.

Veamos algunas diferencias entre Mongodb (base de datos NO relacional) y SQL (base de datos relacional)

Mongodb	**SQL**
Base de datos	Base de datos
Colecciones	Tablas
Documentos	Filas (registros)
Campos	Columnas

Usamos Mongodb para registrar cada cambio hecho a algún documento del modelo de negocio. Los documentos persisten en un servicio de Mongo instalado dentro de una red privada virtual dentro de Azure para evitar exponer la base de datos al mundo exterior.

Como **VULCANO** está pensado para soportar un gran flujo de usuarios y transacciones, Mongo nos ofrece una gran escalabilidad y la gestión de grandes volúmenes de datos. Como su desarrollo se ha basado en la necesidad de crear sistemas de gestión capaces de trabajar con datos no estructurados o semi-estructurados es una ventaja grande para el sistema.

Las principales colecciones de documentos en este momento se ilustran en el siguiente diagrama:

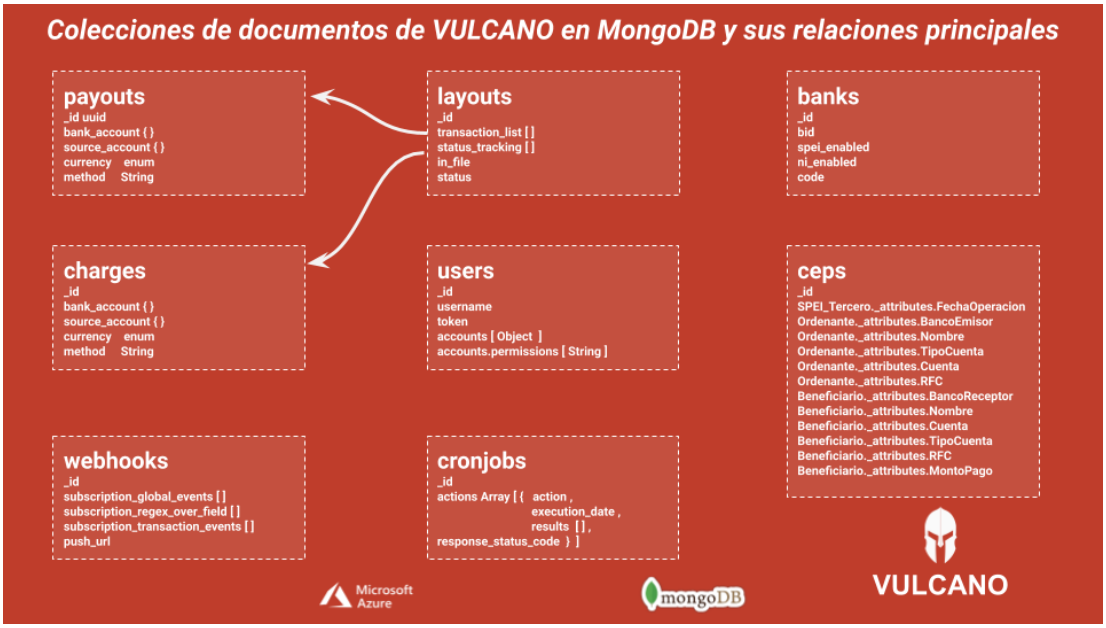


Ilustración 10 Colecciones de documentos de VULCANO

A continuación, veamos ejemplos de documentos para ilustrar mejor el manejo de información VULCANO.

Ejemplo Documento de Colección MongoDB VULCANO **payouts**


```
_id" : ObjectId("5e837ea75741a90019fcc3cb"),
bank_account" : {
  "creation_date" : {
     "$date" : 1585675943720
  },
  "clabe" : "072180000112157634",
  "holder_name" : "Anonimo",
  "bank_code" : "072",
  "bank_name" : "BANORTE"
,
source_account" : {
  "account_type" : "40",
  "bank_code" : "014",
  "account_number" : "014180220006507057"
,
currency" : "MXN",
method" : "bank_account_clabe",
operation_type" : "out",
```

```
bank_operation_type" : "spei",
transaction_type" : "payout",
status" : "in_progress",
creation_date" : {
  "$date" : 1585675943720
,
delivery_date" : {
  "$date" : 1585675943720
,
delivered" : true,
amount" : 0.01,
description" : "Validacion poltvcdr4",
__v" : 0
```


Ilustración 11 Ejemplo Documento de Colección MongoDB VULCANO payouts

Ejemplo Documento de Colección MongoDB VULCANO **layouts**


```
_id" : ObjectId("5e857f307d21890019ca1f33"),
header_data" : {
  "record_type" : "01",
  "operation_code" : "60",
  "participant_bank" : "014",
  "direction" : "E",
  "service" : "2",
  "operation_date" : {
     "$date" : 1585807130954
  },
  "batch_rejection_cause" : "00",
  "sequence_number" : "0000001",
  "batch_number" : "0200006",
  "currency_code" : "01",
  "modality" : "1"
,
transaction_list" : [
  "5e73f9f55741a90019fcc3c9"
,
creation_date" : {
  "$date" : 1585807130954
,
in_file" : "tran020420200601_45617856.in2",
status" : "layoutUploaded",
status_tracking" : [
  {
     "_id" : ObjectId("5e857f307d21890019ca1f34"),
     "status_change_date" : {
        "$date" : 1585807152348
     },
     "status" : "layoutGenerated"
  },
  {
     "_id" : ObjectId("5e857f3dd1ffc6001b1c7edd"),
     "status_change_date" : {
        "$date" : 1585807118101
     },
     "status" : "layoutEncrypted"
  },
  {
     "_id" : ObjectId("5e857f405b7229001904d14f"),
     "status_change_date" : {
        "$date" : 1585807128264
     },
     "status" : "layoutUploaded"
  }
```

```
__v" : 2
```


Ilustración 12 Ejemplo Documento de Colección MongoDB VULCANO layouts

La colección ***layouts*** en la base de datos contiene en su campo "transaction_list" un arreglo de apuntadores hacia el campo _id de documentos en la colección ***payouts*** y ***charges***.

Ejemplo Documento de Colección MongoDB VULCANO ***users***


```
{
    "_id" : ObjectId("5d38e1358060f20013246966"),
    "username" : "admin43",
    "password" : "aR4ndomP4$$wordGoesHereXW2",
    "token" : "e8hfyrbxsywrncmkq286cxsko276cmfs51qzoplsde5",
    "email" : "sebastian@vulcano.com",
    "auth_provider" : "database",
    "accounts" : [
        {
            "_id" : ObjectId("5d38e1358060f20013246968"),
            "permissions" : [
                "get.payouts",
                "get.payouts.*"
            ],
            "account_type" : "bank_account",
            "bank_code" : "014",
            "number" : "12345812387"
        },
        {
            "_id" : ObjectId("5d38e1358060f20013246967"),
            "permissions" : [
                "get.payouts",
                "get.payouts.*",
                "post.payouts.send-tef",
                "post.payouts.send-spei"
            ],
            "account_type" : "clabe",
            "number" : "014180220006507057"
        }
    ],
    "__v" : 0
}
```


Ilustración 13 Ejemplo Documento de Colección MongoDB VULCANO users

La colección de usuarios contiene un arreglo llamado "permissions" que consiste en una lista que especifica mediante una notación muy compacta las acciones que el usuario puede realizar sobre los recursos del servidor.

Es importante mencionar que la notación de los permisos que se definen aquí parte del hecho que estamos usando el paradigma REST (*Resourceful state stransfer*) en el diseño del api de VULCANO. También se utiliza el poder de expresiones regulares para definir con precisión los recursos afectados. El ejemplo de arriba expresa que el usuario puede hacer una petición con el verbo ***GET*** a cualquier recurso debajo la ruta ***<ip_servidor>/payouts*** así como a la ruta ***<ip_servidor>/payouts*** en si. En el ejemplo también se muestra que el usuario puede realizar peticiones que alteren el estado del servidor mediante el verbo ***POST*** a los recursos ubicados en la ruta url ***<ip_servidor>/payouts/send-tef*** y ***<ip_servidor>/payouts/send-spei*** . Sin embargo, el

usuario, en el ejemplo no puede usar el verbo ***REQUEST***, ***DELETE*** o ***PATCH*** sobre ninguno de los recursos anteriormente mencionados.

Ejemplo Documento de Colección MongoDB VULCANO ***ceps***


```
{
   "_id" : ObjectId("5e862bddf8ec3c001923b90a"),
   "cep_xml_content" : {
      "_declaration" : {
         "_attributes" : {
            "version" : "1.0",
            "encoding" : "UTF-8"
         }
      },
      "SPEI_Tercero" : {
         "_attributes" : {
            "FechaOperacion" : "2020-04-02",
            "Hora" : "11:42:32",
            "ClaveSPEI" : "40072",
            "sello" : "CDzaDqrGTn+XlyrK/zhBN+cwRAbidHwSHHcNuvFlEQOeuemBFkRk1pQN6HaoFOc5afmPHrouigQ0j7DJa7DHZE11vS+7WH8
+8YS9o7A0Wjh1gX4R2HqnPhrdk1Vs5bMW2J1Vt5YhbUmt8TschoroRJopNWP7R0dPTafQg8Cut0MKAViU2v226kjx57ZTzaMQu4ue3epcj4CU+
IIL7MegMwnxwgvWy0rFqpGeXWVEaQIF9v4N/49oeih7jjX5Y3QvNVZeAfwDudsL+lkIZQRw2bUlntVW86oWsuESNqIBFcMmesvGmC/TXz9kifozk
E8MhyItMAZn5EuzETVt5NyOFA==",
            "numeroCertificado" : "00001000000410948563",
            "cadenaCDA"                    :                    "||1|02042020|02042020|114232|40072|SANTANDER|ISBIT                    S.A
C.V|40|014180220006507057|IIN131003RTA|BANORTE|JAIME SANCHEZ CORTINA|
40|072180000112157634|SACJ650430CY9|Validacion poltvcdr4|0|0.01|00001000000410948563||
CDzaDqrGTn+XlyrK/zhBN+cwRAbidHwSHHcNuvFlEQOeuemBFkRk1pQN6HaoFOc5afmPHrouigQ0j7DJa7DHZE11vS+7WH8+8YS9o7A0Wjh1gX4R2H
Phrdk1Vs5bMW2J1Vt5YhbUmt8TschoroRJopNWP7R0dPTafQg8Cut0MKAViU2v226kjx57ZTzaMQu4ue3epcj4CU+IIL7MegMwnxwgvWy0rFqpGeXWV
QIF9v4N/49oeih7jjX5Y3QvNVZeAfwDudsL+lkIZQRw2bUlntVW86oWsuESNqIBFcMmesvGmC/TXz9kifozkE8MhyItMAZn5EuzETVt5NyOFA==",
            "claveRastreo" : "2020040240014 HDH0000456373500"
         },
         "Beneficiario" : {
            "_attributes" : {
               "BancoReceptor" : "BANORTE",
               "Nombre" : "JAIME SANCHEZ CORTINA",
               "TipoCuenta" : "40",
               "Cuenta" : "072180000112157634",
               "RFC" : "SACJ650430CY9",
               "Concepto" : "Validacion poltvcdr4",
               "IVA" : 0,
               "MontoPago" : 0.01
            }
         },
         "Ordenante" : {
            "_attributes" : {
               "BancoEmisor" : "SANTANDER",
               "Nombre" : "ISBIT S.A DE C.V",
               "TipoCuenta" : "40",
               "Cuenta" : "014180220006507057",
               "RFC" : "IIN131003RTA"
            }
         }
      }
   },
   "cep_pdf_format" : "/cep-02-04-20202020040240014HDH0000456373500.pdf",
   "cep_xml_format" : "/cep-02-04-20202020040240014HDH0000456373500.xml",
   "__v" : 0
}
```


Ilustración 14 Ejemplo Documento de Colección MongoDB VULCANO "ceps"

Ejemplo Documento de Colección MongoDB VULCANO “***banks***”


```
{
   "_id" : ObjectId("5d38eb31823b7e0011b57214"),
   "currency" : "MXN",
   "spei_enabled" : false,
   "ni_enabled" : false,
   "creation_date" : {
      "$date" : 1564011148437
   },
   "bid" : "637",
   "code" : "order",
   "name" : "ORDER",
   "description" : "ORDER EXPRESS CASA DE CAMBIO SA DE CV",
   "deposit" : false,
   "withdraw" : true,
   "institution_number" : "90637",
   "transfer_key" : "90637",
   "activity" : "IFNB",
   "tef" : {
      "available" : false
   },
   "__v" : 0
}
```


Ilustración 15 Ejemplo Documento de Colección MongoDB VULCANO “banks”

En el ejemplo anterior es importante notar que el campo “activity” indica si la cuenta pertenece a una institución financiera no bancaria o a un banco tradicional.

Ejemplo Documento de Colección MongoDB VULCANO “***cronjobs***”


```
{
   "_id" : ObjectId("5d3725a912454e00198cbccb"),
   "actions" : [
      {
         "_id" : ObjectId("5d3725a912454e00198cbcce"),
         "action" : "generateLayouts",
         "execution_date" : {
            "$date" : 1563895208790
         },
         "status" : true,
         "layouts_to_process" : 0,
         "response_status_code" : "200",
         "results" : [ ]
      },
      {
         "_id" : ObjectId("5d3725a912454e00198cbccd"),
         "action" : "encryptLayouts",
         "execution_date" : {
            "$date" : 1563895208790
         },
         "status" : true,
         "layouts_to_process" : 0,
         "response_status_code" : "200",
         "results" : [ ]
      },
      {
         "_id" : ObjectId("5d3725a912454e00198cbccc"),
         "action" : "uploadLayouts",
         "execution_date" : {
            "$date" : 1563895208790
         },
         "status" : true,
         "layouts_to_process" : 0,
         "response_status_code" : "200",
```

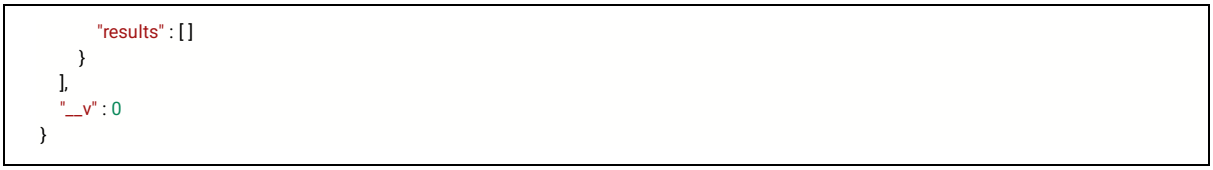


Ilustración 16 Ejemplo Documento de Colección MongoDB VULCANO "cronjobs"

La colección **"cronjobs"** implementa una "pila" o "cola" de actividades pendientes que se tienen que realizar periodicamente y le da seguimiento al cambio de estado de cada actividad. Por ejemplo, la tarea ***generateLayouts*** se ejecuta cada 3 minutos. El algoritmo debe checar si hay pagos pendientes de encriptar (acción "***encryptLayouts***") y si ese es el caso genera el archivo de transacciones y el campo de "***layoutsToProcess"*** será igual a la cantidad pendiente de pagos por agregar a un "**layout"**.

El algoritmo periódicamente revisa si existen ***layouts*** encriptados pendientes de enviar mediante SFTP (acción "uploadLayouts"). En caso de existir archivos en este supuesto su envio es programado mediante la adición en la colección ***"cronjobs"*** de una entrada cuyo campo ***layoutsToProcess*** será igual al numero de archivos pendientes de enviar y los archivos enviados via SFTP seran aquellos que fueron señalados en el arreglo "results" de la entrada correspondiente a la "action" antecedente ***"encryptLayouts".***

Diseño de Arquitectura de Solución

La arquitectura de Vulcano TMS es un sistema modular que corre como clúster en Kubernetes. Los contenedores del clúster a su vez están conectados a una base de datos externa en Mongo.

La arquitectura de **Vulcano** es un complejo sistema que integra métodos rigurosos para asegurar una alta seguridad y velocidad en las transacciones. Vulcano está diseñado con la escalabilidad en mente, impidiendo que los procesos puedan saturarse. Para esto, se utilizó la funcionalidad de ***Docker*** para generar aislamiento al compartimentalizar los procesos en contenedores independientes, Kubernetes nos permite generar sets de réplicas de los contenedores. Las réplicas de contenedores pueden ser incrementados o disminuidos en número, robusteciendo el sistema en caso de aumentar el número de operaciones, permitiendo escalabilidad. ***Vulcano*** es dinámico, los procesos son creados de forma automática y destruidos en caso de falta de flujo transaccional, reduciendo costos operativos.

La corrida de procesos se divide en varios contenedores que ejecutan las operaciones en ambientes aislados en paralelo. Si un proceso fuera a detenerse, este se renueva y se notifica al orquestador.

Es una importante meta de diseño la capacidad de Vulcano para crear réplicas de cada contenedor y al salvar los estados e información durante cada etapa de procesos en la base de datos.

El proceso que ocurre al enviar un pago con Vulcano es el siguiente:

PASO 1. El cliente envía petición (JSON REST API) que es recibida por Balanceador de Carga NGINX/ingress que rutea al módulo de autenticación de autorización (basado en RBAC).
PASO 2. Si la autenticación y autorización es exitosa, la petición es transferida al módulo, que realiza la validación de la transacción (por ejemplo, fondos suficientes para ejecutar orden de pago) al ambiente de Vulcano TMS.
PASO 3. El siguiente contenedor genera un layout con formato CECOBAN (http://cecoban.com/).
PASO 4. El tercer contenedor encripta la información y la guarda directamente en la base de datos.
PASO 5. El siguiente contenedor lo envía el Layout. Si uno falla, el demonio va a permitir que el proceso inicie donde el último proceso se quedó. Es un checkpoint. La base de datos guarda el estado de la transacción en cada transición de estado. El contenedor establece un túnel entre el cortafuegos de vulcano y el del banco, por medio de un mismo IP.
PASO 6. Envía el layout por el TÚNEL entre los cortafuegos de vulcano y el del banco.
PASO 7. Recuperación de Layout de salida periódicamente con resultados de procesamiento de transacciones por el banco.
PASO 8. Desencriptar y procesar Layout de salida periódicamente con resultados de procesamiento de transacciones por el banco, actualizando cambios de estado de transacciones en la base de datos.
PASO 9. Envío de notificaciones con cambios de estado mediante webhooks a suscriptores (sistemas que necesitan desencadenar acciones en la lógica de negocios a partir de eventos, como la liquidación exitosa de un pago o el rechazo del mismo, por ejemplo).

Vulcano traduce los protocolos usados por diferentes bancos a un API más simple universal (REST JSON) que no requiere ser cambiado cada vez que una integración de conexión a un nuevo BANCO es establecida o eliminada.

Vulcano permite una separación de preocupaciones entre las aplicaciones de la organización y la tesorería, lo que tiene el efecto de agilizar el desarrollo de nuevos productos.

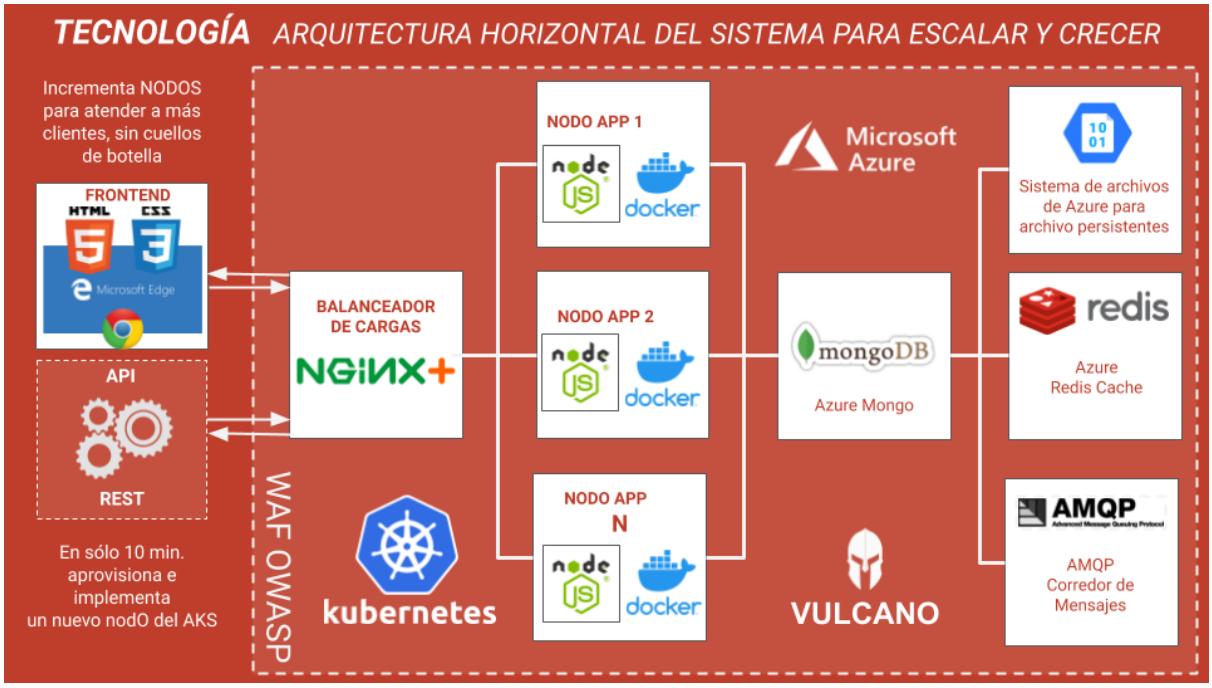


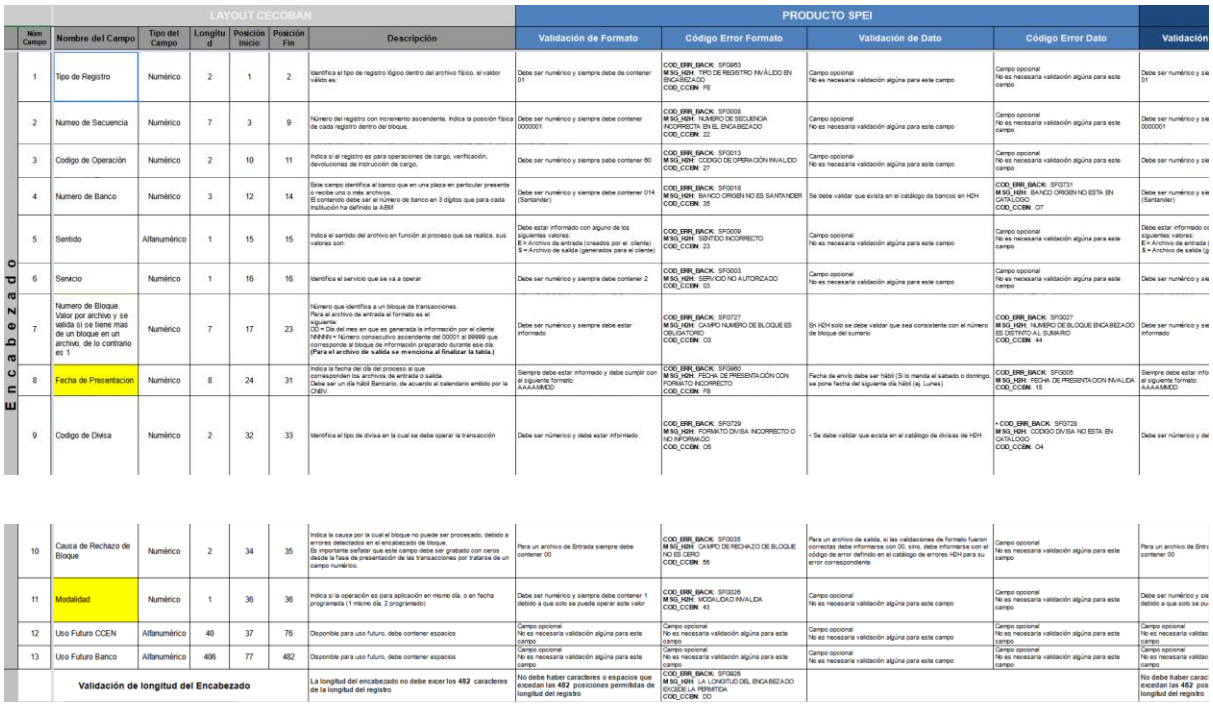
Ilustración 17 TECNOLOGÍA; arquitectura horizontal del sistema para escalar y

Una característica esencial de la solución es que debe ser capaz de operar con múltiples bancos cada uno con su correspondiente integración H2H.

Ilustración 18 Caso de estudio: tesorería Exchange

Implementación con banca transaccional de Santander México

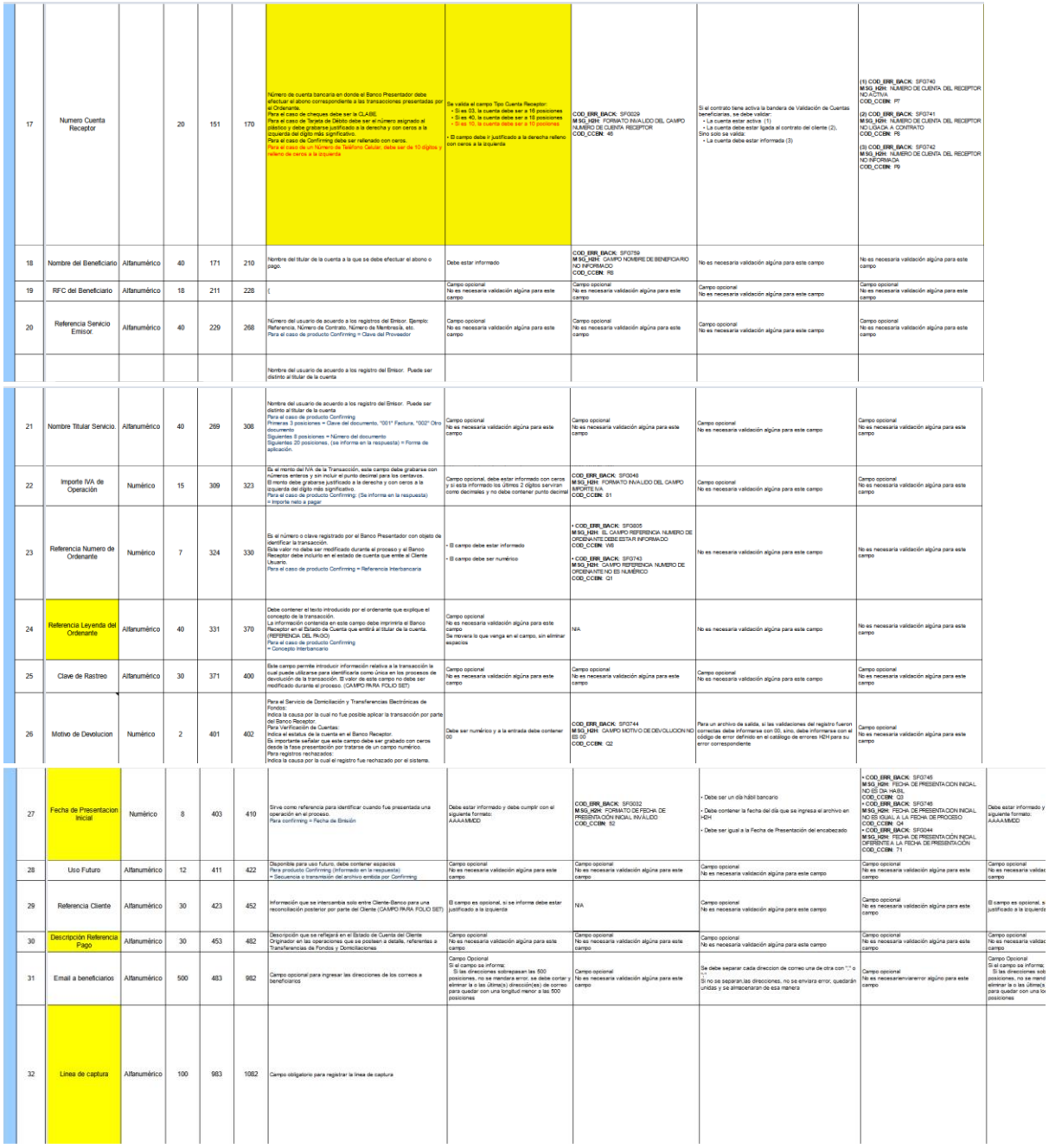
El esquema de comunicación de Santander está basado en el intercambio de archivos de texto en el formato CECOBAN los cuales tienen un encabezado de 9 campos. A los archivos con este formato nos referimos como *"Layouts"*.



		LAYOUT CECOBAN						PRODUCTO SPEI				
	Núm Campo	Nombre del Campo	Tipo del Campo	Longitud	Posición Inicio	Posición Fin	Descripción	Validación de Formato	Código Error Formato	Validación de Dato	Código Error Dato	Validación
Encabezado	1	Tipo de Registro	Numérico	2	1	2	Identifica el tipo de registro lógico dentro del archivo físico, el valor válido es:	Debe ser numérico y siempre debe de contener 01	COD_ERR_BACK: SF0063 M SG_H2H: TPO DE REGISTRO INVÁLDO EN ENCABEZADO COD_CCEN: FE	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Debe ser numérico y siempre 01
	2	Numeo de Secuencia	Numérico	7	3	9	Número del registro con incremento ascendente. Indica la posición física de cada registro dentro del bloque.	Debe ser numérico y siempre debe contener 0000001	COD_ERR_BACK: SF0008 M SG_H2H: NUMERO DE SIEUENCIA INCORRECTA EN EL ENCABEZADO COD_CCEN: 22	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Debe ser numérico y siempre 0000001
	3	Codigo de Operación	Numérico	2	10	11	Indica si el registro es para operaciones de cargo, verificación, devoluciones de instrucción de cargo.	Debe ser numérico y siempre sebe contener 60	COD_ERR_BACK: SF0013 M SG_H2H: CODIGO DE OPERACIÓN INVALIDO COD_CCEN: 27	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Debe ser numérico y siempre
	4	Numero de Banco	Numérico	3	12	14	Este campo identifica al banco que en una plaza en particular presenta o recibe uno o más archivos. El contenido debe ser el número de banco en 3 dígitos que para cada institución ha definido la ABM	Debe ser numérico y siempre debe contener 014 (Santander)	COD_ERR_BACK: SF0018 M SG_H2H: BANCO ORIGEN NO ES SANTANDER COD_CCEN: 35	Se debe validar que exista en el catálogo de bancos en H2H	COD_ERR_BACK: SF0731 M SG_H2H: BANCO ORIGEN NO ESTA EN CATALOGO COD_CCEN: O7	Debe ser numérico y siempre (Santander)
	5	Sentido	Alfanumérico	1	15	15	Indica el sentido del archivo en función al proceso que se realiza, sus valores son:	Debe estar informado con alguno de los siguientes valores: E = Archivo de entrada (creados por el cliente) S = Archivo de salida (generados para el cliente)	COD_ERR_BACK: SF0009 M SG_H2H: SENTIDO INCORRECTO COD_CCEN: 23	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Debe estar informado con siguientes valores: E = Archivo de entrada S = Archivo de salida (g
	6	Senácio	Numérico	1	16	16	Identifica el servicio que se va a operar	Debe ser numérico y siempre debe contener 2	COD_ERR_BACK: SF0003 M SG_H2H: SERVICIO NO AUTORIZADO COD_CCEN: 03	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Debe ser numérico y siempre
	7	Numero de Bloque. Valor por archivo y se valida si se tiene mas de un bloque en un archivo, de lo contrario es 1	Numérico	7	17	23	Número que identifica a un bloque de transacciones. Para el archivo de entrada el formato es el siguiente: DD = Día del mes en que es generada la información por el cliente NNNNN = Número consecutivo ascendente del 00001 al 99999 que corresponde al bloque de información preparado durante ese día. (Para el archivo de salida se menciona al finalizar la tabla.)	Debe ser numérico y siempre debe estar informado	COD_ERR_BACK: SF0727 M SG_H2H: CAMPO NUMERO DE BLOQUE ES OBLIGATORIO COD_CCEN: O8	En H2H solo se debe validar que sea consistente con el número de bloque del sumario	COD_ERR_BACK: SF0027 M SG_H2H: NUMERO DE BLOQUE ENCABEZADO ES DISTINTO AL SUMARIO COD_CCEN: 44	Debe ser numérico y siempre informado
	8	Fecha de Presentacion	Numérico	8	24	31	Indica la fecha del día del proceso al que corresponden los archivos de entrada o salida. Debe ser un día hábil Bancario, de acuerdo al calendario emitido por la CNBV.	Siempre debe estar informado y debe cumplir con el siguiente formato: AAAAMMDD	COD_ERR_BACK: SF0950 M SG_H2H: FECHA DE PRESENTACIÓN CON FORMATO INCORRECTO COD_CCEN: FB	Fecha de envío debe ser hábil (Si lo manda el sabado o domingo, se pone fecha del siguiente día hábil (ej. Lunes)	COD_ERR_BACK: SF0035 M SG_H2H: FECHA DE PRESENTACION INVALIDA COD_CCEN: 16	Siempre debe estar info el siguiente formato: AAAAMMDD
	9	Codigo de Divisa	Numérico	2	32	33	Identifica el tipo de divisa en la cual se debe operar la transacción	Debe ser número y debe estar informado	COD_ERR_BACK: SF0729 M SG_H2H: FORMATO DIVISA INCORRECTO O NO INFORMADO COD_CCEN: O5	- Se debe validar que exista en el catálogo de divisas de H2H	* COD_ERR_BACK: SF0728 M SG_H2H: CODIGO DIVISA NO ESTA EN CATALOGO COD_CCEN: O4	Debe ser número y de
	10	Causa de Rechazo de Bloque	Numérico	2	34	35	Indica la causa por la cual el bloque no puede ser procesado, debido a errores detectados en el encabezado de bloque. Es importante señalar que este campo debe ser grabado con ceros desde la fase de presentación de las transacciones por tratarse de un campo numérico.	Para un archivo de Entrada siempre debe contener 00	COD_ERR_BACK: SF0035 M SG_H2H: CAMPO DE RECHAZO DE BLOQUE NO ES CERO COD_CCEN: 56	Para un archivo de salida, si las validaciones de formato fueron correctas debe informarse con 00, sino, debe informarse con el código de error definido en el catálogo de errores H2H para su error correspondiente	Campo opcional No es necesaria validación alguna para este campo	Para un archivo de Entr contener 00
	11	Modalidad	Numérico	1	36	36	Indica si la operación es para aplicación en mismo día, o en fecha programada (1 mismo día, 2 programado)	Debe ser numérico y siempre debe contener 1 debido a que solo se puede operar este valor	COD_ERR_BACK: SF0026 M SG_H2H: MODALIDAD INVALIDA COD_CCEN: 43	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Debe ser numérico y siempre debido a que solo se pu
	12	Uso Futuro CCEN	Alfanumérico	40	37	76	Disponible para uso futuro, debe contener espacios	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validac campo
	13	Uso Futuro Banco	Alfanumérico	406	77	482	Disponible para uso futuro, debe contener espacios	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validac campo
		Validación de longitud del Encabezado					La longitud del encabezado no debe exicer los 482 caracteres de la longitud del registro	No debe haber caracteres o espacios que excedan las 482 posiciones permitidas de longitud del registro	COD_ERR_BACK: SF0826 M SG_H2H: LA LONGITUD DEL ENCABEZADO EXCEDE LA PERMITIDA COD_CCEN: OO			No debe haber carac excedan las 482 pos longitud del registro

Los registros con las ordenes de pago a ser enviadas al banco tienen 32 campos de los son rellenados con espacio en blanco en caso de no ser utilizados por una transacción en particular y no deben superar una longitud máxima de 1082 posiciones.

		LAYOUT CECOBAN					PRODUCTO SPEI			
	Núm Campo	Nombre del Campo	Tipo del Campo	Longitud	Posición Inicio	Posición Fin	Descripción	Validación de Formato	Código Error Formato	Validación de Dato	Código Error Dato
	1	Tipo de Registro	Numérico	2	1	2	Identifica el tipo de registro lógico dentro del archivo físico	Debe ser numérico y siempre debe contener 02	COD_ERR_BACK: SFG014 M SG_H2H: TIPO DE REGISTRO INVÁLIDO COD_CCEN: 41	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
	2	Numero de Secuencia	Numérico	7	3	9	Número del registro con incremento ascendente. Indica la posición física de cada registro dentro del bloque.	Los valores que pueden tener después del Encabezado de Bloque son: 0000002, 0000003, etc. dependiendo el número de líneas de detalle y debe ser consecutivo ascendente y el número consecutivo debe estar justificado a la derecha, relleno de ceros a la izquierda hasta alcanzar las 7 posiciones del campo	COD_ERR_BACK: SFG015 M SG_H2H: NUMERO DE SECUENCIA EN EL DETALLE INCORRECTA COD_CCEN: 29	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
	3	Codigo de Operación	Numérico	2	10	11	Indica si el registro es para operaciones de cargo, verificación de cuentas, resultado de la verificación, devoluciones de instrucción de cargo.	Debe ser numérico y siempre sebe contener 60	COD_ERR_BACK: SFG013 M SG_H2H: CODIGO DE OPERACIÓN INVALIDO COD_CCEN: 27	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
	4	Codigo de Divisa	Numérico	2	12	13	Identifica el tipo de divisa en la cual se debe operar la transacción según el caso	Debe ser número y debe estar informado	COD_ERR_BACK: SFG729 M SG_H2H: FORMATO DIVISA INCORRECTO O NO INFORMADO COD_CCEN: O6	• Debe de ser 01 • Se debe validar que exista en el catálogo de divisas de H2H • Se debe validar que sea igual a la del encabezado	• COD_ERR_BACK: SFG802 M SG_H2H: EL PRODUCTO NO PUEDE OPERAR CON DIVISA EN DOLARES COD_CCEN: W3 • COD_ERR_BACK: SFG728 M SG_H2H: CODIGO DIVISA NO ESTA EN CATALOGO COD_CCEN: O4 • COD_ERR_BACK: SFG053 M SG_H2H: CÓDIGO DE DIVISA INCONSISTENTE CON EL ENCABEZADO COD_CCEN: 80
	5	Fecha de Transferencia	Numérico	8	14	21	Fecha en que se envía la instrucción Para confirming = Fecha de Emisión del documento	Debe estar informado y debe cumplir con el siguiente formato: AAAAMMDD	COD_ERR_BACK: SFG016 M SG_H2H: FORMATO DE FECHA DE TRANSFERENCIA INVÁLIDO COD_CCEN: 31	• Debe ser un día hábil bancario • No debe ser menor al día en curso • Debe ser igual a la Fecha de Aplicación	• COD_ERR_BACK: SFG017 M SG_H2H: FECHA DE TRANSFERENCIA NO PERMITIDA (DÍA INHÁBIL) COD_CCEN: 32 • COD_ERR_BACK: SFG042 M SG_H2H: FECHA DE TRANSFERENCIA MENOR A LA FECHA DE PROCESO COD_CCEN: 04 • COD_ERR_BACK: SFG730 M SG_H2H: FECHA DE TRANSFERENCIA NO ES IGUAL A FECHA DE APLICACIÓN COD_CCEN: O6
	6	Banco Presentador	Numérico	3	22	24	Código de tres dígitos asignado a Banco Santander por Banco de México	Siempre debe contener 014 (Santander)	COD_ERR_BACK: SFG018 M SG_H2H: BANCO ORIGEN NO ES SANTANDER COD_CCEN: 35	• Se debe validar que exista en el catálogo de bancos en H2H • Debe ser el mismo que el del encabezado	• COD_ERR_BACK: SFG021 M SG_H2H: BANCO ORIGEN NO ESTA EN CATALOGO COD_CCEN: 39 • COD_ERR_BACK: SFG797 M SG_H2H: BANCO ORIGEN NO ES IGUAL AL
	7	Banco Receptor	Numérico	3	25	27	Código de tres dígitos asignado al Banco receptor de los fondos. Para Productos	Debe estar informado y ser numérico	COD_ERR_BACK: SFG019 M SG_H2H: BANCO RECEPTOR CON FORMATO INCORRECTO COD_CCEN: 36	• Se valida su existencia en el catálogo de bancos de H2H (1) • **Si TIPO CUENTA RECEPTORA es "40"**: El campo debe coincidir con las primeras 3 posiciones de la cuenta clabe (2) Solo si esta validación se cumple, realizar las siguientes validaciones: Si el valor del campo es "014" Se valida que se tenga activa la bandera de cambio de producto Si se tiene activa, realiza el cambio de producto a TMS Si no se tiene activa, mandar error (3) Si el valor es diferente de "014" se verifica que el banco opera el producto: Si el banco opera el producto, se procesa como SPEI Si no opera el producto SPEI, mandar error (4) • **Si TIPO CUENTA RECEPTORA es "03" o "10"** Si el valor es diferente de "014" se verifica que el banco opera el producto: Si el banco opera el producto, se procesa como SPEI Si no opera el producto SPEI, mandar error (4) Si el valor es "014" Mandar error (3)	(1) COD_ERR_BACK: SFG021 M SG_H2H: BANCO RECEPTOR NO ESTA EN CATALOGO COD_CCEN: 38 (2) COD_ERR_BACK: SFG049 M SG_H2H: CODIGO BANCO DE CUENTA CLABE NO CORRESPONDE CON BANCO RECEPTOR COD_CCEN: 59 (3) COD_ERR_BACK: SFG794 M SG_H2H: BANCO RECEPTOR NO PUEDE SER SANTANDER COD_CCEN: V6 (4) COD_ERR_BACK: SFG795 M SG_H2H: EL BANCO RECEPTOR NO PUEDE OPERAR EL PRODUCTO COD_CCEN: V8
	8	Importe	Numérico	15	28	42	Es el monto de la transacción, esta campo debe grabarse con números enteros y sin incluir el punto decimal para los centavos.	• Debe estar informado y los últimos 2 dígitos servirán como decimales y no debe contener punto decimal • El campo debe ir justificado a la derecha rellena con ceros sa la izquierda	COD_ERR_BACK: SFG732 M SG_H2H: FORMATO DE IMPORTE INVALIDO COD_CCEN: O6	Se debe validar que sea mayor a cero	COD_ERR_BACK: SFG020 M SG_H2H: IMPORTE INVALIDO, DEBE SER MAYOR A CERO COD_CCEN: 37
	9	Uso Futuro CCEN	Alfanumérico	16	43	58	Disponible para uso futuro, debe contener espacios Para producto Confirming = Clave de registro Documento por pagar, 1 dígito alineado a la derecha relleno de ceros.	Campo opcional	N/A	N/A	N/A
											• COD_ERR_BACK: SFG052
	10	Tipo de Operación	Numérico	2	59	60	Modalidad del servicio a la cual se va a direccionar la transacción.	Debe ser numérico Debe estar informado	COD_ERR_BACK: SFG733 M SG_H2H: FORMATO DE CAMPO TIPO DE OPERACION INCORRECTO O NO INFORMADO COD_CCEN: O9	Siempre debe contener: 01 = TRANSFERENCIA SPEI • Se debe validar que el producto este contratado	• COD_ERR_BACK: SFG052 M SG_H2H: TIPO DE OPERACIÓN INVÁLIDO COD_CCEN: 85 • COD_ERR_BACK: SFG785 M SG_H2H: PRODUCTO NO CONTRATADO COD_CCEN: U7
	11	Fecha de Aplicación	Numérico	8	61	68	Fecha en la que el Banco Receptor debe efectuar el cargo en la cuenta del Cliente Usuario. En caso de no ser el cargo exitoso, el banco receptor debe devolver a más tardar en T + 1 la transacción. Para producto confirming = Fecha de vencimiento del documento,	Debe estar informado y debe cumplir con el siguiente formato: AAAAMMDD	COD_ERR_BACK: SFG022 M SG_H2H: FORMATO DE FECHA DE APLICACIÓN INVÁLIDO COD_CCEN: 41	• Debe ser un día hábil bancario • No debe ser menor al día en curso • Debe ser igual a la Fecha de Transferencia Si la Fecha es del día en curso, el producto se procesara el mismo día **Si la fecha es posterior, debe ser menor o igual a los días naturales parametrizados en el contrato, de cumplirse esto, el producto quedara programado hasta cumplirse la fecha se aplicación,**	• COD_ERR_BACK: SFG043 M SG_H2H: FECHA DE APLICACIÓN NO PERMITIDA (DÍA INHÁBIL) COD_CCEN: 68 • COD_ERR_BACK: SFG734 M SG_H2H: FECHA DE APLICACION MENOR A LA FECHA DE PROCESO COD_CCEN: P1 • COD_ERR_BACK: SFG735 M SG_H2H: FECHA DE APLICACIÓN NO ES IGUAL A FECHA DE TRANSFERENCIA COD_CCEN: P2 • COD_ERR_BACK: SFG931 M SG_H2H: FECHA DE APLICACIÓN EXCEDE EL NUMERO DE DIAS A PROGRAMAR PARAMETRIZADO[COD_CCEN: D7
	12	Tipo Cuenta Ordenante	Numérico	2	69	70	Categoría de la Cuenta a la que se debe efectuar el cargo correspondiente: 01 para cuenta de 11 dígitos y para Contrato Confirming 03 para número de tarjeta de 16 dígitos 40 para cuenta clabe de 18 dígitos	Debe ser número y debe estar informado	COD_ERR_BACK: SFG736 M SG_H2H: FORMATO DE CAMPO TIPO DE CUENTA ORDENANTE INCORRECTO O NO INFORMADO COD_CCEN: P3	Solo debe contener: 01 ó 40	COD_ERR_BACK: SFG023 M SG_H2H: TIPO DE CUENTA DEL ORDENANTE INVALIDO COD_CCEN: 40
Detalle	13	Numero de cuenta ordenante	Numérico	20	71	90	Número de cuenta bancaria en donde el Banco Presentador debe efectuar el cargo correspondiente a las transacciones presentadas por el Ordenante. Para el caso de cheques debe ser el Código Cuenta Cliente. Para el caso de Tarjeta de Débito debe ser el número asignado al plástico. Para el caso de Cuenta clabe deben ser los 18 dígitos de la clabe interbancaria Para el caso de Confirming debe ser el contrato confirming	• El campo debe estar informado Se valida el campo Tipo Cuenta Ordenante: • Si es 01, la cuenta debe ser a 11 posiciones • Si es 40, la cuenta debe ser a 18 posiciones. • El campo debe ir justificado a la derecha rellena con ceros sa la izquierda	COD_ERR_BACK: SFG025 M SG_H2H: FORMATO INVALIDO DEL CAMPO NUMERO DE CUENTA ORDENANTE COD_CCEN: 42	• La cuenta debe estar activa • La cuenta debe estar ligada al contrato del cliente • La divisa de la cuenta debe ser igual pesos mexicanos (no puede ser dolares) Si la cuenta es de 18 posiciones (clabe), se deben tomar aparte de la posicion 9 hasta la 19 (11 posiciones), se debe validar que las 3 primeras posiciones sea "014" (Banco Santander)	• COD_ERR_BACK: SFG737 M SG_H2H: NUMERO DE CUENTA DEL ORDENANTE NO ACTIVA COD_CCEN: P4 • COD_ERR_BACK: SFG738 M SG_H2H: NUMERO DE CUENTA DEL ORDENANTE NO LIGADA A CONTRATO COD_CCEN: P5 • COD_ERR_BACK: SFG802 M SG_H2H: EL PRODUCTO NO PUEDE OPERAR CON DIVISA EN DOLARES COD_CCEN: W3 • COD_ERR_BACK: SFG859 M SG_H2H: EL BANCO INDICADO EN LA CUENTA CLABE DEL ORDENANTE DEBE SER SANTANDER COD_CCEN: FA
	14	Nombre Ordenante	Alfanumérico	40	91	130	Nombre del Emisor del Recibo (Razón Social)	Debe estar informado	COD_ERR_BACK: SFG045 M SG_H2H: NOMBRE DEL ORDENANTE INVÁLIDO, CONTIENE ESPACIOS COD_CCEN: 74	No es necesaria validación alguna para este campo	No es necesaria validación alguna para este campo
	15	RFC Ordenante	Alfanumérico	18	131	148	Corresponde al Registro Federal de Causantes del Emisor que origina la transacción. La información contenida en este campo debe imprimirla el Banco Receptor en el estado de Cuenta que emite al titular de la cuenta	Campo opcional	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	
	16	Tipo Cuenta Receptor	Numérico	2	149	150	Categoría de la Cuenta a la que se debe efectuar el abono correspondiente 00 para producto Orden de Pago y Confirming 01 para cuenta de 11 dígitos 03 para cuenta de 16 dígitos 40 para cuenta clabe de 18 dígitos 10 para números de teléfono celular	Debe ser número y debe estar informado	COD_ERR_BACK: SFG739 M SG_H2H: FORMATO DE CAMPO TIPO DE CUENTA RECEPTOR INCORRECTO O NO INFORMADO COD_CCEN: P6	Solo puede contener los siguientes valores: "03", "40" o "10"	COD_ERR_BACK: SFG028 M SG_H2H: TIPO DE CUENTA DEL RECEPTOR INVÁLIDO COD_CCEN: 45



17	Numero Cuenta Receptor		20	151	170	Número de cuenta bancaria en donde el Banco Presentador debe efectuar el abono correspondiente a las transacciones presentadas por el Ordenante. Para el caso de cheques debe ser la CLABE. Para el caso de Tarjeta de Débito debe ser el número asignado al plástico y debe grabarse justificado a la derecha y con ceros a la izquierda del dígito más significativo. Para el caso de Confirming debe ser rellenado con ceros. Para el caso de un Número de Teléfono Celular, debe ser de 10 dígitos y relleno de ceros a la izquierda	Se valida el campo Tipo Cuenta Receptor: • Si es 03, la cuenta debe ser a 16 posiciones • Si es 40, la cuenta debe ser a 18 posiciones • Si es 10, la cuenta debe ser a 10 posiciones • El campo debe ir justificado a la derecha relleno con ceros a la izquierda	COD_ERR_BACK: SFG029 MSG_H2H: FORMATO INVALIDO DEL CAMPO NUMERO DE CUENTA RECEPTOR COD_CCBI: I45	Si el contrato tiene activa la bandera de Validación de Cuentas beneficiarias, se debe validar: • La cuenta estar activa (1) • La cuenta debe estar ligada al contrato del cliente (2), Sino solo se valida: • La cuenta debe estar informada (3)	(1) COD_ERR_BACK: SFG740 MSG_H2H: NUMERO DE CUENTA DEL RECEPTOR NO ACTIVA COD_CCBI: P7 (2) COD_ERR_BACK: SFG741 MSG_H2H: NUMERO DE CUENTA DEL RECEPTOR NO LIGADA A CONTRATO COD_CCBI: P5 (3) COD_ERR_BACK: SFG742 MSG_H2H: NUMERO DE CUENTA DEL RECEPTOR NO INFORMADA COD_CCBI: P9
18	Nombre del Beneficiario	Alfanumérico	40	171	210	Nombre del titular de la cuenta a la que se debe efectuar el abono o pago.	Debe estar informado	COD_ERR_BACK: SFG759 MSG_H2H: CAMPO NOMBRE DE BENEFICIARIO NO INFORMADO COD_CCBI: R8	No es necesaria validación alguna para este campo	No es necesaria validación alguna para este campo
19	RFC del Beneficiario	Alfanumérico	18	211	228	(	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
20	Referencia Servicio Emisor.	Alfanumérico	40	229	268	Número del usuario de acuerdo a los registros del Emisor. Ejemplo: Referencia, Número de Contrato, Número de Membresía, etc. Para el caso de producto Confirming = Clave del Proveedor	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
						Nombre del usuario de acuerdo a los registro del Emisor. Puede ser distinto al titular de la cuenta				
21	Nombre Titular Servicio.	Alfanumérico	40	269	308	Nombre del usuario de acuerdo a los registro del Emisor. Puede ser distinto al titular de la cuenta Para el caso de producto Confirming Primeras 3 posiciones = Clave del documento, "001" Factura, "002" Otro documento Siguientes 8 posiciones = Número del documento Siguientes 20 posiciones, (se informa en la respuesta) = Forma de aplicación.	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
22	Importe IVA de Operación	Numérico	15	309	323	Es el monto del IVA de la Transacción, este campo debe grabarse con números enteros y sin incluir el punto decimal para los centavos. El monto debe grabarse justificado a la derecha y con ceros a la izquierda del dígito más significativo. Para el caso de producto Confirming: (Se informa en la respuesta) = Importe neto a pagar	Campo opcional, debe estar informado con ceros y si esta informado los últimos 2 dígitos servirían como decimales y no debe contener punto decimal	COD_ERR_BACK: SFG048 MSG_H2H: FORMATO INVALIDO DEL CAMPO IMPORTE IVA COD_CCBI: I51	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
23	Referencia Numero de Ordenante	Numérico	7	324	330	Es el número o clave registrado por el Banco Presentador con objeto de identificar la transacción. Este valor no debe ser modificado durante el proceso y el Banco Receptor debe incluirlo en el estado de cuenta que emite al Cliente Usuario. Para el caso de producto Confirming = Referencia Interbancaria	- El campo debe estar informado - El campo debe ser numérico	• COD_ERR_BACK: SFG005 MSG_H2H: EL CAMPO REFERENCIA NUMERO DE ORDENANTE DEBE ESTAR INFORMADO COD_CCBI: I66 • COD_ERR_BACK: SFG743 MSG_H2H: CAMPO REFERENCIA NUMERO DE ORDENANTE NO ES NUMÉRICO COD_CCBI: Q1	No es necesaria validación alguna para este campo	No es necesaria validación alguna para este campo
24	Referencia Leyenda del Ordenante	Alfanumérico	40	331	370	Debe contener el texto introducido por el ordenante que explique el concepto de la transacción. La información contenida en este campo debe imprimirla el Banco Receptor en el Estado de Cuenta que emitirá al titular de la cuenta. (REFERENCIA DEL PAGO) Para el caso de producto Confirming = Concepto Interbancario	Campo opcional No es necesaria validación alguna para este campo Se movera lo que venga en el campo, sin eliminar espacios	N/A	No es necesaria validación alguna para este campo	No es necesaria validación alguna para este campo
25	Clave de Rastreo	Alfanumérico	30	371	400	Este campo permite introducir información relativa a la transacción la cual puede utilizarse para identificarla como única en los procesos de devolución de la transacción. El valor de este campo no debe ser modificado durante el proceso. (CAMPO PARA FOLIO SET)	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
26	Motivo de Devolucion	Numérico	2	401	402	Para el Servicio de Domiciliación y Transferencias Electrónicas de Fondos: Indica la causa por la cual no fue posible aplicar la transacción por parte del Banco Receptor. Para Verificación de Cuentas: Indica el estatus de la cuenta en el Banco Receptor. Es importante señalar que este campo debe ser grabado con ceros desde la fase presentación por tratarse de un campo numérico. Para registros rechazados: Indica la causa por la cual el registro fue rechazado por el sistema.	Debe ser numérico y a la entrada debe contener 00	COD_ERR_BACK: SFG744 MSG_H2H: CAMPO MOTIVO DE DEVOLUCION NO ES 00 COD_CCBI: Q2	Para un archivo de salida, si las validaciones del registro fueron correctas debe informarse con 00, sino, debe informarse con el código de error definido en el catálogo de errores H2H para su error correspondiente	Campo opcional No es necesaria validación alguna para este campo
27	Fecha de Presentacion Inicial	Numérico	8	403	410	Sirve como referencia para identificar cuando fue presentada una operación en el proceso. Para confirming = Fecha de Emisión	Debe estar informado y debe cumplir con el siguiente formato: AAAAMMDD	COD_ERR_BACK: SFG032 MSG_H2H: FORMATO DE FECHA DE PRESENTACIÓN INICIAL INVÁLIDO COD_CCBI: I52	• Debe ser un día hábil bancario • Debe contener la fecha del día que se ingresa el archivo en H2H • Debe ser igual a la Fecha de Presentación del encabezado	• COD_ERR_BACK: SFG745 MSG_H2H: FECHA DE PRESENTACION INICIAL NO ES DIA HABIL COD_CCBI: Q3 • COD_ERR_BACK: SFG746 MSG_H2H: FECHA DE PRESENTACION INICIAL NO ES IGUAL A LA FECHA DE PROCESO COD_CCBI: Q4 • COD_ERR_BACK: SFG044 MSG_H2H: FECHA DE PRESENTACIÓN INICIAL DIFERENTE A LA FECHA DE PRESENTACIÓN COD_CCBI: T1	Debe estar informado y siguiente formato: AAAAMMDD
28	Uso Futuro	Alfanumérico	12	411	422	Disponible para uso futuro, debe contener espacios Para producto Confirming (Informado en la respuesta): = Secuencia o transmisión del archivo emitida por Confirming	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
29	Referencia Cliente	Alfanumérico	30	423	452	Información que se intercambia solo entre Cliente-Banco para una reconciliación posterior por parte del Cliente (CAMPO PARA FOLIO SET)	El campo es opcional, si se informa debe estar justificado a la izquierda	N/A	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
30	Descripción Referencia Pago	Alfanumérico	30	453	482	Descripción que se reflejará en el Estado de Cuenta del Cliente Originador en las operaciones que se postean a detalle, referentes a Transferencias de Fondos y Domiciliaciones.	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
31	Email a beneficiarios	Alfanumérico	500	483	982	Campo opcional para ingresar las direcciones de los correos a beneficiarios	Campo Opcional Si el campo se informa: - Si las direcciones sobrepasan las 500 posiciones, no se mandara error, se debe cortar y eliminar la o las última(s) dirección(es) de correo para quedar con una longitud menor a las 500 posiciones	Campo opcional No es necesaria validación alguna para este campo	Se debe separar cada direccion de correo una de otra con "," o ; Si no se separan las direcciones, no se enviara error, quedarán unidas y se almacenaran de esa manera	Campo opcional No es necesariaenviararror alguno para este campo	Campo Opcional Si el campo se informa: - Si las direcciones sob posiciones, no se mand eliminar la o las últimas para quedar con una lo posiciones
32	Linea de captura	Alfanumérico	100	983	1082	Campo obligatorio para registrar la linea de captura				

Ilustración 19 LAYOUT CECOBAN registros SPEI

Cada layout al final tiene un sumario/resumen que este compuesto por 8 campos que sirven para validar que la información contenida en el layout es correcta.

		LAYOUT CECOBAN						PRODUCTO SPEI			
	Núm Campo	Nombre del Campo	Tipo del Campo	Longitud	Posición Inicio	Posición Fin	Descripción	Validación de Formato	Código Error Formato	Validación de Dato	Código Error Dato
Sumario	2	Numero de Secuencia	Numérico	7	3	9	Número del registro con incremento ascendente. Indica la posición física de cada registro dentro del bloque.	Debe ser numérico de 7 posiciones y debe ser el último número consecutivo ascendente del archivo, el número consecutivo debe estar justificado a la derecha, relleno de ceros a la izquierda, hasta alcanzar la longitud de 7 posiciones del campo	COD_ERR_BACK: SF0010 MSG_H2H: NUMERO DE SECUENCIA INCORRECTA EN EL SUMARIO COD_CCEN: 24	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
	3	Codigo de Operación	Numérico	2	10	11	Indica si el registro es para operaciones de cargo, abono, verificación de cuentas, resultado de la verificación, devoluciones de instrucciones de cargo	Debe ser numérico de 2 posiciones y siempre debe contener 60	COD_ERR_BACK: SF0013 MSG_H2H: CODIGO DE OPERACIÓN INVALIDO COD_CCEN: 27	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
	4	Numero de Bloque.	Numérico	7	12	18	Identifica a que bloque pertenece el sumario de bloque Valor por archivo y se valida si se tiene mas de un bloque en un archivo, de lo contrario es 1. Formato DDNNNNN	Debe ser numérico de 7 posiciones y siempre debe estar informado	COD_ERR_BACK: SF0027 MSG_H2H: CAMPO NUMERO DE BLOQUE ES OBLIGATORIO COD_CCEN: C0	Se debe validar que sea consistente con el número de bloque del encabezado	COD_ERR_BACK: SF0027 MSG_H2H: NUMERO DE BLOQUE ENCABEZADO ES DISTINTO AL SUMARIO COD_CCEN: 44
	5	Numero de Operaciones	Numérico	7	19	25	La función de este campo es permitir el control y la verificación del total de registros dentro del bloque; su contenido es el número total de registros detalle dentro de ese bloque (tipo de registros 02). Para fines prácticos es el número total de instrucciones registradas en el bloque.	- Debe estar informado y debe ser número - El campo debe ir justificado a la derecha relleno con ceros sa la izquierda	COD_ERR_BACK: SF0748 MSG_H2H: FORMATO DEL CAMPO NUMERO DE OPERACIONES ES INCORRECTO COD_CCEN: C6	Debe contener el número de líneas de detalle (Tipo de Registro = 02) del mismo número de bloque del archivo	COD_ERR_BACK: SF0011 MSG_H2H: TOTAL DE OPERACIONES INCORRECTO COD_CCEN: 25
	6	Importe Total Operaciones	Numérico	18	26	43	Es el monto total de todos los importes de los Registros Detalle, considerando 16 Enteros y 2 Decimales, este campo debe grabarse con números enteros y sin incluir el punto decimal para los centavos.	- Debe estar informado y los últimos 2 dígitos serviran como decimales y no debe contener punto decimal - El campo debe ir justificado a la derecha relleno con ceros sa la izquierda	COD_ERR_BACK: SF0961 MSG_H2H: FORMATO DEL CAMPO IMPORTE TOTAL DE OPERACIONES ES INCORRECTO COD_CCEN: FC	Debe contener la suma de los campos de Importe de las líneas de detalle (Tipo de Registro = 02) del mismo número de bloque del archivo	COD_ERR_BACK: SF0012 MSG_H2H: IMPORTE TOTAL DE OPERACIONES INCORRECTO COD_CCEN: 26
	7	Uso Futuro CCEN	Alfanumérico	40	44	83	Disponible para uso futuro, debe contener espacios	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
	8	Uso Futuro Banco	Alfanumérico	399	84	482	Disponible para uso futuro, debe contener espacios	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo	Campo opcional No es necesaria validación alguna para este campo
		Validación de longitud del Sumario					La longitud del Sumario no debe exceder los **482** caracteres de la longitud del registro	No debe haber caracteres o espacios que excedan las **482** posiciones permitidas de longitud del registro	COD_ERR_BACK: SF0827 MSG_H2H: LA LONGITUD DEL SUMARIO EXCEDE LA LONGITUD PERMITIDA COD_CCEN: DF		

Las tablas anteriores solo muestran el protocolo CECOBAN utilizado por Santander y otros bancos mexicanos parcialmente (lo que corresponde a su producto SPEI).
La especificación completa incluye también pagos de una amplia variedad de servicios que se pueden automatizar:

- PRODUCTO TEF
- PRODUCTO TRANSFERENCIA MISMO BANCO
- PRODUCTO NOMINA MISMO BANCO
- PRODUCTO ALTA PAGOS DE CONFIRMING
- PRODUCTO IMPUESTOS FEDERALES
- PRODUCTO PAGOS REFERENCIADOS
- PRODUCTO APORTACIONES OBRERAS PATRONALES

La especificación completa del Layout CECOBAN para operaciones en Moneda Nacional esta disponible en esta liga y código QR:
https://docs.google.com/spreadsheets/d/1l_LDo4uxVvzjQx-yWzLbHmWP9e5Xph01uKktKCd_pkk/edit#gid=1689382279

La especificación completa del Layout CECOBAN para operaciones en dólares americanos (SPID) esta disponible en esta liga y código QR:
https://docs.google.com/spreadsheets/d/1LqBljhhijOxJfZs71Dgy6RoV-dtVzkqO/edit#gid=291746021

En base a los protocoloes anteriores VULCANO genera un archivo de Texto plano concatenando los campos con la información almacenada en MongoDB de operaciones en cola pendientes de ejecución. La concatenación no lleva ningún separador por lo que la posición de cada carácter define su significado. Se tiene especial cuidado en rellenar con espacios en blanco los campos no utilizados y rellenar los espacios sobrantes de un

campo. Hay algunos campos numéricos donde se debe omitir el punto decimal y expresar montos de dinero en centavos, no en pesos. Una vez compilado y validado el layout este es encriptado y enviado sobre un canal VPN dentro del cual se expone un servidor SFTP protegido Infraestructura de Llave Pública.

El Flujo operativo de envío de archivos del servidor del Cliente al servidor de Host to Host del Banco es el siguiente:

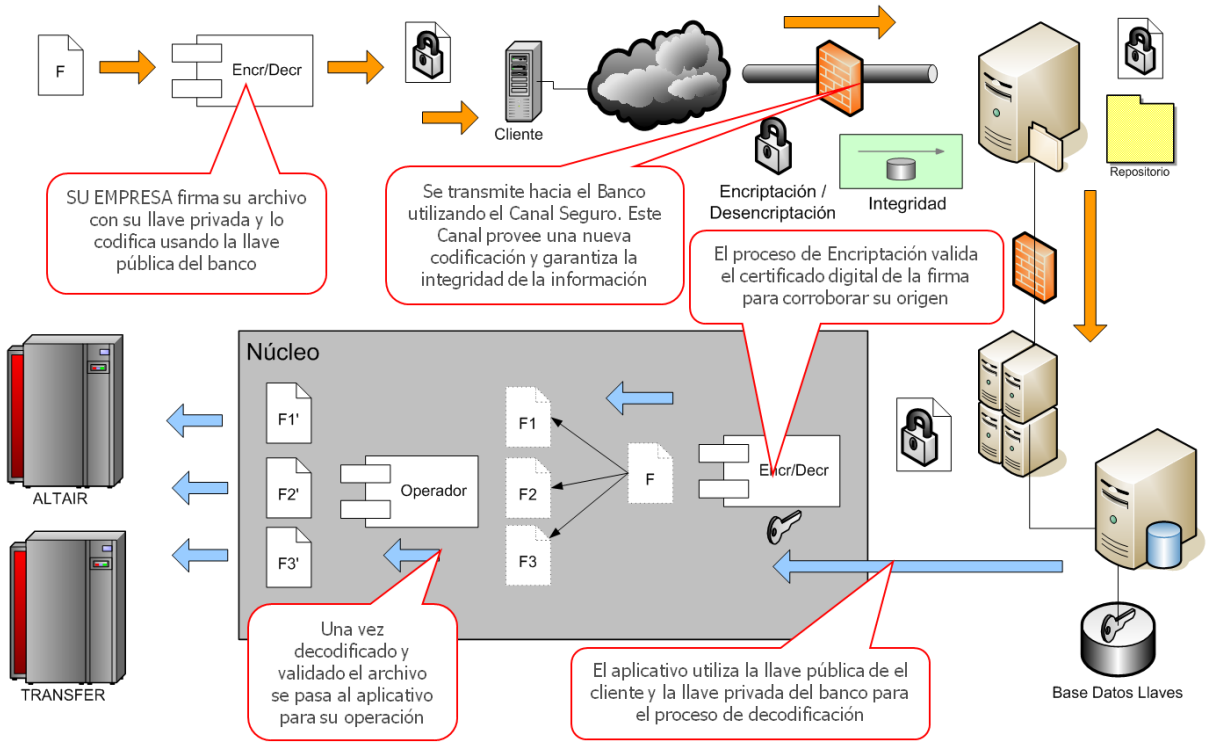


Ilustración 20 Flujo operativo de envío de archivos del servidor del Cliente al servidor

El Flujo operativo de recepción de archivos de respuesta del servidor de Host to Host del Banco hacia el servidor del Cliente es el siguiente:

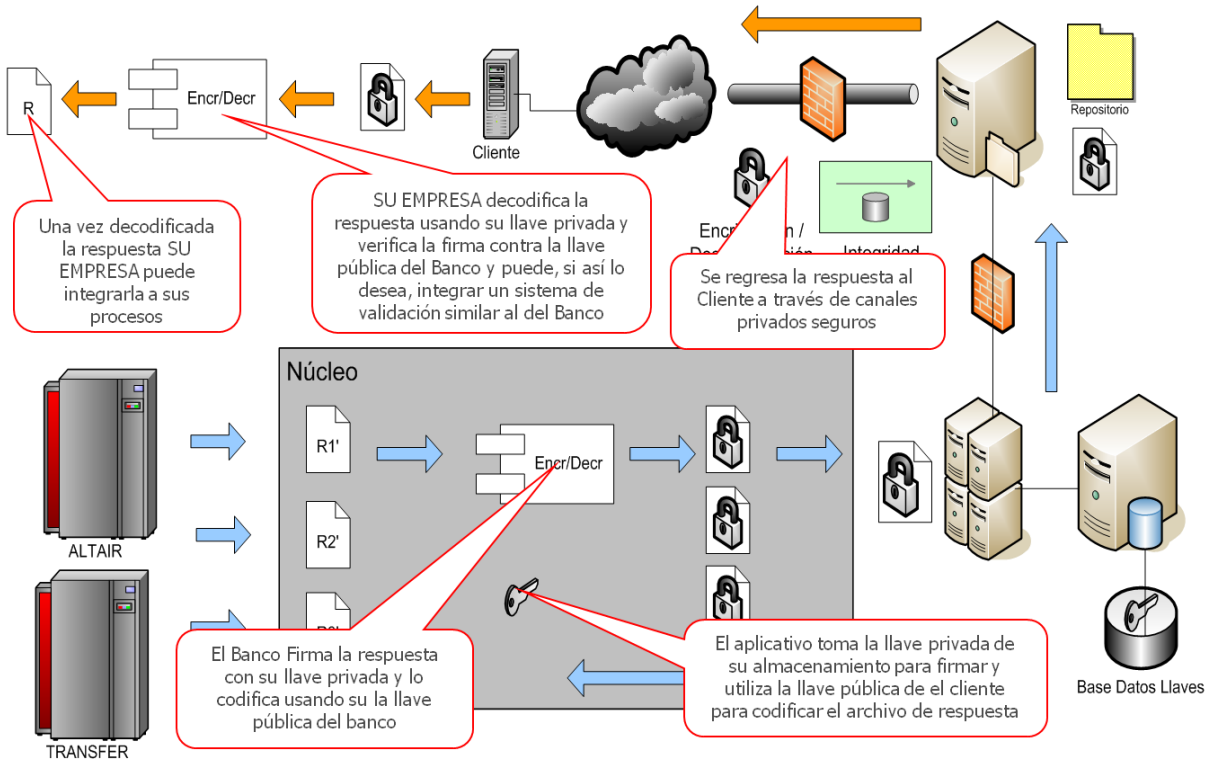


Ilustración 21 Flujo operativo de recepción de archivos de respuesta del servidor de

El siguiente diagrama obtenido de la Documentación muestra el Detalle Técnico para la construcción de VPN Site to Site y el Flujo Operativo del H2H para el caso específico de Santander, sin embargo, es un esquema prácticamente idéntico al de otras instituciones como Monex, Banregio, Bancomer o HSBC por nombrar algunas con las que hicimos pruebas.

Ilustración 22 Detalle Técnico para la construcción de VPN Site to Site y el Flujo

De acuerdo con la documentación proporcionada por Santander, los Requerimientos Técnicos para establecer una conexión H2H son:

1. Servidor dedicado con IP fija y sistema operativo Windows 7, 8, 10, 2008 Server o 2012 Server.

2. Contar con una IP fija en el servidor dedicado y un Concentrador de VPN (Firewall) con IP pública para realizar el túnel de comunicación site to site.
3. 10 GB de espacio libre en disco duro.
4. Acceso a Internet de banda ancha desde el servidor dedicado.
5. Configurar en el Concentrador de VPN (Firewall) las políticas de acceso y ruteo para la dirección IP Privada 192.240.110.98 del servidor Productivo de H2H Santander.

Así funciona Host to Host:

1. Las propuestas de pago se generan y autorizan desde VULCANO.
2. VULCANO genera los archivos de pago y los coloca en el servidor dedicado del cliente.
3. H2H traduce de forma automatizada los archivos que se vayan colocando en el servidor dedicado del Cliente y los envía al servidor de H2H de **Santander**.
4. **Santander** toma los archivos de pago recibidos con la instrucción de pago, ejecuta y genera la respuesta la cual es transmitida el servidor dedicado de H2H de **Santander**.
5. VULCANO toma la respuesta y genera sus procesos de conciliación bancaria con los pagos aceptados y rechazados.

En la arquitectura de VULCANO, se propone crear microservicios independientes para las actividades relacionadas con generar y procesar los archivos (***vulcano-layout-gen-pro***), encriptar los archivos (***vulcano-santander-crypto***) y transmitirlos a servidor dedicado para el Cliente en infraestructura de SANTANDER (***vulcano-santander-sftp***) con el fin de que los usuarios (Robots o Humanos) no tengan que preocuparse por los detalles de implementación y sus complicaciones inherentes.

Preparación de proceso de “Integración Continua” con VSTS

Dentro de la organización **ISBIT SA DE CV**, la metodología utilizada en los procesos de desarrollo y administración de aplicaciones de software es conocida como “DevOps” (Daniels, J. 2016) que es un paradigma bajo el cual se logra mayor eficacia de los recursos (humanos) disponibles al tender un puente y conectar mediante herramientas como **VSTS** las actividades de “Desarrollo” y “Operaciones”.

En el ciclo de trabajo para VULCANO, se definieron Tuberías de Despliegue que tienen el efecto de desencadenar procesos para automatizar la ejecución de pruebas de calidad y el despliegue en un *ambiente de pruebas* casi idéntico a *producción* cada vez que un cambio al código fuente es guardado en el repositorio de GIT, o más llanamente: cada vez que se realiza un “comitt” (Chacon, S. & Straub, B., 2022).

El siguiente archivo de configuración en el DSL de VSTS sirve para construir una imagen ejecutable del microservicio “vulcano-dashboard” y empujarla a un ACR (Repositorio de Contenedores de

Azure) desde donde podrá ser descargado posteriormente por el Orquestador de Contenedores en la Etapa de Despliegue (“Release”) en el ambiente de Calidad (QA) o producción.

Archivo de Configuración de Tubería **azure-pipelines.yml** para vulcano-dashboard

```
# Docker
# Build a Docker image
# https://docs.microsoft.com/azure/devops/pipelines/languages/docker

trigger:
 - master

resources:
 - repo: self

variables:
 tag: '$(Build.BuildId)'
 imageName: 'vulcano_front'

stages:
 - stage: Build
   displayName: Build image
   jobs:
    - job: Build
      displayName: Build
      pool:
       vmImage: 'ubuntu-latest'
      steps:
       - task: Docker@2
         displayName: Build an image
         inputs:
          repository: $(imageName)
          command: buildAndPush
          dockerfile: '**/Dockerfile'
          containerregistrytype: 'Container Registry'
          dockerRegistryConnection: 'vulcano_acr_dev'
          containerRegistry: 'vulcano_acr_dev'
          buildArguments: |
           NPM_TOKEN=0************************8f1
           DOCKER_ENV=dev
          imageName: 'vulcano9cb1.azurecr.io/vulcano_front:$(Build.BuildId)'
          includeLatestTag: true
          tags: |
           $(tag)
           latest
```

Ilustración 23 Archivo de Configuración de Tubería azure-pipelines.yml para vulcano-dashboard

Todos los demás microservicios de **VULCANO,** siguen el mismo patrón y tienen definidas sus tuberías correspondientes para construir las imágenes de sus contenedores y almacenarlas en un ACR (Azure Container Registry). Ver el anexo D para obtener información sobre las tuberías para la etapa de creación y distribución de imágenes de los contenedores de los otros microservicios que componen **VULCANO**.

Existe otra tubería VSTS adicional, que ejecuta comandos de la herramienta **HELM** para desplegar la aplicación **VULCANO** conformada por las distintas imágenes en el Cluster de Kubernetes objetivo. Para tal efecto es necesario que la configuración de la aplicación esté “empaquetada” con el

estándar conocido como “Helm Chart” que es un conjunto de archivos en formato YAML, como los siguientes que forman parte de repositorio **vulcano-charts**:

HELM template **santander-crypto-service.yaml** (de repositorio vulcano-charts)

```
# Vulcano container handling Santander Encription of file to be sent via  Host to Host connections
#
#=======================================
apiVersion: apps/v1
kind: Deployment
metadata:
 name: {{ .Release.Name }}-{{ .Values.santanderCrypto.name }}
 labels:
   app: {{ .Release.Name }}-{{ .Values.santanderCrypto.name }}
   chart: {{ template "vulcano.chart" . }}
   release: {{ .Release.Name }}
   heritage: {{ .Release.Service }}
spec:
 replicas: {{ .Values.santanderCrypto.replicaCount }}
 selector:
   matchLabels:
     app: {{ .Release.Name }}-{{ .Values.santanderCrypto.name }}
     release: {{ .Release.Name }}
 template:
   metadata:
     labels:
       app: {{ .Release.Name }}-{{ .Values.santanderCrypto.name }}
       release: {{ .Release.Name }}
   spec:
     containers:
     - name: {{ .Release.Name }}-{{ .Values.santanderCrypto.name }}
       image: "{{ .Values.santanderCrypto.repository }}:{{ .Values.santanderCrypto.tag }}"
       imagePullPolicy: {{ .Values.pullPolicy }}
       ports:
       - name: crypto
         containerPort: {{ .Values.santanderCrypto.port }}
         protocol: TCP
       livenessProbe:
         httpGet:
           path: /probes/liveness
           port: crypto
         initialDelaySeconds: 20
         timeoutSeconds: 2
         periodSeconds: 20
       readinessProbe:
         httpGet:
           path: /probes/readiness
           port: crypto
         initialDelaySeconds: 5
         timeoutSeconds: 2
         periodSeconds: 20
       env:
         - name: NODE_ENV
           value: {{ .Values.env }}
         - name: DB_NAME
           value: {{ .Values.databases.vulcano }}
         - name: KEY_VAULT_NAME
           value: {{ .Values.keyVault.name }}
         - name: MONGO_URI_NAME
```

```
          value: {{ .Values.keyVault.mongoUriName }}
        - name: MONGO_URI_VERSION
          value: {{ .Values.keyVault.mongoUriVersion }}
        - name: ACCOUNT_KEY_NAME
          value: {{ .Values.keyVault.accountKeyName }}
        - name: ACCOUNT_KEY_VERSION
          value: {{ .Values.keyVault.accountKeyVersion }}
        - name: CRYPTO_PASS_NAME
          value: {{ .Values.keyVault.cryptoPassName }}
        - name: CRYPTO_PASS_VERSION
          value: {{ .Values.keyVault.cryptoPassVersion }}
        - name: ENCODED_CERTIFICATE_NAME
          value: {{ .Values.keyVault.encodedCertificateName }}
        - name: ENCODED_CERTIFICATE_VERSION
          value: {{ .Values.keyVault.encodedCertificateVersion }}
        - name: ENCODED_PRIVATE_KEY_NAME
          value: {{ .Values.keyVault.encodedPrivateKeyName }}
        - name: ENCODED_PRIVATE_KEY_VERSION
          value: {{ .Values.keyVault.encodedPrivateKeyVersion }}
        - name: CERTIFICATE_PATH
          value: {{ .Values.santanderCrypto.certificatePath }}
        - name: PRIVATE_KEY_PATH
          value: {{ .Values.santanderCrypto.privateKeyPath }}
        - name: CIFRADO_ENTRADA_DIR
          value: {{ .Values.santanderCrypto.cifradoEntrada }}
        - name: CIFRADO_SALIDA_DIR
          value: {{ .Values.santanderCrypto.cifradoSalida }}
        - name: DECODE_ENTRADA_DIR
          value: {{ .Values.santanderCrypto.decodeEntrada }}
        - name: DECODE_SALIDA_DIR
          value: {{ .Values.santanderCrypto.decodeSalida }}
        - name: GENERATED_LAYOUTS_DIR_AF
          value: {{ .Values.volume.generatedLayoutsAF }}
        - name: ENCRYPTED_LAYOUTS_DIR_AF
          value: {{ .Values.volume.encryptedLayoutsAF }}
        - name: DOWNLOADED_NOTIFICATIONS_DIR_AF
          value: {{ .Values.volume.downloadedNotificationsAF }}
        - name: DECRYPTED_NOTIFICATIONS_DIR_AF
          value: {{ .Values.volume.decryptedNotificationsAF }}
         - name: DOWNLOADED_RESPONSES_DIR_AF
          value: {{ .Values.volume.downloadedResponsesAF }}
        - name: DECRYPTED_RESPONSES_DIR_AF
          value: {{ .Values.volume.decryptedResponsesAF }}
        - name: DOWNLOADED_REPORTS_DIR_AF
          value: {{ .Values.volume.downloadedReportsAF }}
        - name: DECRYPTED_REPORTS_DIR_AF
          value: {{ .Values.volume.decryptedReportsAF }}
        - name: STORAGE_ACCOUNT
          value: {{ .Values.volume.storageAccount }}
        - name: SHARE
          value: {{ .Values.volume.share }}
        {{- $root := . }}
        {{- range $ref, $values := .Values.secrets }}
        {{- range $key, $value := $values }}
        - name: {{ $ref | upper }}_{{ $key | upper }}
          valueFrom:
            secretKeyRef:
              name: {{ template "vulcano.fullname" $root }}-{{ $ref }}
              key: {{ $key }}
        {{- end }}
        {{- end }}
      resources:
```

```
{{ toYaml .Values.resources | indent 12 }}
  {{- with .Values.imagePullSecrets }}
    imagePullSecrets:
{{ toYaml . | indent 8 }}
  {{- end }}
  {{- with .Values.affinity }}
    affinity:
{{ toYaml . | indent 8 }}
  {{- end }}
  {{- with .Values.tolerations }}
    tolerations:
{{ toYaml . | indent 8 }}
  {{- end }}
```

Ilustración 24 HELM template santander-crypto-service.yaml (de repositorio vulcano-charts)

4. Metodología y recursos empleados

Mecanismos para conexión a la banca transaccional mediante red privada virtual denominado en la industria como "HOST to HOST" o H2H

Host to Host nos ayuda a establecer una conexión punto a punto entre cualquier banco y su sistema interno (ERP) para operar de forma automática diversos servicios transaccionales y agilizar sus procesos de conciliación.

La manera en cómo funciona es la siguiente:

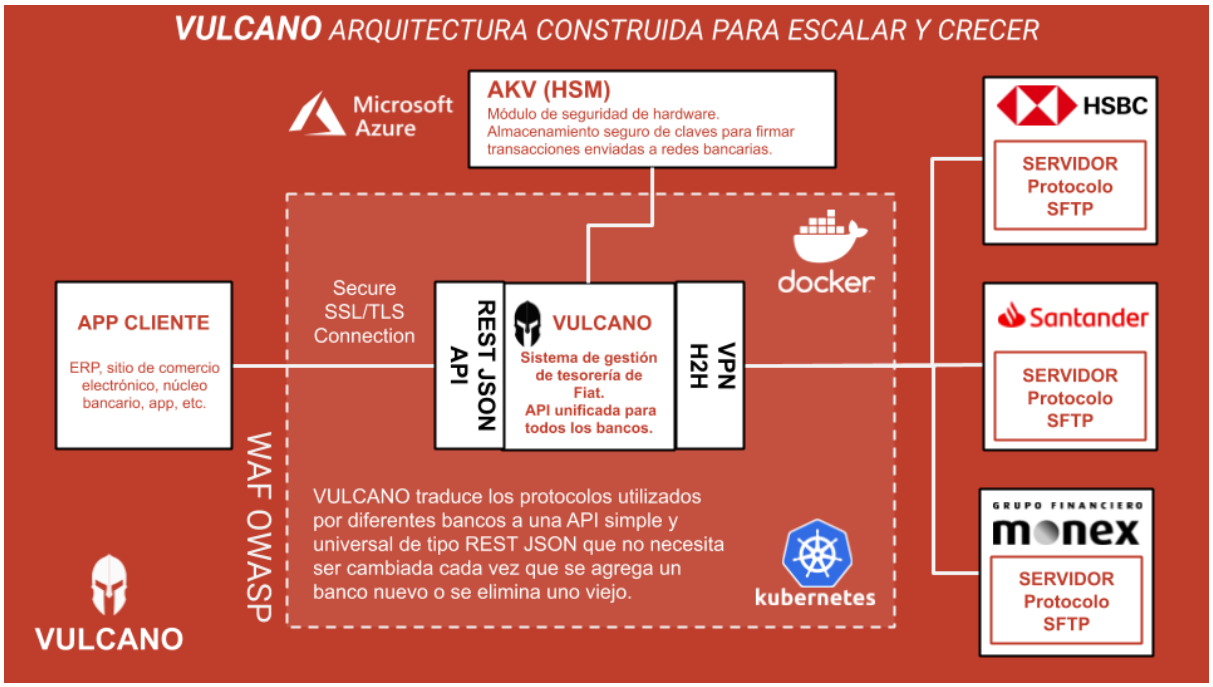


Ilustración 25 Arquitectura construida para escalar y crecer

A partir de la información de ERP de la persona, se envía la orden de transacción a Vulcano para que gestione la tesorería de fiat y genere una única API tipo Json para todos los bancos, posteriormente, Vulcano manda la instrucción directamente al banco por medio de VPN.

La ventaja de usar H2 es que permite enviar y validar archivos las 24 horas del día, sin embargo, es importante considerar que cada operación tiene un horario particular. El servicio está disponible los 365 días del año, generan las instrucciones de pago directamente desde su sistema interno (ERP) y se notifica sobre los pagos vía email tanto para el pagador y el beneficiario.

En el Anexo C se encuentra la documentación de H2H Santander.

Paradigma de Microservicios y Contenedorización de aplicaciones en la nube

Veamos un ejemplo concreto de cómo la información fluye para primero enviar transacciones SPEI y después recuperar información de cambio de estatus de transacciones, entre los distintos Microservicios, en lugar de en un algoritmo monolítico.

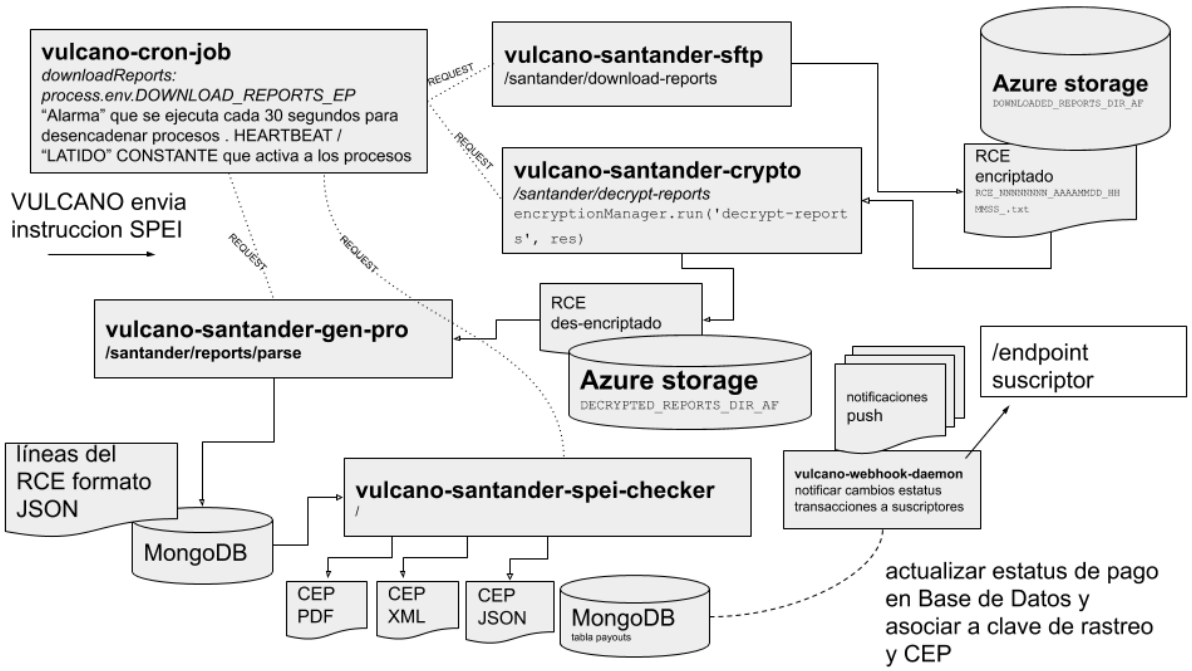


Ilustración 26 Actualizar estatus de pago y asociar a clave de rastreo y CEP

La actividad de enviar una instrucción de pago es una tarea asíncrona e independiente de la tarea de recuperar la información de pagos realizados por cuentas de empresas administradas por **VULCANO** o por cuentas de terceros.

Posteriormente (a enviar una orden de pago) es necesario recuperar la información del estado de la transacción periódicamente conforme este cambie. La transacción podría fallar por una gran variedad de razones o tener éxito.

En caso de tener éxito (un SPEI o SPID) es necesario recuperar los comprobantes electrónicos de pago (CEP) de la transacción y asociarlos a la misma en la base de datos para consulta futura y auditoría.

El modelo en la BD de cada reporte de cobranza enriquecido tiene varias transiciones de estado que el siguiente diagrama ilustra:

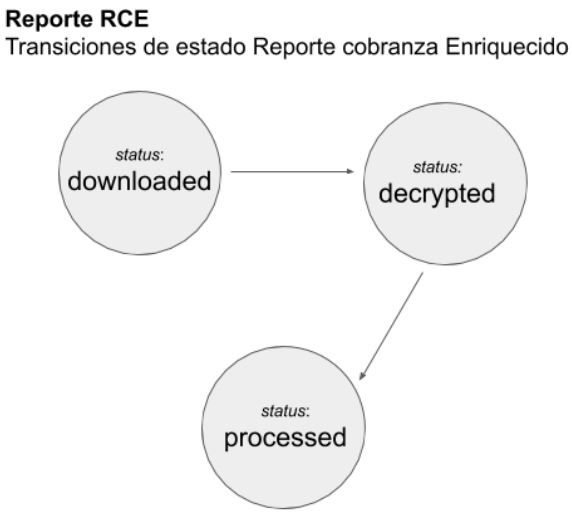


Ilustración 27 Transiciones Estado Reporte RCE

Cada transacción contenida en el RCE (Reporte Cobranza Enriquecido) también es representada en la base de datos por un documento que tiene los cambios de estado ilustrados en el siguiente diagrama:

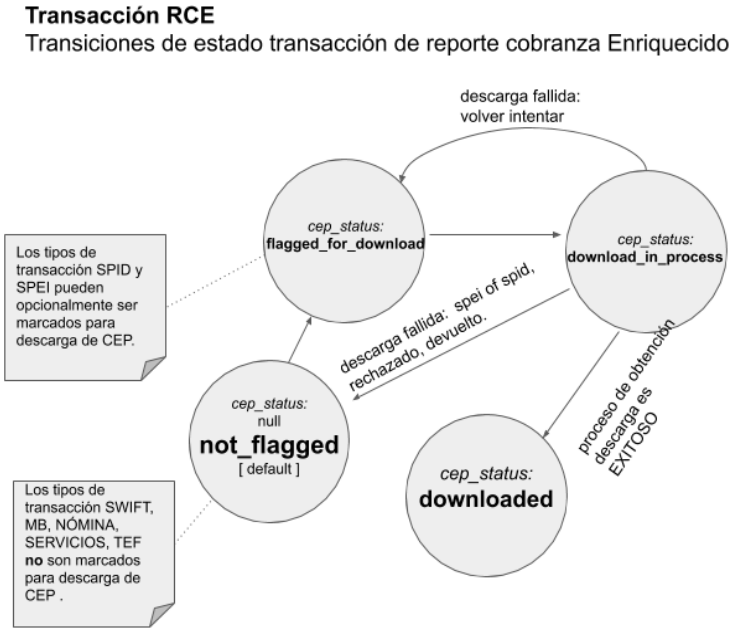


Ilustración 28 Transacción RCE (transiciones de estado de transacción y CEP asociado)

El diagrama anterior muestra que existen 3 estados "transitorios" del "CEP" asociado (***flagged_for_download***, ***not_flagged*** y ***download_in_process***) y 1 estado "absorvente" (***downloaded***). Esto significa que una vez que el estado del CEP es ***downloaded*** ya no cambiará mas. Sin embargo, un CEP con estatus ***flagged_for_download*** puede transicionar a ***download_in_process***.

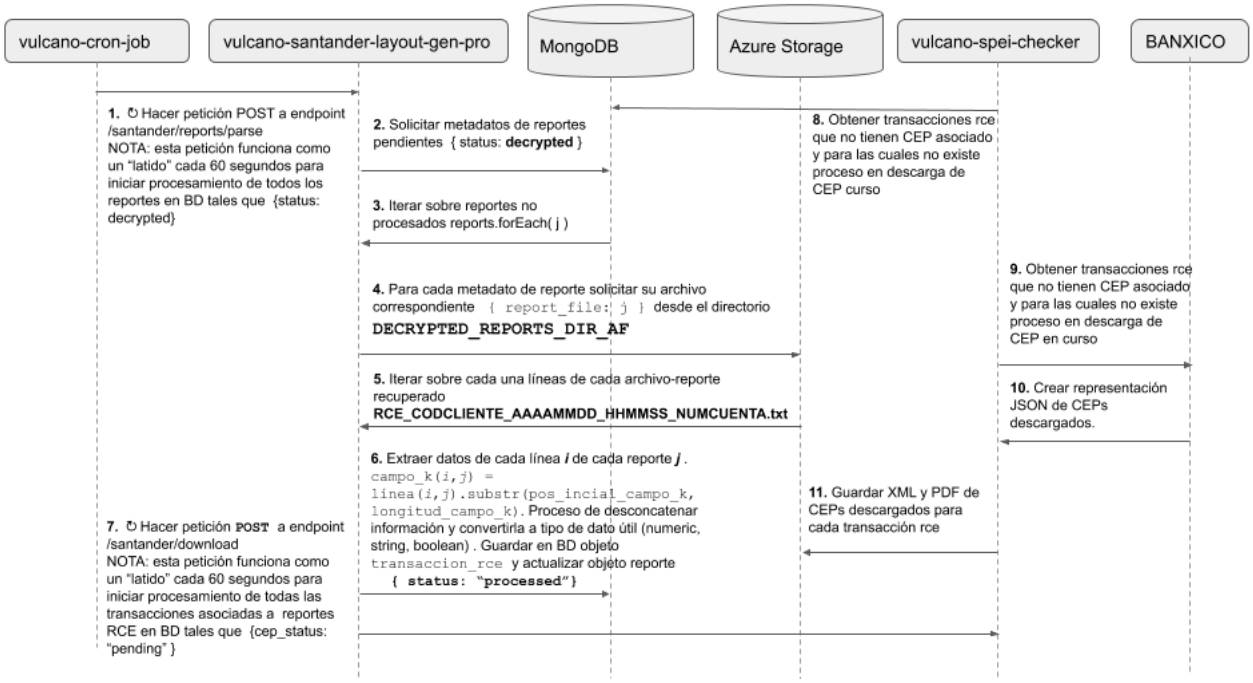


Ilustración 29 Pasos a seguir para la obtención del CEP

La explicación y diagramas anteriores muestran el proceso a nivel lógico sin entrar en detalles de implementación. La implementación del Paradigma de Microservicios en el caso de VULCANO, depende de 3 tecnologías clave sobre las que profundizaremos a continuación. A saber; Docker, Kubernetes y Azure.

Docker

Docker implementa una API de alto nivel para proporcionar contenedores livianos que ejecutan procesos de manera aislada.

Construido sobre las facilidades proporcionadas por el kernel Linux (principalmente ***cgroups*** y ***namespaces***), un contenedor Docker, a diferencia de una máquina virtual, no requiere incluir un sistema operativo independiente. En su lugar, se basa en las funcionalidades del kernel y utiliza el aislamiento de los recursos (CPU, memoria, el bloque E/S, red, etc.) y los ***namespaces*** separados, con el fin de aislar la vista de una aplicación del sistema operativo.

Docker accede a la virtualización del kernel Linux, ya sea directamente a través de la biblioteca lib container (disponible desde Docker 0.9), o indirectamente a través de ***libvirt***, LXC o ***systemd-nspawn***.

Mediante el uso de contenedores, los recursos pueden ser aislados y los servicios restringidos. Se otorga a los procesos la capacidad de tener una visión casi completamente privada del sistema operativo con su propio identificador de espacio de proceso, la estructura del sistema de archivos, y las interfaces de red. Contenedores múltiples comparten el mismo núcleo, pero cada contenedor puede ser restringido a utilizar solo una cantidad definida de recursos como CPU, memoria y I/O (entrada/"input" y salida/"output" de datos).

Usar Docker para crear y gestionar contenedores puede simplificar la creación de sistemas altamente distribuidos, permitiendo que múltiples aplicaciones, las tareas de los trabajadores y otros procesos funcionen de forma autónoma en una única máquina física o en varias máquinas virtuales. Esto permite que el despliegue de nodos se realice a medida que se dispone de recursos o cuando se necesiten más nodos, lo que permite una plataforma como servicio (PaaS - Platform as a Service) de estilo de despliegue y ampliación de los sistemas. Docker también simplifica la creación y el funcionamiento de las tareas de carga de trabajo o las colas y otros sistemas distribuidos.

Se tomó la decisión de constituir ***VULCANO*** como una combinación de Microservicios, cada uno a su vez es soportado por una o varias réplicas de un contenedor Docker.

El siguiente ejemplo es el ***Dockerfile*** del microservicio ***vulcano-santander-crypto*** y muestra claramente cómo se utiliza esta tecnología para crear un ambiente de ejecución aislado

que contiene las versiones específicas del software requerido por Banco Santander en su manual de Conexión H2H, tales como versiones de JAVA antiguas compatibles con software de Cifrado y versión de linux de 32 Bits.

Dockerfile de Microservicio (vulcano-santander-crypto)

```
# Inherit docker image with node 32 bits
FROM i386/node AS base
LABEL maintainer="Sebastian Acosta<sebastian@isbit.co>"
# Create work directory
WORKDIR /src
# Bundle app source
COPY . .

FROM base AS testing
# Set environment
ENV NODE_ENV="test"
# Set Build-Arg to Authenticate to NPM Registry
ARG NPM_TOKEN
#Install all node packages and run tests
# RUN npm install && npm test

# Release node
FROM base AS release
# Set environment
ARG DOCKER_ENV
ENV NODE_ENV=${DOCKER_ENV}
# Remove test packages
RUN rm -rf test/
# Install Java
RUN mkdir ./java
COPY ./software/jre-8u20-linux-i586.tar.gz ./java
RUN cd /src/java && tar zxvf jre-8u20-linux-i586.tar.gz && rm jre-8u20-linux-i586.tar.gz
COPY ./software/local_policy.jar ./java/jre1.8.0_20/lib/security
COPY ./software/US_export_policy.jar ./java/jre1.8.0_20/lib/security
ENV PATH="/src/java/jre1.8.0_20/bin:${PATH}"
# Make CifradoEntrada, CifradoSalida, DecodeEntrada and DecodeSalida dirs
RUN cd apicifrado && mkdir CifradoEntrada && mkdir CifradoSalida && mkdir DecodeEntrada && mkdir DecodeSalida
# Set Build-Arg to Authenticate to NPM Registry
ARG NPM_TOKEN
# Install app dependencies without dev packages
RUN npm install --production
# Remove .npmrc from container
RUN rm -f .npmrc

EXPOSE 33440 33440
CMD ["npm", "start"]
```

Ilustración 30 Dockerfile (vulcano-santander-crypto)

La ventaja de los contenedores es que cada microservicio puede tener ambientes totalmente distintos. Por ejemplo, el microservicio ***vulcano-santander-sftp***, a diferencia del microservicio ***vulcano-santander-crypto***, utiliza la última versión de "node" que a su vez depende de una capa previa en su *Dockerfile* con la versión de 64 bits de linux.

Dockerfile de Microservicio (vulcano-santander-sftp)

```
FROM node:latest AS base
LABEL maintainer="Sebastian Acosta <sebastian@isbit.co>"
# Create app directory
WORKDIR /app
# Bundle app source
COPY . .
# Testing node
FROM base AS testing
# Set environment
ENV NODE_ENV="test"
# Set Build-Arg to Authenticate to NPM Registry
ARG NPM_TOKEN
# Install all node packages and run tests
RUN npm install && npm test
# Release node
FROM base AS release
# Set environment
ARG DOCKER_ENV
ENV NODE_ENV=${DOCKER_ENV}
# Remove test modules and .env
RUN rm -rf test/
# Set Build-Arg to Authenticate to NPM Registry
ARG NPM_TOKEN
# Install app dependencies without dev packages
RUN npm install --production
# Remove .npmrc from container
RUN rm -f .npmrc
EXPOSE 33436 33436
CMD ["npm", "start"]
```

Ilustración 31 Dockerfile de Microservicio (vulcano-santander-sftp)

El equipo de Desarrollo y Sistemas de la empresa ***ISBIT S.A. de C.V.*** experimentó grandes ventajas al utilizar ***Docker*** como piedra angular para construir ***VULCANO***. El paradigma de contenedores y la Tecnología ***Docker*** nos permitió utilizar la herramienta adecuada para cada trabajo. Cada banco proporciona sus requisitos técnicos de forma muy distinta, por lo que no fue factible fue utilizar un enfoque monolítico. El ambiente que se requiere para la interoperabilidad con cada banco puede variar en diferentes temas desde lenguajes de programación, versiones de software usadas, versiones y tipos de Sistemas Operativos, arquitectura de procesador (64 bits, o 32 bits? AMD, Intel?), marca y modelo de cortafuegos, estándares en cifrados, ente otras cosas

Los archivos en formato "***Dockerfile"*** de los demás microservicios están incluidos por completez en el **Anexo D (Código Fuente)**.

Kubernetes

Kubernetes es una plataforma portable y extensible de código abierto para administrar

cargas de trabajo y servicios, un “orquestador” de contenedores. Kubernetes facilita la automatización y la configuración declarativa. Tiene un ecosistema grande y en rápido crecimiento. El soporte, las herramientas y los servicios para Kubernetes están ampliamente disponibles.

Kubernetes ofrece un entorno de administración **centrado en contenedores**, orquesta la infraestructura de cómputo, redes y almacenamiento para que las cargas de trabajo de los usuarios no tengan que hacerlo. Esto ofrece la simplicidad del paradigma ***Plataforma como Servicio*** (PaaS) con la flexibilidad de la ***Infraestructura como Servicio*** (IaaS) y permite la portabilidad entre proveedores de infraestructura (“Kubernetes Components”, The Kubernetes Authors, 2020).

Los flujos de trabajo de las aplicaciones pueden optimizarse para acelerar el tiempo de desarrollo. Una solución de orquestación propia puede ser suficiente al principio, pero suele requerir una automatización robusta cuando necesita escalar. Es por ello que Kubernetes fue diseñada como una plataforma para poder construir un ecosistema de componentes y herramientas que hacen más fácil el desplegar, escalar y administrar aplicaciones.

Las etiquetas, o Labels, les permiten a los usuarios organizar sus recursos como deseen. Las anotaciones les permiten asignar información arbitraria a un recurso para facilitar sus flujos de trabajo y hacer más fácil a las herramientas administrativas inspeccionar el estado.

Además, el Plano de Control de Kubernetes usa las mismas APIs que usan los desarrolladores y usuarios finales. Los usuarios pueden escribir sus propios controladores, como por ejemplo un planificador o scheduler, usando sus propias APIs desde una herramienta de línea de comandos.

Este diseño ha permitido que otros sistemas sean construidos sobre Kubernetes. A continuación, un diagrama general de la arquitectura de Kubernetes donde destaca el rol de ***etcd*** para guardar el estatus de los recursos en el clúster (Juggery, L., 2019).

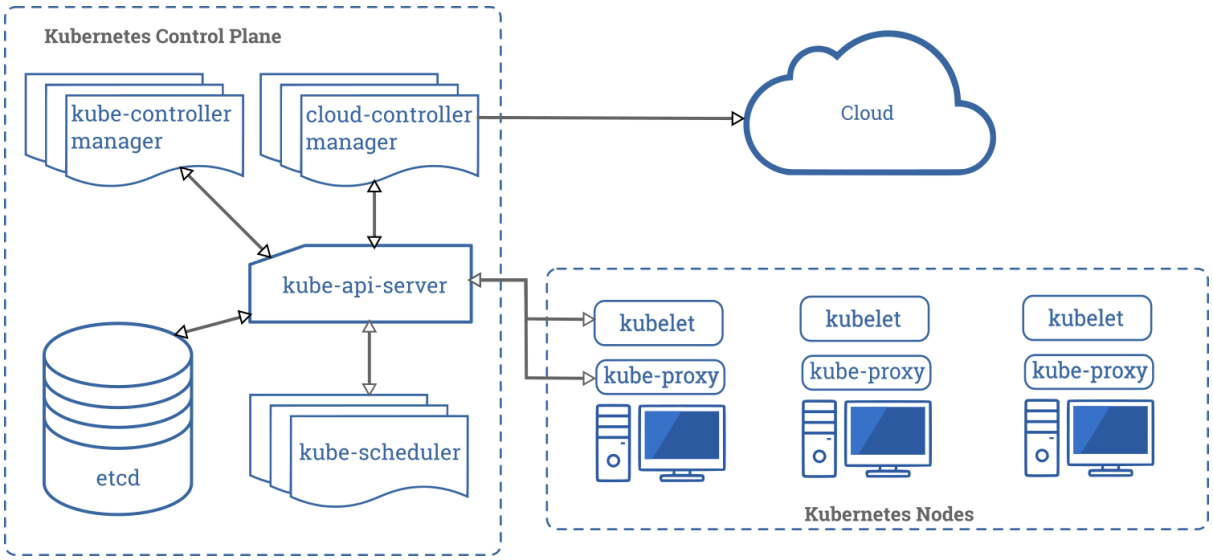


Ilustración 32 Diagrama de Arquitectura de Kubernetes reproducida de documentación

Las aplicaciones de producción real abarcan varios contenedores. Esos contenedores deben implementarse en varios hosts de servidores. La seguridad de los contenedores tiene varias capas y puede ser complicada. Aquí es donde Kubernetes nos ayuda, ofreciéndonos la capacidad de organización y gestión necesaria para implementar contenedores a escala para estas cargas de trabajo. El sistema de organización de Kubernetes nos permite diseñar servicios de aplicaciones que abarcan varios contenedores, programar esos contenedores en un clúster, ampliarlos y gestionar su estado a lo largo del tiempo.

La manera propuesta por Kubernetes es desplegar contenedores basados en virtualización a nivel del sistema operativo, en vez del hardware. Estos contenedores están aislados entre ellos y con el servidor anfitrión: tienen sus propios sistemas de archivos, no ven los procesos de los demás y el uso de recursos puede ser limitado. Son más fáciles de construir que una máquina virtual, y porque no están acoplados a la infraestructura y sistema de archivos del anfitrión, pueden migrarse entre diferentes nubes y distribuciones de sistema operativo.

Ya que los contenedores son pequeños y rápidos, una aplicación puede ser empaquetada en una imagen de contenedor. Esta relación uno a uno entre aplicación e imagen nos abre un abanico de beneficios al usar contenedores. Con contenedores, podemos crear imágenes inmutables al momento de la compilación en vez del despliegue ya que las aplicaciones no necesitan componerse junto al resto del ***stack*** ni atarse al entorno de infraestructura de producción. Generar una imagen de contenedor al momento de la compilación permite tener un entorno consistente en las diferentes etapas desde *desarrollo*, *QA* y hasta *producción*. De igual forma, los contenedores son más transparentes que las máquinas virtuales y eso hace que el monitoreo y la administración sean más fáciles. Esto se aprecia más cuando los ciclos de vida de los contenedores son administrados por la infraestructura en vez de un proceso supervisor escondido en el

contenedor. Por último, ya que solo hay una aplicación por contenedor, administrar el despliegue de la aplicación se reduce a administrar el contenedor.

Aplicación de kubernetes en VULCANO

Los microservicios más importantes que maneja VULCANO son tres y cada uno tiene una función diferente, a continuación, la descripción.

Microservicios	Función
vulcano-layout-gen-pro-service 	Genera y procesa la información convirtiéndola en un layout, que es un archivo de texto que contiene instrucciones sobre transacciones que se le envían al banco o viceversa.
vulcano-<banco>-crypto-service 	Desencripta o encripta el archivo layout que obtenemos del banco o que enviamos al banco. Es importante notar que existen múltiples microservicios de esta familia. Uno para cada banco con el que se tiene contratado el servicio H2H hasta el momento: ***vulcano-santander-crypto-service,*** ***vulcano-banorte-crypto-service,*** ***vulcano-monex-crypto-service,*** ***vulcano-bbva-crypto-service,*** ***vulcano-hsbc-crypto-service y*** ***vulcano-banregio-crypto-service***
vulcano-<banco>-sftp-service 	Este microservicio se encarga de los detalles de comunicación con el banco mediante una VPN y la subida y bajada de archivos “layout” mediante el protocolo Secure File Transfer Protocol. Este microservicio establece el puente al Dispositivo Cortafuegos correspondente que tiene una IP estática en la Lista Blanca del Banco. Se mantienen varias IPs estáticas para cada uno de los siguientes microservicios en esta familia: ***vulcano-santander-sftp-service,*** ***vulcano-banorte-sftp-service,***

	vulcano-monex-sftp-service, ***vulcano-bbva-sftp-service,*** ***vulcano-hsbc-sftp-service y*** ***vulcano-banregio-sftp-service***
vulcano-webhooks-service 	Nos ayuda a notificar a los involucrados en tiempo real sobre sus transacciones actualizándose cuando se produce el evento.

Ilustración 33 Microservicios orquestados por Kubernetes que componen VULCANO

Es importante mencionar que cada banco tiene sus propios protocolos de información, por lo tanto, se necesita hacer un microservicio diferente para cada uno.

Docker nos permite correr cargas de trabajo de manera ***contenerizada*** como ya explicamos en la sección de Docker. Una vez que los microservicios se encuentran en un contenedor entra Kubernetes a orquestarlos, es decir, nos permite manejar esas aplicaciones a través de muchos "microservidores". Imaginemos que en algún momento la parte de "***vulcano-santander-sftp-service***" falla o deja de funcionar, pues resulta que Kubernetes crea réplicas para que en cuanto falle una entre una nueva a sustituirla.

Para mayor claridad, veamos el funcionamiento de Kubernetes en dos diagramas, el primero indica el proceso en donde el banco recibe la instrucción y el segundo cuando VULCANO la recibe:

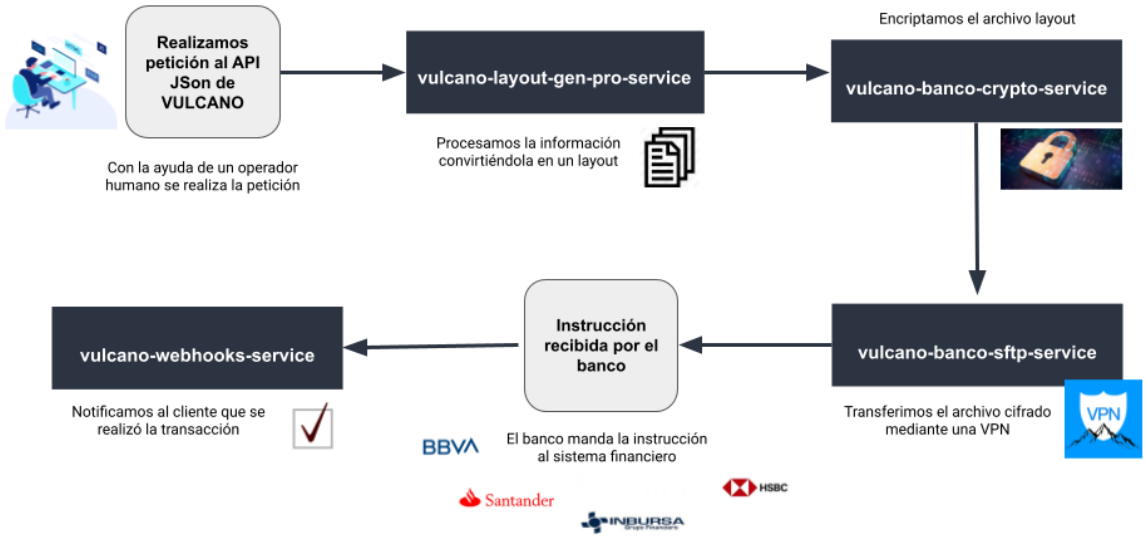


Ilustración 34 Uso de kubernetes en Vulcano; el banco recibe la instrucción

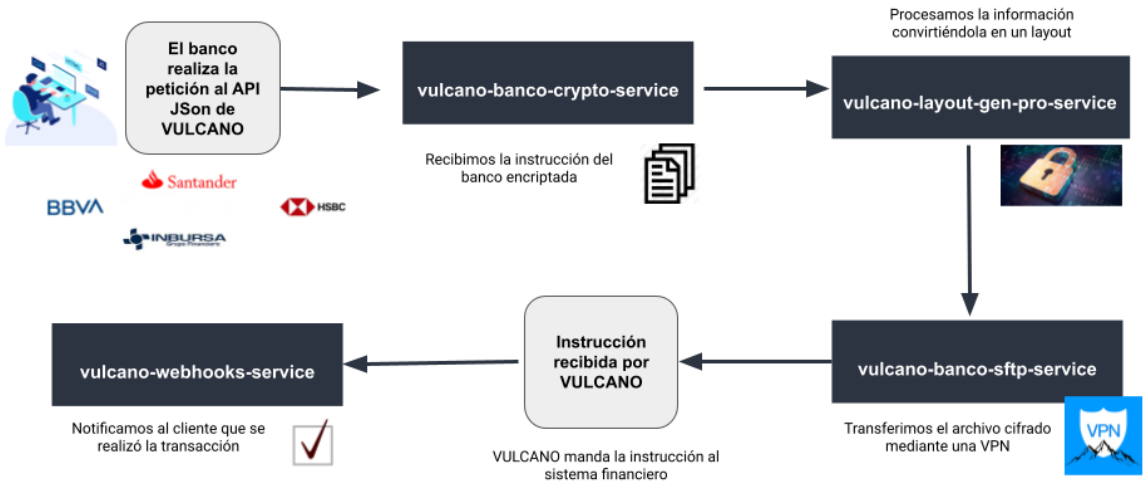


Ilustración 35 Uso de kubernetes en Vulcano; Vulcano recibe la instrucción

Concepto de Estado Deseado y Sistema que se Autorepara

Desde un principio nos propusimos utilizar el estado del arte en automatización para que VULCANO pudiera funcionar de forma totalmente desatendida y así evitar la intervención del personal de la empresa ISBIT SA DE CV en caso de una condición de error.

A continuación, se describe un concepto clave en Kubernetes que encaminó al equipo de desarrollo de ISBIT SA DE CV a crear un sistema capaz de aumentar su capacidad de procesamiento de transacciones de manera dinámica al experimentar un aumento en demanda (sin saturarse y fallar) y sea capaz de liberar recursos en caso de una disminución de demanda (para economizar el gasto y mejorar el presupuesto destinado a infraestructura en la Nube Pública Azure de la Empresa ISBIT SA DE CV). Asi mismo al tratarse VULCANO de un sistema de misión crítica, los participantes en el diseño y desarrollo del software, decimos que VULCANO debería de recuperarse de manera automática de fallos de redes, máquinas virtuales y otras condiciones inesperadas.

De acuerdo con el autor Michael Hausenblas(2019) el bucle de control de Kubernetes es como sigue:

1. Leer el estado de los recursos, preferentemente mediante mecanismo basado en *eventos* (ver la sección "watches" en el Capítulo 3 del citado libro de Michael Hausenblas)
2. Cambiar el estado de los objetos en el *cluster* o el Mundo Externo al *cluster*. Por ejemplo, lanzar un ***pod***, crear un ***endpoint*** de red o hacer una consulta a una API en la red. (ver la sección "Cambiar Objetos del Cluster o el Mundo Externo" del Capítulo 1 del citado libro de Michael Hausenblas).
3. Actualizar el estatus del recurso en el Paso 1 mediante el servidor de API en **etc.** (ver la sección sobre "Concurrencia Optimista" del citado libro de Michael Hausenblas para más detalles sobre esto).
4. Repetir ciclo; regresar al paso 1.

Sin importar que tan simple o complejo sea el controlador de Kubernetes utilizado, los tres pasos (leer el estado del recurso > cambiar el mundo para converger al estado deseado > actualizar el status del recurso en ***etcd***) siempre serán los mismos. El Bucle de Control es mostrado en la siguiente figura tomada de Programming Kubernetes (Hausenblas M., O'Reilly, USA, 2019), donde se pueden apreciar las partes móviles típicas, con el principal ciclo del controlador en el centro:

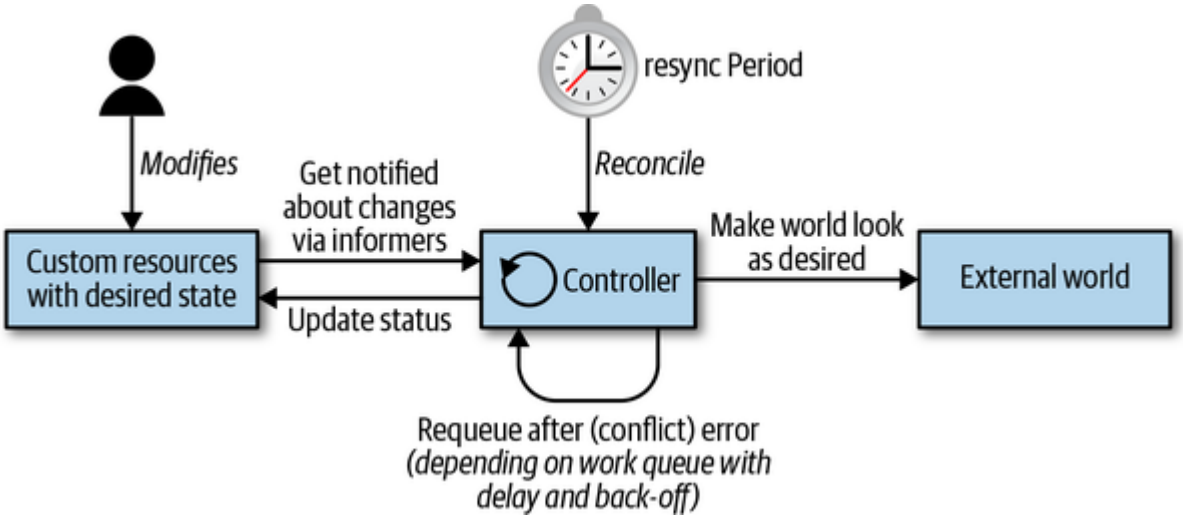


Ilustración 36 Bucle de control

El ciclo principal está corriendo continuamente dentro del proceso del controlador. Este proceso normalmente está corriendo dentro de un pod del cluster.

A continuación, se muestra el resultado del comando ***kubectl*** utilizado en entorno de QA para conocer el estado de los despliegues en el cluster con su correspondiente *estado deseado.* Se puede observar que, por ejemplo, para el microservicio ***vulcano-spei-checker*** el estado deseado es la existencia de 7 instancias de *pods* y que en el momento de ejecución del comando existían 7 disponibles. Si por cualquier razón, una instancia de ***vulcano-spei-checker*** fallara , el controlador, para lograr alcanzar el estado deseado, creará una nueva instancia que reiniciara el proceso en el punto del procesamiento último en ser guardado.

En el caso particular de **VULCANO** para poder reanudar las tareas en los momentos correctos, guardamos cada cambio de verión de nuestro objeto de negocio o modelos, en nuestra base de datos Mongodb tal como se explica en la sección sobre modelado de base de datos del capítulo 3 del presente trabajo.

Queremos evitar un pago dos veces, porque el proceso de enviar pagos falla en algún punto, o volver a descargar un ***cep*** que ya fue descargado pero tuvimos una falla al procesarlo y guardarlo en la base de datos. Estos son solo algunos ejemplos, por lo que es importante el control de versión de los objetos de negocio.

Comando para conocer estatus de Despliegues de microservicios de VULCANO
kubectl get deployments -A

NAMESPACE	NAME	READY	UP-TO-DATE	AVAILABLE	AGE
default	aks-ssh	1/1	1	1	287d
ing-front	nginx-nginx-ingress-controller	1/1	1	1	70d
ing-front	nginx-nginx-ingress-default-backend	1/1	1	1	70d
ingress-basic	cert-manager	1/1	1	1	17d
ingress-basic	cert-manager-cainjector	1/1	1	1	17d
ingress-basic	cert-manager-webhook	1/1	1	1	17d
kube-system	coredns	2/2	2	2	383d
kube-system	coredns-autoscaler	1/1	1	1	383d
kube-system	dashboard-metrics-scraper	1/1	1	1	80d
kube-system	kubernetes-dashboard	1/1	1	1	12d
kube-system	metrics-server	1/1	1	1	383d
kube-system	omsagent-rs	1/1	1	1	383d
kube-system	tiller-deploy	1/1	1	1	72d
kube-system	tunnelfront	1/1	1	1	383d
santander	vulcano	2/2	2	2	13d
santander	vulcano-external-front	1/1	1	1	13d
santander	vulcano-external-front-ng	1/1	1	1	13d
santander	vulcano-layout-gen-pro	3/3	3	3	13d
santander	vulcano-santander-crypto	4/4	4	4	13d
santander	vulcano-santander-sftp	4/4	4	4	13d
santander	vulcano-spei-checker	7/7	7	7	13d

Azure

Azure es la plataforma de aplicaciones en la nube de microsoft, esto significa que está diseñada para ser un lugar en donde se pueda ejecutar una aplicación a escala fuera de internet. Al ser centrada en aplicaciones, implica que administramos todo el ciclo de vida de nuestra aplicación, desde el diseño inicial, el desarrollo, y las pruebas de la aplicación para su implementación en la nube, todo esto tocando un simple botón para supervisar y escalar dicha aplicación cuando se ejecuta en internet.

Para el desarrollo del software, ocupamos Azure Kubernetes Software (AKS) este es un servicio de contenedor que simplifica la implementación, la administración y el uso de Kubernetes como un servicio orquestador de contenedores totalmente administrado.

Entre las principales funciones que maneja Azure AKS son:

- Actualizaciones automatizadas de Kubernetes:
- Plano de control autoderrapable
- etcd con respaldo en SSD automatizado, H/A, copia de seguridad/restauración
- Redes personalizadas (Azure VNET, CNI)
- Escalado de clusters
- TLS en todas partes respaldado por Azure KeyVault
- RBAC y Azure AD integrados
- Clusters híbridos (futuros)

En resumen, Azure es una nube pública de pago por uso que nos permite compilar, implementar y administrar rápidamente aplicaciones en una red global de centros de datos de Microsoft. Azure fue la mejor opción para poder desplegar el software.

Librerías de software

Para el desarrollo de VULCANO utilizamos dependencias npm previamente desarrolladas.

Node Package Manager (npm) es un gestor de paquetes, el cual nos facilita el trabajo al momento de usar Node, ya que gracias a él podremos tener cualquier librería disponible con solo una línea de código, también nos ayuda a administrar nuestros módulos, distribuir paquetes y agregar dependencias de una manera sencilla. Cuando instalamos nuevos paquetes lo que hace npm es instalarlo de manera local en nuestro proyecto dentro de la carpeta node_modules, pero nosotros podemos decirle que lo instale de manera global de ser necesario.

En el apartado de Kubernetes, hice mención de los microservidores más importantes para el funcionamiento de Vulcano, los cuales son: vulcano-layout-gen-pro-service, vulcano-banco-crypto-service, vulcano-banco-sftp-service, vulcano-webhooks-service.

Las dependencias usadas para la implementación de cada microservicio son:

```
"azure-storage": "^2.10.3",
"body-parser": "^1.15.0",
"debug": "^2.2.0",
"dotenv": "^7.0.0",
"express": "^4.14.0",
"eyes": "^0.1.8",
"mongoose": "^5.9.6",
"morgan": "^1.10.0",
"nodemon": "~1.17.3",
"request": "^2.88.2",
"request-promise": "^4.2.5",
"ssh2-sftp-client": "^2.5.2"
"csvtojson": "^2.0.8",
"express-validator": "^5.3.0",
"joi": "^13.7.0",
"js-yaml": "^3.13.1",
"shelljs": "^0.8.3",
"url": "^0.11.0",
"xml-js": "^1.6.9",
"xml2js": "^0.4.19"
```

```
"cookie-parser": "~1.4.3",
"applicationinsights": "^1.7.6",
```

Es importante mencionar que muchas librerías funcionan para un mismo microservicio, es decir, estas librerías en general son las que usamos para todos los microservicios.

Diseño de servicio web (API) basado en patrón de diseño REST

Veamos el formato de los Mensajes Transaccionales utilizados. Vulcano expone una API JSON REST con estructura como el de los siguientes ejemplos

	Mensaje Enviar Pago https://mit-vulcano.eastus.cloudapp.azure.com/api/payouts/send-spei
	Header de Autenticación
	Vulcano-Account-Id Vulcano-User-Token
	Cuerpo de Petición

	<pre>{
	  "method":"bank_account_clabe",
	  "to":{
	  "number":"044180001021643787",
	  "holder_name":"Sebastian Acosta Checa"
	  },
	  "amount": 39.6,
	  "description":"Pago a tercero G5"
	}</pre>

	Respuesta Exitosa

	<pre>{
	  "bank_account": {
	  "creation_date": "2019-10-01T02:24:52.363Z",
	  "clabe": "044180001021643787",
	  "holder_name": "Sebastian Acosta Checa",
	  "bank_code": "044",
	  "bank_name": "SCOTIA BANK INVERLAT"
	  },
	  "source_account": {
	  "account_type": "40",
	  "bank_code": "014",
	  "account_number": "014180220006507057"
	  },
	  "currency": "MXN",
	  "method": "bank_account_clabe",
	  "operation_type": "out",
	  "bank_operation_type": "spei",
	  "transaction_type": "payout",
	  "status": "in_progress",
	  "creation_date": "2019-10-01T02:24:52.363Z",</pre>

```
    "delivery_date": "2019-10-01T02:24:52.363Z",
    "delivered": false,
    "_id": "5d92b8f4c5f967001751599c",
    "amount": 39.6,
    "description": "Pago a tercero G5",
    "__v": 0
  }
```


Mensaje Listar Pagos
https://mit-vulcano.eastus.cloudapp.azure.com/api/payouts

Header de Autenticación

Vulcano-Account-Id
Vulcano-User-Token

Cuerpo de Petición

Respuesta Exitosa


```
[
  {
    "bank_account": {
      "creation_date": "2019-08-25T15:02:24.564Z",
      "clabe": "072180004200788982",
      "holder_name": "Felipito Acosta",
      "bank_code": "072",
      "bank_name": "BANORTE"
    },
    "source_account": {
      "account_type": "40",
      "bank_code": "014",
      "account_number": "014180220006507057"
    },
    "currency": "MXN",
    "method": "bank_account_clabe",
    "operation_type": "out",
    "bank_operation_type": "spei",
    "transaction_type": "payout",
    "status": "in_progress",
    "creation_date": "2019-08-25T15:02:24.564Z",
    "delivery_date": "2019-08-25T15:02:24.564Z",
    "delivered": true,
    "_id": "5d62a300557a4e00117b3d2c",
    "amount": 10,
    "description": "Pago a tercero",
    "__v": 0
  },
  {
    "bank_account": {
      "creation_date": "2019-08-25T17:07:28.197Z",
      "clabe": "072180004859128430",
      "holder_name": "Mara Checa",
      "bank_code": "072",
```

```
        "bank_name": "BANORTE"
      },
      "source_account": {
        "account_type": "40",
        "bank_code": "014",
        "account_number": "014180220006507057"
      },
      "currency": "MXN",
      "method": "bank_account_clabe",
      "operation_type": "out",
      "bank_operation_type": "spei",
      "transaction_type": "payout",
      "status": "in_progress",
      "creation_date": "2019-08-25T17:07:28.197Z",
      "delivery_date": "2019-08-25T17:07:28.197Z",
      "delivered": true,
      "_id": "5d62c05022cabc0011c38c64",
      "amount": 15,
      "description": "Pago a tercero",
      "__v": 0
    },
    {
      "bank_account": {
        "creation_date": "2019-08-25T17:48:03.769Z",
        "clabe": "044180001021643787",
        "holder_name": "Sebastian Acosta Checa",
        "bank_code": "044",
        "bank_name": "SCOTIA BANK INVERLAT"
      },
      "source_account": {
        "account_type": "40",
        "bank_code": "014",
        "account_number": "014180220006507057"
      },
      "currency": "MXN",
      "method": "bank_account_clabe",
      "operation_type": "out",
      "bank_operation_type": "spei",
      "transaction_type": "payout",
      "status": "in_progress",
      "creation_date": "2019-08-25T17:48:03.769Z",
      "delivery_date": "2019-08-25T17:48:03.769Z",
      "delivered": true,
      "_id": "5d62c9d3b856470011c76ee9",
      "amount": 16,
      "description": "Pago a tercero",
      "__v": 0
    },
    {
      "bank_account": {
        "creation_date": "2019-08-25T18:01:32.549Z",
        "clabe": "044180001021643787",
        "holder_name": "Sebastian Acosta Checa",
        "bank_code": "044",
        "bank_name": "SCOTIA BANK INVERLAT"
      },
      "source_account": {
        "account_type": "40",
        "bank_code": "014",
        "account_number": "014180220006507057"
      },
      "currency": "MXN",
```

```
        "method": "bank_account_clabe",
        "operation_type": "out",
        "bank_operation_type": "spei",
        "transaction_type": "payout",
        "status": "in_progress",
        "creation_date": "2019-08-25T18:01:32.549Z",
        "delivery_date": "2019-08-25T18:01:32.549Z",
        "delivered": true,
        "_id": "5d62ccfcb856470011c76eea",
        "amount": 18,
        "description": "Pago a tercero",
        "__v": 0
    },
]
```


Diseño de interfaz web (GUI) basado en patrón de diseño SPA

En la búsqueda de lograr los objetivos de fomentar el acceso a la información de la tesorería no solamente para los usuarios Robots, sino también los usuarios humanos de VULCANO, se decidió crear una Interfaz Gráfica de Usuario (**GUI**, por sus siglas en inglés) para complementar la Interfaz de Programación de Aplicaciones (**API**, por sus siglas en inglés). El reto principal de lograr entregar los servicios de VULCANO mediante el canal GUI y el canal API, consiste en reutilizar microservicios tanto para componer la API, como la GUI. El equipo de ISBIT SA DE CV consideró, después del debido análisis, que para resolver el reto planteado sería indispensable separar por completo el código cliente del código cliente, mediante el enfoque SPA (Single Page Application).

Cuando un usuario de VULCANO interactúa con el GUI aplicación primero, su navegador descarga código cliente (aplicación de Angular) desde un endpoint público, que sirve los archivos estáticos que le constituyen. Este endpoint es servido por el microservicio ***vulcano-external-front-ng*** mediante la aplicación ANGULAR, con un servidor NGINX. Una vez descargada la aplicación ANGULAR, el navegador web (por ejemplo Chrome o Mozilla) del usuario se conecta a otro endpoint distinto (EL BACKEND) que es un API JSON REST servido por el microservicio **vulcano-external-front.** La aplicación cargada en cliente (conjunto de CSS, HTML y JAVASCRIPT en el Navegador de la máquina local del usuario) intercambia mensajes JSON con el backend sin nunca “refrescar” la página, lo cual mejora la experiencia del usuario al aumentar el “tiempo de respuesta” o “responsividad” a sus interacciones.

Usamos la extensión Angular-Material[1] para darle un aspecto moderno y ergonómico a la interfaz gráfica de VULCANO, generando un resultado visual como el que se puede apreciar con siguientes pantallazos:

[1] "Angular Material." https://material.angular.io/. Fecha de acceso 5 may.. 2020.

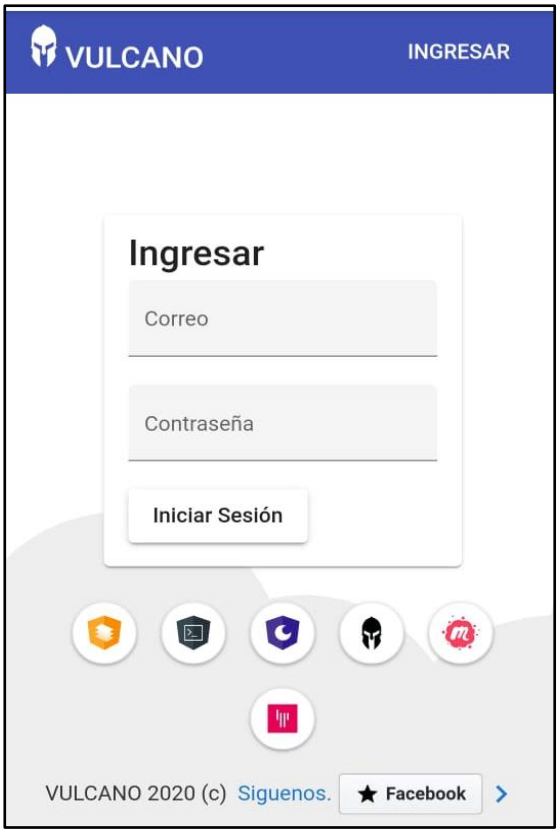


Ilustración 37 Resultado visual de la página de Vulcano: contenido

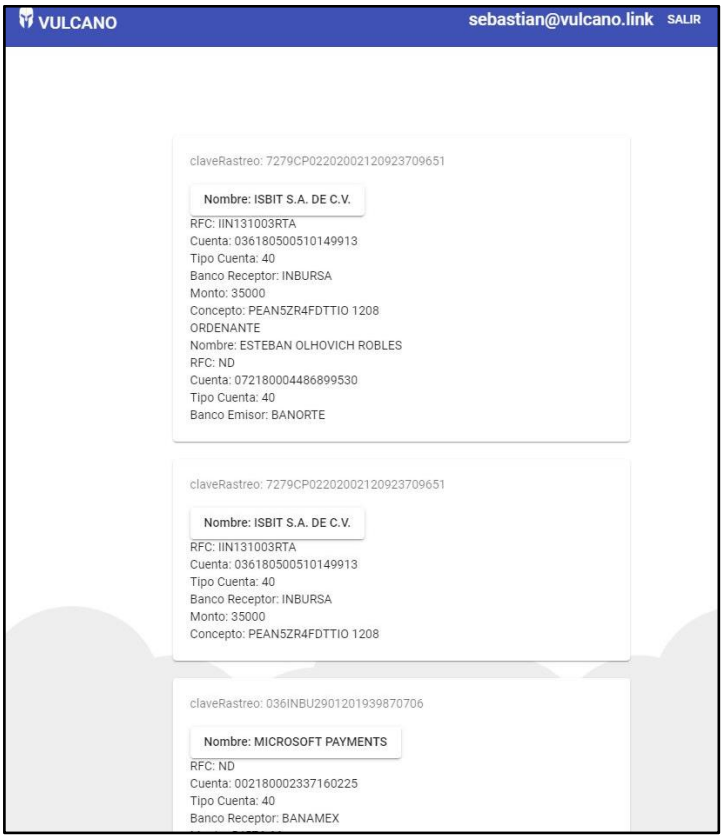


Ilustración 38 Interfaz Grafica de Vulcano – explorador de CEPs

Mitigación de Riesgos de Ciberseguridad

Autorización

Al desarrollar el software, se estableció un sistema de acceso para cada usuario basado en roles, restringiendo los permisos de cada uno para acceder a la información, por medio del paradigma “Role based access control” (RBAC). Los detalles sobre su funcionamiento se encuentran en el anexo B.

Autenticación

Es el mecanismo para poder acceder a la aplicación. Se ocupó el JSON Web Token (JWT), este tipo de token es generado del lado del servidor.

Está compuesto por tres secciones; el Header, Payload y Signature. En el siguiente esquema, se muestra la implementación de cada Componente del JWT.

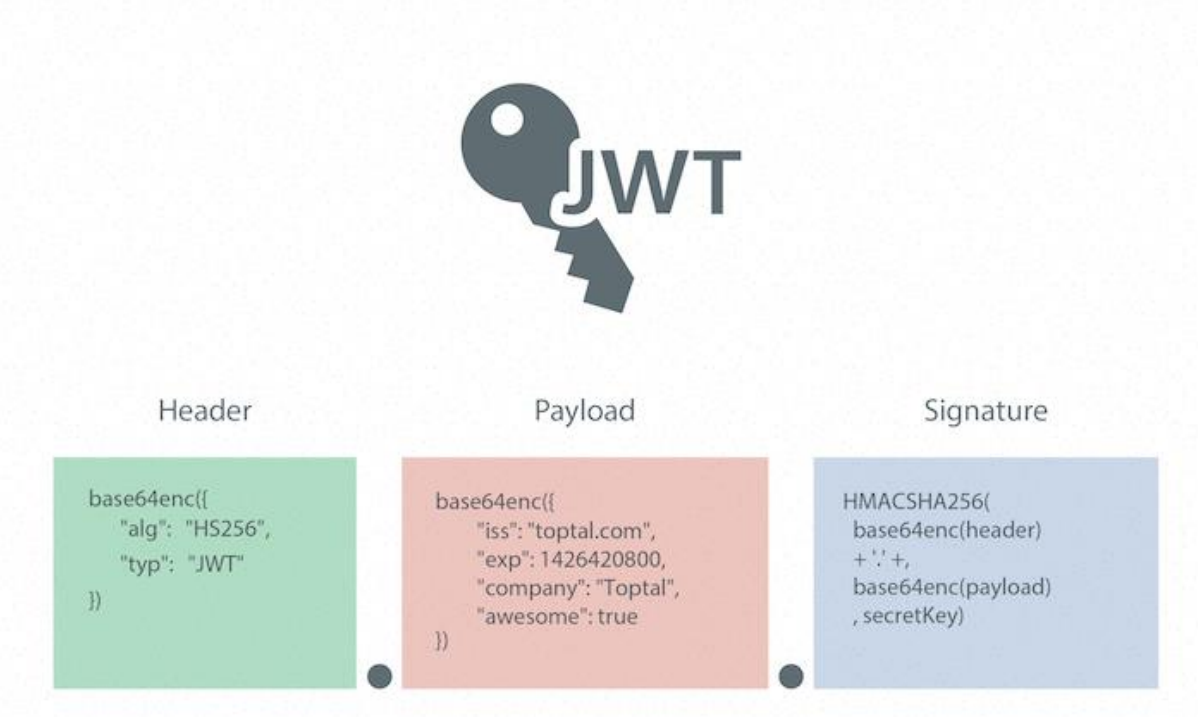


Ilustración 39 JWT; header, payloado y signature

1. Header (encabezado): es un JSON que contiene el tipo de token y el algoritmo de firma que se utiliza, como HMAC SHA256 o RSA.

2. Payload (carga útil): es un JSON que contiene información referente a la persona que se está autenticando. Hay tres tipos de reclamaciones:

Registradas: se trata de un conjunto de reclamaciones predefinidas que no son obligatorias, pero sí recomendadas, para proporcionar un conjunto de reclamaciones útiles e interoperables. Algunos de ellos son: iss(emisor), exp (tiempo de vencimiento), sub(asunto), aud (audiencia) y otros.

Públicas: estos pueden ser definidos a voluntad por aquellos que usan JWT. Pero para evitar colisiones, deben definirse en el Registro de tokens web JSON de IANA o definirse como un URI que contiene un espacio de nombres resistente a colisiones.

Privadas: estos son reclamos personalizados creados para compartir información entre las partes que acuerdan usarlos y no son reclamos *registrados* ni *públicos*.

3. Verify signature (firma): en esta parte se debe tomar el encabezado y la carga útil codificados, una llave privada, el algoritmo especificado en el encabezado y firmarlo. La firma se utiliza para verificar que el mensaje no se haya modificado en el camino y, en el caso de los tokens firmados con una clave privada, también puede verificar que el remitente del JWT es quien dice ser.

Resultado

Al final tenemos tres cadenas de URL Base64 separadas por puntos que se pueden pasar fácilmente en entornos HTML y HTTP, mientras que son más compactas en comparación con estándares basados en XML como SAML.

Una vez que tenemos el encabezado, la carga útil y está firmada por una clave privada, obtenemos un código de este estilo, en donde cada color representa una sección.

eyKJdfNGkslElKibHfOpdmspD7C938H.kd9HD9Alcckd8CNKICjv8sInco7HDK.6pFjdnfrAN4cKJSco9mJhGr7aNen

Después de que el usuario logra autenticarse, así es como funciona JWT

En la autenticación, cuando el usuario inicia sesión con éxito con sus credenciales, se devolverá un JSON Web Token. Dado que los tokens son credenciales, se debe tener mucho cuidado para evitar problemas de seguridad. Por lo regular no se conservan los tokens más del tiempo necesario. Tampoco debe almacenar datos confidenciales de la sesión en el

almacenamiento del navegador debido a la falta de seguridad.

Siempre que el usuario desee acceder a una ruta o recurso protegido, el agente de usuario debe enviar el JWT, normalmente en el encabezado de **autorización** utilizando el esquema de **portador**. El contenido del encabezado debe tener el siguiente aspecto: Authorization: Bearer <token>

Las rutas protegidas del servidor buscarán un JWT válido en la autorización y, si está presente, el usuario podrá acceder a los recursos protegidos.

Así es como funciona paso a paso:

1. La aplicación o el cliente solicita autorización al servidor de autorización. Esto se realiza a través de uno de los diferentes flujos de autorización.

2. Si toda la información es correcta, el servidor de autorización devuelve un token de acceso a la aplicación.

3. La aplicación utiliza el token de acceso para acceder a un recurso protegido (como una API).

Con los tokens firmados, toda la información contenida en el token está expuesta a los usuarios u otras partes, aunque no puedan cambiarla. Esto significa que no debe poner información secreta dentro del token.

Lista blanca de IP's

Para reducir ataques de Denegación de Servicios y otros riesgos asociados a hacer disponible la API de VULCANO desde internet es necesario implementar uso del controlador **nginx-ingress** para restringir el acceso por IP (lista blanca de ip) para un servicio implementado en un clúster de Kubernetes (AKS).

Veamos cómo para un servicio implementado en el clúster AKS, se restringe el acceso a solamente ciertas IP de origen del cliente (Ip Whitelisting).

El resto de esta sección asume que el clúster de AKS Kubernetes está disponible, se tiene la herramienta **HELM**[2] instalada, ya que es necesario ejecutar el comando *helm init* con éxito antes de proceder a configurar lista Blanca de IP, así como ejecutar otros comandos de *helm* durante el proceso de configuración.

[2] "Helm Upgrade - Helm." https://helm.sh/docs/helm/helm_upgrade/. Fecha de acceso 6 may.. 2020

Lo primero que hacemos, es instalar el controlador nginx-ingress usando helm. La página de github para el HELM Chart del controlador nginx-ingress está disponible en GITHUB con el siguiente link o escaneándolo mediante el QR.

https://github.com/kubernetes/charts/tree/master/stable/nginx-ingress.

El valor por defecto del parámetro de configuración **controller.service.externalTrafficPolicy** en el Helm Chart de nginx es "Cluster" y necesitamos cambiar este valor a "Local". Con el valor por defecto "Cluster" el controlador de *ingresos,* no observa la ip original de la petición del cliente, sino una IP interna. Después de cambiar el parámetro **controller.service.externalTrafficPolicy** a "Local" , el controlador ingress obtiene las IP originales sin modificación de los paquetes de información y peticiones de clientes. El cambio se logra mediante el siguiente comando:

```
helm upgrade nginx stable/nginx-ingress --set controller.service.externalTrafficPolicy=Local --namespace ing-front
```

Para información adicional sobre las banderas y opciones usadas en el Comando anterior, consultar la sección correspondiente en la Documentación Oficial de Helm.

El comando anterior asume la existencia preexistente del Helm Chart nginx en el espacio de nombres "ing-front" como en un cluster de pruebas que contiene un Charts de nombre *nginx* como en el siguiente cuadro:

```
sebastians-air:~ sebastian$ helm ls –all-namespaces
NAME            NAMESPACE       REVISION  UPDATED                                  STATUS    CHART                 APP VERSION
cert-manager    ingress-basic1  1         2020-04-10 19:24:29.247936 -0500 CDT     deployed  cert-manager-v0.13.0  v0.13.0
nginx           ing-front       1         2020-02-18 01:28:20.26454181 +0000 UTC   deployed  nginx-ingress-1.30.3  0.28.0
vulcano         santander       1         2020-04-15 04:29:40.512039 +0000 UTC     deployed  vulcano-v0.2.5
```

Comando para obtener lista de *servicios* en el cluster de kubernetes y su salida

```
kubectl get svc -A
NAMESPACE     NAME                                 TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)                      AGE
default       kubernetes                           ClusterIP      10.0.0.1       <none>         443/TCP                      393d
ing-front     nginx-nginx-ingress-controller       LoadBalancer   10.0.153.236   52.179.4.245   80:30908/TCP,443:32020/TCP   79d
ing-front     nginx-nginx-ingress-default-backend  ClusterIP      10.0.210.167   <none>         80/TCP                       79d
ingress-basic cert-manager                         ClusterIP      10.0.217.223   <none>         9402/TCP                     26d
ingress-basic cert-manager-webhook                 ClusterIP      10.0.125.249   <none>         443/TCP                      26d
kube-system   dashboard-metrics-scraper            ClusterIP      10.0.217.46    <none>         8000/TCP                     90d
kube-system   healthmodel-replicaset-service       ClusterIP      10.0.133.212   <none>         25227/TCP                    240d
kube-system   kube-dns                             ClusterIP      10.0.0.10      <none>         53/UDP,53/TCP                393d
kube-system   kubernetes-dashboard                 ClusterIP      10.0.100.72    <none>         443/TCP                      393d
kube-system   metrics-server                       ClusterIP      10.0.227.171   <none>         443/TCP                      393d
kube-system   tiller-deploy                        ClusterIP      10.0.241.125   <none>         44134/TCP                    81d
santander     external-front-ng-service            ClusterIP      10.0.176.167   <none>         80/TCP                       22d
santander     external-front-service               ClusterIP      10.0.173.135   <none>         3000/TCP                     22d
santander     spei-checker-service                 ClusterIP      10.0.32.201    <none>         33438/TCP                    22d
santander     vulcano                              ClusterIP      10.0.154.209   <none>         8080/TCP                     22d
santander     vulcano-layout-gen-pro-service       ClusterIP      10.0.32.205    <none>         33439/TCP                    22d
santander     vulcano-santander-crypto-service     ClusterIP      10.0.32.225    <none>         33440/TCP                    22d
santander     vulcano-santander-sftp-service       ClusterIP      10.0.32.215    <none>         33436/TCP                    22d
```

Para aplicar el mecanismo de Lista Blanca de IP 's, usamos “anotaciones” en los archivos de configuración YAML de un recurso *Ingress* en Kubernetes.

En concreto necesitamos aplicar la anotación (**nginx.ingress.kubernetes.io/whitelist-source-range)** tal como es detallado en la Guía adjunta al paquete helm de nginx-ingress, la cual podemos encontrarla en el siguiente link o escaneándolo mediante el QR.
https://github.com/kubernetes/ingress-nginx/blob/87d1b8bbf28265386b339e65d8943d8c3f8582ed/docs/user-guide/nginx-configuration/annotations.md#whitelist-source-range.

El siguiente archivo de documentación del entorno de Pruebas de VULCANO muestra cómo utilizar la directiva (texto subrayado en color amarillo) para limitar las redes con posibilidad de hacer peticiones a Balanceador de Carga:

nginx-cert-mgr-tls-ingress.yml archivo en formato YAML con configuración

ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 annotations:
   cert-manager.io/cluster-issuer: letsencrypt-prod
   kubernetes.io/ingress.class: nginx
   nginx.ingress.kubernetes.io/rewrite-target: /$1
   nginx.ingress.kubernetes.io/ssl-redirect: "false"
   nginx.ingress.kubernetes.io/proxy-connect-timeout: '600'
   nginx.ingress.kubernetes.io/proxy-send-timeout: '600'
   nginx.ingress.kubernetes.io/proxy-read-timeout: '600'
   nginx.ingress.kubernetes.io/whitelist-source-range: 49.36.X.X/32
 generation: 1
 name: vulcano
 namespace: santander
 selfLink: /apis/extensions/v1beta1/namespaces/santander/ingresses/vulcano
spec:
 rules:
  - host: vulcano.link
    http:
      paths:
        - backend:
            serviceName: external-front-ng-service
            servicePort: 80
          path: /(.*)
        - backend:
            serviceName: external-front-service
            servicePort: 3000
          path: /dashboard_api/(.*)
  - host: api.vulcano.link
    http:
      paths:
        - backend:
            serviceName: vulcano
            servicePort: 8080
          path: /(.*)
        - backend:
            serviceName: spei-checker-service
            servicePort: 33438
          path: /cep/(.*)
 tls:
   - hosts:
       - vulcano.link
       - api.vulcano.link
     secretName: tls-secret-vulcano-link
status:
 loadBalancer:
   ingress:
     - ip: 52.179.4.245
```

Con la configuración anterior las peticiones realizadas desde redes que no estén en la lista autorizada resultan en una respuesta HTTP/1.1 **403 Forbidden** mientras que las que provenientes de ip autorizadas resultan en HTTP/1.1 **200 OK .**

El documento anterior también muestra cómo el ingress tiene reglas para canalizar el tráfico a microservicios diferentes dependiendo de la ruta (del URL en una partición). La configuración subrayada en morado muestra como peticiones relacionadas con el API con canalizadas al microservicios *vulcano* y *spei-checker-service* , mientras que configuracion subrayada en verde muestra como peticiones relacionadas con GUI son canalizadas a microservicio *external-front-service* y *external-front-ng-service* .

Pruebas unitarias y de integración en entornos de calidad antes de despliegue en producción

La existencia de pruebas unitarias y pruebas de integración es esencial ya que esta herramienta nos permite saber si algunas funciones o componentes del sistema dejan de funcionar cada vez que realizamos un cambio. Las pruebas automatizadas también pueden ser integradas al proceso de CI para solamente permitir el despliegue en producción cuando tenemos confianza de que todo funciona sin necesidad de manualmente corroborarlo cada vez ya que sería una tarea monótona y muy exhaustiva en términos de "horas hombre".

Encriptado de api endpoint externo con TLS/SSL

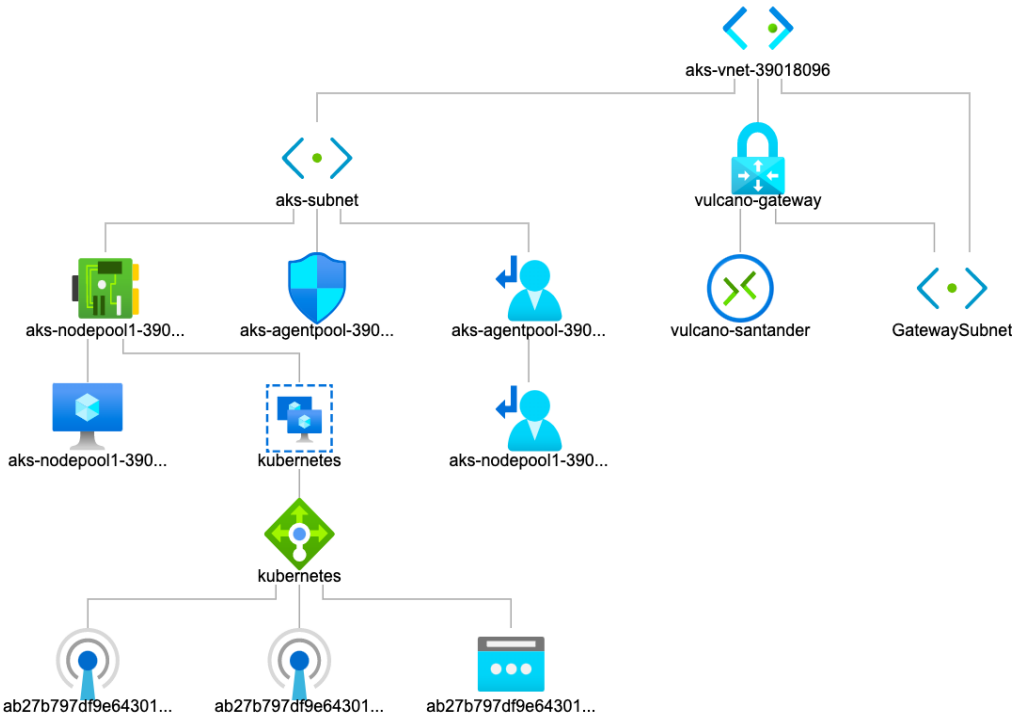


Ilustración 40 Topología de la Red del cluster de VULCANO

Del diagrama de la topología de la Red del cluster de VULCANO el elemento más importante a destacar es que las comunicación con Bancos no es mediante túneles y la comunicación con clientes de VULCANO (humanos o robots) es mediante internet en IPs, interfaces y redes diferentes.

La comunicación con el Banco fluye a través de un "Virtual Network Gateway" que une dos redes diferentes (vulcano-gateway representada con candado azul arriba) , la red del Banco

con la red del cluster Kubernetes. Por otro lado la comunicación con clientes de VULCANO fluye mediante el balanceador y las IPs públicas conectadas a este (representado por rombo verde en diagrama de arriba).

Autenticación con Servidores de Santander mediante certificados PKI

PKI es la infraestructura de clave pública, se usa para la comunicación segura entre la firma y el certificado digitales. La firma digital nos ayuda a garantizar la autoría e integridad de los documentos digitales y la posibilidad de demostrar estas propiedades ante terceros permitiendo que estos gocen de características que únicamente eran propias de los documentos en formato duro certificados digitales, ante esto, no nace una pregunta ¿Cómo se hace o cómo se puede asegurar que una clave pública pertenece a un usuario dado?

Es necesario poder vincular la clave pública de un usuario con su identidad y para esto surge el concepto de certificado digital, el cual contiene la identidad de usuario, clave pública del usuario, periodo de validez del certificado, identidad de la autoridad certificadora y la firma digital de certificado. De esta manera, se alcanzan los cuatro objetivos de la seguridad informática que son; autenticidad (la firma digital tendrá la misma validez que la manuscrita), confidencialidad(de la información transmitida entre las partes), integridad (capacidad de detectar si un documento firmado ha sido manipulado) y no repudio de un documento firmado digitalmente.

Entre la comunicación segura entre cliente y servidor aparecen nuevos interlocutores; autoridad de certificación cuya misión es emitir certificados digitales a la autoridad de registro, que es la responsable de asegurar al solicitante que el del certificado es quien dice ser.

Autoridad de validación: responsable de comprobar la validez de los certificados digitales emitidos

Repositorios: almacenes de certificados emitidos y aquellos que han sido revocados y han dejado de ser válidos.

Funcionamiento:
1. Un nuevo usuario se registra para obtener un certificado
2. El sistema de registro inicia la captura de información de registro y activa la generación de claves
3. Devuelve la clave pública y la información de registro a la autoridad de registro
4. Se hace una petición de certificado a la autoridad de certificación
5. La autoridad de certificación firma la petición válida
6. Envía el certificado a la autoridad de certificación
7. La autoridad de registro entrega entonces el certificado firmado al usuario

8. El certificado ha sido ya emitido y finalmente es publicado en un directorio

Encriptado de archivos (layouts) transferidos desde VULCANO a Santander

El microservicio ***vulcano-santander-crypto*** es el encargado de encriptar los archivos que posteriormente serán enviados al banco por el microservicio ***vulcano-santander-sftp***. Veamos cómo funciona el proceso de *encriptado* de archivos de manera general. Para más información del algoritmo completo ver Anexo D.

CmpApiCifrado.sh

```
#!/bin/bash
# Cambiar $API_SECURITY_PATH por la ruta de java a utilizar para la ejecucion de la aplicacion
#      Ejemplo:/usr/java/bin
# Cambiar $PATH_CERT_DES Ruta de certificado del banco
#   Ejemplo /home/xxx/apicifrado/certificadoBanco.cert
# Cambiar $PATH_CERT_REM Ruta de certificado del banco
#   Ejemplo /home/xxx/apicifrado/llavePrivada.cert
# Cambiar $PATH_PKEY_REM por ruta de llave privada
#   Ejemplo /home/xxx/apicifrado/llavePrivada.p12
# Cambiar $REM_PWD por contrase�a de llave privada cifrada en Base64
# Cambiar $JAR_PATH por la ruta de instalacion del API
# cambiar $OPC_TRANSF por el valor que se requiere para transferir el archivo (S o N)

# Usage: sh CpmApiCifrado.sh source_file_path destination_file_path base_64_encoded_password

java -cp /src/apicifrado/bcpkix-jdk15on-148.jar:/src/apicifrado/bcprov-jdk15on-148.jar:/src/apicifrado/CmpApiCifrado.jar:/src/apicifrado/GestorCertificadosClient.jar:/src/apicifrado/jsch-0.1.49.jar:/src/apicifrado/log4j-1.2.8.jar:/src/apicifrado/sha3.jar:/src/commons-codec-1.9.jar:/src/apicifrado/shiro-1.2.1-isban.jar mx.isban.ArchivosCifrados.tool.SecureFile -e /src/apicifrado/certs/HOSTTOHOSTPRODHSM.cer /src/apicifrado/certs/H2H45617856089000003056.cer /src/apicifrado/certs/SANTANDER_CERT.p12 $3 $1 $2 N
```

El microservicio ***vulcano-santander-crypto*** también es el encargado de *des-encriptar* archivos recibidos por el microservicio ***vulcano-santander-sftp*** provenientes del banco tales como notificaciones, respuestas, estados de cuenta y reportes de cobranza, entre otros. Veamos cómo funciona el proceso de *desencriptado* de archivos de manera general. Para más información del algoritmo completo ver **Anexo D**.

Microservicio vulcano-santander-crypto

obtener secretos de AKV

-mongoUri
-accountKey
-cryptoPass
-cryptoPass
-encodedPrivateKey

app.js

servicios ofrecidos (endpoints)

encrypt-layouts
decrypt-notifications
decrypt-responses
decrypt-statements
decrypt-reports

app.js

Obtener secretos de AKV	Servicios ofrecidos (endpints)
mongoUri	encrypt-layouts
accountKey	decrypt-notifications
cryptoPass	decrypt-responses
cryptoPass	decrypt-statements
encodedPrivateKey	decrypt-reports

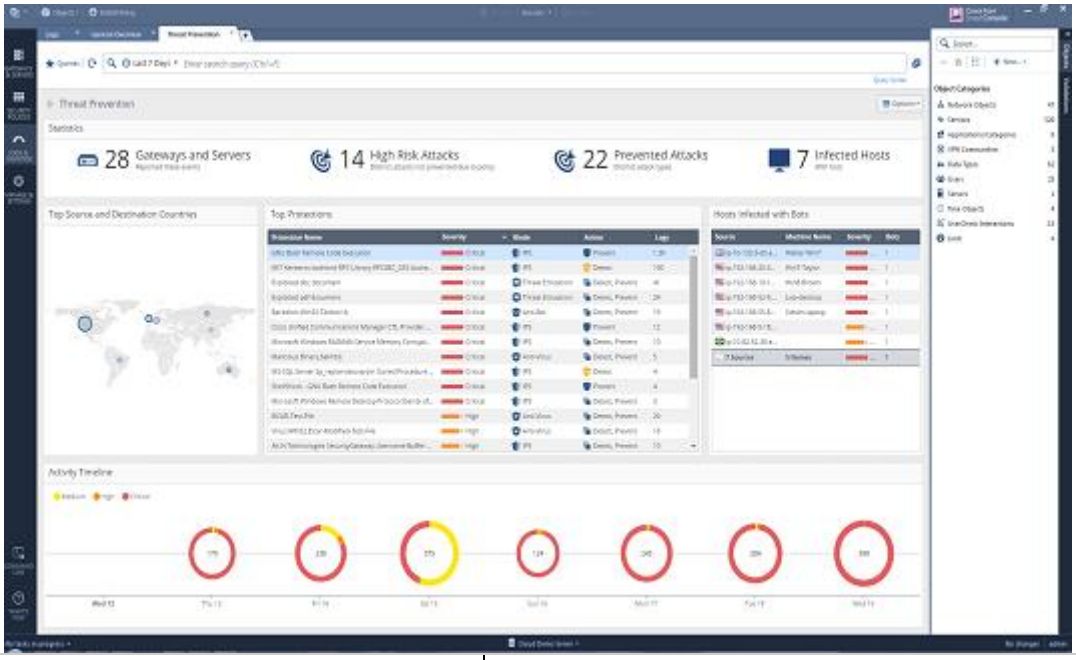
CmpApiCifradoDecode.sh [$1 documento_origen] [$2 documento_destino] [$3 clave]

```
#!/bin/bash
# Cambiar $API_SECURITY_PATH por la ruta de java a utilizar para la ejecucion de la aplicacion
#      Ejemplo:/usr/java/bin
# Cambiar $PATH_CERT_DES Ruta de certificado del banco
#          Ejemplo /home/xxx/apicifrado/certificadoBanco.cert
# Cambiar $PATH_PKEY_REM por ruta de llave privada
#          Ejemplo /home/xxx/apicifrado/llavePrivada.p12
# Cambiar $REM_PWD por contrasea de llave privada cifrada en Base64
# Cambiar $JAR_PATH por la ruta de instalacion del API

# Usage: sh CpmApiCifradoDecode.sh source_file_path destination_file_path base_64_encoded_password

java -cp /src/apicifrado/bcpkix-jdk15on-148.jar:/src/apicifrado/bcprov-jdk15on-148.jar:/src/apicifrado/CmpApiCifrado.jar:/src/apicifrado/GestorCertificadosClient.jar:/src/apicifrado/jsch-0.1.49.jar:/src/apicifrado/log4j-1.2.8.jar:/src/apicifrado/sha3.jar:/src/apicifrado/commons-codec-1.9.jar:/src/apicifrado/shiro-1.2.1-isban.jar mx.isban.ArchivosCifrados.tool.SecureFile -d /src/apicifrado/certs/HOSTTOHOSTPRODHSM.cer /src/apicifrado/certs/SANTANDER_CERT.p12 $3 $1 $2
```

Autenticación con Servidores de Santander mediante VPN

La conexión de los nodos del cluster de VULCANO con los servidores SFTP dentro de la red de Banco Santander ocurre mediante una conexión VPN tipo “Site to Site” con los siguientes parámetros:

Parámetro de Configuración de Cortafuegos para Red Privada Virtual entre VULCANO y SANTANDER		
VPN Settings	**VPN-Santander:**	**VPN-Partner:**
Network Settings		
Peer IP	**170.169.98.20**	
VPN Product	**Site to Site**	**Site to Site**
IKE V2 Policy		
Message Encryption algorithm	**AES-256**	**AES-256**
Message integrity algorithm	**SHA-256**	**SHA-256**
Peer Authentication Method		**Presharedkey**
DH-Group	**Group 19 (256-bit ECP)**	**Group 19 (256-bit ECP)**
IKE Lifetime	**86400s**	**86400s**
Supports Aggressive Mode	**No**	**No**
IPSec Parameters		
Mechanism for payload encryption	**ESP**	**ESP**
ESP Transform	**AES-256**	**AES-256**
Data Integrity	**SHA-256**	**SHA-256**
Security Association (SA) Lifetime	**3600s**	**3600s**
VPN Parameters		
Network Address for encrypt domain	**192.240.110.98**	
Source IP / Destination IP	**192.240.110.98**	
Service	**Icmp, ssh**	**Icmp, ssh**
VPN Test Machine Parameters		
IP Address		
Services		



NOTAS: Las llaves pre compartidas son acordadas manualmente por teléfono	Detalles de algoritmo AES utilizado para encripcion: https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf

Es importante notar que un cortafuegos de grado industrial que soporta todos los parámetros anteriores es el *CHECKPOINT™*, además de ser la marca de "appliance" usada en el perímetro de seguridad de Santander. Por la razón anterior, en el ambiente de desarrollo de VULCANO para simular de manera realista la infraestructura de SANTANDER en las pruebas de integración usamos la versión VIRTUALIZADA de Checkpoint disponible en el mercado de soluciones de Azure, el cual tiene un panel de control que permite monitorear todos los aspectos del trafico en la red como se aprecia en la siguiente captura de pantalla de su panel de control:

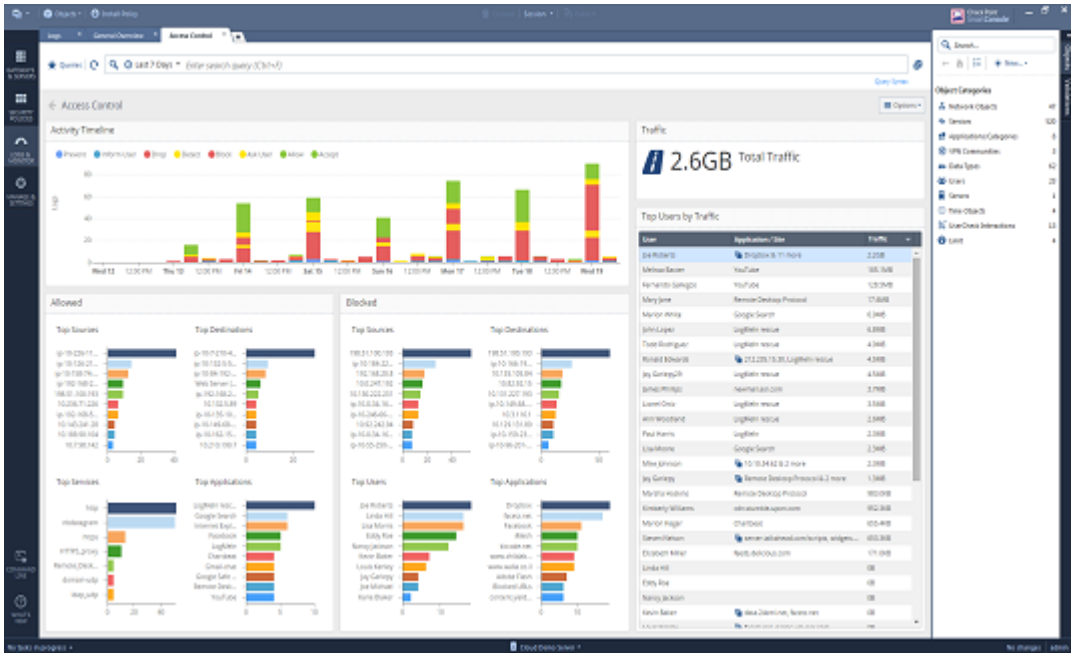


Ilustración 41 CHECKPOINT Dispositivo Cortafuegos de Hardware

Es importante destacar que el canal de comunicación con Santander mediante la VPN está encriptado con la longitud máxima soportada por el algoritmo “Estándar de Encriptación Avanzada” (AES, por sus siglas en inglés).

La mayoría de los cálculos del algoritmo AES se hacen en un campo finito determinado. AES opera en una matriz de 4×4 bytes, llamada *state*.

La guía de integración de Vulcano se encuentra en el anexo E.

Algoritmo AES

El Pseudocódigo del Algoritmo AES de acuerdo con la especificación publicada por NIST el 26 de noviembre de 2001 es el siguiente:

- Expansión de la clave usando el esquema de claves de Rijndael (Rijmen, V. & Daemen J., 2003).
- Etapa inicial:

1. **AddRoundKey**

- Rondas:

1. SubBytes — en este paso se realiza una sustitución no lineal donde cada byte es reemplazado con otro de acuerdo con una tabla de búsqueda. Ver definición de tabla de Búsqueda (o "Hashtable") en el Capítulo 3 (páginas 253 - 280) del Libro "Introducción a los Algoritmos" (Thomas H. Cormen, charles E., MIT PRESS, USA, 1990). En la fase de **SubBytes**, cada byte en el *state* es reemplazado con su entrada en una tabla de búsqueda fija de 8 bits, S; $b_{ij} = S(a_{ij})$.

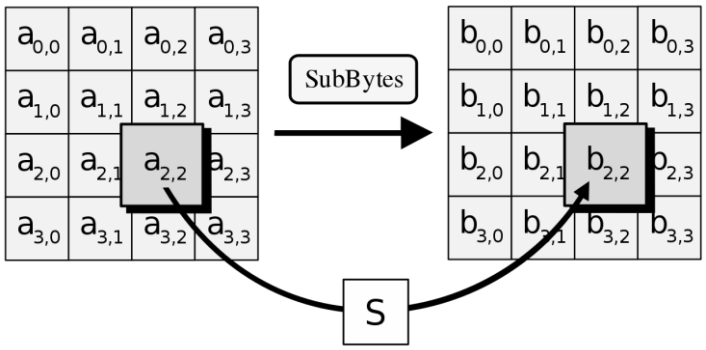


Ilustración 42 SubBytes

2. ShiftRows — en este paso se realiza una transposición donde cada fila del *«state»* es rotada de manera cíclica un número determinado de veces. En el paso **ShiftRows**, los bytes en cada fila del state son rotados de manera cíclica hacia la izquierda. El número de lugares que cada byte es rotado difiere para cada fila.

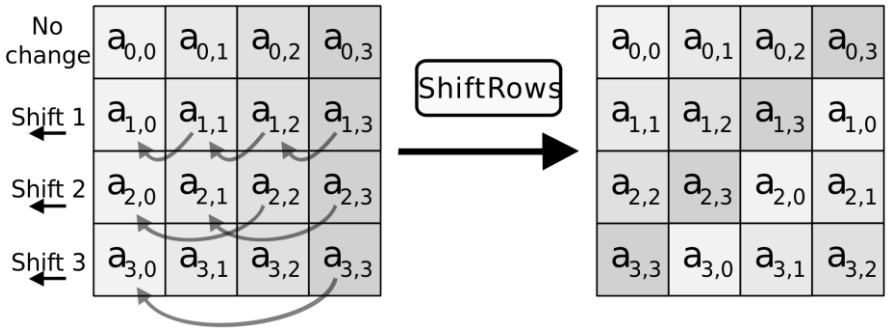


Ilustración 43 ShiftRows

3. MixColumns — operación de mezclado que opera en las columnas del *«state»*, combinando los cuatro bytes en cada columna usando una transformación lineal. En el paso **MixColumns**, cada columna del state es multiplicada por un polinomio constante *c(x)*.

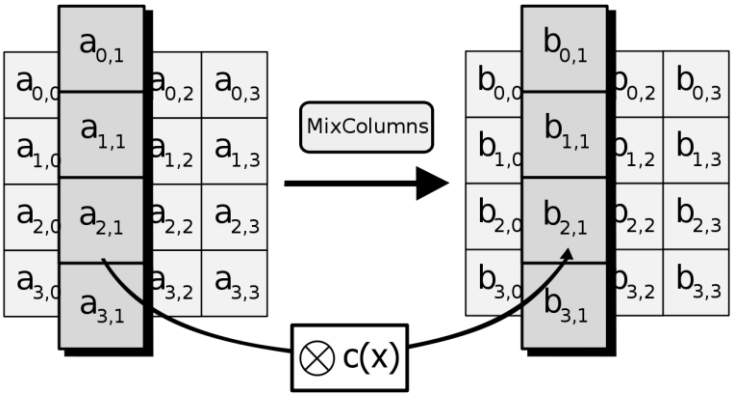


Ilustración 44 MixColumns

4. AddRoundKey — cada byte del «state» es combinado con la clave «round»; cada clave «round» se deriva de la clave de cifrado usando una *iteración de la clave*. En el paso **AddRoundKey**, cada byte del state se combina con un byte de la subclave usando la operación XOR (⊕).

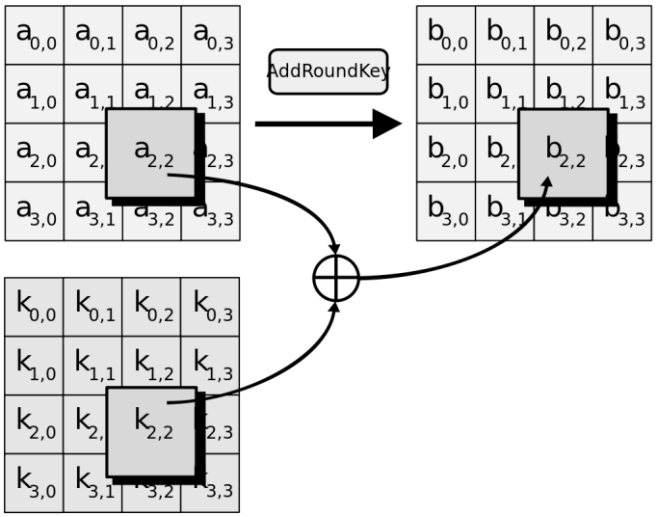


Ilustración 45 AddRoundKey

- Etapa final:

1. **SubBytes**
2. **ShiftRows**
3. **AddRoundKey**

La interfaz de la programación de VULCANO se encuentra en el anexo F.

5. Resultados obtenidos

Con base en los objetivos ya mencionados en su respectiva sección, VULCANO fue creado para aumentar la seguridad de la tesorería de la empresa ISBIT. A continuación, expondré una de sus aplicaciones más importantes para la empresa.

Caerus es un software que nos permite calcular los arbitrajes financieros de activos virtuales, es el cerebro que nos ayuda a pensar en qué momento comprar o vender estos activos en cualquier parte del mundo, dependiendo del precio que exponga cada mercado. VULCANO es una herramienta que Caerus ocupa para automatizar las operaciones permitiendo la rentabilidad de las transacciones.

En el siguiente diagrama se muestra el proceso de arbitraje; supongamos que en una zona/país A tenemos una cuenta de banco con dinero FIAT pero nos damos cuenta de que en otra zona/país B el precio de alguna criptomoneda es mayor que en la zona A, entonces queremos transferir el dinero a dicha zona para obtener una ganancia, para esto el dinero que tenemos en la zona A lo convertimos en crypto y posteriormente lo transferimos en el mismo formato a la zona B. Una vez estando en la zona B, convertimos el activo a dinero FIAT, para que después, por medio de VULCANO H2H se transfiera a nuestro banco y así obtener el dinero que teníamos desde una zona lejana A. La interferencia de vulcano nos ayuda a que el proceso sea rápido, seguro y eficiente.

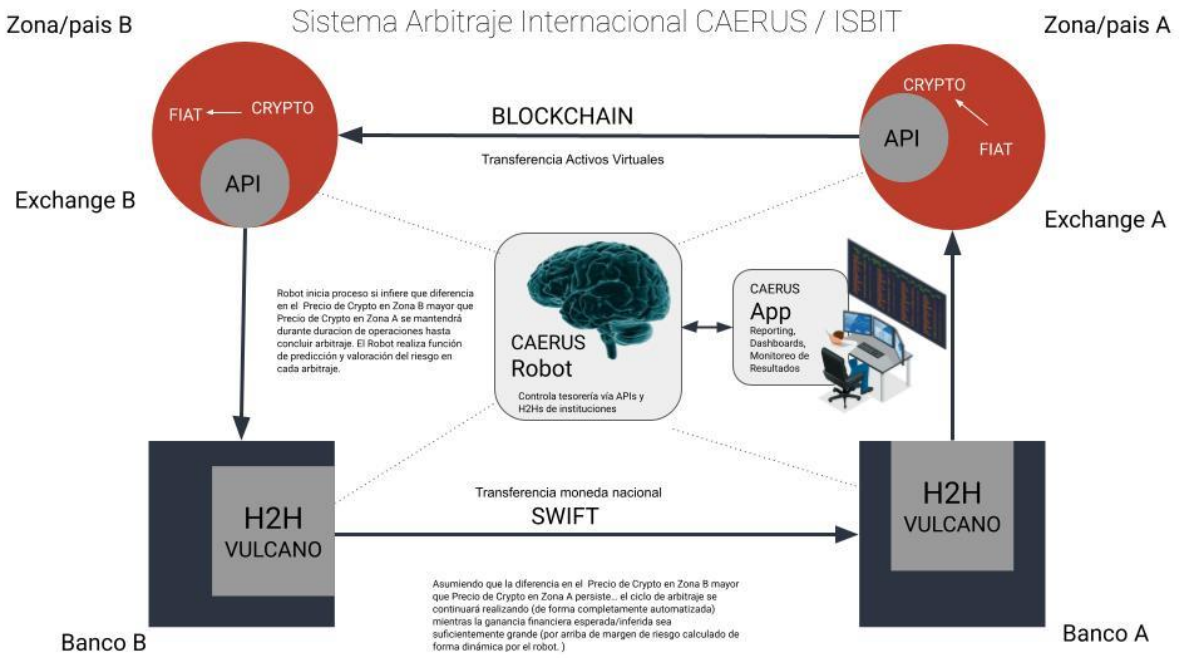


Ilustración 46 Sistema de arbitraje internacional CAERUS/ISBIT

Ahora veamos los resultados obtenidos para esta aplicación:

● La seguridad en las transacciones de una zona A a otra zona B, aumentó considerablemente, ya que al transferir el dinero no existe un rango de riesgo desde su punto de partida hasta su destino.

● La velocidad en cada transacción es casi instantánea, la operación tarda menos de un minuto.

● VULCANO facilitó la tarea del cumplimiento de la empresa debido a que se tiene fácil acceso a la información de las transacciones realizadas para generar automáticamente los reportes y así poder declararlos al Servicio de Administración Tributaria en el tiempo establecido.

● Se redujo el tiempo que nos toma un ciclo de arbitraje, automatizando las transferencias y reduciendo la necesidad de un operador humano en un 48.61%. Esto implica que podemos realizar 2.057, es decir, aproximadamente dos veces más ciclos exactamente en el mismo tiempo, lo que nos indica que tenemos el doble de ganancias del fondo administrado.

● Se disminuyó el riesgo de errores operativos; como robos de dinero, error en precios, abuso de confianza, captura de datos, en general cualquier error generado por una persona.

● VULCANO nos ayudó a reducir de gran manera las comisiones impuestas por los bancos.

● La implementación del software fue diseñada con el fin de facilitar su instalación y mantenimiento. Además, el sistema tiene poca dependencia entre sus componentes, por lo tanto, nos proporciona la escalabilidad del software, es decir, cuenta con la capacidad de adaptación y respuesta con respecto al rendimiento del mismo.

A continuación, mostraré las gráficas estadísticas de las operaciones más importantes, mes con mes a partir del año 2016 a la fecha.

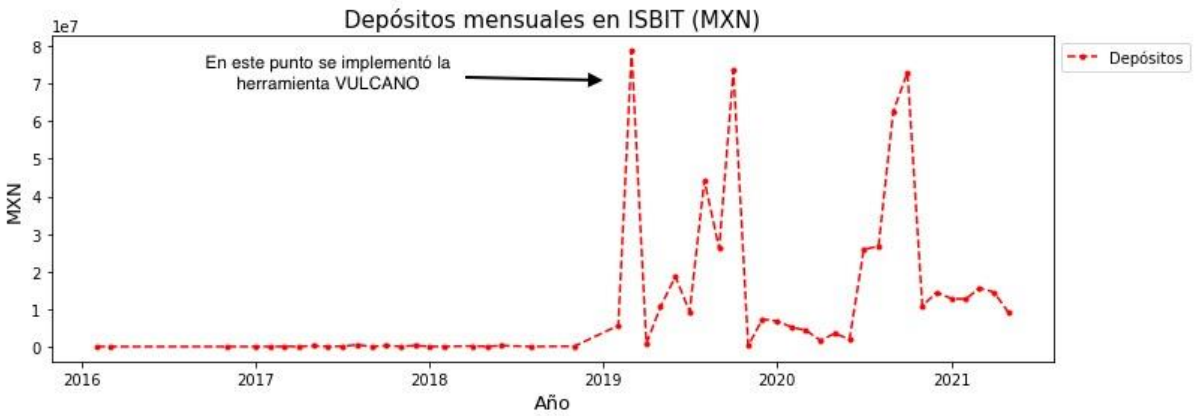


Ilustración 47 Depósitos mensuales en ISBIT (MXN)

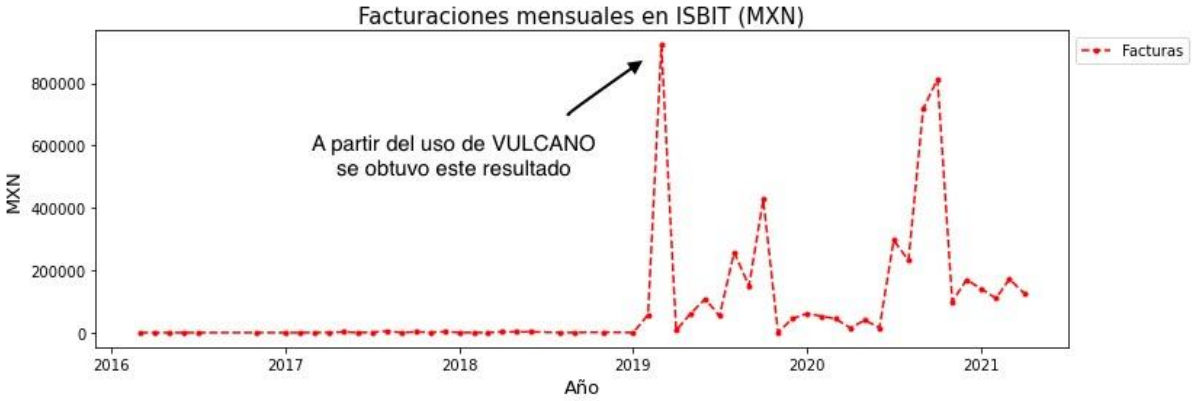


Ilustración 48 Facturaciones mensuales en ISBIT (MXN)

Ilustración 49 Intercambios mensuales en ISBIT (BTC)

Al analizar las gráficas, podemos notar que en las tres pasa algo similar; al usar la herramienta VULCANO el cambio más significante se encuentra a inicios del año 2019, que fue el momento de su implementación.

En el anexo G, se encuentran las gráficas de los retiros realizados, las transacciones y el número de depósitos, así como las consultas al sistema para la realización de cada gráfica.

Año Fiscal	2016	2017	2018	2019	2020	2021
Volumen de intercambio MXN	$1,571.26	$1,290,656.16	$587,797.38	$275,648,506.15	$253,284,273.91	$316,604,949.57
Volumen de intercambio US	$78.56	$64,532.81	$29,389.87	$13,782,425.31	$12,664,213.70	$15,830,247.48
Incremento VOLUMEN (x veces)		NA	0.46	468.95	0.92	1.25
% Margen (Ingreso por Comisiones)	2.14%	1.12%	4.11%	0.76%	1.01%	1.01
% Margen (Ingreso por Arbitraje / MM)	0.00%	0.00%	0.00%	0.12%	4.07%	4.06%

Ilustración 50 Volumen Intercambio de Activos Virutales contra Moneda Nacional Historico Anual en plataforma ISBIT EXCHANGE

El volumen de intercambio anual aumentó más de **468 veces** como resultado de implementar VULCANO a finales de diciembre de 2018.

La innovación, investigación y desarrollo invertido en la creación de VULCANO tuvo un impacto significativo en la rentabilidad y eficiencia de la empresa, lo cual se puede constatar en los Estados Financieros de ISBIT SA DE CV.

Concepto de ingreso(egreso) ISBIT SA DE CV / AÑO	2016	2017	2018	2019	2020	2021
Facturación de comisiones operaciones en plataforma	33.555135	$14,446.22	24,165.33	2,089,553.83	2,553,027.76	3,141,500.66
% INCREMENTO FACTURACION COMISIONES	0.00%	42952.19%	67.28%	8546.91%	22.18%	23.05%
Margen Profit por cada ciclo arbitraje	0.00%	0.00%	0.00%	0.12%	4.07%	4.07%
Rotacion Inventario (ciclos mensuales)			0.27	45.94	1.95	2.00
AUM (inventario propio) portafolios activos propios			$180,000.00	$500,000.00	$10,800,000.00	$21,349,440.00
Crecimiento anual AUM ops arbitraje/MM	0	0	0%	177.78%	2060.00%	97.68%
Ingresos ops arbitraje/MM/especulacion	$0.00	$0.00	$0.00	$320,000.00	$10,300,000.00	$10,549,440.00
Ventas Consultoria/Desarrollo/implementacion			870,200.00	0.00	0.00	0.00
Ingresos Brutos (antes costos, impuestos)			$894,365.33	$2,409,553.83	$12,853,027.76	$13,690,940.66
Costos			500,000.00	1,200,000.00	1,350,000.00	1,518,750.00
Gastos			450,200.00	1,200,900.00	1,320,000.00	1,450,911.82
Ingresos Netos			-$55,834.67	$8,653.83	$10,183,027.76	$10,721,278.84

Ilustración 51 Cambios en Estado de Resultados ISBIT SA DE CV

6. Conclusiones

La creación de VULCANO resultó ser más beneficioso de lo que imaginé, al principio fue pensado para el beneficio de la empresa ISBIT, buscando el apoyo de un sistema que me ayudara a hacer simples transferencias bancarias, pero al ponerlo en marcha me di cuenta de que su potencial daba para poder acoplarlo como herramienta de apoyo para Caerus al realizar arbitraje. Es un sistema que puede ser acoplado a necesidades particulares.

En la actualidad y lo que ha sido más común hasta el momento, es hacer transferencias por medio de intermediarios como Conekta, Openpay, PayPal, SafetyPay, entre otros. Estas empresas cobran comisiones considerablemente altas.

A continuación, se muestra la comisión y los días hábiles en los que tarda en llegar el depósito de cada intermediario en comparación a VULCANO.

Intermediario/ Solución	Comisión Recepción Pago Cobranza	Días hábiles de liquidación
Conekta	3.9%	4 dias
Openpay	2.9% + $2.5 MXN	3 dias
PayPal	3.5%	3-4 dias
VULCANO	0%	0.000208-0.04 dias

Lo anterior representa un problema para las empresas que desean transferir cantidades grandes de dinero a otra parte del mundo, definitivamente la tarifa será absurdamente alta además debe considerar el tiempo de llegada del dinero y su asistencia al banco para realizar la operación o bien, para configurar una autorización de transferencia. Es evidente que lo anterior implica dinero, tiempo y esfuerzo que puede ser ahorrado gracias a VULCANO.

Vulcano cumplió con las expectativas esperadas, la creación de este sistema me ha ahorrado tiempo de un equipo de 2 a 5 personas operando la tesoreria de manea manual y estoy seguro de que puede ser muy útil para otras empresas. Ahora queda seguir a la expectativa de las necesidades que se vayan presentando para seguir implementándolo para ampliar y mejorar su rendimiento.

La implementación de VULCANO en ISBIT SA DE CV permitió ofrecer a los clientes opciones de depósito y retiro de fondos de menor costo. Antes de implementar VULCANO forzosamente necesitamos trasladar el costo de procesar un depósito o retiro al cliente final.

Pudimos validar de primera mano la Ley de Oferta y Demanda de la Economía. La ley de la oferta y la demanda es el principio básico sobre el que se basa una economía de mercado. Este principio refleja la relación que existe entre la demanda de un producto y la cantidad ofrecida de ese producto teniendo en cuenta el precio al que se vende el producto (Varian, H., 1992).

Así, según el precio que haya en el mercado de un bien, los oferentes (los que venden) están dispuestos a fabricar un número determinado de ese bien. Al igual que los demandantes (los que compran) están dispuestos a comprar un número determinado de ese bien, dependiendo del precio.

Según esta teoría, la ley de la demanda establece que, manteniéndose todo lo demás constante (ceteris paribus), la cantidad demandada de un bien disminuye cuando el precio de ese bien aumenta. Por el otro lado, la ley de la oferta indica que, manteniéndose todo lo demás constante (ceteris paribus), la cantidad ofrecida de un bien aumenta cuando lo hace su precio.

Así, la curva de la oferta y la curva de la demanda muestran como varía la cantidad ofrecida o demandada, respectivamente, según varía el precio de ese bien.

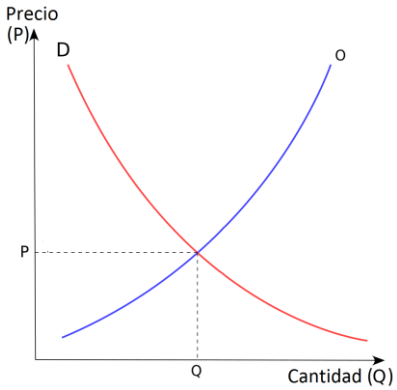


Ilustración 52 Ley de Oferta y Demanda en la Economía

Para entender cómo se puede llegar al punto de equilibrio hay que hablar de dos situaciones de exceso:

1) ***Exceso de oferta:*** Cuando existe exceso de oferta, el precio al que se están ofreciendo los productos es mayor que el precio de equilibrio. Por tanto, la cantidad ofrecida es

mayor que la cantidad demandada. Con lo consiguiente, los oferentes bajarán los precios para aumentar las ventas.

2) ***Exceso de demanda:*** Por el lado contrario, cuando existe escasez de productos, significa que el precio del bien ofrecido es menor que el precio de equilibrio. La cantidad demandada es mayor que la cantidad ofrecida. De modo que los oferentes aumentarán el precio, dado que hay muchos compradores para pocas unidades del bien para que el número de demandantes disminuya, y se establezca el punto de equilibrio.

Trasladando a un gráfico los comportamientos de la oferta y demanda que acabamos de explicar, se comprende que la curva de oferta (***O***, línea azul) sea creciente y la curva de demanda (***D***, línea roja) sea decreciente. El punto donde se cruzan se conoce como equilibrio de mercado. (Varian, H., 1992)

Si partimos del punto inicial en el que se demanda la cantidad Q1 de un bien al precio P1, y debido a alguna causa externa se produce un aumento en la demanda hasta la cantidad Q2, el precio del bien aumentará hasta situarse en P2. (Varian, H., 1992)

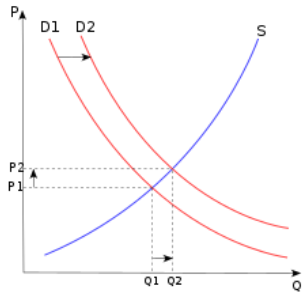


Ilustración 53 Curva Demanda, Efecto en Precio de Mercado por Aumento en la Cantidad de Demanda

Si ocurre por el contrario que los vendedores por alguna razón disminuyen su producción (por ejemplo, El “Bitcoin Halving” provoca que la producción de monedas de los Mineros disminuya), en la gráfica observaremos un movimiento de la curva de oferta (O) a la izquierda y por tanto, aumenta el precio del bien en cuestión y con ello la demanda se verá reducida.

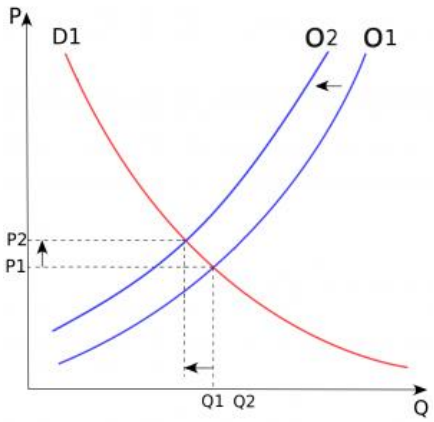


Ilustración 54 Curva de Oferta, Efecto en el Precio de Equilibrio por Disminución de la cantidad de Oferta

Una disminución en el Precio ***P*** de un servicio/producto implica un aumento en la cantidad ***Q*** de **demanda** de los consumidores.

Concretamente, observamos que una disminución del **86.19%** en el precio de nuestro producto/servicio aumentó la demanda para el mismo **468.95 veces** (ventas 2019/ventas 2018) en el transcurso de un año.

A pesar de bajar los precios al cliente final logramos aumentar la facturación de comisiones cobradas en el Exchange **85.4 veces** (en 2019 a comparación de 2018).

Implementar VULCANO también permitió abrir nuevas líneas de negocio que antes no eran factibles (Arbitraje Financiero, CAERUS).

Integrar VULCANO con METIS permitió evitar multas y sanciones que pueden oscilar entre el 10% al 100% del valor de una operación o 800,000 mxn.

7. Referencias

Secretaria de Hacienda y Credito Publico. 9 marzo 2018, LRITF, Diario Oficial de la Federación. Sitio web: https://www.dof.gob.mx/nota_detalle.php?codigo=5515623&fecha=09/03/2018

Congreso de Los Estados Unidos Mexicanos. Texto Vigente de Ley publicada el 17 de Octubre de 2012, Ultima reforma publicada en DOF 20 Mayo 2021 , LEY FEDERAL PARA LA PREVENCIÓN E IDENTIFICACIÓN DE OPERACIONES CON RECURSOS DE PROCEDENCIA ILÍCITA, Diario Oficial de la Federación. Sitio web: https://www.diputados.gob.mx/LeyesBiblio/pdf/LFPIORPI_200521.pdf

Congreso de los Estados Unidos Mexicanos. 30 Noviembre 2020, Acuerdo por el que se modifican las Reglas de Carácter General a que se refiere la Ley Federal para la Prevencion e Identificación de Operaciones con Recursos de Procedencia Ilicita ,Diario Oficial de la Federación. Sitio web: https://dof.gob.mx/nota_detalle.php?codigo=5606232&fecha=30/11/2020

Leyva, J., & Soto, G. (1 Junio 2018). Este es el relato secreto de un hacker que estuvo ligado con el robo histórico a la banca mexicana. 29 Abril 2020, de EL FINANCIERO Sitio web: https://www.elfinanciero.com.mx/bloomberg-businessweek/este-es-el-relato-secreto-de-un-hacker-que-estuvo-ligado-con-el-robo-historico-a-la-banca-mexicana/

Schwaber, K., & Southerland, J. (2020). The Scrum Guide. 18 abril 2022, de SCRUM.org Sitio web: https://scrumguides.org/scrum-guide.html

Avellaneda, M., & Stoikov, S. (2006). High-frequency trading in a limit order book. 18 Abril 2022, de CORNELL Sitio web: https://people.orie.cornell.edu/sfs33/LimitOrderBook.pdf

Mitchell, R. (2015). Web Scraping with Python: Collecting Data from the Modern Web. Estados Unidos de America: O'REILLY.

Perriault, N. (2016). CasperJS: Navigation scripting & testing for PhantomJS and SlimerJS. 3 mayo 2022, de CasperJS Contributors Sitio web: https://www.casperjs.org/

Comité de Pagos e Infraestructuras del Mercado. (28 Marzo 2016). Divulgación del cumplimiento y adopción de los Principios para las Infraestructuras del Mercado Financiero. 10 Mayo 2022 12:45pm, de Banco de Mexico Sitio web: https://www.banxico.org.mx/sistemas-de-pago/d/%7B89B6CCF0-6070-7389-3DD5-B27AC4ECD9D1%7D.pdf

Hidayat, A. (2020). PhantomJS Documentation. 3 mayo 2022, de PhantomJS Contributors Sitio web: https://phantomjs.org/documentation/

Schenker G.N., (2018). Containerize your Apps with Docker and Kubernetes. USA: Packt Publishing.

Matthias K., Kane S.P. (junio 2015). Docker - Up & Running - Shipping Reliable Containers in Production. Estados Unidos: O'Reilly Media.

Negus C. (2015). Docker Containers - Build and Deploy with Kubernetes, Flannel, Cockpit, and Atomic. USA: Pearson Education.

Goasguen S. (2015). Docker Cookbook - Solutions and Examples for Building Distributed Applications. USA: O'Reilly Media.

Baire A. (2017). Docker Tutorial. Francia: Université de Rennes.

Nickoloff J. (2016). Docker in Action. Nueva York: Manning Publications Co.

Miell I., Sayers A.H. (2016). Docker in Practice. Nueva York: Manning Publications Co.

Vohra D. (2016). Kubernetes - Microservices with Docker. Canadá: Apress.

Huss R., Ibryam B. (2017), Kubernetes Patterns, Lean Publishing

Cesar de la Torre. (2016). Containerized Docker Application Lifecycle with Microsoft Platform and Tools. USA: Microsoft Corp.

Chodorow, Kristina. (2013). MongoDB: The Definitive Guide, Second Edition. Estados Unidos de América: O'Reilly Media.

Brown, Ethan. (2014). Web Development with Node and Express, First Edition. Estados Unidos: O'Reilly Media.

Desarrolladores de Node JS. (1 Julio de 2019). Documentación Oficial de NodeJs. Recuperado en enero 2020, USA. Sitio web: https://nodejs.org/api/documentation.html

MongoDB I. (2009). Documentación Oficial de Azure Mongodb. Recuperado en 2020, de Microsoft Corporation, Sitio web: https://docs.microsoft.com/en-us/azure/cosmos-db/partition-data

Gobierno de México. Lineamientos de Prevención de Lavado de Dinero para Actividades Vulnerables. 18 abril de 2020. Sitio web: https://sppld.sat.gob.mx/pld/interiores/sppld.html

Davis J., Daniels R. (2016). Effective DevOps. USA: O'Reilly.

Abhishek Verma, Luis Pedrosa, Madhukar R. Korupolu, David Oppenheimer, Eric Tune, John Wilkes. (2015). Proceedings of the European Conference on Computer Systems (EuroSys) . Bordeaux, France: ACM.

Marko Luksa. (20 enero 2018). Kubernetes in Action. Slovenia: Manning Publications.

Hightower K. (2019). Kubernetes: Up and Running, 2nd Edition. USA: O'Reilly.

Hausenblas M., Schimanski S. (2019). Programming Kubernetes. USA: O'Reilly. Sitio web:

https://learning.oreilly.com/library/view/programming-kubernetes/9781492047094/ch01.html

Federal Information Processing Standards Publication 197, (26 Abril 2011) NIST, USA

Daemen J., Rijmen V. (2003). The Rijndael Block Cipher. USA: NIST.

Sebastián Peyrott Spread the word. (2013). Introduction to JSON Web Tokens. Recuperado en 2020, de JWT Sitio web: https://jwt.io/introduction

Internet Security Research Group. (2020). Tipos de Desafios Lets Encrypt. Diciembre 2021, de LINUX FOUNDATION Sitio web: https://letsencrypt.org/docs/challenge-types/

Chacon, S. & Straub B. (2022). Pro Git. USA: APRESS. Sitio web: https://github.com/progit/progit2/releases/download/2.1.339/progit.pdf

Leavitt, S. & Staheli, D. (2019). ¿Qué es Azure Pipelines?. 29 Abril 2020, de MICROSOFT CORPORATION Sitio web: https://docs.microsoft.com/es-mx/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops|

Varian, H. (1992). Microeconomic Analysis, THIRD EDITION. Estados Unidos de America: Norton & Company, Inc.

NIST. (2001). Specification for the ADVANCED ENCRYPTION STANDARD (AES), Publication 197. Enero 2020, de Federal Information Processing Standards Sitio web: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

Rijmen V., Daemen J.. (2003). AES Proposal: The Rijndael Block Cipher. 8 junio 2021, de NIST Sitio web: https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf

Juggery, L. (2019). A Closer Look at Etcd: The Brain of a Kubernetes Cluster. 12 Agosto 2020, de MEDIUM - BETTER PROGRAMMING Sitio web: https://betterprogramming.pub/a-closer-look-at-etcd-the-brain-of-a-kubernetes-cluster-788c8ea759a5

8 Bibliografía

Acosta, S. & Nuñez R. (febrero 2022). Política de Cookies de ISBIT. 6 Enero de 2022, de ISBIT S.A. de C.V. Sitio web: https:///isbit.exchange/vulcano/unam/Politica_de_Cookies_ISBIT_2022.pdf

Acosta, S. & Núñez, R.. (febrero 2016). Términos y Condiciones de Uso de ISBIT EXCHANGE. 24 Marzo 2022, de ISBIT S.A. de C.V. Sitio web: https://isbit.exchange/vulcano/unam/Terminos_y_condiciones_ISBIT_2022.pdf

9 Glosario de Palabras

VSTS: Microsoft Visual Studio Team System (VSTS) es un conjunto de herramientas de gestión del ciclo de vida de desarrollo de software que abordan las necesidades de una variedad de roles dentro de la organización.

DSL: La línea de abonado digital o línea de suscriptor digital, *Digital Subscriber Line* (DSL), es una familia de tecnologías que proporcionan el acceso a Internet mediante la transmisión de datos digitales a través del par trenzado de hilos de cobre convencionales de la red telefónica básica o conmutada, constituida por las líneas de abonado: ADSL, ADSL2, ADSL2+, SDSL, IDSL, HDSL, SHDSL, VDSL y VDSL2

Helm Chart: Helm lo ayuda a administrar las aplicaciones de Kubernetes: los gráficos de Helm lo ayudan a definir, instalar y actualizar incluso la aplicación de Kubernetes más compleja. Los gráficos son fáciles de crear, versionar, compartir y publicar, así que comience a usar Helm y deje de copiar y pegar.

YAML: es un formato de serialización de datos legible por humanos inspirado en lenguajes como XML, C, Python, Perl, así como el formato para correos electrónicos especificado en RFC 2822 (publicaciones RFC).

VPN: es una tecnología de red que permite conectar uno o más ordenadores en una red privada virtual, a través de una red pública como Internet, sin necesidad de que los ordenadores estén conectados físicamente entre sí o de que estén en un mismo lugar.

Layout: es la representación de un plano sobre el cual se va a dibujar la distribución de un espacio específico o determinado. El layout puede tomarse como base para una página web, pues es a partir de ese plan o diseño que esta se irá a desarrollar.

MongoDB: es la más representativa de las bases de datos conocidas como NoSQL, acrónimo de Not only SQL. También podemos denominarla con el término de base de datos documental, ya que lo que almacenamos son puros documentos JSON y no registros, como sucede en las tablas de las bases de datos relacionales.

Docker: es una herramienta diseñada para beneficiar tanto a desarrolladores, testers, como administradores de sistemas, en relación a las máquinas, a los entornos, en sí donde se ejecutan las aplicaciones software, los procesos de despliegue, etc.

Kubernetes: es una plataforma open source que automatiza las operaciones de los contenedores de Linux. Elimina muchos de los procesos manuales involucrados en la implementación y escalabilidad de las aplicaciones en contenedores.

SPA: son las siglas de "Single Page Application". Es un tipo de aplicación web donde todas las pantallas las muestra en la misma página, sin recargar el navegador.

GUI: es una interfaz entre la persona y la máquina. Representa el código del *backend* de un sistema de la forma más clara posible para el usuario con el fin de simplificarle las tareas diarias. Para esto, son muy importantes los iconos y las imágenes, ya que solo estos permiten una aplicación universal e independiente del texto. Por ejemplo, casi todo el mundo sabe cómo es un icono de wifi, mientras que la palabra varía mucho en los distintos idiomas.

NGINX: Nginx es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico. Es software libre y de código abierto, licenciado bajo la Licencia BSD simplificada; también existe una versión comercial distribuida bajo el nombre de Nginx Plus.

CSS: es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

HTML: lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de las siglas que corresponden a HyperText Markup Language, es decir, "Lenguaje de Marcas de Hipertexto".

10. Anexos

Anexo A. Comprobante electrónico de pago

Éste es un comprobante de una transferencia SPEI (Sistema de pagos electrónicos interbancarios) autentificado por Banxico.

El **CEP** garantiza que la transacción entre clientes/proveedores se llevó de manera exitosa. A partir de un **CEP** se puede obtener la información necesaria para que los clientes registren este movimiento en el sistema Comercial y de esta forma puedan llevar un mejor control de sus transacciones.

El **CEP** cuenta con la siguiente información:

- Fecha de operación en **SPEI**
- Fecha calendario de abono en la cuenta beneficiaria
- Hora calendario de abono en la cuenta beneficiaria.
- Banco emisor del pago
- Titular de la cuenta ordenante
- CLABE / Tarjeta de débito/Número de celular de la cuenta ordenante
- RFC o CURP registrado para la cuenta ordenante
- Banco receptor del pago
- Titular de la cuenta beneficiaria
- CLABE / Tarjeta de débito/Número de celular de la cuenta beneficiaria
- RFC o CURP registrado para la cuenta ordenante
- Concepto del pago
- Monto (peso en moneda nacional)
- IVA (pesos en moneda nacional)
- Número de referencia
- Clave de rastreo

A continuación, el ejemplo de un CEP en formato pdf y en xml

SPEI SISTEMA DE PAGOS ELECTRÓNICOS INTERBANCARIOS
Pagos más rápidos y seguros

Fecha de consulta 24 de marzo de 2021
Hora de consulta 16:53:44 horas

COMPROBANTE ELECTRÓNICO DE PAGO

Fecha de operación en el SPEI®	24 de marzo de 2021	Monto	$ 28,789.35
Fecha de abono en la cuenta beneficiaria*	24 de marzo de 2021	IVA	$ 0.00
Hora de abono en la cuenta beneficiaria*	16:42:41 horas	Referencia numérica	2021032
Concepto del pago	2Q MARZO 2021 servicios de administracio	Clave de rastreo	036INBU2403202166817925

Ordenante		Beneficiario	
Institución emisora del pago	INBURSA	Institución receptora del pago	BBVA BANCOMER
Titular de la cuenta	ISBIT S.A. DE C.V.	Titular de la cuenta	SWITCH SOLUCIONES HUMANAS SA DE CV
RFC/CURP	IIN131003RTA	RFC/CURP	SSH170609776
CLABE,Tarjeta de débito,Número de celular	036180500199863911	CLABE,Tarjeta de débito,Número de celular	012180001108570255

*La hora de abono corresponde al huso horario que rige en la Ciudad de México.

Número de Serie del Certificado de Seguridad de la institución receptora del pago
00001000000411613442

Cadena Original (información del pago):
||01|24032021|24032021|164241|40012|INBURSA|ISBIT S.A. DE C.V.|40|036180500199863911|IIN131003RTA|BBVA BANCOMER|SWITCH SOLUCIONES HUMANAS SA DE CV|40|012180001108570255|SSH170609776|2Q MARZO 2021 servicios de administracio|00000000000000.00|00000000028789.35|00001000000411613442||

Sello Digital (firma provista por la institución receptora del pago):
KBXBGbGEnk/gSz8pNwaThbQLIQZydvyluf4Ez0YNn5N0GnWIsgJHdpkYBJoznvc+c1o2ZU+4qOdECKoqeYQwZtK6csNtkn0rIzmMJ7HjbPdtCBAscrpvuhHONX39IBaUwIyIb+SJg7uGf1xpH/etbq3AXfILRKBIZNbfttfUpB/f8y+96KmmvouY3T27K0zcRINaXZ9W/DQUC6L2G0oWCLHdg8TUgVaXd1RodkQNIKQc/42Q4Z7d9g52Hv/4Aaj0NfnwtR7W8Wn1QkIQ2zXH12D7eWOgy/2hG5AsqcI59NRyiS6XEdP64ylCHNze62mS1Kt+2Ltgy87sIIJ2wdJTxg==

Ilustración 55 Ejemplo de CEP

CEP.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<SPEI_Tercero FechaOperacion="2021-03-24" Hora="16:42:41"
ClaveSPEI="40012"
sello="KBXBGbGEnk/gSz8pNwaThbQLIQZydvyluf4Ez0YNn5N0GnWIsgJHdpkYB
Joznvc+c1o2ZU+4qOdECKoqeYQwZtK6csNtkn0rIzmMJ7HjbPdtCBAscrpvuhHO
NX39IBaUwIyIb+SJg7uGf1xpH/etbq3AXfILRKBIZNbfttfUpB/f8y+96KmmvouY3T
27K0zcRINaXZ9W/DQUC6L2G0oWCLHdg8TUgVaXd1RodkQNIKQc/42Q4Z7d9g
52Hv/4Aaj0NfnwtR7W8Wn1QkIQ2zXH12D7eWOgy/2hG5AsqcI59NRyiS6XEdP6
4ylCHNze62mS1Kt+2Ltgy87sIIJ2wdJTxg=="
numeroCertificado="00001000000411613442"
```

```
cadenaCDA="||01|24032021|24032021|164241|40012|INBURSA|ISBIT S.A. DE
C.V.|40|036180500199863911|IIN131003RTA|BBVA BANCOMER|SWITCH
SOLUCIONES HUMANAS SA DE
CV|40|012180001108570255|SSH170609776|2Q MARZO 2021 servicios de
administracio|000000000000000.00|000000000028789.35|000010000004116
13442||KBXBGbGEnk/gSz8pNwaThbQLIQZydvyluf4Ez0YNn5N0GnWIsgJHdpkY
BJoznvc+c1o2ZU+4qOdECKoqeYQwZtK6csNtkn0rIzmMJ7HjbPdtCBAscrpvuhH
ONX39IBaUwIyIb+SJg7uGf1xpH/etbq3AXfILRKBIZNbfttfUpB/f8y+96KmmvouY
3T27K0zcRINaXZ9W/DQUC6L2G0oWCLHdg8TUgVaXd1RodkQNlKQc/42Q4Z7d
9g52Hv/4Aaj0NfnwtR7W8Wn1QkIQ2zXH12D7eWOgy/2hG5AsqcI59NRyiS6XEd
P64ylCHNze62mS1Kt+2Ltgy87sIIJ2wdJTxg=="
claveRastreo="036INBU2403202166817925">
  <Beneficiario BancoReceptor="BBVA BANCOMER" Nombre="SWITCH
SOLUCIONES HUMANAS SA DE CV" TipoCuenta="40"
Cuenta="012180001108570255" RFC="SSH170609776" Concepto="2Q MARZO
2021 servicios de administracio" IVA="000000000000000.00"
MontoPago="000000000028789.35"/>
  <Ordenante BancoEmisor="INBURSA" Nombre="ISBIT S.A. DE C.V."
TipoCuenta="40" Cuenta="036180500199863911" RFC="IIN131003RTA"/>
</SPEI_Tercero>
```

Anexo B. RBAC control de acceso basado en roles

La gestión del acceso a los recursos de la nube es una función fundamental para cualquier organización que utilice la nube. El control de acceso basado en roles de Azure (Azure RBAC) lo ayuda a administrar quién tiene acceso a los recursos de Azure, qué pueden hacer con esos recursos y a qué áreas tienen acceso.

Azure RBAC es un sistema de autorización creado en Azure Resource Manager que proporciona una administración de acceso detallada de los recursos de Azure.

Azure RBAC nos ayuda a:

- Permitir que un usuario administre máquinas virtuales en una suscripción y otro usuario administre redes virtuales
- Permitir que un grupo de DBA administre bases de datos SQL en una suscripción
- Permitir que un usuario administre todos los recursos en un grupo de recursos, como máquinas virtuales, sitios web y subredes.
- Permitir que una aplicación acceda a todos los recursos de un grupo de recursos

Una *asignación de rol* es el proceso de adjuntar una definición de rol a un usuario, grupo, entidad de servicio o identidad administrada en un ámbito particular con el propósito de otorgar acceso. El acceso se otorga mediante la creación de una asignación de funciones y el acceso se revoca al eliminar una asignación de funciones.

Anexo C. Documentación de H2H Santander

Hojas de Cálculo (especificación de formatos intercambio datos y protocolos)

Layout CECOBAN SPID	https://docs.google.com/spreadsheets/d/1LqBljhhijOxJfZs71Dgy6RoV-dtVzkqO/edit#gid=291746021 https://docs.google.com/spreadsheets/d/1LqBljhhijOxJfZs71Dgy6RoV-dtVzkqO/edit#gid=291746021
Layout CECOBAN (SPEI, TEF, NÓMINA INTERBANCARIA, IMP FEDERALES)	https://docs.google.com/spreadsheets/d/1l_LDo4uxVvzjQx-yWzLbHmWP9e5Xph01uKktKCd_pkk/edit#gid=1689382279
Layout RCE - Reporte Cobranza Enriquecido	https://docs.google.com/spreadsheets/d/1c64TWT668NfO4xqjPcL0_tAXYnlfaQc1hlS-2cjYMFw/edit#gid=1836950563
Layout Órdenes de Pago	https://docs.google.com/spreadsheets/d/1lnlG8ulwLD1zVfuXTxu7N53zSFclue1p/edit#gid=989355549
Claves de Rechazo	https://docs.google.com/spreadsheets/d/1WPx2jV1GZIhddSMiuJMuuQDcTIEUZi-v/edit#gid=452799893
Consulta de Transferencias	https://docs.google.com/spreadsheets/d/1LJbFu--od4zjzXg0N68c79fUFeAyL8_3/edit#gid=1895405240

Documentos PDF y Procesador de Texto

Presentación Comercial	https://drive.google.com/open?id=1YhDcX5zTfrvZkzuAlUuTh9l8hb0HrDnM
Manual Configuracion SSH/SFTP	https://drive.google.com/open?id=1xN2NV22nqErogda6q1RK8oTv2eGkrXr1
Manual Instalación API JAR Encripción	https://drive.google.com/open?id=13f_emr72ZiPJVgrXsmihkFp6YKl5ft2e
Manual Operativo Frontend	https://drive.google.com/open?id=1NLoiXbQVKSUWnqme7GfRRB1YsWIhuSkA
Monitoreo y Contingencia Operativa	https://drive.google.com/open?id=1recRun54PnGIcexRZAkOziOIuTmbSSxx

Anexo D. Código fuente del sistema de gestión de tesorería desarrollado

Microservicio vulcano-santander-layout-gen-pro

vulcano-santander-layout-gen-pro Dockerfile

```
FROM node:latest
LABEL maintainer="Sebastian Acosta - ISBIT SA DE CV <sebastian@isbit.co>"
# Create app directory
WORKDIR /app
# Authenticate to NPM Registry
ARG NPM_TOKEN
# Bundle app source
COPY . .
# Install app dependencies
RUN npm install
RUN rm -f .npmrc
EXPOSE 33437 33437
CMD [ "npm", "start" ]
```

vulcano-santander-layout-gen-pro - server.js

```
require('dotenv').config()
const express = require('express')
const bodyParser = require('body-parser')
const validator = require('express-validator')
const akvHandler = require('@sebastian-isbit/akv_handler')
const probesRouter = require('./routes/probes')
const layoutRoutes = require('./routes/layout.routes')
const port = process.env.PORT || 33439

const app = express()
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true }))
app.use(validator())

// app.use('/api', apiRouter)
app.use('/santander', layoutRoutes)
app.use('/probes', probesRouter)

app.get('/', (req, res) => {
 return res.status(200).send('Santander Layout Generator and Procesor\'s TMS API V1.0')
})

if (process.env.NODE_ENV != 'local'){
 const keyVaultName = process.env.KEY_VAULT_NAME
 const mongoUriSecret = process.env.MONGO_URI_NAME + process.env.MONGO_URI_VERSION
 const accountKeySecret = process.env.ACCOUNT_KEY_NAME + process.env.ACCOUNT_KEY_VERSION
 // Retrieve sensitive information from Azure Key Vault
 akvHandler.getSecrets(keyVaultName, [mongoUriSecret, accountKeySecret])
   .then((secrets) => {
    app.set('mongoUri', secrets[mongoUriSecret])
    app.set('accountKey', secrets[accountKeySecret])
    console.log("Keys were retrieved successfully!")
   })
   .catch((err) => {
    console.log("Failed to get the keys", err)
   })
} else {
// Database connection handler
 const connectToDatabase = require('@sebastian-isbit/db-connection-handler')
 const mongoUri = process.env.MONGO_URI || 'mongodb://localhost/'

 app.set('mongoUri', mongoUri)
 app.set('accountKey', process.env.ACCOUNT_KEY)

 console.log("App using database at " + mongoUri)
}

// error handler
app.use(function (err, req, res, next) {
 res.status(err.status || 500)
 res.json({
   message: err.message,
   error: {}
 })
})

app.set('port', port)
```

```
var server = app.listen(app.get('port'), function () {
  console.log('Express server listening on port ' + server.address().port)
})
```

vulcano-santander-layout-gen-pro - layout.controller.js

```
const PayoutModel = require('@sebastian-isbit/vulcano-models').payout
const LayoutModel = require('@sebastian-isbit/vulcano-models').layout
const Settings = require('@sebastian-isbit/vulcano-models').definitions
const blf = require('../lib/BankLayoutFactory')
const Joi = require('joi')
const moment = require('moment')
const azurefileHandler = require('@sebastian-isbit/azurefile-handler')
const santanderClientAccountID = process.env.SANTANDER_CLIENT_ACCOUNT_ID || '089000003056'
// Database connection handler
var connectToDatabase = require('@sebastian-isbit/db-connection-handler')
require('dotenv').config()
const LayoutController = {
 payoutSchema () {
  return Joi.object().keys({
   payoutId: Joi.string().required()
  })
 },
 layoutSchema () {
  return Joi.object().keys({
   in_file: Joi.string().required(),
   transaction_list: Joi.array().min(1),
   header_data: {
    record_type: Joi.string().length(2).regex(/^[0-9]+$/).required(),
    sequence_number: Joi.string().length(7).required(),
    operation_code: Joi.string().length(2).regex(/^[0-9]+$/).required(),
    participant_bank: Joi.string().length(3).regex(/^[0-9]+$/).required(),
    direction: Joi.string().length(1).required(),
    service: Joi.string().length(1).required(),
    currency_code: Joi.string().length(2).regex(/^[0-9]+$/).required(),
    modality: Joi.string().length(1).regex(/^[0-9]+$/).required(),
    batch_number: Joi.string().required()
   }
  })
 },
 async generateLayoutFile (res, payouts, fileService, lastBatchIndex, operationCode =
Settings.OperationCodes.CREDIT_INSTRUCTION) {
  const share = process.env.SHARE
  const azurefileDirectory = process.env.GENERATED_LAYOUTS_DIR_AF

  try {
   // Retrieve DB Connection String
   var mongoUri = res.app.get('mongoUri')

   await connectToDatabase(mongoUri, process.env.DB_NAME)
  } catch (err) {
   throw {
    reason: 'databaseError',
    err: err.message
   }
  }

  // get current sequence number from previous layout number
```

```
  let batchNumber = '0000' + parseInt(lastBatchIndex)
  batchNumber = batchNumber.substr(batchNumber.length - 5)
  batchNumber = moment().format('DD') + batchNumber


   let dateESStr = moment().format('DDMMYYYY')
  let fileName = 'tran' + dateESStr + batchNumber.substr(batchNumber.length - 2) + operationCode + '_' +
santanderClientAccountID + '.in2'

  let layoutInstance = blf.buildLayoutInstance(payouts, batchNumber,res.app.get('mongoUri'))

  console.log('Layout :', layoutInstance.toString())

  try {
    return new Promise(resolve => {
      fileService.createFileFromText(share, azurefileDirectory, fileName, layoutInstance.toString(), function(err, result,
response) {
        if (err) {
          reject({
            reason: 'fsError',
            stack: err.message
          })
        } else {
          result = {
            in_file: fileName,
            transaction_list: layoutInstance.getTransactionList(),
            header_data: {
              record_type: layoutInstance.operationType(),
              sequence_number: layoutInstance.getHeader().sequenceNumber(),
              operation_code: layoutInstance.getHeader().operationCode(),
              participant_bank: layoutInstance.getHeader().bankNumberCode(),
              direction: layoutInstance.getHeader().directionCode(),
              service: layoutInstance.getHeader().serviceCode(),
              batch_number: batchNumber,
              currency_code: layoutInstance.getHeader().bankCurrencyCode(),
              modality: layoutInstance.getHeader().executionModality(),
            }
          }

          console.log(`File with generated layout has been writen to: ${fileName}`)
          console.log(result)

          resolve(result)
        }
      })
    })
  } catch (err) {
    throw {
      reason: 'fsError',
      stack: err.message
    }
  }
 },
 async generateLayout (req, res) {
  let lastLayout
  // Retrieve DB Connection String
  const mongoUri = res.app.get('mongoUri')
  // Create azurefile service
  const storageAccount = process.env.STORAGE_ACCOUNT
  const share = process.env.SHARE
  const accountKey = res.app.get('accountKey')
  const azurefileDirectory = process.env.GENERATED_LAYOUTS_DIR_AF
```

```
  try {
    var directories = [azurefileDirectory]
    var fileService = await azurefileHandler.connectToAzurefile(storageAccount, accountKey, share, directories)
  } catch(err) {
    return res.status(500).send({
      err: err
    })
  }

  // assure the database connection is active
  try {
    await connectToDatabase(mongoUri, process.env.DB_NAME)
  } catch (err) {
    return res.status(500).send({
      err: {
        reason: 'databaseError',
        err: err.message
      }
    })
  }

  // find all payouts to process
  PayoutModel.find({delivered: false/*, delivery_date: {$lte: Date.now()}*/}, async (err, payouts) => {
    if (err) {
      console.log('Error found. Payout not fount')
      return res.status(500).send({
        err: {
          reason: 'databaseError',
          err: err.message
        }
      })
    }

    // obtain last previous stored layout for finding last sequence number used
    try {
      var today = new Date(moment().format('YYYY-MM-DD'))
      var tomorrow = new Date(moment().add(1, 'days').format('YYYY-MM-DD'))

      lastLayout = await LayoutModel.findOne({"status_tracking": {$elemMatch: {'status':'layoutGenerated',
"status_change_date": {$lt: tomorrow, $gte: today}}}}, {}, { sort: { "header_data.batch_number": -1 }, limit: 1 })
    } catch (err) {
      return res.status(500).send({
        err: {
          reason: 'databaseError',
          err: err.message
        }
      })
    }

    // start processing all payouts in order to obtain the correspondent layouts
    try {
      let promises = []
      let layoutsResult = {
        toProcess: payouts.length,
      }

      console.log(layoutsResult)

        if (!payouts.length) {
        console.log('Nothing found')
```

```
      return res.status(200).send(layoutsResult)
    }

    // let secuenceNumber = lastLayout ? parseInt(lastLayout.header_data.sequence_number) : 0
    let batchNumber = 0;

    if (lastLayout) {
      batchNumber = lastLayout.header_data.batch_number.substr(2);
      batchNumber = parseInt(batchNumber)
    }
    payouts.forEach((payout) => {
      batchNumber++
      console.log('Payout to be processed: ' + JSON.stringify(payout) + ' with index ' + batchNumber)

      let promise = new Promise ((resolve, reject) => {
        let payoutProccessResult = {
          id: payout._id,
          success: false,
        }
        LayoutController.generateLayoutFile(res,                    [payout],fileService,                    batchNumber,
blf.translateBankOperationType(payout.bank_operation_type)).then(layout => {
          Joi.validate(layout, LayoutController.layoutSchema(), async (err, value) => {
            if (!err) {
              layout.status = 'layoutGenerated'
              layout.status_tracking = []
              layout.status_tracking.push({
                status: 'layoutGenerated',
                status_change_date: Date.now()
              })

              const layoutModel = new LayoutModel(layout)

              layoutModel.save(err => {
                if (err) {
                 payoutProccessResult.err = {
                   reason: 'databaseError',
                   stack: err.message
                 }

                  resolve(payoutProccessResult)
                }

                payout.delivered = true
                payout.save()

                payoutProccessResult.id = layout.in_file
                payoutProccessResult.success = true
                resolve(payoutProccessResult)
              })
            } else {
              console.log('Error found. Bad layout generated.')
              payoutProccessResult.err = {
                reason: 'malformedDataError',
                stack: 'Received a non valid data for layout. Details: ' + JSON.stringify(err.details)
              }

              resolve(payoutProccessResult)
            }
          })
        }).catch((err) => {
          console.log('Error found. Unknown.' + err.stack)
```

```
            payoutProccessResult.err = err
            resolve(payoutProccessResult)
          })
        })

        promises.push(promise)
      })

      layoutsResult.results = await Promise.all(promises)

      return res.status(200).json(layoutsResult)
    } catch (err) {
      return res.status(500).send({
        err: {
          reason: 'malformedDataError',
          err: err.message
        }
      })
    }
  })
},
// retrieve all layouts made
async getAllLayouts (req, res) {

  // Retrieve DB Connection String
  var mongoUri = res.app.get('mongoUri')

  // assure the database connection is active
  try {
    await connectToDatabase(mongoUri, process.env.DB_NAME)
  } catch (err) {
    return res.status(500).send({
      err: {
        reason: 'databaseError',
        err: err.message
      }
    })
  }

  LayoutModel.find({}, (err, layouts) => {
    if (err) { return res.status(404).send(err) }

    return res.json(layouts)
  })
},
// retrieve information for an specific layout
async getLayout (req, res) {
  let id = req.params.id

  // Retrieve DB Connection String
  var mongoUri = res.app.get('mongoUri')

  // assure the database connection is active
  try {
    await connectToDatabase(mongoUri, process.env.DB_NAME)
  } catch (err) {
    return res.status(500).send({
      err: {
        reason: 'databaseError',
        err: err.message
      }
    })
```

```
  }

  LayoutModel.findOne({ _id: id }, (err, layout) => {
    if (err) { return res.status(404).send(err) }

    return res.json(layout)
  })
},
// confirm a layout validation
async confirmValidLayout (req, res) {
  let id = req.params.id

  let layout = {
    notification_file: req.params.notification_file
  }

  // Retrieve DB Connection String
  var mongoUri = res.app.get('mongoUri')

  // assure the database connection is active
  try {
    await connectToDatabase(mongoUri, process.env.DB_NAME)
  } catch (err) {
    return res.status(500).send({
      err: {
        reason: 'databaseError',
        err: err.message
      }
    })
  }

  LayoutModel.update({ _id: id }, layout, (err, result) => {
    if (err) { return res.status(500).send(err) }

    return res.status(200).send({ msg: `The layout with id ${id} has been successfully updated.` })
  })
},
// process result for an in-progress layout
async setResultFileLayout (req, res) {
  let id = req.params.id

  let layout = {
    out_file: req.params.out_file
  }

  // Retrieve DB Connection String
  var mongoUri = res.app.get('mongoUri')

  // assure the database connection is active
  try {
    await connectToDatabase(mongoUri, process.env.DB_NAME)
  } catch (err) {
    return res.status(500).send({
      err: {
        reason: 'databaseError',
        err: err.message
      }
    })
  }

  LayoutModel.update({ _id: id }, layout, (err, result) => {
    if (err) { return res.status(500).send(err) }
```

```
    return res.status(200).send({ msg: `The layout with id ${id} has been successfully updated.` })
   })
  },
  // remove all layouts from database
  async removeAllLayouts (req, res) {

   // Retrieve DB Connection String
   var mongoUri = res.app.get('mongoUri')

   // assure the database connection is active
   try {
    await connectToDatabase(mongoUri, process.env.DB_NAME)
   } catch (err) {
    return res.status(500).send({
     err: {
      reason: 'databaseError',
      err: err.message
     }
    })
   }

   LayoutModel.deleteMany({}, (err) => {
    if (err) { return res.status(404).send(err) }

    const result = {}
    result['done'] = true
    result['msg'] = 'All data was removed from collection'

    return res.json(result)
   })
  }
}

module.exports = LayoutController
```

vulcano-santander-layout-gen-pro BankLayout.js

```
const blHeader = require('./BankLayoutHeader').EntryBankLayoutHeader
const blDetail = require('./BankLayoutDetail')
const blSummary = require('./BankLayoutSummary')
const Settings = require('@sebastian-isbit/vulcano-models').definitions

function BankLayout(operationType, blockNumber, presentationDate, payouts = [], scheduled = false) {
 // this._blockNumber = blockNumber
 // this._presentationDate = presentationDate
 // this._scheduled = scheduled
 this._operationType = operationType

 this._header = new blHeader(blockNumber, presentationDate, scheduled)
 this._details = []

 var amount = 0.00
 for (var i=0; i<payouts.length; i++) {
  var detail = new blDetail(payouts[i],i,presentationDate,this._operationType,scheduled)
```

```
    amount += payouts[i].amount
    this._details.push(detail)
  }

  this._summary = new blSummary(blockNumber,2+payouts.length,amount)
}

BankLayout.prototype.operationType = function() {
  if (!this._operationType)
    throw Error('This is an abstract class and could be created as an instance')

  return this._operationType
}

BankLayout.prototype.getHeader = function() {
  if (!this._operationType)
    throw Error('This is an abstract class and could be created as an instance')

  return this._header
}

BankLayout.prototype.getDetails = function() {
  if (!this._operationType)
    throw Error('This is an abstract class and could be created as an instance')

  return this._details
}

BankLayout.prototype.getTransactionList = function() {
  if (!this._operationType)
    throw Error('This is an abstract class and could be created as an instance')

  var tl = [];

  for (var i=0; i<this._details.length; i++) {
    var payout = this._details[i].getPayoutData()

    if (payout)
      tl.push(payout._id)
  }

  return tl
}

BankLayout.prototype.getSummary = function() {
  if (!this._operationType)
    throw Error('This is an abstract class and could be created as an instance')

  return this._summary
}

BankLayout.prototype.toString = function() {
  if (!this._operationType)
    throw Error('This is an abstract class and could be created as an instance')

  var layoutContent = []

  layoutContent.push(this._header.toString())

  for (var i=0; i<this._details.length; i++)
    layoutContent.push(this._details[i].toString())
```

```
 layoutContent.push(this._summary)

 return layoutContent.join('\n')
}

function SPEIBankLayout(blockNumber, presentationDate, payouts = [], scheduled = false) {
 BankLayout.call(this,  Settings.OperationTypes.INTERBANK_SPEI_TRANSFERENCE,  blockNumber,presentationDate,
payouts,scheduled)
}

SPEIBankLayout.prototype = Object.create(BankLayout.prototype);

function TEFBankLayout(blockNumber, presentationDate, payouts = [], scheduled = false) {
 BankLayout.call(this,  Settings.OperationTypes.TEF_CREDIT_TRANSFERENCE,  blockNumber,presentationDate,
payouts,scheduled)
}

TEFBankLayout.prototype = Object.create(BankLayout.prototype);

function TMBBankLayout(blockNumber, presentationDate, payouts = [], scheduled = false) {
 BankLayout.call(this,  Settings.OperationTypes.SANTANDER_ACCOUNT_TRANSFERENCE,
blockNumber,presentationDate, payouts,scheduled)
}

TMBBankLayout.prototype = Object.create(BankLayout.prototype);

function NIBankLayout() {
 BankLayout.call(this, Settings.OperationTypes.INTERBANK_ROSTER_TRANSFERENCE, blockNumber,presentationDate,
payouts,scheduled)
}

NIBankLayout.prototype = Object.create(BankLayout.prototype);

function NMBBankLayout() {
 BankLayout.call(this,  Settings.OperationTypes.SANTANDER_ACCOUNT_ROSTER_TRANSFERENCE,
blockNumber,presentationDate, payouts,scheduled)
}

NMBBankLayout.prototype = Object.create(BankLayout.prototype);

function SPIDBankLayout() {
 BankLayout.call(this, '', blockNumber,presentationDate, payouts,scheduled)
}

SPIDBankLayout.prototype = Object.create(BankLayout.prototype);

module.exports = {
 BankLayout,
 NIBankLayout,
 TMBBankLayout,
 NMBBankLayout,
 TEFBankLayout,
 SPEIBankLayout
}
```

vulcano-santander-gen-pro BankLayoutDetail.js

```
const validateBankingDay = false;
const definitions = require('./BankLayoutDefinitions')
```

```
const moment = require('moment')
const Settings = require('@sebastian-isbit/vulcano-models').definitions

// BankLayoutDetail
function BankLayoutDetail(payout, payoutIndex, presentationDate, operationType, scheduled = false) {
 // if (!(payout instanceof LayoutModel))
 //   throw new Error('Invalid Layout instace')
 //
 // if (!blockNumber || (blockNumber > definitions.maxBlockNumber) || (blockNumber < definitions.minBlockNumber) )
 //   throw new Error('Invalid Block Number value')

 if (!presentationDate || !presentationDate.length || !presentationDate.match(/\d\d\d\d\d\d\d\d/))
  throw new Error('Invalid Presentation Date value')

 var date = moment(presentationDate, definitions.layoutDateFormat).utcOffset('-300')

 if (validateBankingDay && (date.day() == 0 || date.day() == 6))
  throw new Error('Presentation Date is not a banking operation date')

 if (scheduled) // presentationDate != moment().format(layoutDateFormat)
  this._executionModality = 2

 switch (operationType) {
  case Settings.OperationTypes.INTERBANK_SPEI_TRANSFERENCE:
  case Settings.OperationTypes.TEF_CREDIT_TRANSFERENCE:
  case Settings.OperationTypes.INTERBANK_ROSTER_TRANSFERENCE:
   if (payout.method == 'bank_account') // bank_account is the correct validation not allowed (01)
    throw Error('Operation type not allowed for this payment type.')

   // código del banco receptor
   this._receptorBankNumberCode = payout.bank_account.bank_code
   break;
  default:
   if (payout.bank_account.bank_code != definitions.hostBankCode)
    throw Error('Operation type not allowed due to wrong destination bank code.')

   if (payout.method == 'debit_card')
    throw Error('Operation type not allowed for this payment type.')

   this._receptorBankNumberCode = definitions.hostBankCode
   this._targetAccountNumber = (payout.bank_account.clabe.length == 18) ? payout.bank_account.clabe.substr(6, 11) : payout.bank_account.clabe
   break;
 }

 switch (payout.method) {
  case 'bank_account_clabe':
   if (payout.bank_account.clabe.length < 18)
    throw Error('Wrong clabe length. It must be 18 digits length.')

   this._targetAccountNumber = payout.bank_account.clabe
   this._targetAccountType = Settings.ReceptorAccountTypes.INTERBANK_CLABE_PAYMENT_CODE
   break;
  case 'bank_account':
   if (payout.bank_account.clabe.length < 18) // valitaion number must be 11. This is temporal
    throw Error('Wrong clabe length. It must be 18 digits length.')

   this._targetAccountNumber = payout.bank_account.clabe.substr(7, 11)
   this._targetAccountType = Settings.ReceptorAccountTypes.CLIENT_ACCOUNT_CODE
   break;
  case 'debit_card':
   // TODO: validate length of payment type
   this._targetAccountType = Settings.ReceptorAccountTypes.DEBIT_CARD_CODE
   break;
  case 'cellular':
   // TODO: validate length of payment type
   this._targetAccountType = Settings.ReceptorAccountTypes.CELLULAR_PHONE_CODE
   break;
  default:
   throw Error('Payment type not allowed.')
   break;
 }

 // orden de pago para crear el layout asociado
 this._payoutItem = payout;
 // tipo de bloque
 this._recordType = Settings.RecordTypes.DETAILS_BLOCK
 // número de secuencia inicial del bloque
 this._startSequenceNumber = 2
 // número de secuencia relativa a la inicial del bloque
 this._relativeSequenceNumber = payoutIndex;
 // código de operación (60 para un pago)
 this._operationCode = Settings.OperationCodes.CREDIT_INSTRUCTION
 // tipo de moneda de la operación
 // MXN - 01, USD - 02
 this._operationCurrency = '01'//Settings.CurrencyCodes.MXN_CURRENCY_CODE
 // fecha de realizada la transafernci a
 // _transactionDate
 // código del banco ordenante (014)
 this._bankNumberCode = definitions.hostBankCode
 // monto de la operación
 // _operationAmount // value given in payout
 // uso futuro
 this._futureUseField = ' '.repeat(16)
 // tipo de operación
 // SPEI - 01, NI - 02, TEF - 97, TMB - 98, NMB - 99
 this._transactionOperationType = operationType
 // fecha de aplicación del cargo
 // _executionDate // value calculated
 // número de cuenta del ordenante
```

```
  // (cuenta - 01, clabe - 40)
  this._sourceAccountType = Settings.OriginData.ORIGIN_ACCOUNT_TYPE
  // número de cuenta del ordenante
  // (cuenta - 11 dígitos, clabe - 18)
  this._sourceAccountNumber = Settings.OriginData.ORIGIN_ACCOUNT_NUMBER
  // nombre del dueño de la cuenta del ordenante
  this._sourceAccountOwner = 'ISBIT S.A DE C.V'
  // RFC del ordenante
  this._sourceAccountOwnerRFC = Settings.OriginData.ORIGIN_ACCOUNT_BANK_RFC
  // tipo de cuenta del beneficiario
  // (cuenta - 01, tarjeta - 03, clabe - 40, celular - 10)
  //_targetAccountType
  // número de cuenta del beneficiario
  // (cuenta - 11 dígitos, tarjeta - 16, clabe - 18, celular - 10)
  // _targetAccountNumber
  // nombre del dueño de la cuenta del beneficiario
  this._targetAccountOwner = payout.bank_account.holder_name
  // RFC del beneficiario
  this._targetAccountOwnerRFC = ''
  // referencia servicio emisor: VULCANO
  this._senderServiceReference = 'VULCANO'
  // nombre titular del servicio: ISBIT
  this._ownerServiceName = 'ISBIT'
  // IVA Operación
  this._operationIVA = '0.00'
  // referencia leyenda ordenante
  // _ordererReferenceLegend // given in payout data
  // referencia número ordenante
  // _ordererReferenceNumber // fixed by now using blockNumber
  // clave de rastreo
  // _trackingNumber // configured from data
  // motivo de la devolución
  this._rejectionCause = '00'
  // fecha de presentación
  // _presentationDate
  // uso futuro
  this._anotherFutureUseField = ' '.repeat(12)
  // referencia del cliente
  // _clientReference // fixed by now
  // descripción de la referencia del pago
  this._paymentReferenceDescription = 'ACCION DE PAGO CON VULCANO'
  // lista de correos de los beneficiarios
  this._beneficiaryEmailAddresses = []
  // línea de captura
  this._captureLine = ''
}

BankLayoutDetail.prototype.getPayoutData = function() {
  return (this._payoutItem) ? this._payoutItem : undefined;
}

BankLayoutDetail.prototype.recordType = function() {
  return this._recordType
}

BankLayoutDetail.prototype.sequenceNumber = function() {
  var sequenceNumber = '0'.repeat(6) + (this._startSequenceNumber + this._relativeSequenceNumber)

  return sequenceNumber.substr(sequenceNumber.length - 7, 7)
}

BankLayoutDetail.prototype.operationCode = function() {
  return this._operationCode
}

BankLayoutDetail.prototype.operationCurrency = function() {
  return this._operationCurrency
}

BankLayoutDetail.prototype.transactionDate = function() {
  return moment(this._payoutItem.creation_date).format(definitions.layoutDateFormat)
}

BankLayoutDetail.prototype.bankNumberCode = function() {
  return this._bankNumberCode;
}

BankLayoutDetail.prototype.receptorBankNumberCode = function() {
  return this._receptorBankNumberCode
}

BankLayoutDetail.prototype.operationAmount = function() {
  var amount = '0'.repeat(15) + (this._payoutItem.amount * 100)

  return amount.substr(amount.length - 15, 15)
}

BankLayoutDetail.prototype.transactionOperationType = function() {
  return this._transactionOperationType
}

BankLayoutDetail.prototype.executionDate = function() {
  return moment().utcOffset(-300).format(definitions.layoutDateFormat)
}

BankLayoutDetail.prototype.sourceAccountType = function() {
  return this._sourceAccountType
}
```

```
BankLayoutDetail.prototype.sourceAccountNumber = function() {
  var accountNumber = '0'.repeat(20) + this._sourceAccountNumber

  // console.log('accont -', accountNumber.substr(accountNumber.length - 20, 20))

  return accountNumber.substr(accountNumber.length - 20, 20)
}

BankLayoutDetail.prototype.sourceAccountOwner = function() {
  var accountOwner = this._sourceAccountOwner + ' '.repeat(40)

  return accountOwner.substr(0, 40)
}

BankLayoutDetail.prototype.sourceAccountOwnerRFC = function() {
  var accountOwnerRFC = this._sourceAccountOwnerRFC + ' '.repeat(18)

  return accountOwnerRFC.substr(0, 18)
}

BankLayoutDetail.prototype.targetAccountType = function() {
  return this._targetAccountType
}

BankLayoutDetail.prototype.targetAccountNumber = function() {
  var accountNumber = '0'.repeat(20) + this._targetAccountNumber

  return accountNumber.substr(accountNumber.length - 20, 20)
}

BankLayoutDetail.prototype.targetAccountOwner = function() {
  var accountOwner = this._targetAccountOwner + ' '.repeat(40)

  return accountOwner.substr(0, 40)
}

BankLayoutDetail.prototype.targetAccountOwnerRFC = function() {
  var accountOwnerRFC = this._targetAccountOwnerRFC + ' '.repeat(18)

  return accountOwnerRFC.substr(0, 18)
}

BankLayoutDetail.prototype.senderServiceReference = function() {
  var serviceReference = this._senderServiceReference + ' '.repeat(40)

  return serviceReference.substr(0, 40)
}

BankLayoutDetail.prototype.ownerServiceName = function() {
  var serviceOwner = this._ownerServiceName + ' '.repeat(40)

  return serviceOwner.substr(0, 40)
}

BankLayoutDetail.prototype.operationIVA = function() {
  var amount = '0'.repeat(15) + (this._operationIVA * 100)

  return amount.substr(amount.length - 15, 15)
}

BankLayoutDetail.prototype.ordererReferenceLegend = function() {
  var ordererReferenceLegend = this._payoutItem.description + ' '.repeat(40)

  return ordererReferenceLegend.substr(0, 40)
}

BankLayoutDetail.prototype.ordererReferenceNumber = function() {
  return this.blockNumber() // TODO: definir un código de 7 dígitos para este campo
}

BankLayoutDetail.prototype.trackingNumber = function() {
  var noise = ''
  var alphabet = '0123456789'
  var date = moment().utcOffset(-300).format(definitions.layoutDateFormat)

  for ( var i = 0; i < 3; i++ ) {
    noise += alphabet.charAt(Math.floor(Math.random() * 10));
  }

  return `${this._ownerServiceName}${this._senderServiceReference}${date}${this.ordererReferenceNumber()}${noise}`

  // return ' '.repeat(30)
}

BankLayoutDetail.prototype.rejectionCause = function() {
  return this._rejectionCause
}

BankLayoutDetail.prototype.presentationDate = function() {
  return moment().utcOffset(-300).format(definitions.layoutDateFormat)
}

BankLayoutDetail.prototype.clientReference = function() {
  // return this._clientReference // TODO: preguntar posteriormente a Joel
  return ' '.repeat(30)
}

BankLayoutDetail.prototype.paymentReferenceDescription = function() {
  var paymentReferenceDescription = this._paymentReferenceDescription + ' '.repeat(30)
```

```
  return paymentReferenceDescription.substr(0, 30)
}

BankLayoutDetail.prototype.beneficiaryEmailAddresses = function() {
  var beneficiaries = this._beneficiaryEmailAddresses.join(',') + ' '.repeat(500)

  return beneficiaries.substr(0, 500)
}

BankLayoutDetail.prototype.captureLine = function() {
  var captureLine = this._captureLine + ' '.repeat(100)

  return captureLine.substr(0, 100)
}

BankLayoutDetail.prototype.blockNumber = function() {
//   var blockNumber = '0'.repeat(6) + this._blockNumber
//
//   return blockNumber.substr(blockNumber.length - 7, 7)
  return this.sequenceNumber()
}

BankLayoutDetail.prototype.toString = function() {
  var                                                          content                                                          =
`${this.recordType()}${this.sequenceNumber()}${this.operationCode()}${this.operationCurrency()}${this.transactionDate()}${this.bankNumberCode()}${this.receptorBankNumberCode()}${this.
operationAmount()}${this._futureUseField}${this.transactionOperationType()}${this.executionDate()}${this.sourceAccountType()}${this.sourceAccountNumber()}${this.sourceAccountOwner()}
${this.sourceAccountOwnerRFC()}${this.targetAccountType()}${this.targetAccountNumber()}${this.targetAccountOwner()}${this.targetAccountOwnerRFC()}${this.senderServiceReference()}${t
his.ownerServiceName()}${this.operationIVA()}${this.ordererReferenceNumber()}${this.ordererReferenceLegend()}${this.trackingNumber()}${this.rejectionCause()}${this.presentationDate()}${t
his._anotherFutureUseField}${this.clientReference()}${this.paymentReferenceDescription()}${this.beneficiaryEmailAddresses()}${this.captureLine()}`//
  //   var   content   =   `${this.recordType()}   ${this.sequenceNumber()}   ${this.operationCode()}   ${this.operationCurrency()}   ${this.transactionDate()}   ${this.bankNumberCode()}
${this.receptorBankNumberCode()}   ${this.operationAmount()}   ${this._futureUseField}   ${this.transactionOperationType()}   ${this.executionDate()}   ${this.sourceAccountType()}
${this.sourceAccountNumber()}  ${this.sourceAccountOwner()}  ${this.sourceAccountOwnerRFC()}  ${this.targetAccountType()}  ${this.targetAccountNumber()}  ${this.targetAccountOwner()}
${this.targetAccountOwnerRFC()}  ${this.senderServiceReference()}  ${this.ownerServiceName()}  ${this.operationIVA()}  ${this.ordererReferenceLegend()}  ${this.ordererReferenceNumber()}
${this.trackingNumber()}   ${this.rejectionCause()}   ${this.presentationDate()}   ${this._anotherFutureUseField}   ${this.clientReference()}   ${this.paymentReferenceDescription()}
${this.beneficiaryEmailAddresses()} ${this.captureLine()}`

  console.log(content, ' ', content.length)

  if (content.length !== definitions.detailsLength)
   throw new Error(`Not valid details layout for payout ${this._payoutItem._id}`)

  return content
}

module.exports = BankLayoutDetail
```

Microservicio vulcano-santander-crypto

vulcano-santander-crypto   app.js

```
'use strict'
var debug = require('debug')
var express = require('express')
var logger = require('morgan')
var bodyParser = require('body-parser')
var cryptoRouter = require('./routes/crypto')
var probesRouter = require('./routes/probes')
// Package to retrieve secrets from Azure Key Vault
var akvHandler = require('@sebastian-isbit/akv_handler')
var app = express()
app.use(logger('dev'))
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: false }))
app.use('/santander', cryptoRouter)
app.use('/probes', probesRouter)
if(process.env.NODE_ENV !== 'local' && process.env.NODE_ENV !== 'test') {
 // Define Key Vault Name, and secrets name and version
 const keyVaultName = process.env.KEY_VAULT_NAME
 const mongoUriSecret = process.env.MONGO_URI_NAME + process.env.MONGO_URI_VERSION
 const accountKeySecret = process.env.ACCOUNT_KEY_NAME + process.env.ACCOUNT_KEY_VERSION
 const cryptoPassSecret = process.env.CRYPTO_PASS_NAME + process.env.CRYPTO_PASS_VERSION
 const encodedCertificateSecret = process.env.ENCODED_CERTIFICATE_NAME + process.env.ENCODED_CERTIFICATE_VERSION
 const encodedPrivateKeySecret = process.env.ENCODED_PRIVATE_KEY_NAME + process.env.ENCODED_PRIVATE_KEY_VERSION
```

```
 // Retrieve sensitive information from Azure Key Vault
 akvHandler.getSecrets(keyVaultName, [mongoUriSecret, accountKeySecret, cryptoPassSecret, encodedCertificateSecret,
encodedPrivateKeySecret])
   .then((secrets) => {
    app.set('mongoUri', secrets[mongoUriSecret])
    app.set('accountKey', secrets[accountKeySecret])
    app.set('cryptoPass', secrets[cryptoPassSecret])
    app.set('encodedCertificate', secrets[encodedCertificateSecret])
    app.set('encodedPrivateKey', secrets[encodedPrivateKeySecret])
   })
   .catch((err) => {
    console.log("Failed to get the key", err);
   })
} else if (process.env.NODE_ENV === 'local') {
 const path = require('path')
 require('dotenv').config({path: path.resolve(__dirname, './.env')})
 var localConf = require('./localConf.json')

 app.set('mongoUri', localConf.mongoUri)
 app.set('accountKey', localConf.accountKey)
 app.set('cryptoPass', localConf.cryptoPass)
 app.set('encodedCertificate', localConf.encodedCertificate)
 app.set('encodedPrivateKey', localConf.encodedPrivateKey)
}
// catch 404 and forward to error handler
app.use(function (req, res, next) {
 var err = new Error('Not Found')
 err.status = 404
 next(err)
})
// error handlers
// development error handler
// will print stacktrace
if (app.get('env') === 'development') {
 app.use(function (err, req, res, next) {
   res.status(err.status || 500)
   res.json({
     message: err.message,
     error: {}
   })
 })
}
// production error handler
// no stacktraces leaked to user
app.use(function (err, req, res, next) {
 res.status(err.status || 500)
 res.json({
   message: err.message,
   error: {}
 })
})
app.set('port', process.env.PORT || 33440)
var server = app.listen(app.get('port'), function () {
 debug('Express server listening on port ' + server.address().port)
})
```

vulcano-santander-crypto utils/encryptionManager.js

```
// Process to handle encryption module
var childProcesses = require('./childProcesses')
// Filesystem operations
var fsOperations = require('./fsOperations')
// Controllers
var controller = require('../controllers/controller')
// Database connection handler
var connectToDatabase = require('@sebastian-isbit/db-connection-handler')
```

```
// Azurefile handler
var azurefileHandler = require('@sebastian-isbit/azurefile-handler')
module.exports.run = async (action, res) => {
 var params = {}
 params.project = {'_id': false, 'in_file': true}
 switch (action) {
  case 'encrypt-layouts':
   params.model = 'layout',
   params.query = {'status': 'layoutGenerated'}
   params.status = 'layoutEncrypted'
   params.script = 'cd apicifrado && sh CmpApiCifrado.sh '
   params.sourceDirectory = process.env.GENERATED_LAYOUTS_DIR_AF
   params.destinationDirectory = process.env.ENCRYPTED_LAYOUTS_DIR_AF
   params.tempSourceDirectory = process.env.CIFRADO_ENTRADA_DIR
   params.tempDestinationDirectory = process.env.CIFRADO_SALIDA_DIR
   params.id_file = 'in_file'
   params.layoutExt = '.in2'
   params.ext = '.in2'
   break
  case 'decrypt-notifications':
   params.model = 'layout',
   params.query = {'status': 'notificationDownloaded'}
   params.status = 'notificationDecrypted'
   params.script = 'cd apicifrado && sh CmpApiCifradoDecode.sh '
   params.sourceDirectory = process.env.DOWNLOADED_NOTIFICATIONS_DIR_AF
   params.destinationDirectory = process.env.DECRYPTED_NOTIFICATIONS_DIR_AF
   params.tempSourceDirectory = process.env.DECODE_ENTRADA_DIR
   params.tempDestinationDirectory = process.env.DECODE_SALIDA_DIR
   params.id_file = 'in_file'
   params.layoutExt = '.in2'
   params.ext = '.not'
   break
  case 'decrypt-responses':
   params.model = 'layout',
   params.query = {'status': 'responseDownloaded'}
   params.status = 'responseDecrypted'
   params.script = 'cd apicifrado && sh CmpApiCifradoDecode.sh '
   params.sourceDirectory = process.env.DOWNLOADED_RESPONSES_DIR_AF
   params.destinationDirectory = process.env.DECRYPTED_RESPONSES_DIR_AF
   params.tempSourceDirectory = process.env.DECODE_ENTRADA_DIR
   params.tempDestinationDirectory = process.env.DECODE_SALIDA_DIR
   params.id_file = 'in_file'
   params.layoutExt = '.in2'
   params.ext = '.out'
   break
  case 'decrypt-statements':
   params.model = 'statement'
   params.query = {'status': 'statementDownloaded'}
   params.status = 'statementDecrypted'
   params.script = 'cd apicifrado && sh CmpApiCifradoDecode.sh '
   params.sourceDirectory = process.env.DOWNLOADED_STATEMENTS_DIR_AF
   params.destinationDirectory = process.env.DECRYPTED_STATEMENTS_DIR_AF
   params.tempSourceDirectory = process.env.DECODE_ENTRADA_DIR
   params.tempDestinationDirectory = process.env.DECODE_SALIDA_DIR
   params.id_file = 'statement_file'
   params.layoutExt = '.edocta'
   params.ext = '.edocta'
   break
  case 'decrypt-reports':
   params.model = 'report'
   params.query = {'status': 'downloaded'}
   params.status = 'decrypted'
   params.script = 'cd apicifrado && sh CmpApiCifradoDecode.sh '
   params.sourceDirectory = process.env.DOWNLOADED_REPORTS_DIR_AF
   params.destinationDirectory = process.env.DECRYPTED_REPORTS_DIR_AF
   params.tempSourceDirectory = process.env.DECODE_ENTRADA_DIR
   params.tempDestinationDirectory = process.env.DECODE_SALIDA_DIR
   params.id_file = 'report_file'
   params.layoutExt = '.txt'
   params.ext = '.txt'
```

```
    break
}
// Define responseObject var
var responseObject = {}
// Retrieve DB Connection String
var mongoUri = res.app.get('mongoUri')
try {
  // Check connection to DB, stablish a new one if needed
  await connectToDatabase(mongoUri, process.env.DB_NAME)
  // Get list of elements with the expected status
  var filesList = await controller.getList(params.query, params.project, params.model)
} catch (errorObject) {
  responseObject.err = errorObject
  // If the connection to the DB fails, send response
  return res.status(500).send(responseObject)
}
// Set counter of layouts to process
responseObject.toProcess = filesList.length
if (filesList.length === 0) {
  // If there are no elements matching query, send response
  return res.status(200).send(responseObject)
}
// Retrieve crypto password
var pass = res.app.get('cryptoPass')
try {
  // Retrieve encoded certificate contents
  var certificateName = process.env.CERTIFICATE_PATH
  var encodedCertificate = res.app.get('encodedCertificate')
  var privateKeyName = process.env.PRIVATE_KEY_PATH
  var encodedPrivateKey = res.app.get('encodedPrivateKey')
  // Generate certificates files
  await fsOperations.createCertificateFile(encodedCertificate, certificateName)
  await fsOperations.createCertificateFile(encodedPrivateKey, privateKeyName)
} catch(errorObject) {
  // If it's not possible to generate the certificates, send response
  responseObject.err = errorObject
  return res.status(500).send(responseObject)
}
// Retrieve azurefile account key
var accountKey = res.app.get('accountKey')
var storageAccount = process.env.STORAGE_ACCOUNT
var share = process.env.SHARE
try {
  var directories = [params.sourceDirectory, params.destinationDirectory]
  var fileService = await azurefileHandler.connectToAzurefile(storageAccount, accountKey, share, directories)
} catch(errorObject) {
  responseObject.err = errorObject
  return res.status(500).send(responseObject)
}
const promises = filesList.map(async(file) => {
    var result = {}
    result.id = file[params.id_file]
    var fileName = result.id.replace(params.layoutExt, params.ext)
    try {
      // Retrieve generated layout from azurefile to local file system
      await fsOperations.getFileToEncrypt(fileService, share, params.sourceDirectory, params.tempSourceDirectory, fileName)
      // Generate and execute encryption script
      var script = params.script + params.tempSourceDirectory + fileName + ' ' + params.tempDestinationDirectory + fileName + ' ' + pass
      console.log('SCRIPT: ' + JSON.stringify(script,null,2))
      await childProcesses.execShell(script)
      // Save encrypted layout to azurefile
      await fsOperations.saveEncryptedFile(fileService, share, params.destinationDirectory, params.tempDestinationDirectory, fileName)
      // Update layout status in DB
      await controller.updateStatus(file, params.status, params.model)
      result.success = true
      return result
    } catch (err) {
      // Make element of the response object
      console.log(err)
      result.success = false
```

```
      result.err = {
        reason: err.reason,
        stack: err.stack
      }
      return result
    }
  })
  responseObject.results = await Promise.all(promises)
  try {
    // Delete certificates files
    await fsOperations.deleteCertificateFile(certificateName)
    await fsOperations.deleteCertificateFile(privateKeyName)
  } catch(errorObject) {
    // Send a warning if deletion fails
    responseObject.warnings = errorObject
  }
  return res.status(200).send(responseObject)
}
```

vulcano-santander-crypto utils/fsOperations.js

```
var fs = require('fs')
// Copy a file from the azurefile to CifradoEntrada directory
module.exports.getFileToEncrypt = (fileService, share, volumeDirectory, podDirectory, fileName) => {
  return new Promise((resolve, reject) => {
    fileService.getFileToLocalFile(share, volumeDirectory, fileName, podDirectory + fileName, (err) => {
      if (!err) {
        console.log('Generated layout %s has been retrieved', fileName)
        resolve()
      } else {
        console.log({err})
        var errorObject = {
          reason: 'fsError',
          stack: err.message,
        }
        reject(errorObject)
      }
    })
  })
}
// Copy a file from CifradoSalida directory to the azurefile
module.exports.saveEncryptedFile = (fileService, share, volumeDirectory, podDirectory, fileName) => {
  return new Promise((resolve, reject) => {
    fileService.createFileFromLocalFile(share, volumeDirectory, fileName, podDirectory + fileName, (err) => {
      if (!err) {
        console.log('Encrypted layout %s has been saved', fileName)
        resolve()
      } else {
        console.log({err})
        var errorObject = {
          reason: 'fsError',
          stack: err.message,
        }
        reject(errorObject)
      }
    })
  })
}
// Generate certificate files
module.exports.createCertificateFile = (encodedCert, filePath) => {
  return new Promise((resolve, reject) => {
    var cert = new Buffer(encodedCert, 'base64')
    fs.writeFile(filePath, cert, (err) => {
      if(!err) {
        console.log('Certificate %s has been generated', filePath)
        resolve()
```

```
    } else {
      var errorObject = {
        reason: 'certificateError',
        stack: err.message
      }
      reject(errorObject)
    }
  })
 })
}
// Delete certificate files
module.exports.deleteCertificateFile = (filePath) => {
 return new Promise((resolve, reject) => {
  fs.unlink(filePath, (err) => {
    if (!err) {
      console.log('Certificate %s has been deleted', filePath)
      resolve()
    } else {
      var errorObject = {
        reason: 'certificateError',
        stack: err.message
      }
      reject(errorObject)
    }
  })
 })
}
```

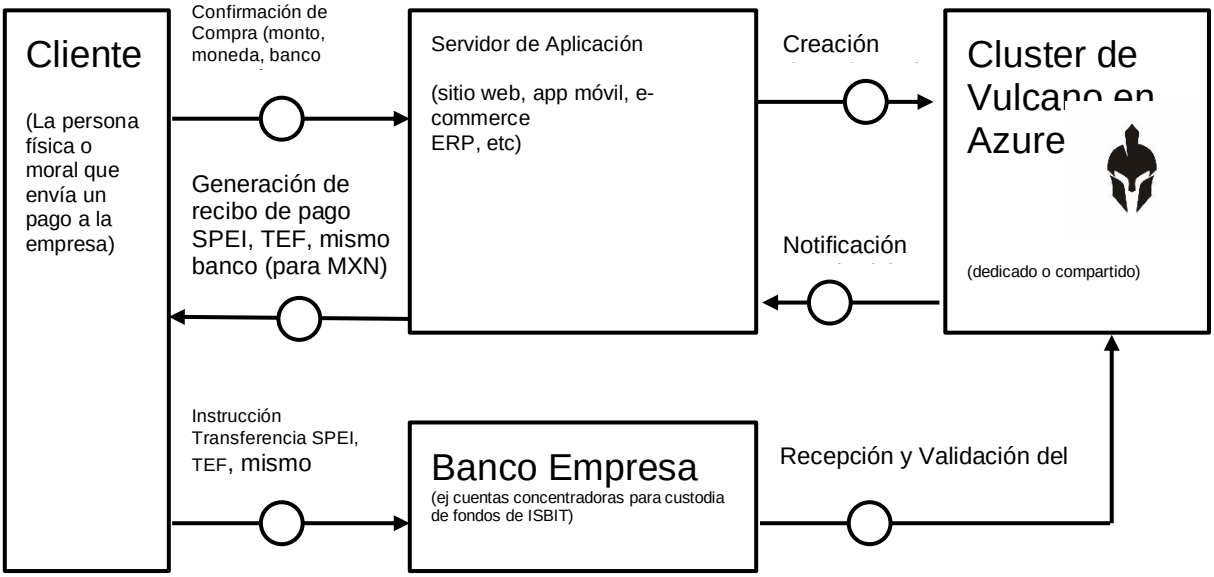
Anexo E. Guía de Integración de VULCANO

A lo largo de este anexo, todos los ejemplos programables están en gema para Ruby.

Cobros en Banco

El objetivo de esta guía es explicar paso a paso como generar referencias de pago de un banco, mediante las cuales los clientes podrán realizar transferencias bancarias a través de una transferencia del mismo banco, TEF o a través de SPEI para otros bancos en el caso de MXN y SPID en caso de USD.

Flujo para realizar cargos en banco



Pasos:

1. El cliente confirma la compra en tu sitio web seleccionando como medio de pago Banco.
2. Desde tu servidor de aplicación se crea una referencia creando un cargo en **VULCANO.**
3. Con los datos obtenidos al crear la referencia se crea un recibo de pago personalizado.
4. El cliente realiza el pago en el portal de su banco.
5. **VULCANO** válida y recibe el pago.
6. **VULCANO** notifica la recepción del pago a tu servidor.

En esta guía veremos los pasos número 2 y 3, para el paso número 6 consulta la sección de notificaciones

Crear cargo; paso 2

Para poder recibir un pago a través de un banco es necesario crear un cargo indicando en el campo `method` el tipo `bank_account` de la siguiente manera:

```
@vulcano=VulcanoApi.new("mzdtln0bmtms6o3kck8f","sk_e568c42a6c384b7ab02cd47d2e407cab")
@charges=@vulcano.create(:charges)
request_hash={
   "method" => "bank_account",
   "amount" => 200.00,
```

```
  "description" => "Cargo con banco",
  "order_id" => "oid-00051"
 }

response_hash=@charges.create(request_hash.to_hash)
```

Al momento de crearse el cargo se regresará una objeto transacción.

Respuesta:


```
{
 "id" : "t6utz9dywve6zipnppys",
 "description" : "Cargo con banco",
 "error_message" : null,
 "authorization" : null,
 "amount" : 100,
 "operation_type" : "in",
 "payment_method" : {
  "type" : "bank_transfer",
  "bank" : "Santander",
  "agreement" : "1411217",
  "clabe" : "00000000000000000000",
  "name" : "11094690394055678934"
 },
 "order_id" : "oid-00051",
 "transaction_type" : "charge",
 "creation_date" : "2013-12-05T17:50:09-06:00",
 "currency" : "MXN",
 "status" : "in_progress",
 "method" : "bank_account"
}
```


Con esta información puedes generar una ficha de depósito a tu cliente para que realice un pago mediante una transferencia interbancaria vía SPEI, TEF o mismo banco cuando la variable currency es igual a “MXN”, y en el caso en que el valor de currency es “USD” transferencia mismo banco o SPID.

Para más información consulta la referencia de cargos

Recibo de pago para el cliente; paso 3

El recibo de pago para el cliente deberá de incluir la siguiente información:

1. Nombre de la empresa
   - El nombre de tu empresa

2. Fecha límite de pago

○ El día límite que tu cliente tiene para pagar. El tiempo máximo permitido por Vulcano es de 90 días.

3. Monto a pagar

○ El monto exacto a pagar por tu cliente.

4. Instrucciones de pago

○ El recibo debe incluir las instrucciones de pago tanto para pagar desde el mismo banco con el que tu empresa tiene un convenio “Host to Host” como para pagar desde otros bancos, de acuerdo con las siguientes secciones

Instrucciones de pago desde el mismo banco con el que se tiene convenio “Host to Host” (transferencia mismo banco).

1. El cliente deberá ingresar a su banca en línea y dentro del menú seleccione la cuenta de cargo que se va a usar en la divisa correcta y que coincida con la divisa en la ficha generada por VULCANO.
2. Seleccionar la opción de “transferencia entre cuentas mismo banco”
3. Ingrese los datos de registro para concluir con la operación:

○ Referencia

○ Monto exacto a pagar

○ Concepto

Instrucciones de pago para otros bancos (transacción interbancaria)

1. El proceso conceptualmente es el mismo, pero con ligeras diferencias si la ficha se generó para recibir dinero a una cuenta de USD vía SPID o para recibir dinero a una cuenta de MXN vía SPEI o TEF. La cantidad enviada deberá ser en la divisa en la que esté denominada la ficha y desde una cuenta en la misma divisa que la indicada en la ficha generada por VULCANO.
2. Deberá registrar la cuenta beneficiaria del pago con los siguientes datos:

○ Nombre del banco destino (el banco con el que se establece contrato H2H)

○ Número de cuenta CLABE

○ Nombre de beneficiario

3. Ingresar a la sección de transferencias o pagos a terceros y proporcionar los datos de la transferencia, monto y concepto del pago.
   - Monto exacto a pagar
   - En el concepto de pago colocar la referencia numérica de 20 dígitos

Recibo de pago genérico de VULCANO

Ejemplo del recibo de pago

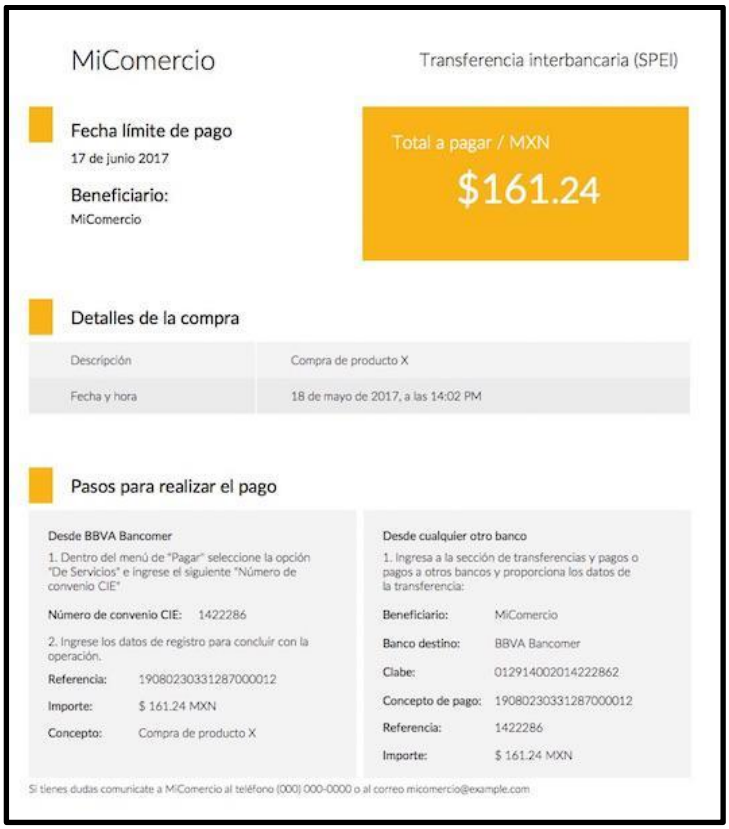


Ilustración 56 Ejemplo del recibo de pago

VULCANO brinda la posibilidad de obtener un recibo de pago genérico con la información necesaria para que tu cliente pueda realizar la transferencia correctamente, para llegar a él basta con estructurar una ruta compuesta de la siguiente manera:

{DASHBOARD_PATH}/spei-pdf/{MERCHANT_ID}/{TRANSACTION_ID}

Sandbox: {DASHBOARD_PATH} = https://<id_suscriptor>-vulcano-sandbox.eastus.cloudapp.azure.com/dashboard

Producción: {DASHBOARD_PATH} = https://<id_suscriptor>-vulcano-prod.eastus.cloudapp.azure.com/dashboard

{MERCHANT_ID} = tu id de comerciante

{TRANSACTION_ID} = valor del campo id del objeto transacción regresado al crearse el cargo

Ejemplo:

https://<id_suscriptor>-vulcano-sandbox.eastus.cloudapp.azure.com/mzdtlirufa4oosnkfa3qwo/t6utshgtovmw49dmzwq

Notas:

El PDF del recibo estará disponible únicamente si la transacción se encuentra en estado pendiente. Una vez que el cliente haya realizado el pago o la transacción se haya cancelado el recibo ya no se mostrará.

- Asegúrate que tu integración cumple con los requisitos de compatibilidad de versiones más detalles
- Implementa las Notificaciones para conocer el estado de los pagos en tiempo real

Notificaciones

¿Qué es un Webhook?

Los Webhooks te permiten recibir notificaciones de las transacciones realizadas a algún servicio web que tu configures en tu cuenta. Esto te permite saber, por ejemplo, cuando se ha realizado un cargo a una tarjeta, si un pago que enviaste fue liquidado exitosamente o cuando un depósito se ha realizado con éxito.

Nota: Si deseas manejar los webhooks vía API, consulta la referencia aquí

Objeto Webhook

Cada vez que se realice una transacción, VULCANO enviará un objeto JSON a las URL's registradas para recibir webhooks. VULCANO puede agregar más campos en un futuro, o agregar nuevos valores a

campos existentes, por lo que es recomendado que tu Webhook pueda manejar datos adicionales desconocidos.

Parámetros

El objeto webhook contiene los siguientes datos:

type	string El tipo de evento que generó la notificación
event_date	timestamp Fecha de creación del evento en formato ISO 8601
transaction	string El objeto de Transacción relacionado con el evento. No enviado en notificaciones de tipo verification.
verification_code	string El código de verificación del Webhook. Enviado solamente en notificaciones de tipo verification.

Todas las notificaciones de las transacciones se enviarán a las URL's que tengas registradas. Para distinguir una transacción, utiliza la propiedad `type`

Nota: VULCANO intentará entregar la notificación hasta recibir una respuesta de éxito. Esto puede causar que algunas notificaciones se envíen dos veces, por lo que debes estar preparado para poder recibir la misma notificación más de una vez.

Ejemplo

```
{
   "type" : "charge.succeeded",
```

```
  "event_date" : "2013-11-22T15:09:38-06:00",
  "transaction" : {
    "amount" : 2000.0,
    "authorization" : "801585",
    "method" : "card",
    "operation_type" : "in",
    "transaction_type" : "charge",
    "card" : {
      "type" : "debit",
      "brand" : "mastercard",
      "address" : {
        "line1" : "Calle #1 Interior #2",
        "line2" : null,
        "line3" : null,
        "state" : "Queretaro",
        "city" : "Queretaro",
        "postal_code" : "76040",
        "country_code" : "MX"
      },
      "card_number" : "1881",
      "holder_name" : "Card Holder",
      "expiration_month" : "10",
      "expiration_year" : "14",
      "allows_charges" : true,
      "allows_payouts" : true,
      "creation_date" : "2013-11-22T15:09:32-06:00",
      "bank_name" : "Santander",
      "bank_code" : "012"
    },
    "status" : "completed",
    "id" : "tlxcm4vprtz74qoenuay",
    "creation_date" : "2013-11-22T15:09:33-06:00",
    "description" : "Description",
    "error_message" : null,
    "order_id" : "oid_14235"
  }
}
```


Características de un servicio Webhook válido

- Endpoint: Solo dominios (No IPs). ejemplo: https://notifications.merchant.com
- Puerto: 443/TCP, 1518/TCP, 1519/TCP, 8443/TCP y 10443/TCP
- Protocolo: HTTPS/TLS_1.2
- Certificado: Válido (firmado por CA pública y match con dominio).

Tipos

verification	La notificación contiene el código de verificación del Webhook
charge.created	Se creó un cargo para ser pagado por transferencia bancaria.
charge.succeeded	Indica que el cargo se ha completado (tarjeta, banco o tienda)
charge.refunded	Un cargo a tarjeta fue reembolsado.
payout.created	Se ha programado un pago.
payout.succeeded	Se ha enviado el pago.
payout.failed	El pago fue rechazado.
transfer.succeeded	Se ha realizado una transferencia entre dos clientes.
fee.succeeded	Se ha cobrado una comisión a un cliente.
spei.received	Se han agregado fondos a una cuenta mediante SPEI.
chargeback.created	Se ha recibido un contracargo de un cargo a tarjeta
chargeback.rejected	El contracargo se ha ganado a favor del comercio
chargeback.accepted	El contracargo se ha perdido. Se ha creado una transacción tipo contracargo que descontará los fondos de tu cuenta.

Tu servidor debe estar preparado para aceptar notificaciones de tipos no conocidos, para asegurar una compatibilidad con futuras versiones.

Registro

Para configurar un Webhook sigue los siguientes pasos:

1. Entra al Dashboard de VULCANO utilizando tu correo y tu contraseña
2. Da click en tu nombre para acceder a tu perfil de comercio.
3. En la sección de Webhooks, selecciona la opción +Agregar Webhook.
4. En el formulario que aparece, indica la URL completa de tu webhook, incluyendo el protocolo a usar. Es recomendado utilizar https.
5. Si tu Webhook requiere autenticación HTTP, configurala. Actualmente solo se soporta autenticación HTTP Basic.
6. Da click en el botón Guardar.

Verificación

Al terminar la configuración de registro, **VULCANO** enviará mediante POST un mensaje JSON a la URL indicada, conteniendo un objeto de notificación Webhook. Tu servicio deberá guardar el código de verificación de alguna manera, y regresar el estado 200 OK.

Si por alguna razón requieres que se te envíe de nuevo el código de verificación, selecciona la opción Reenviar Código. Un nuevo código de verificación será generado y enviado a la URL proporcionada.

Una vez que ya tengas el código de verificación del Webhook, selecciona la opción de Verificar e introduce el código proporcionado en el objeto de la notificación. Esto activará el Webhook en **VULCANO**, y empezarás a recibir notificaciones de las transacciones realizadas a partir de ese momento.

Ejemplo:


```
{
  "type" : "verification",
  "event_date" : "2013-11-22T11:04:49-06:00",
  "verification_code" : "UY1qqrxw"
```

}

Implementación

Para implementar tu Webhook, solo tienes que crear un servicio Web con una URL a la que **VULCANO** pueda enviar peticiones `POST`.

Tu Webhook debe manejar los diferentes tipos de notificaciones, incluyendo recibir el código de verificación, para que puedas darlo de alta en VULCANO.

Tu Webhook también deberá poder recibir tipos de notificaciones inesperados, para asegurar compatibilidad con futuras versiones.

Los Webhook deberán regresar un estado `HTTP 200 OK` siempre que reciban una notificación, de otra manera **VULCANO** reintentará el envío continuamente.

Eliminación

En cualquier momento puedes seleccionar la opción -Eliminar desde el dashboard para eliminar un Webhook y dejar de recibir notificaciones a esa URL.

Anexo F. Documentación de interfaz de programación de Vulcano

La interfaz está contenida en un drive, esta cuenta con los siguientes apartados;

API Endpoints, Autenticación, Errores, Cargos, Cargo a Banco, Listado de Cargos, Confirmar Cargo, Pagos o Retiros, Pago a cuenta bancaria, Cancelar un pago, Listado de Pagos, Resumen de Pagos, Objeto Resumen de Pagos, Detalle de Pagos, Clientes, Crear un nuevo cliente, Actualizar un cliente, Obtener un cliente existente, Eliminar un cliente, Listado de clientes, Cuentas Bancarias, Crear una cuenta bancaria, Obtener una cuenta bancaria, Eliminar una cuenta bancaria, Listado de Cuentas Bancarias, Comisiones, Cobra una comisión a la cuenta de un cliente, Devolver Comisión, Webhooks, Objeto Webhoo, Crear Webhook, Obtener un Webhook, Eliminar un, Webhook, Listado de Webhook, Objetos Comunes, Objeto Transacción, Objeto Dirección, Objeto Geolocation, Objeto Transaction Status.

Link del drive y código QR:
https://docs.google.com/document/d/e/2PACX-1vSPdzax9gAeJyroyeY5dMgdSJX0Sv7IVkG4cxzIuCQiNbXX6bJAemg-k7vQ75oC2s8w_SdEa8q48Cw_/pub

Anexo G. Estadísticas sobre transaccionalidad

A continuación, se muestran las gráficas de los retiros, transacciones y número de depósitos y retiros mes con mes a partir del 2016 hasta abril del 2021.

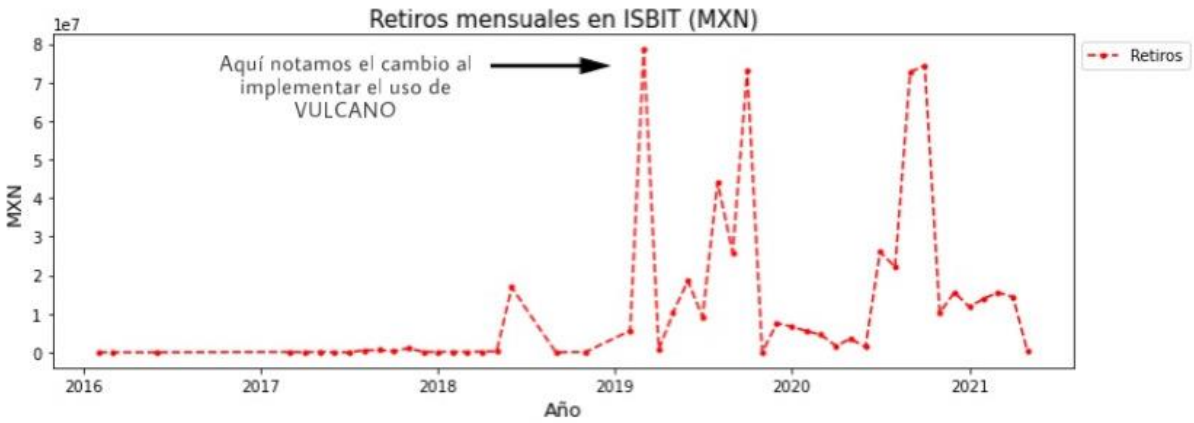


Ilustración 57 Retiros mensuales en ISBIT (MXN

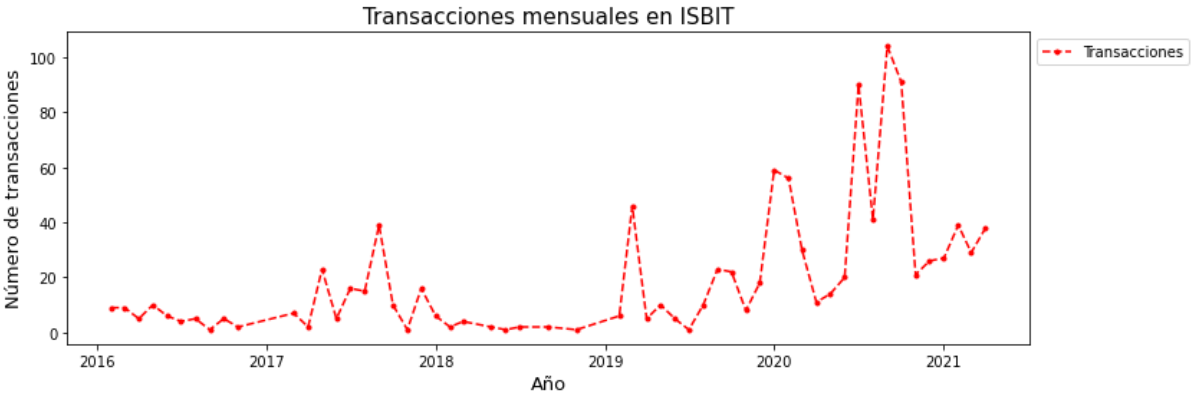


Ilustración 58 Transacciones mensuales en ISBIT

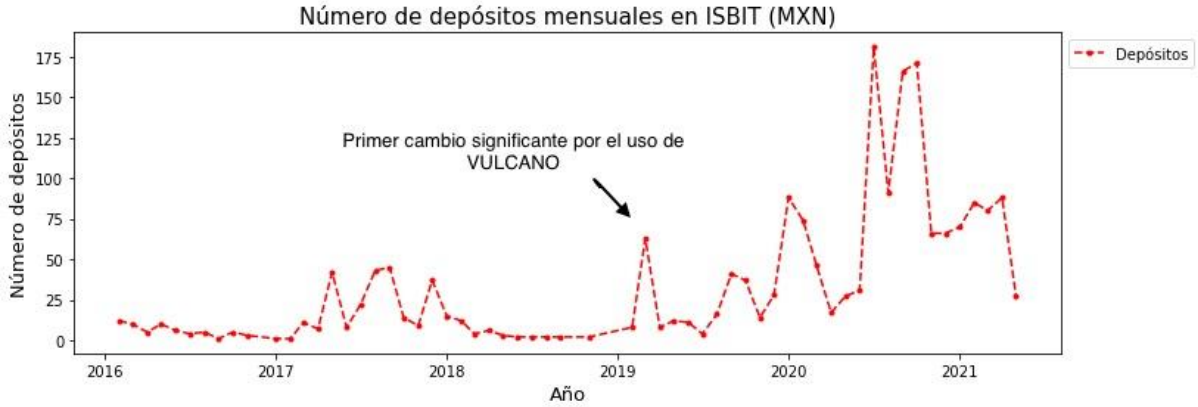


Ilustración 59 Número de depósitos mensuales en ISBIT (MXN)

Ilustración 60 Número de retiros mensuales en ISBIT (MXN).

Consultas:

Consultas utilizadas para generar conjunto de datos a partir del cual se realizaron graficas sobre transaccionalidad de esta sección.

```
-Retiros
/*
Tabla: withdraws
Se usó la columna de created_at para obtener los meses y años.
Se usó la columna de amount para la suma mensual.
Consulta:
SELECT DATE_PART('month',created_at)::INT AS month,
DATE_PART('year',created_at)::INT AS year,
SUM(amount) as Suma
FROM withdraws
GROUP BY month, year
ORDER BY year, month;
*/
--Depósitos
```

```
/*
Tabla: deposits
Se usó la columna de created_at para obtener los meses y años.
Se usó la columna de amount para la suma mensual.
Consulta:
SELECT DATE_PART('month',created_at)::INT AS month,
DATE_PART('year',created_at)::INT AS year,
SUM(amount) as Suma
FROM deposits
GROUP BY month, year
ORDER BY year, month;
*/
--Intercambios
/*
Tabla: trades
Se usó la columna de created_at para obtener los meses y años.
Se usó la columna de volume para la suma mensual.
Consulta:
SELECT DATE_PART('month',created_at)::INT AS month,
DATE_PART('year',created_at)::INT AS year,
SUM(volume) as Suma
FROM trades
GROUP BY month, year
ORDER BY year, month;
*/
--Facturas
/*
Tabla: invoices
Se usó la columna de statement_period_start para obtener los meses y años.
Se usó la columna de total_fee para la suma mensual.
Consultas:
SELECT DATE_PART('month',statement_period_start)::INT AS month,
DATE_PART('year',statement_period_start)::INT AS year,
SUM(total_fee) as Suma
FROM invoices
GROUP BY month, year
ORDER BY year, month;
*/
-No. de depósitos por mes
/*
submitting
```

```
cancelled
accepted
SELECT DATE_PART('month',created_at)::INT AS month,
DATE_PART('year',created_at)::INT AS year,
COUNT(id) as Suma
FROM deposits
WHERE aasm_state = 'accepted'
GROUP BY month, year
ORDER BY year, month;
*/
Suma de retiros por mes
/*
SELECT DATE_PART('month',created_at)::INT AS month,
DATE_PART('year',created_at)::INT AS year,
SUM(amount) as Suma
FROM withdraws WHERE currency = 1
AND aasm_state = 'done'
GROUP BY month, year
ORDER BY year, month;
```

Anexo H. Información Legal

En la institución donde se realizó la **práctica profesional** que dio origen al presente **TRABAJO PROFESIONAL**, la sociedad denominada **"ISBIT SA DE CV"**, Sebastian Acosta Checa tiene los cargos de Director General y de Administrador Único, desde el 3 de Octubre de 2013 como lo acredita la escritura pública **13,974** otorgada ante fe del notario público RICARDO CUEVAS MIGUEL, Titular de la notaría **210** del Distrito Federal.

ISBIT SA DE CV es una persona moral constituida por la escritura número **13,974** de fecha **3 de octubre de 2013**., otorgada ante fe del notario público RICARDO CUEVAS MIGUEL, Titular de la notaría **210** del Distrito Federal.

ISBIT SA DE CV se encuentra registrado en el Registro Federal de Contribuyentes con el código **IIN131003RTA.**

Mediante ASAMBLEA GENERAL EXTRAORDINARIA DE ACCIONISTAS DEL 22 DE NOVIEMBRE DE 2016 se *acuerda por unanimidad cambiar la denominación de la sociedad de "**INTELHIT INTELSELLER**" por el nombre de "**ISBIT**", para lo cual se obtuvo previamente*

la autorización de uso por parte de la Dirección General de Normatividad Mercantil de la Secretaría de Economía marcada con la clave única de documento: ***A20161117132211524****, quedando el mismo tipo societario de SOCIEDAD ANÓNIMA DE CAPITAL VARIABLE.*

Por instrumento número **90,605 del libro 1358,** de fecha 9 de enero de 2017., otorgada ante fe del notario público **MAURICIO GÁLVEZ MUÑOZ**, Titular de la notaría **39** del Distrito Federal, se protocolizó testimonio de la escritura del **CAMBIO DE DENOMINACIÓN** de **"INTELHIT INTELSELLER"**, SOCIEDAD ANÓNIMA DE CAPITAL VARIABLE POR LA DE **"ISBIT"** SOCIEDAD ANÓNIMA DE CAPITAL VARIABLE y la reforma al Artículo primero de los estatutos sociales. Que el testimonio de Cambio de Denominación anterior fue inscrito en el Registro Público de la Propiedad de la Propiedad y el Comercio de la Ciudad de México bajo el Folio Mercantil Electrónico Número **505059-1**

La razón por la que **VULCANO** se presenta como una solución automatizar tareas de auditoría tiene que ver con la alta carga de trabajo que el Cumplimiento Regulatorio en materia de prevención de lavado de dinero supone para la empresa. **ISBIT** SA DE CV se encuentra legalmente obligada a presentar los ***Avisos*** de operaciones de compraventa de Activos Virtuales ya que está registrada como sujeto obligado en el padrón de ***Actividades Vulnerables*** como se acredita con el siguiente documento:

FOLIO: 214643

HACIENDA

SECRETARÍA DE HACIENDA Y CRÉDITO PUBLICO
REGISTRO DE ACTIVIDADES VULNERABLES. ALTA

02/04/2020

I. Datos de Identificación (PM)

RFC:	IIN131003RTA
DENOMINACIÓN O RAZÓN SOCIAL:	ISBIT
FECHA DE CONSTITUCIÓN:	03/10/2013
PAÍS DE CONSTITUCIÓN:	ESTADOS UNIDOS MEXICANOS
NACIONALIDAD:	ESTADOS UNIDOS MEXICANOS

Las acciones representativas de su capital o títulos que representan dicho capital, cotizan en bolsa? SI

II. Datos de contacto:

Clave Lada: 55 **Telefóno:** 55734992 **Celular:** 5514780943 **Correo electrónico:** activaciones@isbit.co

Clave Lada: 55 **Telefóno:** 55734992 **Celular:** 5514780943 **Correo electrónico:** compliance@isbit.co

Clave Lada: 55 **Telefóno:** 55734992 **Celular:** 5514780943 **Correo electrónico:** s@isbit.co

III. Actividades Vulnerables realizadas:

CLAVE ACTIVIDAD	ACTIVIDAD VULNERABLE REALIZADA	FECHA PRIMERA OPERACION
21	OPERACIONES CON ACTIVOS VIRTUALES	03/02/2020

IV. Avisos por Actividades Vulnerables realizadas:

TIPO DE AVISO A PRESENTAR	VENCIMIENTO	FUNDAMENTO	FECHA ALTA
Avisos. Operaciones con Activos Virtuales	A más tardar el día 17 del mes inmediato siguiente a aquél en que se llevó a cabo la operación	Ley Federal para la Prevención e Identificación de Operaciones con Recursos de Procedencia Ilícita: Artículo 17, Fracción XVI	03/02/2020

Por este conducto acepté que la Secretaría de Hacienda y Crédito Público, la Unidad de Inteligencia Financiera o el Servicio de Administración Tributaria, lleven a cabo las notificaciones que correspondan a través de este medio electrónico, en términos de lo señalado en el artículo 6 de las Reglas de Carácter General a que se refiere la Ley Federal para la Prevención e Identificación de Operaciones con Recursos de Procedencia Ilícita.

Asimismo, acepté acceder a estos medios electrónicos de manera periódica y acusar la recepción de las notificaciones correspondientes, ya que reconozco que las notificaciones que se realicen a través de estos medios surtirán efectos al momento de acusar su recepción o a los cinco días hábiles siguientes contados a partir de que la autoridad remita las notificaciones y las haga disponibles a través de este medio electrónico, lo anterior, de conformidad con lo establecido en el artículo 36 de las Reglas de Carácter General antes referidas.

Declaro bajo protesta de decir verdad y conocedor de las penas en que incurren las personas que declaran con falsedad en los términos de lo dispuesto por el artículo 247, fracción I del Código Penal Federal, que todos los datos asentados en esta forma oficial son veridicos y exactos.

Sello Digital de la FIEL:
WHl4Qlg4RS9wMmhJTDB3NktsMkE0VDRPV0pUSlZETi9lY3lGRlREUyt5YXZDckpUcHdyVnZ1MVQ4dDB3WnZ
tMzBJa2dXVmVXVFZQdEdhTzFZOENnNm03SFl3Nm55OGdmQlc5VFlOZXJEK0xuSWlGbFdSa0tXZFJDbWhm
TlJrU1dWMG1TcjNmcCttVkFMRHozZzhYaTNtV3J3NUp2a1I4ZE1VSXMxa1FCYmtQd29rbG9aT1B4R0ZLQnhsb
GtGOURybzlCTE03SXBWR2YwRjRkTEhZamdPRW1mbDVPeHR1VU91QWs4OVFDS1RDYjdZM2tTL2NGQ3B
pWjlmRTlNbVJsZUNNmp5ay9FSmtib0ZDNFpySHd0OHV6QXBiY29PajVtaUd5enFPWGxKaGxBRy9DSFFiYXBj
c3E3TFhHVW1laFFQQnB6Z29zMTJWKzFSSzFrcnF5ZmxnPT0=

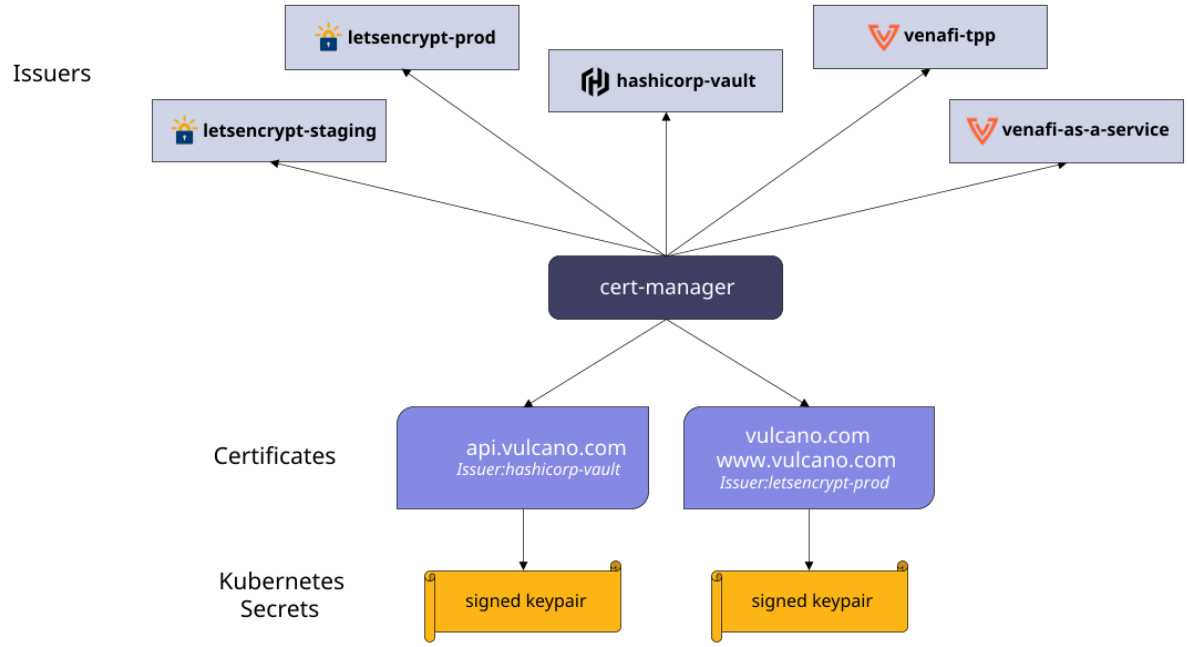
Sello Digital SAT:
SRo/UPxfBM2vK+CVROLhWyi91gBqegXjzFF0NHtSx7ubF+wjaIfHyRrwtm46xvQnS4M3hY8Xk3UDnmkhvyE+we
OlsgBM4SQyz3OS5Y1RNPiCtaQkT+Pj7V/PVZPq+t9fSksRtKG8BeNNYDIOyzZdrGWCIhD7Orejh8XKKdxzl4g=

Ilustración 61 Hacienda; Actividades Vulnerables

Anexo I. Procedimiento para encriptar comunicaciones API JSON con HTTPS mediante *letsencrypt*.

Uso de certificados con LetsEncrypt.org en Application Gateway para clúster AKS de VULCANO

En esta sección se configura su instancia de AKS para aprovechar LetsEncrypt.org y obtener automáticamente un certificado TLS/SSL para el dominio. El certificado se instalará en Application Gateway, que realizará la terminación de SSL/TLS para el clúster de AKS utilizado por VULCANO. La configuración que se describe aquí usa el complemento cert-manager de Kubernetes, que automatiza la creación y la administración de certificados. Es importante mencionar que por motivos de seguridad y confidencialidad en esta sección se remplazo el verdadero domino y endpoint DNS donde se ofrecen los servicios web por "vulcano.com" como un simple marcador de posición. Es decir **vulcano.com** no es un dominio "real" controlado o administrado por ISBIT SA DE CV.



La imagen de arriba fue basada en información del portal del projecto de código abierto "cert-mánager" que puede ser encontrada en https://cert-manager.io/docs .

En este anexo, a modo de documentación, se presentan los pasos para instalar cert-manager en el clúster de VULCANO en un AKS existente.

Gráfico de Helm

Se ejecuta el siguiente script para instalar el gráfico de Helm ("Helm chart") de cert-manager. De este modo:

- Se creará un nuevo espacio de nombres denominado cert-manager en la instancia de AKS utilizada por VULCANO
- Se crean las siguientes ***CRDs*** (Definiciones de Recursos Costumizados):
    0. ***Certificate*** (Certificado de seguridad TLS/SSL),
    1. ***Challenge*** (Desafío para validar identidad servidor),
    2. ***ClusterIssuer*** (Emisor de certificados para Clúster),
    3. ***Issuer*** (Emisor certificados para espacio de nombres),
    4. ***Orden*** (una Orden encapsula varios desafíos ACME)
- Instalación del gráfico de ***cert-manager*** (desde [docs.cert-manager.io)](docs.cert-manager.io)

Rutina de Bash para instalar el grafico de Helm *cert-manager*

```
#!/bin/bash

# (EN)Install the CustomResourceDefinition resources separately
# (ES)Instalar la Definición de Recursos Costumizados (CRD). Es una dependencia necesaria
kubectl apply -f https://raw.githubusercontent.com/jetstack/cert-manager/release-0.14/deploy/manifests/00-crds.yaml

# (EN)Create the namespace for cert-manager
# (ES)Crear espacio de nombres para cert-manager
            kubectl create namespace cert-manager

            # (EN)Label the cert-manager namespace to disable resource validation
            # (ES)Etiquetar el espacio de nombres correspondiente a cert-manager para deshabilitar validación de recursos

            kubectl label namespace cert-manager certmanager.k8s.io/disable-validation=true

            # (EN)Add the Jetstack Helm repository
            # (ES)Agregar el Repositorio mantenido por la organización Jetstack y todos los gráficos de Helm disponibles ahí

            helm repo add jetstack https://charts.jetstack.io

            # (EN)Update your local Helm chart repository cache
            # (ES)Actualizar la memoria temporal de acceso rápido del repositorio de graficas de Helm local

            helm repo update

            # (EN)Install the cert-manager Helm chart
            # (ES)Instalar la grafica de Helm correspondiente a cert-manager

            # Helm v3+
            helm install \
              cert-manager jetstack/cert-manager \
              --namespace cert-manager \
              --version v1.0.4 \
              # --set installCRDs=true

            # Helm v2
            helm install \
              --name cert-manager \
```

```
  --namespace cert-manager \
  --version v1.0.4 \
  jetstack/cert-manager \
  # --set installCRDs=true

#To automatically install and manage the CRDs as part of your Helm
release,
#   you must add the --set installCRDs=true flag to your Helm
installation command.

#Para instalar y administrar de manera automatica Definiciones de
Recursos Costumizados (CRDs) como parte del lanzamiento de un nuevo
grafico de Helm,
#   es necesario agregar el switch --set installCRDs=true al comando
de instalación de Helm.
```

Recurso ClusterIssuer

Crear un recurso ClusterIssuer. cert-manager lo necesita para representar a la entidad de certificación Lets Encrypt en la que se obtendrán los certificados firmados.

Mediante el uso del recurso **ClusterIssuer** sin espacios de nombres, **cert-manager** emitirá certificados que se pueden consumir desde varios espacios de nombres. **Let's Encrypt** usa el protocolo **ACME** para comprobar que controla un nombre de dominio determinado y para emitirle un certificado. Más información sobre la configuración de las propiedades de ClusterIssuer puede ser encontrada aquí: https://cert-manager.io/docs/configuration/ . **ClusterIssuer** indicará a cert-manager que emita certificados mediante el entorno de ensayo de Lets Encrypt que se usa para las pruebas (el certificado raíz no está presente en los almacenes de confianza del explorador o del cliente).

El tipo de desafío predeterminado en el siguiente YAML es **http01**. Otros desafíos están documentados en Tipos de desafíos de lets Encrypt (Internet Security Group, 2020).

Archivo YAML de configuración de Emisor de Certificados del Clúster

```
#!/bin/bash
kubectl apply -f - <<EOF
apiVersion: certmanager.k8s.io/v1alpha1
kind: ClusterIssuer
metadata:
name: letsencrypt-staging
spec:
acme:
    # You must replace this email address with your own.
    # Let's Encrypt will use this to contact you about
expiring
```

```
    # certificates, and issues related to your account.
    email: h2h-aks@vulcano.com>
    # ACME server URL for Let's Encrypt's staging
environment.
    # The staging environment will not issue trusted
certificates but is
    # used to ensure that the verification process is
working properly
    # before moving to production
    server: https://acme-staging-
v02.api.letsencrypt.org/directory
    privateKeySecretRef:
    # Secret resource used to store the account's
private key.
    name: vulcano-issuer-account-key
    # Enable the HTTP-01 challenge provider
    # you prove ownership of a domain by ensuring that a
particular
    # file is present at the domain
    http01: {}
EOF
```


Cree un recurso de entrada para exponer la aplicación guestbook mediante la instancia de Application Gateway con el certificado de Lets Encrypt.

Asegúrese de que su instancia de Application Gateway tiene una configuración de IP pública de front-end con un nombre DNS (ya sea mediante el dominio de azure.com predeterminado o aprovisione un servicio de Azure DNS Zone y asigne su propio dominio personalizado). Observe la anotación certmanager.k8s.io/cluster-issuer: letsencrypt-staging, que indica a cert-manager que procese el recurso de entrada etiquetado.

Importante

El YAML siguiente es un ejemplo de rutina en BASH para crear un recurso Ingress en el clúster que interactue con certificados de cert-manager de prueba para un entorno de calidad (“stage”) y ligar la protección criptográfica al dominio VULCANO.

Script de bash para Crear recurso ingress y ligar al certificado

```
#!/bin/bash
kubectl apply -f - <<EOF
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
name: vulcano-letsencrypt-staging
annotations:
```

```
    kubernetes.io/ingress.class: azure/application-
gateway
    certmanager.k8s.io/cluster-issuer: letsencrypt-
staging
spec:
tls:
- hosts:
    - vulcano.com
    secretName: vulcano-secret-name
rules:
- host: vulcano.com
    http:
    paths:
    - backend:
        serviceName: frontend
        servicePort: 80
EOF
```

Después de unos segundos, se puede acceder al servicio de ***vulcano*** con la dirección URL HTTPS de *Application Gateway* mediante el certificado de **almacenamiento provisional** *Lets Encrypt* emitido automáticamente. Es posible que el explorador web advierta de una entidad de certificación no válida. `CN=Fake LE Intermediate X1` emite el certificado de almacenamiento provisional. Esto indica que el sistema funcionó según lo esperado y está listo para el certificado de producción.

Certificado de producción

Una vez que el certificado de ensayo se haya configurado correctamente, se procede a cambiar a un servidor de producción ACME:

- Se reemplaza la anotación de almacenamiento provisional en el recurso de entrada con: `certmanager.k8s.io/cluster-issuer: letsencrypt-prod`
- Se elimina el `ClusterIssuer` de almacenamiento provisional existente que creó en el paso anterior y cree uno nuevo; para ello, sustituya el servidor ACME del YAML del ClusterIssuer anterior por `https://acme-v02.api.letsencrypt.org/directory`

Expiración y renovación del certificado

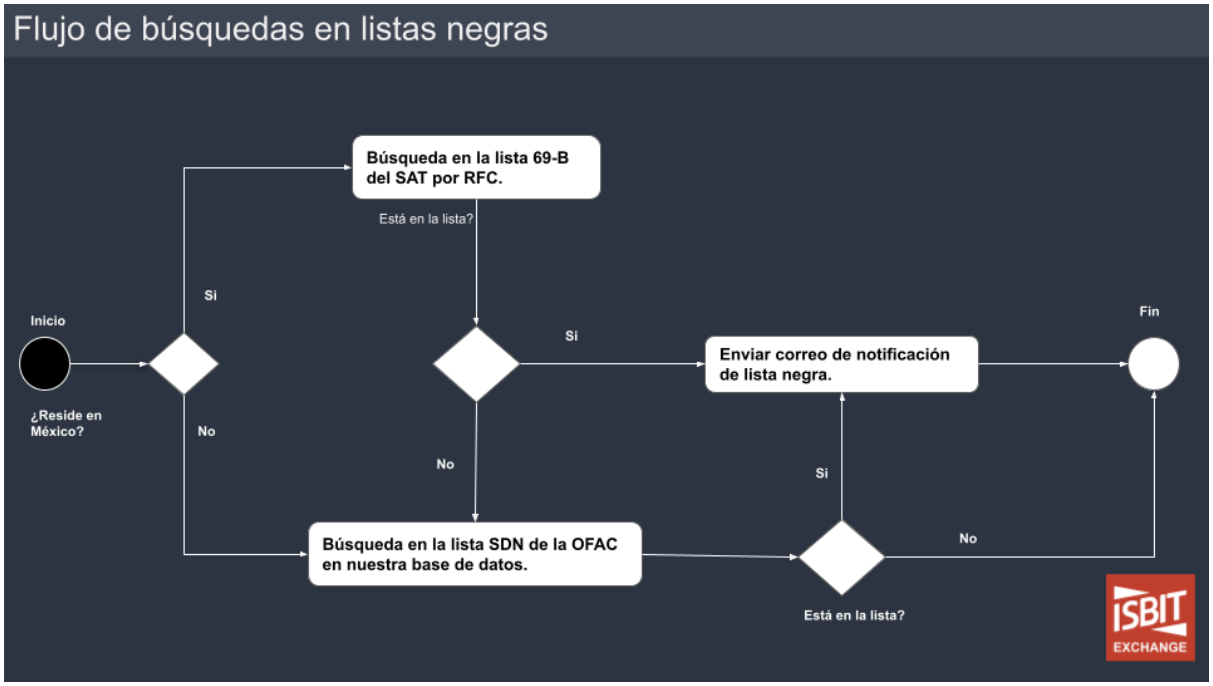
Antes de que expire el certificado de `Lets Encrypt`, `cert-manager` lo actualizará automáticamente en el almacén de secretos de Kubernetes. En ese punto, el controlador de entrada de Application Gateway aplicará el secreto actualizado al que se hace referencia en los recursos de

entrada que está usando para configurar la instancia de ***Application Gateway***.

Anexo J. Presentación Técnica de VULCANO

https://isbit.exchange/vulcano/unam/trabajo_profesional_presentacion.pdf

Anexo K. Prácticas de Prevención de Lavado de Dinero vigentes en la organización ISBIT



La Presentación completa con un resumen de las prácticas de Prevención de Lavado y Financiamiento del Terrorismo implmentadas dentro de la organización puede ser encontrada en el siguiente enlace:

https://s3.eu-west-1.amazonaws.com/isbit.exchange/vulcano/unam/guia-PLD-FT.pdf

https://isbit.exchange/vulcano/unam/guia-PLD-FT.pdf

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

LICENCIATURA EN ACTUARÍA; FACULTAD DE CIENCIAS

Análisis, diseño e implementación de VULCANO: Sistema de Gestión de Tesorería Automatizado en la empresa del sector financiero ISBIT SA DE CV

TRABAJO PROFESIONAL

PARA OBTENER EL GRADO ACADÉMICO DE:

Licenciado en **Actuaría**

PRESENTA

Sebastián Acosta Checa

Dirigido por: Dr. Humberto Andrés Carrillo Calvet

Antecedentes

ISBIT SA de CV es una empresa que fue fundada en el año 2013 por Sebastian Acosta Checa. En el 2016, la institución desarrolló una plataforma de intercambio de Activos Virtuales.

Debido a la demanda de usuarios, el volumen y el número de transacciones de la empresa aumentó, al grado de que era muy difícil llevar el control manual de las operaciones, por lo que fue necesario buscar una solución que automatiza los procesos de tesorería de la empresa, tales como: cobranza, dispersiones, auditoría, cambios de divisas.

ISBIT EXCHANGE

Introducción

VULCANO

A partir de la necesidad de automatizar y optimizar la tesorería de la empresa ISBIT, nació la idea de crear un software denominado “VULCANO”.

Con VULCANO se busca cumplir con los objetivos de hacer más eficaz el funcionamiento de ISBIT aumentando la seguridad de tesorería de la empresa y la velocidad de las operaciones, maximizar la transparencia, así como acrecentar la redundancia y facilitar la instalación, mantenimiento y evolución futura del sistema.

Se pretende explicar a fondo cuáles fueron las bases y las herramientas usadas para lograrlo, así como la metodología y los recursos empleados, como los microservicios, los contenedores, aplicaciones, librerías, mitigación de riesgos de ciberseguridad, el diseño de interfaces y de servicio web, además de los resultados y las conclusiones obtenidas.

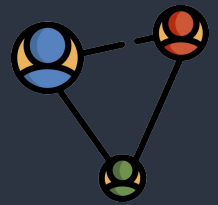
¿Qué es VULCANO?

Es un software que resuelve las ineficiencias y riesgos asociados al manejo de tesorería y sus transacciones realizadas como cobranza, recibo y envío de pagos de empresas asociadas al sector FinTech, e-commerce, PyMEs, entre otras.

Automatización de cobranza, dispersiones

Brinda información en tiempo real accesible mediante una de base de datos indexada para facilitar tareas de auditoria y cumplimineto

Rompe con la necesidad de intermediarios



AUTOMATIZACIÓN DE TRANSACCIONES DE DISTINTOS TIPOS EN LA MISMA API

Sistemas de transferencia de fondos

SPEI

Cuentas en MXN

¿Quién usaría SPEI con VULCANO?

Empresas con un alto volumen y/o frecuencia de transacciones. Por ejemplo:

- Empresas de comercio electrónico
- Financieras (prestamos por internet)
- Fintech
- Aseguradoras
- Agregadores de pago

SPID

Cuentas en USD

¿Quién usaría SPID con VULCANO?

Empresas como:

- Relacionadas con comercio exterior
- Fondos de inversión internacionales
- Arbitraje financiero

SWIFT

Cuentas en otras Divisas

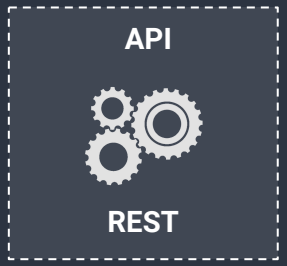
¿Quién usaría SWIFT con VULCANO?

- Personas morales nacionales e internacionales.
- Importadores y Exportadores
- Transmisores de dinero
- Fondos de cobertura y comerciantes
- Gobierno
- Plataformas de activos virtuales

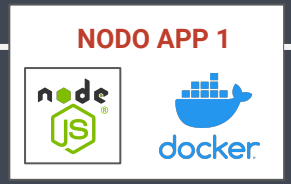
Arquitectura

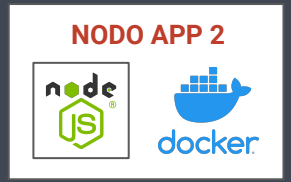
Horizontal del sistema para escalar y crecer

Incrementa NODOS para atender a más clientes, sin cuellos de botella

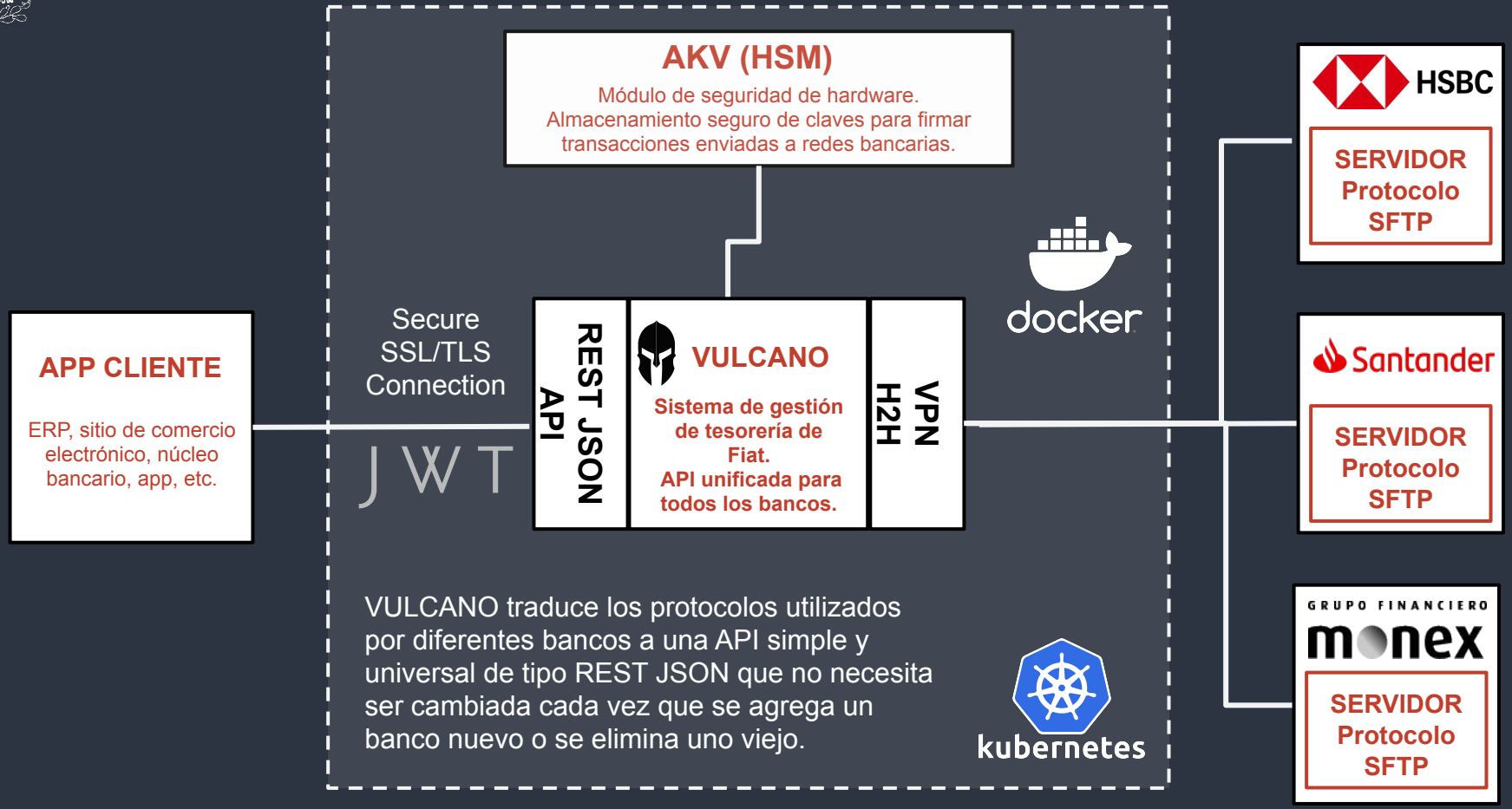


En sólo 10 min. aprovisiona e implementa un nuevo nodO del AKS





WAF OWASP


AKV (HSM)
Módulo de seguridad de hardware.
Almacenamiento seguro de claves para firmar transacciones enviadas a redes bancarias.
APP CLIENTE
ERP, sitio de comercio electrónico, núcleo bancario, app, etc.
Secure SSL/TLS Connection
JWT
REST JSON API
VULCANO
Sistema de gestión de tesorería de Fiat.
API unificada para todos los bancos.
VPN H2H
docker
HSBC
SERVIDOR Protocolo SFTP
Santander
SERVIDOR Protocolo SFTP
GRUPO FINANCIERO monex
SERVIDOR Protocolo SFTP
VULCANO traduce los protocolos utilizados por diferentes bancos a una API simple y universal de tipo REST JSON que no necesita ser cambiada cada vez que se agrega un banco nuevo o se elimina uno viejo.
kubernetes

JWT

Mecanismo para poder acceder a la aplicación

Está compuesto por tres secciones

1

Header (encabezado)

JSON que contiene el tipo de token y el algoritmo de firma que se utiliza

```
base64enc ({
    “alg”; “HS256”,
    “typ”: “jwt”
})
```

2

Payload (carga útil)

JSON que contiene información referente a la persona que se está autenticando

```
base64enc ({
    “iss”; “isbit.co”,
    “exp”: “1578589600”,
    “company”: “ISBIT”,
    “awesome”: true,
   “user”: “4125876”
})
```

- Registradas
- Públicas
- Privadas

```
HMACSHA256 (
   base64enc(header),
   +’.’+
   base64enc(payload)
   ,secretKey)
```

3

Verify signature (firma)

Se toma el header y payload codificados, una llave privada, el algoritmo especificado en el encabezado y se firma

Resultado JWT

eyKJdfNGkslElKibHfOpdmspD7C938H.kd9HD9Alcckd8CNKICjv8sInco7HDK.6pFjdnfrAN4cKJSco9mJhGr7aNe

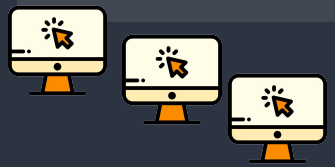
Administra quién tiene acceso a los recursos de Azure, qué pueden hacer con esos recursos y a qué áreas tienen acceso

Microsoft Azure

Nos ayuda a

Que un usuario administre máquinas virtuales en una suscripción y otro usuario administre redes virtuales

Un usuario puede administrar todos los recursos en un grupo de recursos, como máquinas virtuales, sitios web y subredes.



Que una aplicación acceda a todos los recursos de un grupo de recursos

Permitir que un grupo de DBA administre bases de datos SQL en una suscripción

Función de RBAC en VULCANO

El sistema soporta múltiples tipos de usuarios:

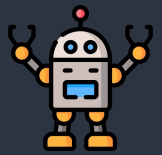
Usuario 1

Perfil 1

Permisos:
enviar transferencias de las cuentas x1, x2, x3

Interfaz de usuario gráfica **GUI**

Tienen acceso a



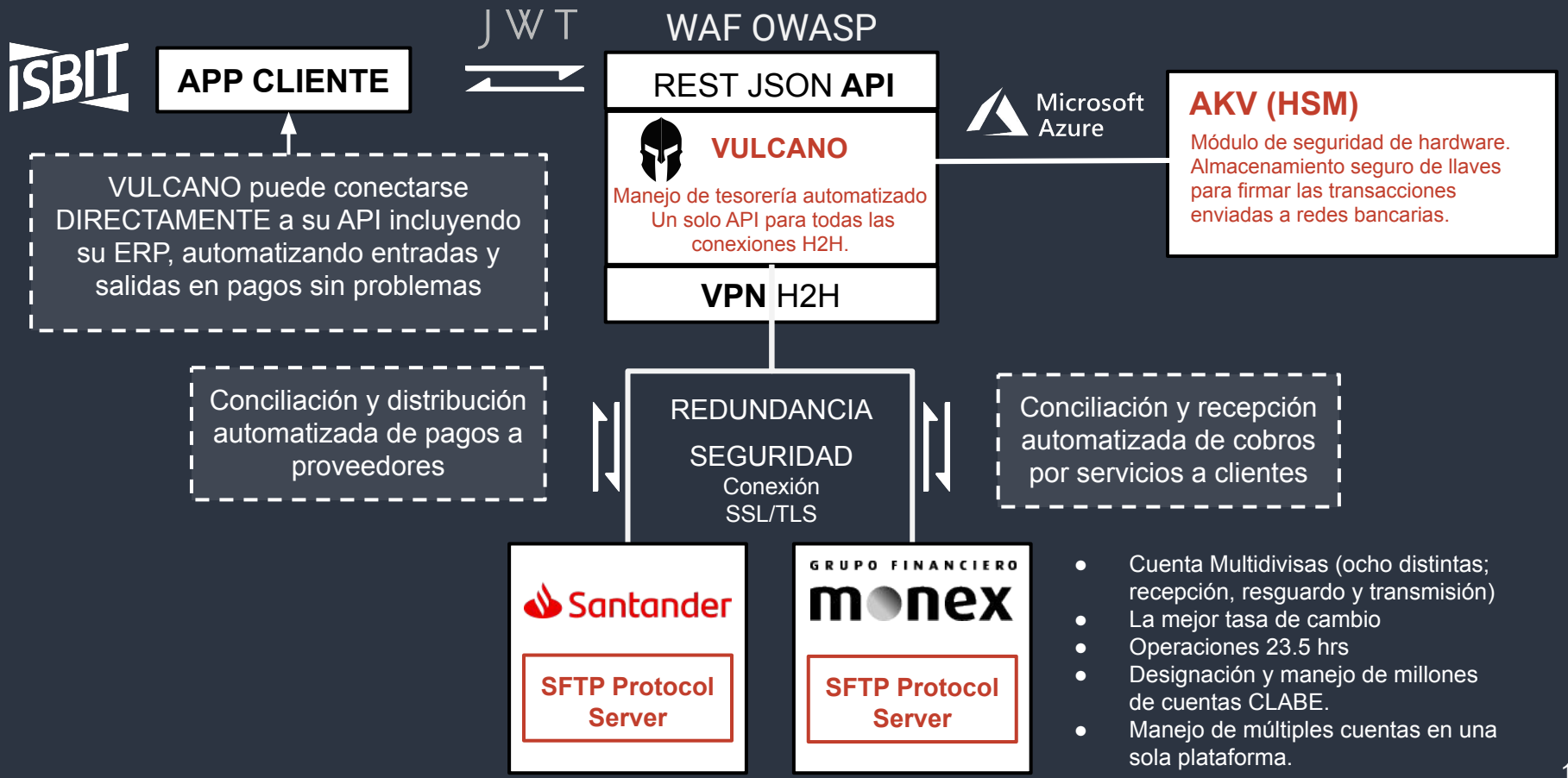
Usuario 2

Perfil 2

Permisos:
leer información contable pero no puede leer transferencias de las cuentas y1, y2, y3

Interfaz de programación de aplicaciones **API**

Cada tipo de usuario asume un rol correspondiente a un perfil, en donde quizá en un sistema puedan editar información y en otro solo puedan leerla



- Cuenta Multidivisas (ocho distintas; recepción, resguardo y transmisión)
- La mejor tasa de cambio
- Operaciones 23.5 hrs
- Designación y manejo de millones de cuentas CLABE.
- Manejo de múltiples cuentas en una sola plataforma.

Proceso de conciliación de cobros automatizado

Cliente

La persona física o moral que envía un pago a la empresa

Confirmación de Compra (monto, moneda, banco receptor)

1

Servidor de aplicación

Aplicación web o móvil. En el caso de ISBIT EXCHANGE, esta aplicación incluye la interfaz gráfica donde se genera una ficha de depósito/cobro y el usuario la puede imprimir.

2

Creación de Referencia

Servidor de VULCANO

Este servidor ofrece el API REST consumida por el servidor de aplicación (e.j ISBIT EXCHANGE, METIS, CAERUS, ERP, etc)

VULCANO

6

Notificación estado del cobro

Generación de recibo de pago SPEI, TEF, mismo banco (para MXN) o SPID (en caso de USD)

3

4

Instrucción Transferencia SPEI, TEF, mismo banco o SPID

Banco de la Empresa

Los fondos de clientes custodiados en cuentas concentradoras

ISBIT EXCHANGE

5

Recepción y Validación del Pago

Validación de Identidad Bancaria

APP CLIENTE

ERP, sitio de comercio electrónico, núcleo bancario, app, etc.

1 Petición con:
- CLABE Interbancaria
- Número de CELULAR
- Número de tarjeta de DÉBITO

VULCANO

Respuesta con: **2**

- RFC del receptor
- Nombre real del receptor
- Nombre del banco receptor
- Plaza del banco receptor

Microsoft Azure

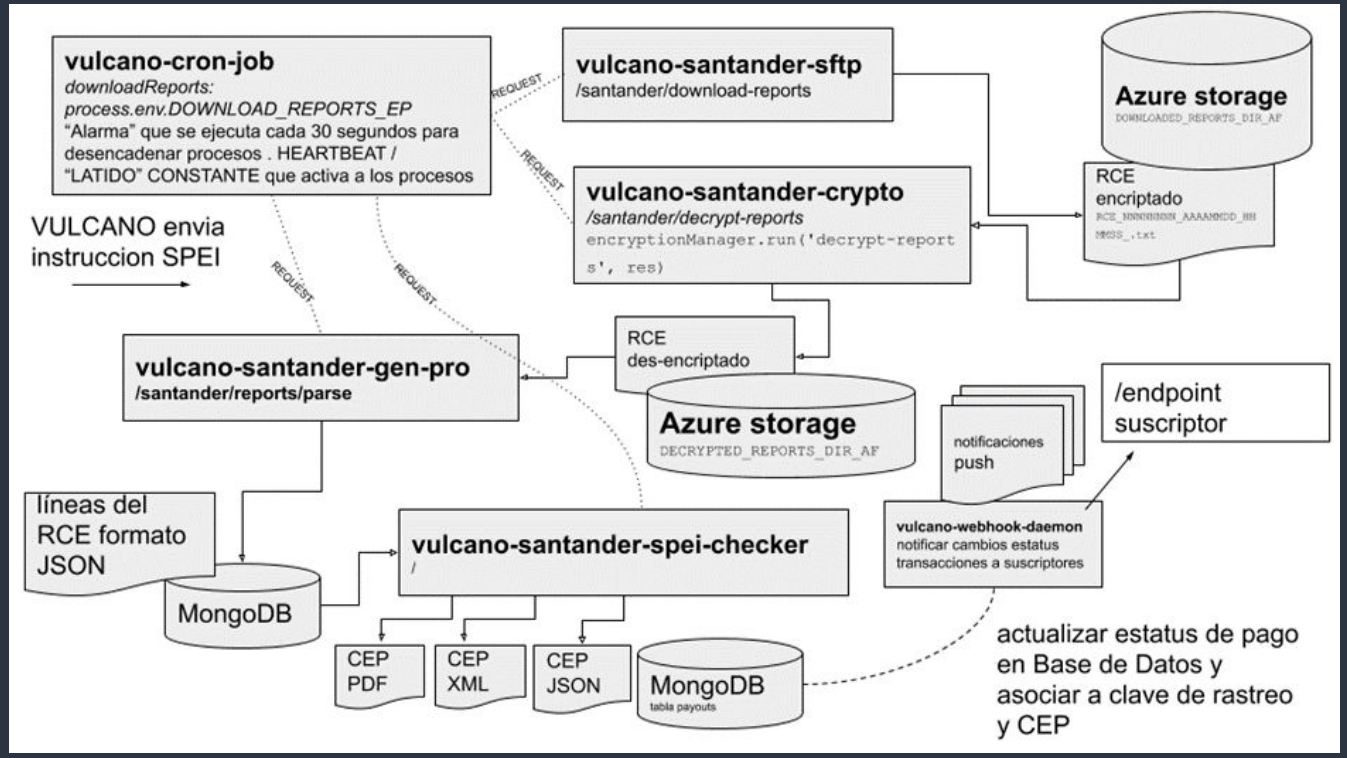
Azure Cosmos mongoDB

3

VALIDACIÓN

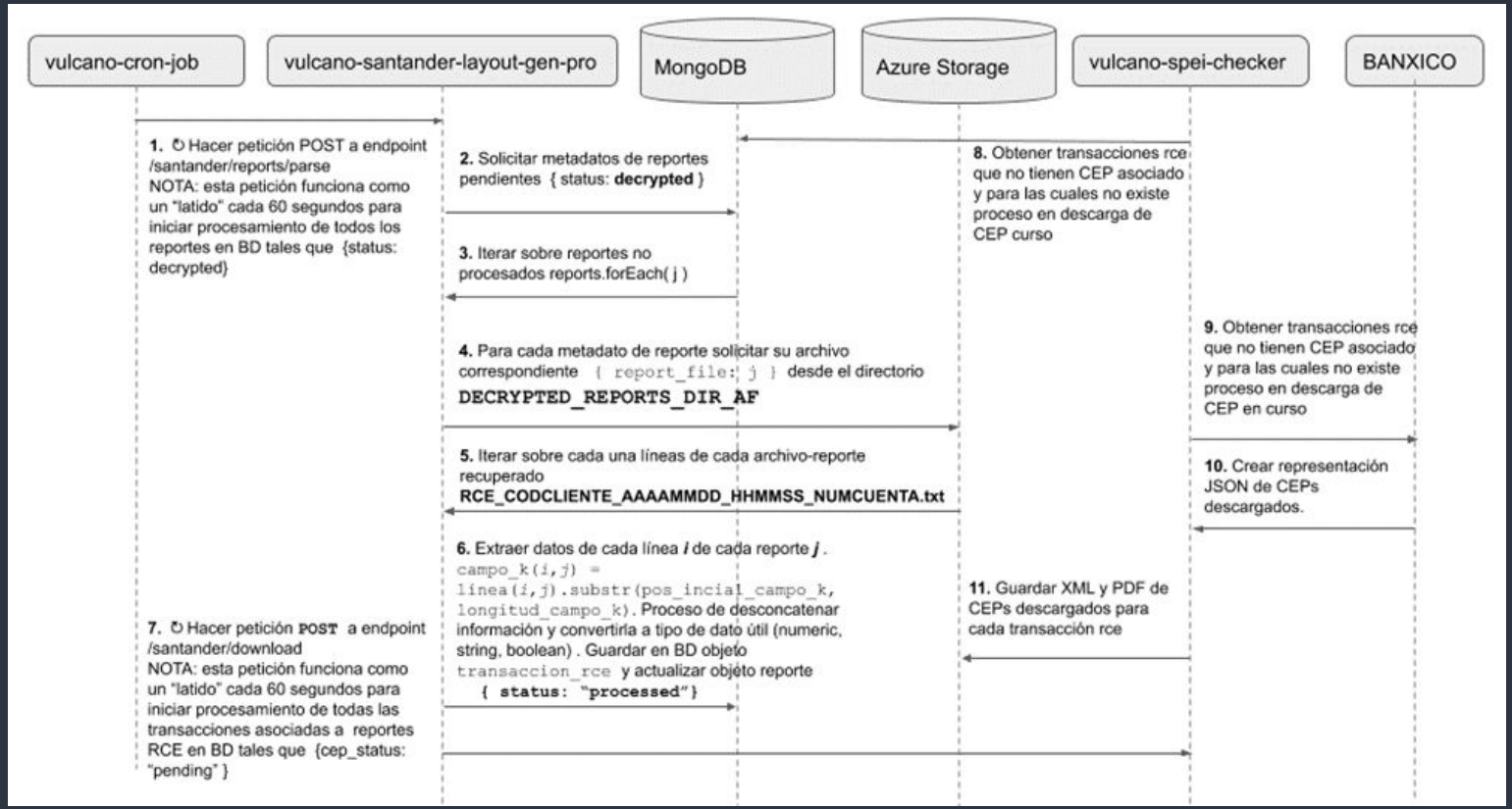
Identidad BENEFICIARIO = Identidad CUENTA BANCARIA ?

Validación de Identidad Bancaria



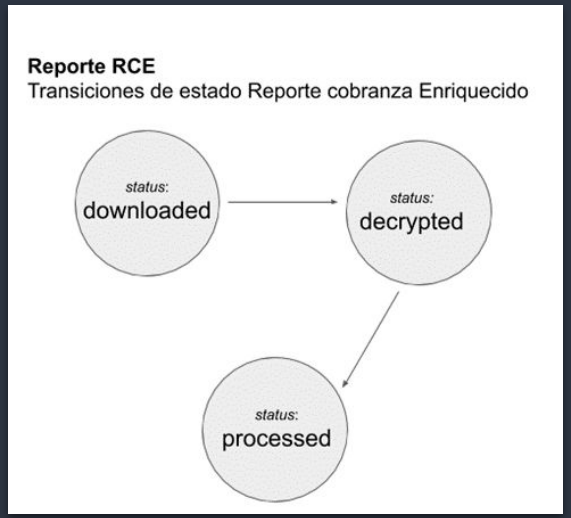
Actualizar estatus de pago y asociar clave de rastreo y CEP a objeto "payout"

Validación de Identidad Bancaria

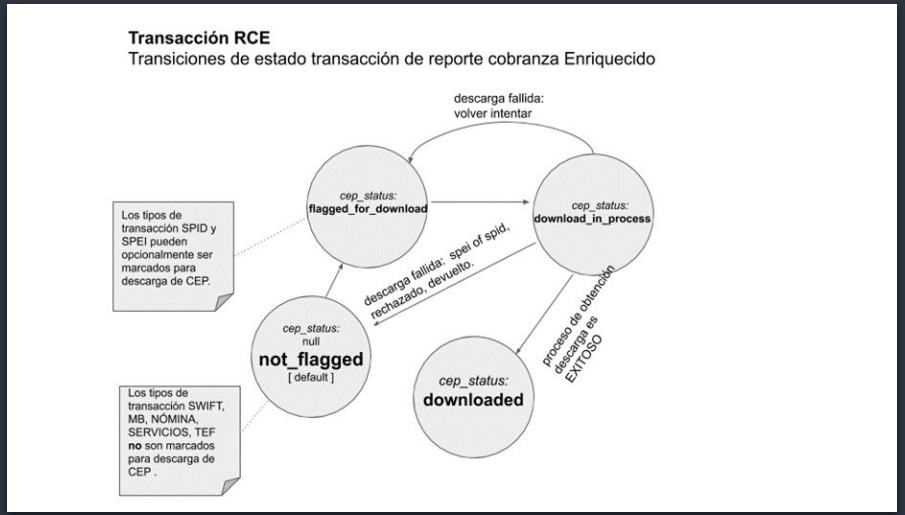


Pasos a seguir para obtención de CEP

Validación de Identidad Bancaria



Transiciones de Estado RCE



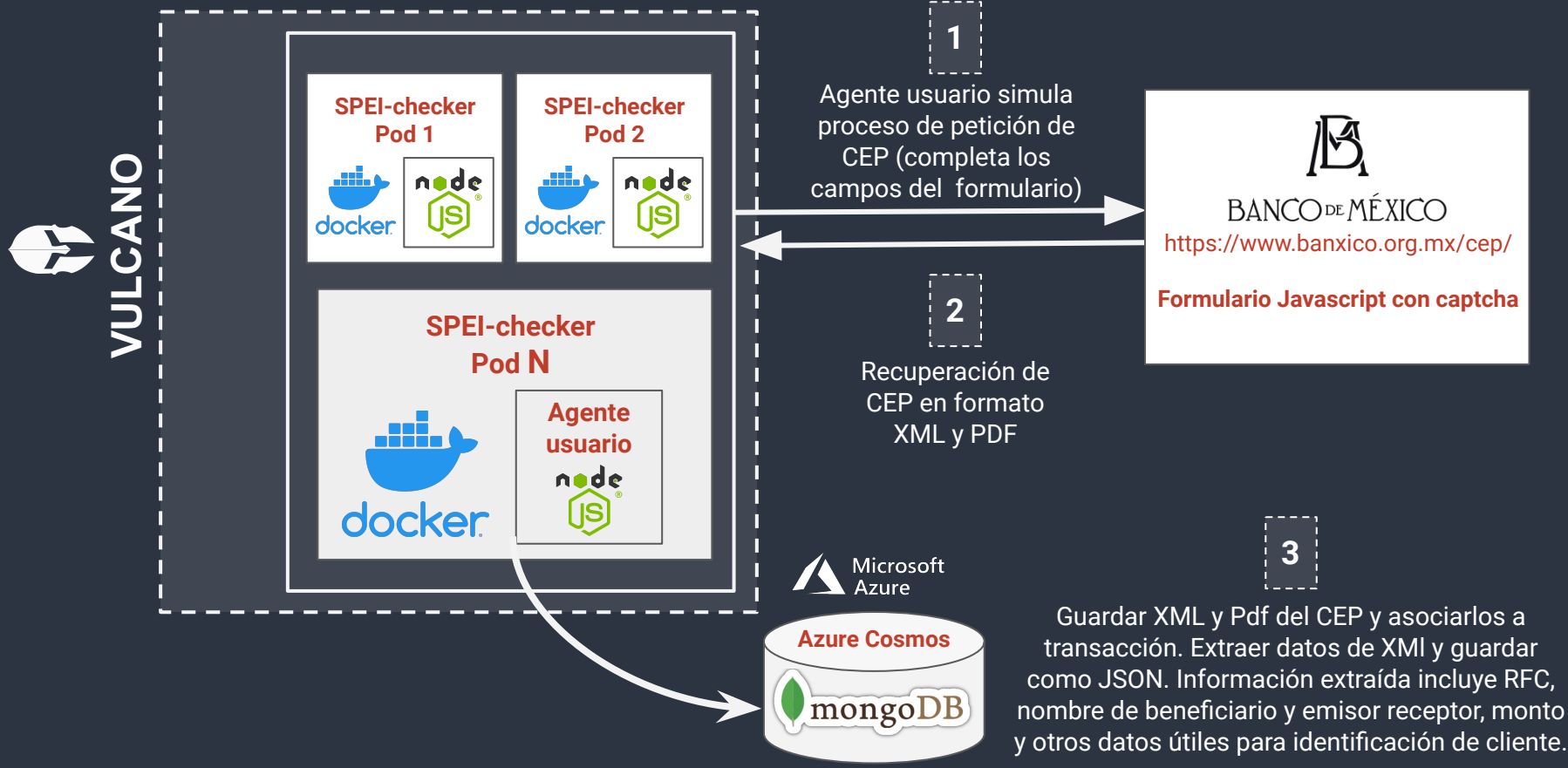
Transiciones de Estado Transacción de RCE

ISBIT EXCHANGE

VULCANO

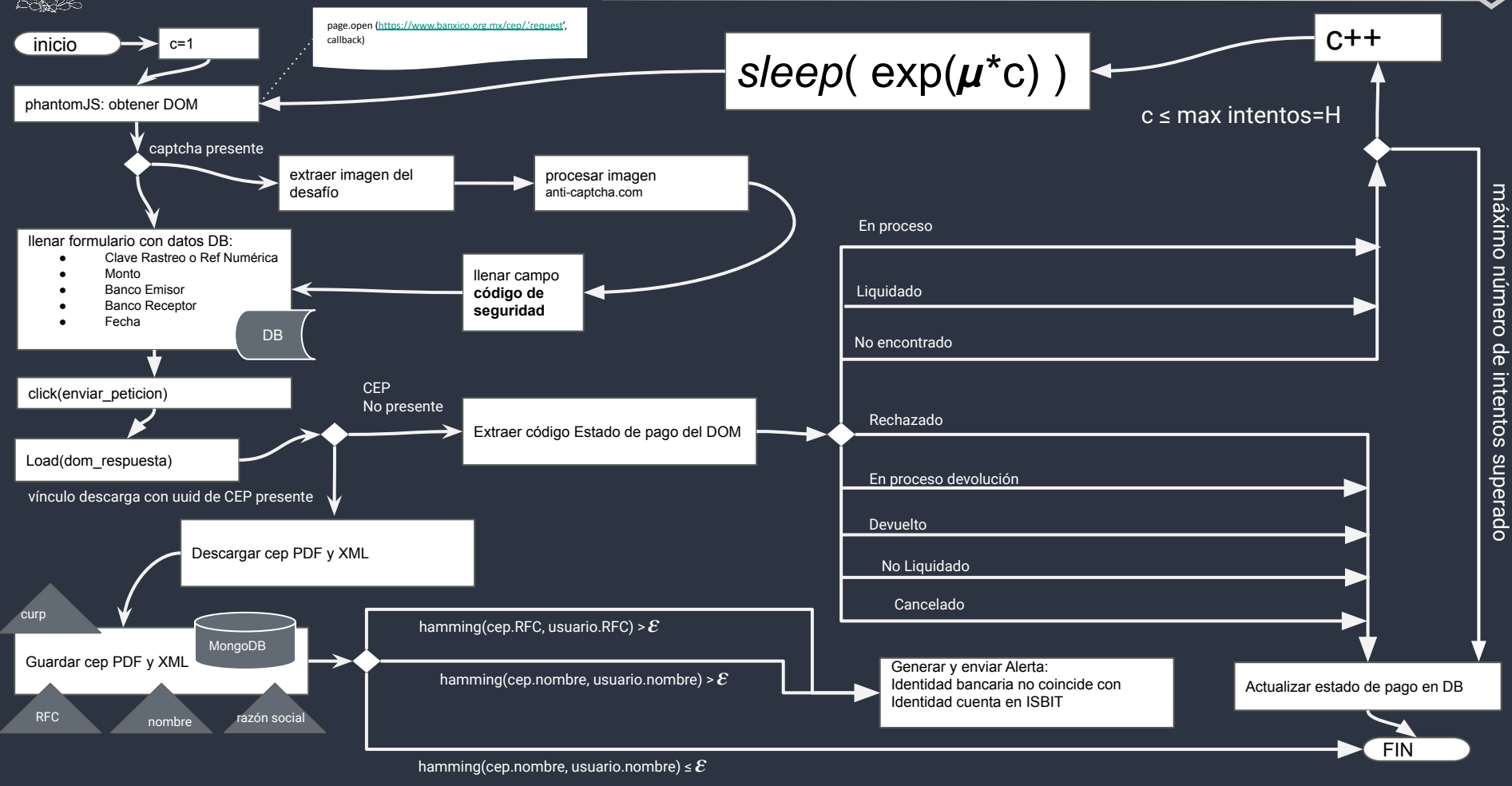
Validación de Identidad Bancaria

Procesamiento automático de recuperación de Comprobantes Electrónicos de Pago



Validación de Identidad Bancaria SPEI-checker

Procesamiento automático de recuperación de Comprobantes Electrónicos de Pago



Distribución Tiempo Espera obtención de CEP

Función de densidad Log-Cauchy:

$$f(x;\mu,\,\sigma) = \frac{1}{x\pi}\left[\frac{\sigma}{(\ln(x)-\mu)^2+\sigma^2}\right] \qquad , x > 0$$

Parámetro de ubicación estimados utilizando muestra de ***243*** transacciones SPEI realizadas en 2018:

$$\hat{\mu} = \begin{cases} \frac{\ln\left(t_{(n+1)/2}\right)+\ln\left(t_{\left(\frac{n+1}{2}\right)+1}\right)}{2} & n\,par \\ \ln\left(t_{(n+1)/2}\right) & n\,non \end{cases} = 1.9797\,segundos$$

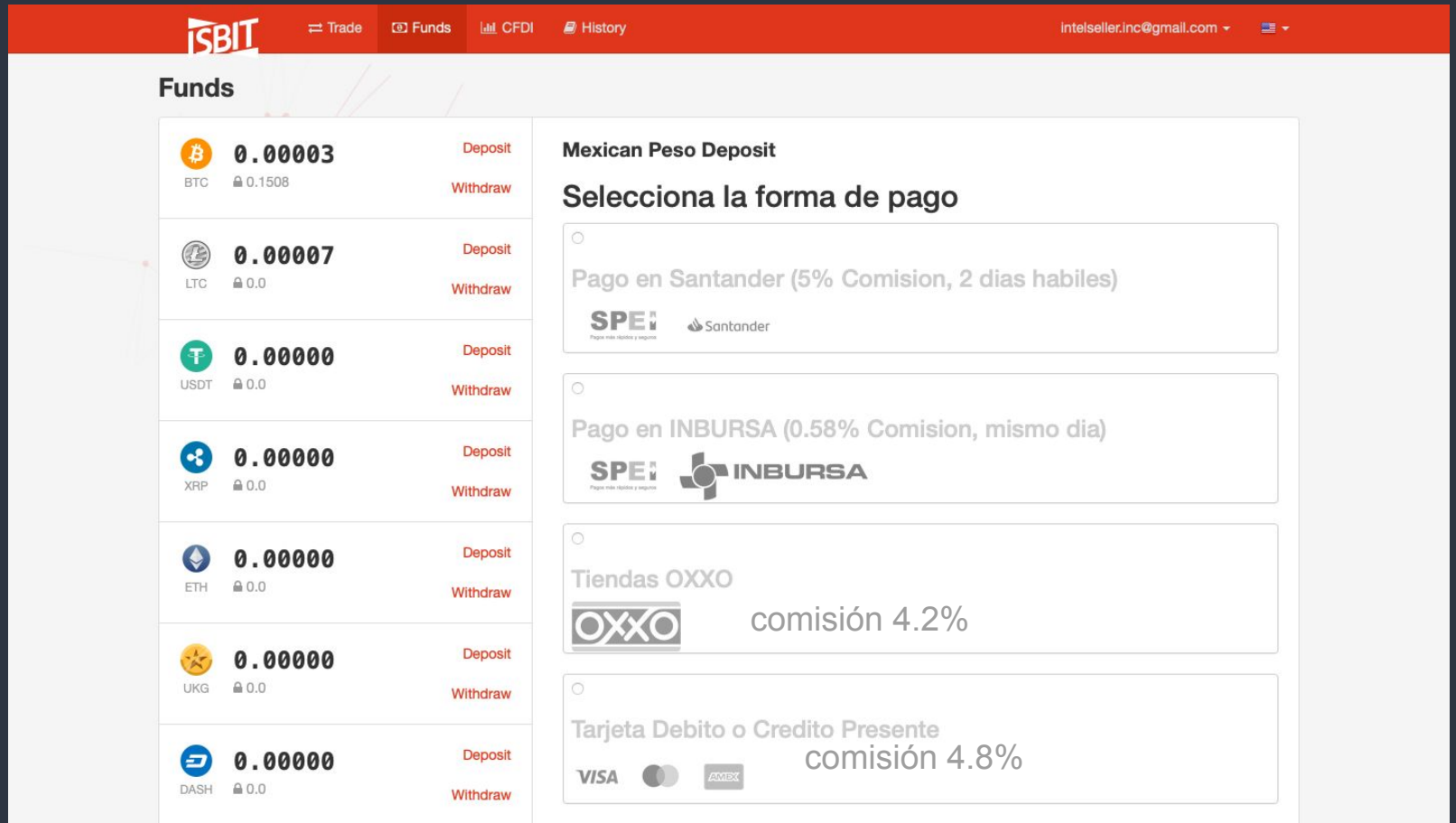
Parámetro de ubicación estimados utilizando muestra de transacciones SPEI realizadas en 2018:

$$\hat{\sigma} = media(|\ln(T_i) - \hat{\mu}|) = 22,767.84\,\text{sec} = \ 6.3244\,horas$$

Durante el periodo comprendido desde enero hasta diciembre de 2018 la empresa realizó el registro de los tiempos de espera para recuperación de **243** CEPs iniciando la cuenta regresiva a partir del momento de la propagación de la transacción por el banco *emisor*. De la muestra de **243 CEPs** se tuvieron 5 devoluciones por distintas causas de error y 16 CEPs cuyo estatus en el SPEI fue “liquidada” o “en proceso de liquidación” pero nunca se obtuvo confirmación y generación del CEP a pesar del éxito del pago. Banxico no mantiene por tiempo indefinido los CEPs disponibles para ser descargados. Por lo que en la muestra nos limitamos a intentar recuperar CEPs hasta por un máximo de *H*=90 días. Entonces podemos pensar que se tuvieron 16+5=21 tiempos de espera “infinitos”.

En base a los datos empíricos recabados por la empresa se calcularon estimadores para los parámetros de ubicación y escala de la distribución.

A continuación se muestra la manera en que el cliente puede elegir la forma de depósito a la cuenta concentradora de ISBIT



Sistema automatizado PLD; METIS

Plataforma que sirve para consultar datos de los clientes de la empresa

Nos ayuda a

Saber el listado de clientes que hicieron alguna operación en la empresa

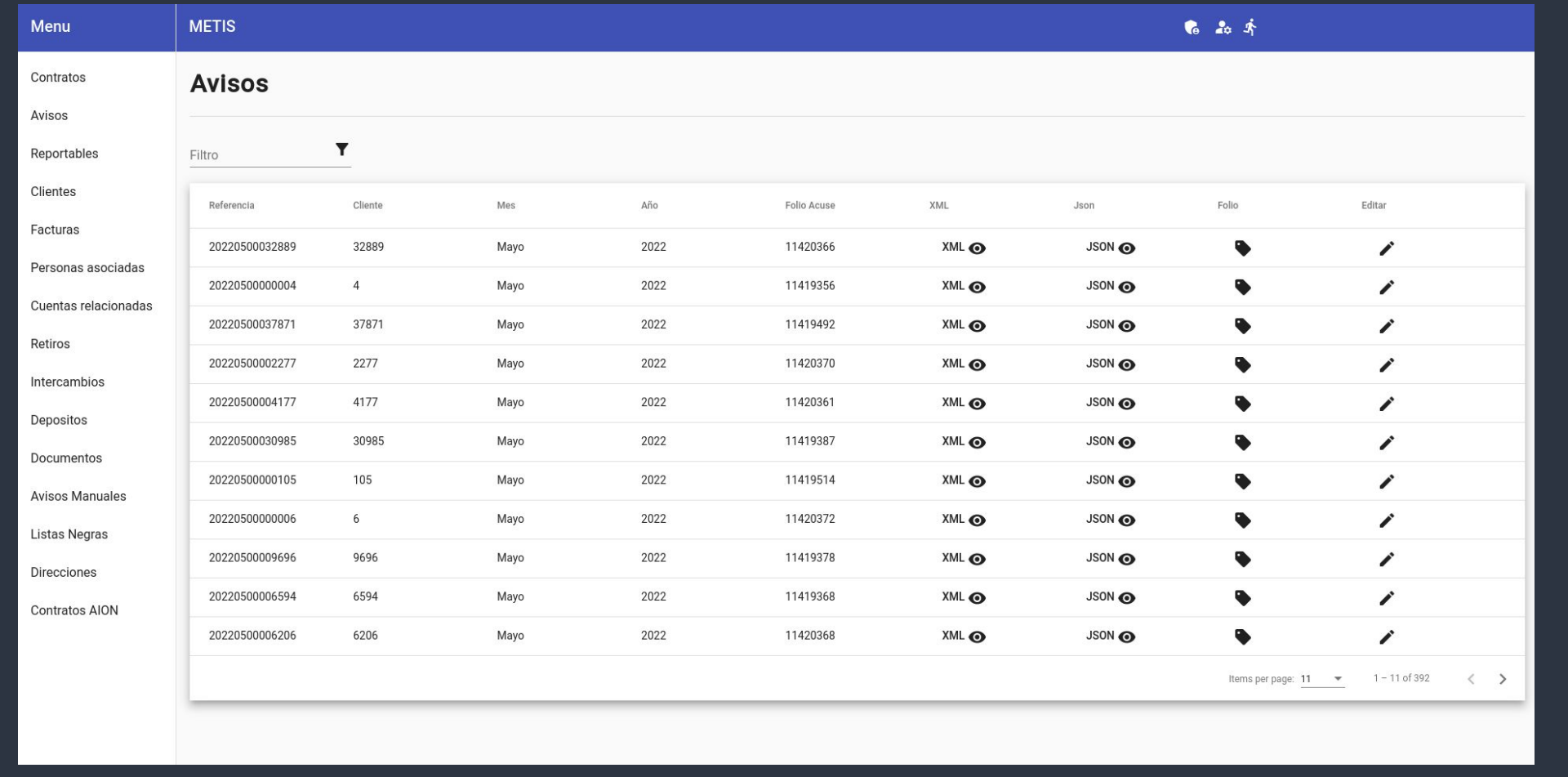
como

- Contratos
- Facturas
- Retiros
- Depósitos

Permite consultar y generar automáticamente avisos de operaciones vulnerables de los clientes en determinado mes y año.

El sistema automatiza el envío al SHCP de avisos

El sistema automatizado usa a VULCANO para determinar origen y destino de recursos (CEPs)


Menu
METIS
Contratos
Avisos
Reportables
Clientes
Facturas
Personas asociadas
Cuentas relacionadas
Retiros
Intercambios
Depositos
Documentos
Avisos Manuales
Listas Negras
Direcciones
Contratos AION
Avisos
Filtro
Referencia
Cliente
Mes
Año
Folio Acuse
XML
Json
Folio
Editar
20220500032889 32889 Mayo 2022 11420366 XML JSON
20220500000004 4 Mayo 2022 11419356 XML JSON
20220500037871 37871 Mayo 2022 11419492 XML JSON
20220500002277 2277 Mayo 2022 11420370 XML JSON
20220500004177 4177 Mayo 2022 11420361 XML JSON
20220500030985 30985 Mayo 2022 11419387 XML JSON
20220500000105 105 Mayo 2022 11419514 XML JSON
20220500000006 6 Mayo 2022 11420372 XML JSON
20220500009696 9696 Mayo 2022 11419378 XML JSON
20220500006594 6594 Mayo 2022 11419368 XML JSON
20220500006206 6206 Mayo 2022 11420368 XML JSON
Items per page: 11
1 – 11 of 392

Menu

METIS

- Contratos
- Avisos
- Reportables
- Clientes
- Facturas
- Personas asociadas
- Cuentas relacionadas
- Retiros
- Intercambios
- Depositos
- Documentos
- Avisos Manuales
- Listas Negras
- Direcciones
- Contratos AION

Depositos

Filtro

Id	Account	Miembro	Moneda	Monto	Cuota	Fund extra	TxId	AASM	Tipo	Metodo
20	80	40	2	0.0013527500000000	0.0000000000000000		fb13058a109d02c808f4e764d9d9967413df294a99f23a5809d1be4ca429e369	accepted	Deposits::Satoshi	
21	5	3	1	200.0000000000000000	0.0000000000000000	inbursa		cancelled	Deposits::Bank	
2	4	2	2	0.0004500000000000	0.0000000000000000		aa2528c8b9842080da991aa6ee1ad6a7c63914503af7f85258c1b7c63ed08144	accepted	Deposits::Satoshi	
1	4	2	2	0.0030000000000000	0.0000000000000000		37b669a59b1eca16bd4c61908b33a25157ac93a6411be2f2265da26d4d78e1a2	accepted	Deposits::Satoshi	
35	4	2	2	0.0004000000000000	0.0000000000000000		894da45fd432d5430a508598d8cc66e6dc6ea9f5f3e814fadab358a958c83883	accepted	Deposits::Satoshi	
34	4	2	2	0.0006282200000000	0.0000000000000000		ad18577d69d177dd7bad97196b171a8ebb841aa28359bcacf3e154c01e62818b	accepted	Deposits::Satoshi	
3	3	2	1	1000.0000000000000000	0.0000000000000000	inbursa	432536546546	accepted	Deposits::Bank	
25	6	3	2	0.0013538700000000	0.0000000000000000		f2df8e2c5ca090a9afd41158090ab0237f72e016558e350760a59680d95995ca	accepted	Deposits::Satoshi	
8	4	2	2	0.0050000000000000	0.0000000000000000		42a6e17dc3dd8f34e2a224491183abd40a86cc12c5c68f70496515e6e5eae06f	accepted	Deposits::Satoshi	
22	6	3	2	0.0017603600000000	0.0000000000000000		e6415ce765b13eaff37f56cc9e48a026233e22bc6c1f30071d9b28df0d45534b	accepted	Deposits::Satoshi	
17	2	1	2	0.0006746300000000	0.0000000000000000		f92bba320346e9e70cb36eaca07e322857cf3a93f10ca1bf1372b640367d553f	accepted	Deposits::Satoshi	

Items per page: 11 1 – 11 of 3306

Menu

METIS

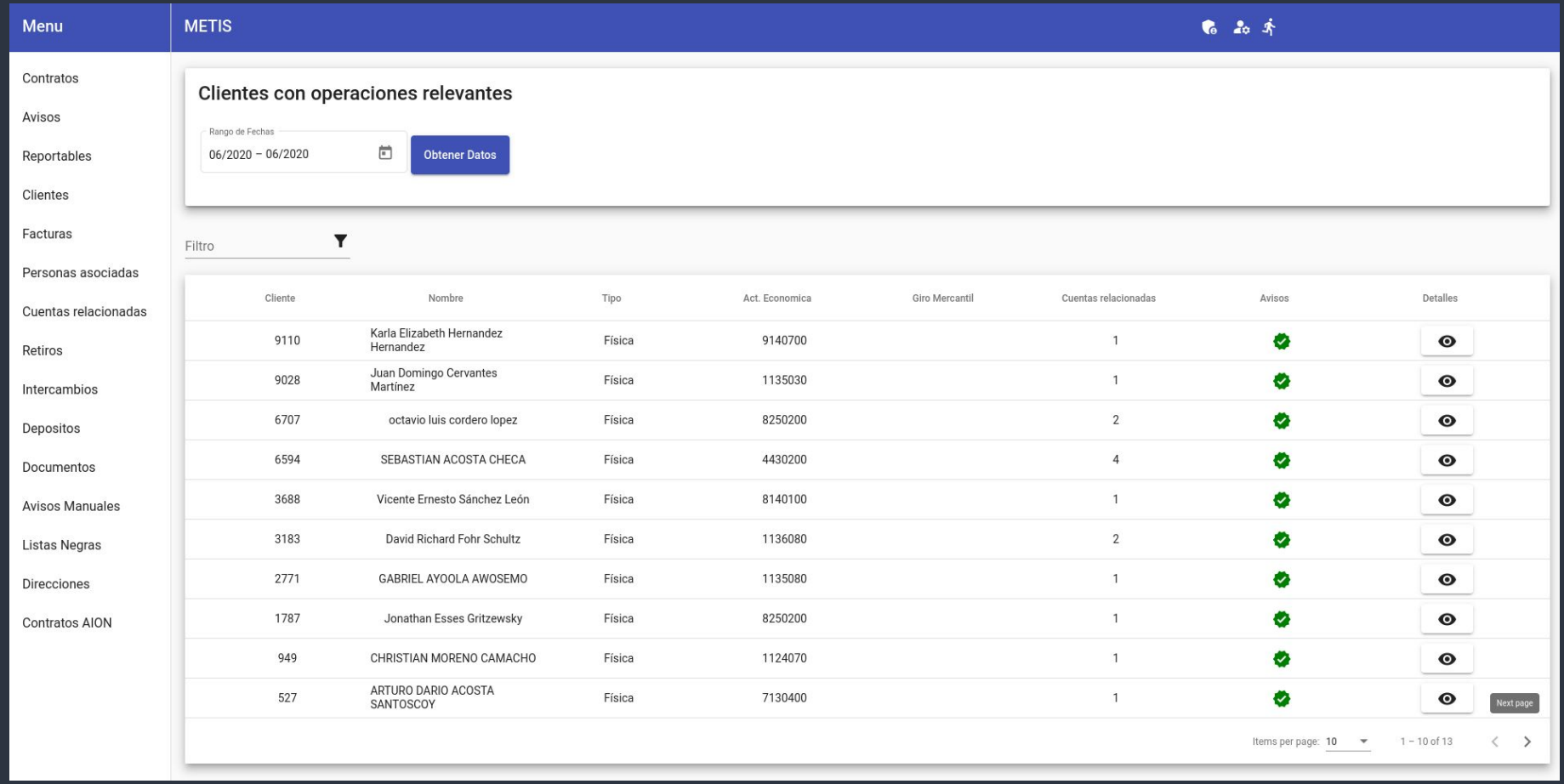
- Contratos
- Avisos
- Reportables
- Clientes
- Facturas
- Personas asociadas
- Cuentas relacionadas
- Retiros
- Intercambios
- Depositos
- Documentos
- Avisos Manuales
- Listas Negras
- Direcciones
- Contratos AION

Retiros

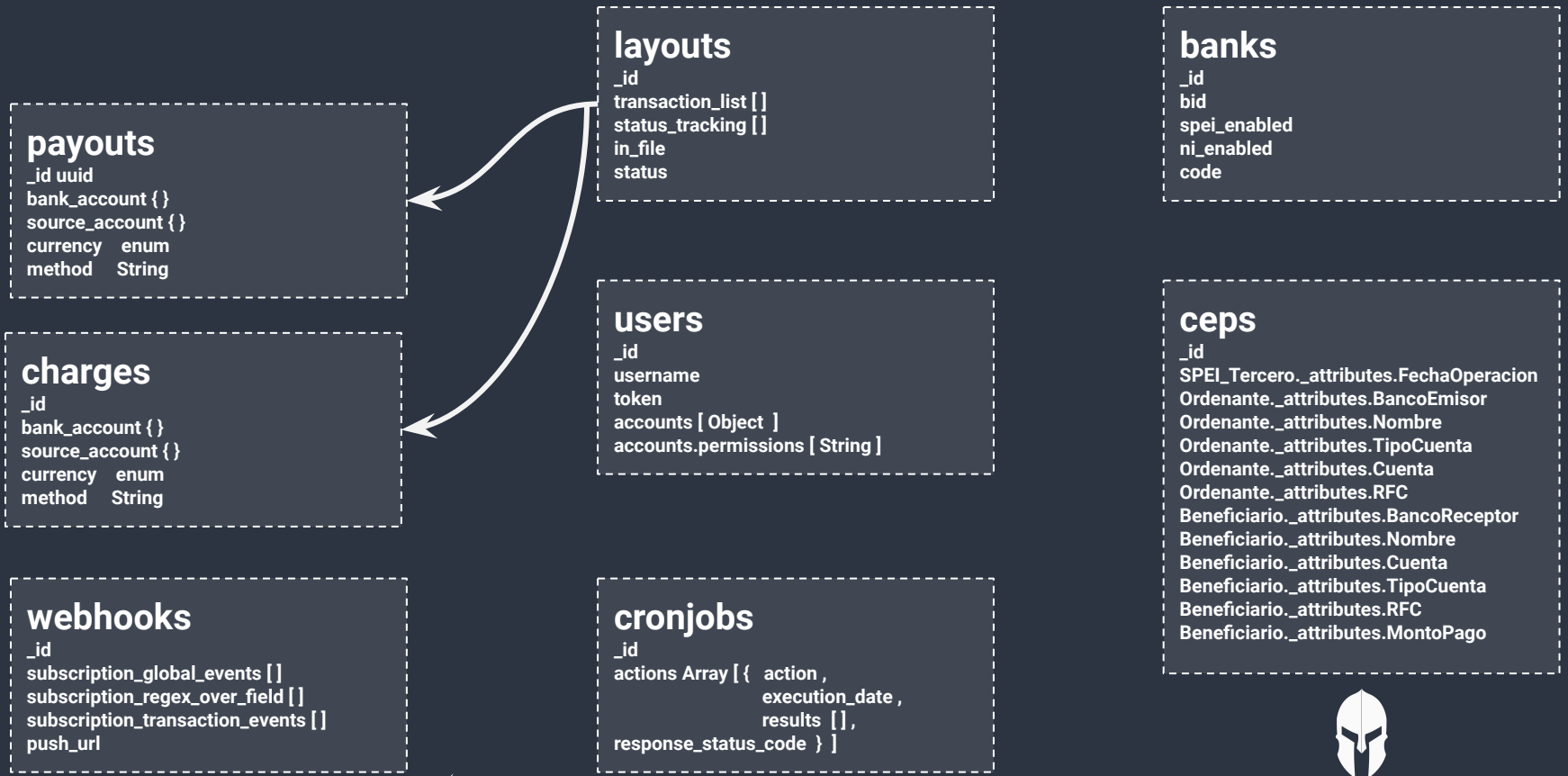
Filtro

Id	SN	Moneda	Monto	Cuota	Fund UID	Fund extra	AASM	Suma	Tipo
17	16050515500017	2	0.0004000000000000	0.0001000000000000	1JLGjmXaEkPste8X8AXkAFQVnb79GM8WzX	cartera_jeronimo_vargas	done	0.0005000000000000	Withdraws::Satoshi
6	16031715470006	2	0.0009000000000000	0.0001000000000000	1KEGUYxU5MmWYoxmfLxLyJpdbhdxg9vjzY	direccion_cartera_android	done	0.0010000000000000	Withdraws::Satoshi
11	16040317560011	2	0.0287266100000000	0.0001000000000000	391hiT2fmvJjtP5xJE6FScSwJEZ7db5bfn	cartera_bitgo1	done	0.0288266100000000	Withdraws::Satoshi
2	16030419560002	2	0.0132300000000000	0.0001000000000000	1KEGUYxU5MmWYoxmfLxLyJpdbhdxg9vjzY	direccion_cartera_android	done	0.0133300000000000	Withdraws::Satoshi
7	16032815320007	2	0.0013142300000000	0.0001000000000000	1H6pN9DaqEhqB2h5Fb3LaP2bGprrj6Pvtx	cartera bitcoin celular	canceled	0.0014142300000000	Withdraws::Satoshi
21	16061620230021	2	0.0009000000000000	0.0001000000000000	1GsXZ2cRTFojvqMomabpa2gSvkvCPTTfAi	monedero android demo bbva	done	0.0010000000000000	Withdraws::Satoshi
14	16050218330014	2	0.0018400000000000	0.0001000000000000	1JSsGT11uSb8p9G1u7JFWvRBLgKn1Ex49	eduardo_castillo	done	0.0019400000000000	Withdraws::Satoshi
3	16031014260003	2	0.0012527500000000	0.0001000000000000	1yCFHRYNBHpj5Dca6LZwpUq9xd6UkDUVk	Retiro Sebas 100316	done	0.0013527500000000	Withdraws::Satoshi
12	16040401470012	2	0.0011300000000000	0.0001000000000000	3Pf1kQcYDCrVqScMwYc4NJjGL3g9R5QCV4	cartera_externa_bitgo	done	0.0012300000000000	Withdraws::Satoshi
8	16032815400008	2	0.0013142300000000	0.0001000000000000	1H6pN9DaqEhqB2h5Fb3LaP2bGprrj6Pvtx	cartera bitcoin celular	done	0.0014142300000000	Withdraws::Satoshi
4	16031623190004	1	103.5600000000000000	0.0000000000000000	072180004200788982	ixe	done	103.5600000000000000	Withdraws::Bank

Items per page: 11 1 – 11 of 3675


Menu
METIS
Contratos
Avisos
Reportables
Clientes
Facturas
Personas asociadas
Cuentas relacionadas
Retiros
Intercambios
Depositos
Documentos
Avisos Manuales
Listas Negras
Direcciones
Contratos AION
Clientes con operaciones relevantes
Rango de Fechas
06/2020 – 06/2020
Obtener Datos
Filtro
Cliente
Nombre
Tipo
Act. Economica
Giro Mercantil
Cuentas relacionadas
Avisos
Detalles
9110
Karla Elizabeth Hernandez Hernandez
Física
9140700
1
9028
Juan Domingo Cervantes Martínez
Física
1135030
1
6707
octavio luis cordero lopez
Física
8250200
2
6594
SEBASTIAN ACOSTA CHECA
Física
4430200
4
3688
Vicente Ernesto Sánchez León
Física
8140100
1
3183
David Richard Fohr Schultz
Física
1136080
2
2771
GABRIEL AYOOLA AWOSEMO
Física
1135080
1
1787
Jonathan Esses Gritzewsky
Física
8250200
1
949
CHRISTIAN MORENO CAMACHO
Física
1124070
1
527
ARTURO DARIO ACOSTA SANTOSCOY
Física
7130400
1
Next page
Items per page: 10
1 – 10 of 13

Colecciones de documentos de VULCANO en MongoDB y sus relaciones principales



Microsoft Azure

mongoDB

VULCANO

Zona/país B

Exchange B

BITFINEX

AION

FIAT ← CRYPTO

API

BLOCKCHAIN

Transferencia Activos Virtuales

CRYPTO → FIAT

API

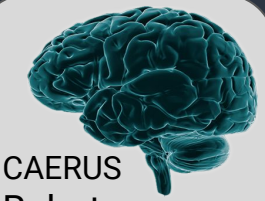
Zona/país A

Exchange A

ISBIT EXCHANGE

kraken

Robot inicia proceso si infiere que diferencia en el Precio de Crypto en Zona B mayor que Precio de Crypto en Zona A se mantendrá durante duracion de operaciones hasta concluir arbitraje. El Robot realiza función de predicción y valoración del riesgo en cada arbitraje.



CAERUS Robot

Controla tesorería vía APIs y H2Hs de instituciones

CAERUS App

Reporting, Dashboards, Monitoreo de Resultados

H2H

VULCANO

Transferencia moneda nacional

SWIFT, SPEI, SPID & TEF

H2H

VULCANO

Banco B

Banco A

Asumiendo que la diferencia en el Precio de Crypto en Zona B mayor que Precio de Crypto en Zona A persiste… el ciclo de arbitraje se continuará realizando (de forma completamente automatizada) mientras la ganancia financiera esperada/inferida sea suficientemente grande (por arriba de margen de riesgo calculado de forma dinámica por el robot.)

ERP

Generación de datos contables e informes.

METIS

Software Cumplimiento (Tareas KYC, AML y CTF)

EXCHANGE

Plataforma de liquidación electrónica

CAERUS

Algoritmo comercial de alta frecuencia de la suite. Bots, API, WEBSockets, Creación de mercado y arbitraje

VULCANO

Sistema de gestión de tesorería de Fiat. API unificada para todos los bancos

Pago distribuido y liquidación al sistema bancario via H2H

VULCANO permite utilizar la información transaccional a los diferentes sistemas Y áreas departamentales de ISBIT

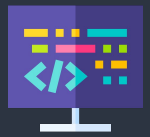
RESULTADOS

La seguridad en las transacciones aumentó considerablemente

Se disminuyó el riesgo de errores operativos

Se redujeron las comisiones impuestas por los bancos.

La implementación fue diseñada para facilitar su instalación, mantenimiento y escalabilidad



Se redujo el tiempo que nos toma un ciclo de arbitraje en un 48.61%

Esto implica que podemos realizar aproximadamente el doble de ciclos exactamente en el mismo tiempo

Comparación de la velocidad en transacciones

Intermediarios

Cuenta Bancaria Cliente

Agregador Pagos

Cuenta Bancaria Empresa

Tiempo de liquidación

de 24 a 72 horas

Tasa de descuento

de 2% a 6%

vs

VULCANO

Cuenta Bancaria Cliente

Tiempo de liquidación

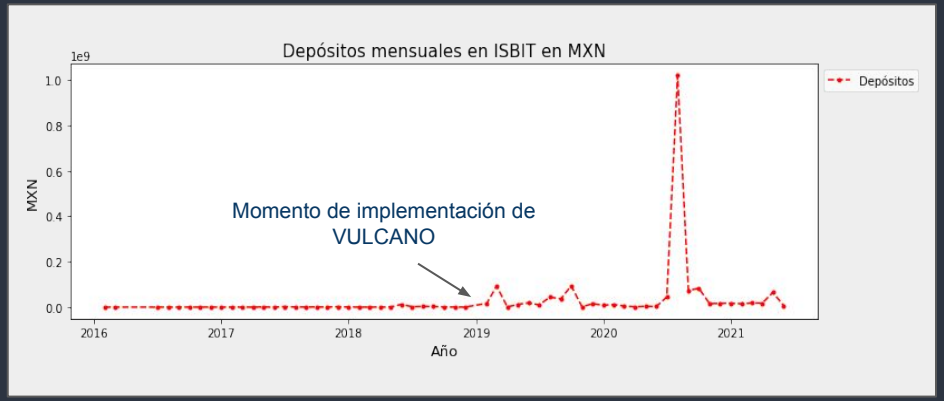
menos de 1 minuto

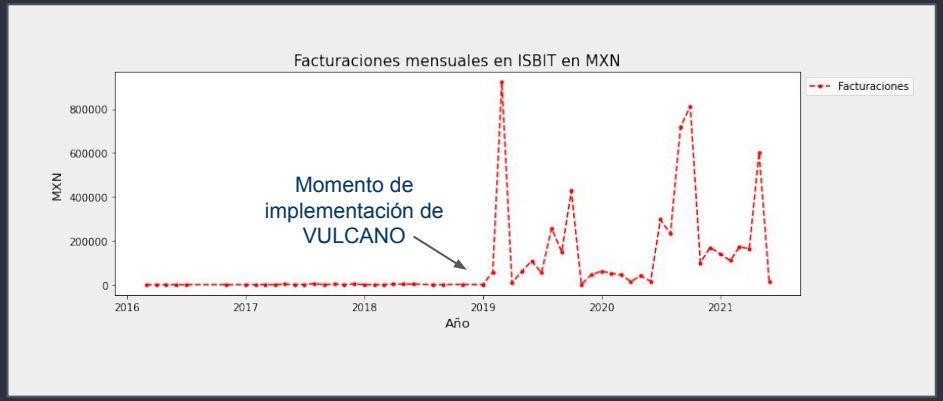
Tasa de descuento

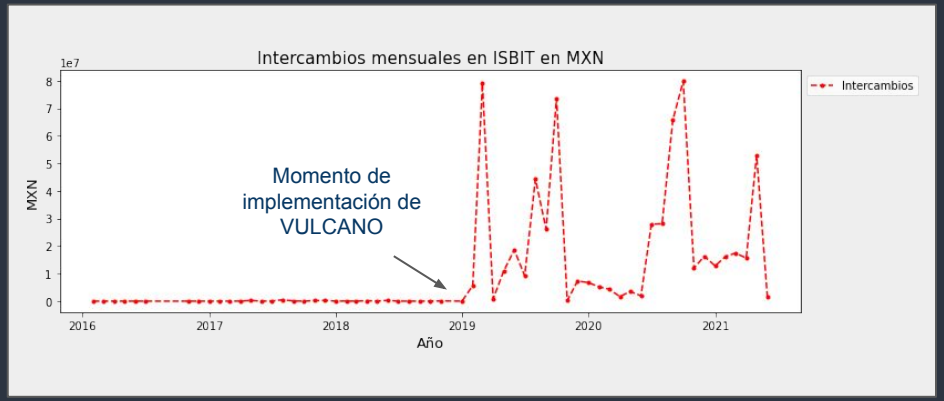
0%

Cuenta Bancaria Empresa

Aumento de la transaccionalidad del negocio







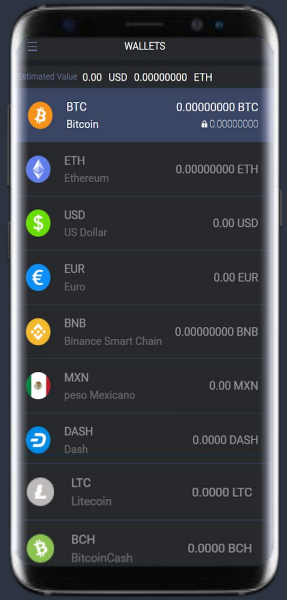
VULCANO se implementó a finales del 2018.

A partir de ese momento, **aumentó significativamente la frecuencia** mensual de transacciones, el volumen mensual y la facturación de comisiones asociada a operaciones.

ISBIT EXCHANGE

Resultados

Año Fiscal	2016	2017	2018	2019	2020	2021
Volumen de intercambio MXN	$1,571.26	$1,290,656.16	$587,797.38	$275,648,506.15	$253,284,273.91	$316,604,949.57
Volumen de intercambio US	$78.56	$64,532.81	$29,389.87	$13,782,425.31	$12,664,213.70	$15,830,247.48
Incremento VOLUMEN (x veces)	NA	NA	0.46	468.95	0.92	1.25
% Margen (Ingreso por Comisiones)	2.14%	1.12%	4.11%	0.76%	1.01%	1.01
% Margen (Ingreso por Arbitraje / MM)	0.00%	0.00%	0.00%	0.12%	4.07%	4.06%



El volumen de intercambio anual aumentó más de **468 veces** como resultado de implementar VULCANO a finales de diciembre de 2018.

La innovación, investigación y desarrollo invertido en la creación de VULCANO tuvo un impacto significativo en la rentabilidad y eficiencia de la empresa, lo cual se puede constatar en los Estados Financieros de ISBIT SA DE CV.

Resultados

Consecuencia de implementar VULCANO a inicios de 2019

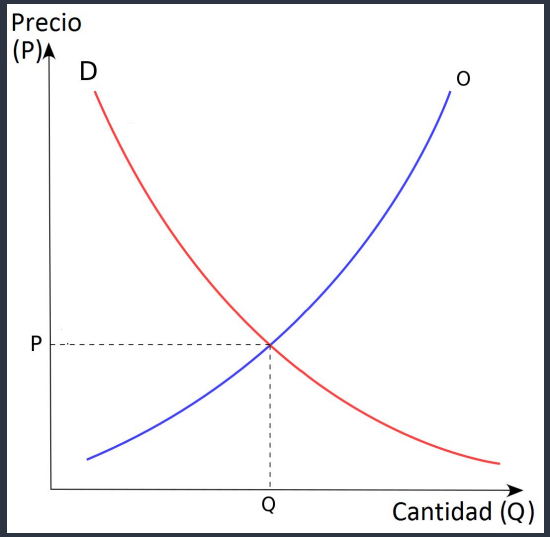
concepto de ingreso	2016	2017	2018	2019	2020	2021
Facturación de comisiones operaciones en plataforma	33.555135	$14,446.22	24,165.33	2,089,553.83	2,553,027.76	3,141,500.66
% INCREMENTO FACTURACIÓN COMISIONES	0.00%	42952.19%	67.28%	8546.91%	22.18%	23.05%
Margen Profit por cada ciclo arbitraje	0.00%	0.00%	0.00%	0.12%	4.07%	4.07%
Rotacion Inventario (ciclos mensuales)			0.27	45.94	1.95	2.00
AUM (inventario propio) portafolio activos propios			$180,000.00	$500,000.00	$10,800,000.00	$21,349,440.00
Crecimiento anual AUM ops arbitraje/MM	0	0	0%	177.78%	2060.00%	97.68%
Ingresos ops arbitraje/MM/especulacion	$0.00	$0.00	$0.00	$320,000.00	$10,300,000.00	$10,549,440.00
Ventas Consultoria/Desarrollo/implementacion			870,200.00	0.00	0.00	0.00
Ingresos Brutos (antes costos, impuestos)			$894,365.33	$2,409,553.83	$12,853,027.76	$13,690,940.66
Costos			500,000.00	1,200,000.00	1,350,000.00	1,518,750.00
Gastos			450,200.00	1,200,900.00	1,320,000.00	1,450,911.82
ingresos Netos			-$55,834.67	$8,653.83	$10,183,027.76	$10,721,278.84

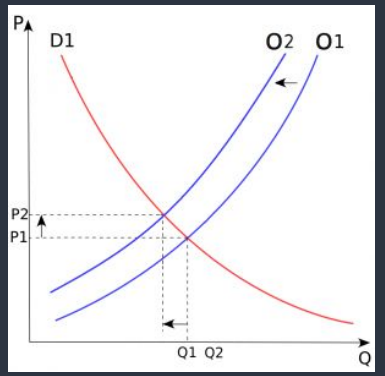
Conclusión

1. La implementación de VULCANO en ISBIT SA DE CV permitió ofrecer a los clientes opciones de depósito y retiro de fondos de menor costo. Antes de implementar VULCANO forzosamente necesitamos trasladar el costo de procesar un depósito o retiro al cliente final. Menos VALOR es extraído por los ***intermediarios*** (agregadores o pasarelas de pago) ahora y este valor es capturado por la empresa y sus clientes en forma de ahorros sustanciales.
2. Pudimos validar de primera mano una Ley de Economía. Una disminución del 86.19% en el precio de nuestro producto/servicio aumentó la demanda para el mismo 468.95 (ventas 2019/ventas 2018) veces en un año.

A pesar de bajar los precios al cliente final logramos aumentar la facturación de comisiones cobradas en el Exchange 85.4 veces (en 2019 a comparación de 2018) .

3. Implementar VULCANO también permitió abrir nuevas líneas de negocio que antes no eran factibles (Arbitraje Financiero, CAERUS).
4. Integrar VULCANO con METIS permitió evitar multas y sanciones que pueden oscilar entre el 10% al 100% del valor de una operación o 800,000 mxn.





ISBIT EXCHANGE

¡GRACIAS!

Sebastian Acosta Checa
Presidente & Director General

s@isbit.co

+52 55 1478 0943

Paseo de la Reforma 296,
PISO 40, Suite B122
Colonia Juárez
Ciudad de México
Código Postal 06600