Universidad Continental

FACULTAD DE INGENIERÍA

Escuela Académico Profesional de Ingeniería de Sistemas e Informática

Trabajo de Suficiencia Profesional

Arquitectura de software basada en microservicios para implementación de la aplicación web de cobranza digital en Financial Systems Company SAC

Hans Richard Villaizán Yamamoto

Para optar el Título Profesional de Ingeniero de Sistemas e Informática

Huancayo, 2019

AGRADECIMIENTO

Agradezco a mis padres y amigos quienes no dejaron de motivarme para realizar este trabajo, recibiendo su apoyo incondicional.

De forma particular quisiera agradecer al Dr. Job Daniel Gamarra Moreno por su revisión exhaustiva de este trabajo; del mismo modo agradecer al Ing. Miguel Ángel Córdova Solís, por las precisiones brindadas a este trabajo; quienes siempre aportaron de manera positiva en mi educación.

DEDICATORIA

Dedico este trabajo a mis padres y hermanos, quienes me apoyaron durante la etapa de formación universitaria y me alentaron a dar continuidad al crecimiento profesional.

ÍNDICE

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

La arquitectura de microservicios es una tendencia que crece rápidamente en el mundo empresarial, dividiendo una aplicación en pequeñas funcionalidades o servicios; los métodos tradicionales de cobranza se han vuelto menos útiles; en contraste los canales digitales están siendo ampliamente utilizados por las empresas y personas, considerando esta oportunidad se integra la arquitectura de microservicios en la aplicación web de cobranza digital, utilizando los canales email, whatsapp, sms y chatbots durante la gestión de cobranza. El objetivo del presente estudio fue diseñar la arquitectura de software basada en microservicios para la implementación en la aplicación web de cobranza digital en la empresa Financial Systems Company SAC. El estudio tiene un diseño no experimental de tipo transversal puesto que para elaborar el diseño de la arquitectura de microservicios se realizó una investigación sobre sus componentes, interacciones y atributos de calidad, se caracterizaron los requisitos de la aplicación web de cobranza digital y por medio del método analítico-sintético se logró diseñar la arquitectura de microservicios e implementarla en la aplicación web de cobranza digital.

El resultado del estudio es una arquitectura basada en microservicios implementada en la aplicación web de cobranza digital cumpliendo los atributos de calidad, apta para uso masivo y escalable. Entre las conclusiones del estudio se menciona que fue importante la incorporación de servicios de terceros para obtener independencia en los microservicios, también que la implementación de la arquitectura de microservicios en la cobranza digital soporta envíos masivos de mensajes por canal sin impactar en el rendimiento, manteniendo su alta disponibilidad.

Palabras clave: Arquitectura de microservicios, microservicios, aplicación web, cobranza digital.

ABSTRACT

The microservices architecture is a fast-growing trend in the business world, dividing an application into small functionalities or services; traditional collection methods have become less useful; In contrast, digital channels are widely used by companies and individuals, considering this opportunity to integrate the microservices architecture into the digital collection web application, using email, whatsapp, sms and chatbots channels during collection management. The objective of the present study was to design the software architecture based on microservices for the implementation in the digital collection web application in Financial Systems Company SAC. The study has a non-experimental cross-sectional design, to elaborate the design of the microservices architecture, a research on its components, interactions and quality attributes was carried out, the requirements of the web application of the digital collection were characterized and by means of The analytical-synthetic method was able to design the microservice architecture and implement it in the web application of the digital collection.

The result of the study is an architecture based on microservices implemented in the web application of digital collection that meets the quality attributes, suitable for a massive use and scalable. Among the conclusions of the study it is mentioned that it was important the incorporation of third party services to obtain independence in the microservices, and also that the implementation of the architecture of microservices in the digital collection allows the mass sending of messages per channel without affecting the performance, maintaining high availability.

Keywords: microservice architecture, microservice, web application, digital collection.

INTRODUCCIÓN

Los servicios financieros prestados a través de medios digitales están generando una revolución disruptiva, el cliente va accediendo a los servicios financieros mediante el celular, las empresas no están utilizando los canales que direccionan a la mejor respuesta del cliente. Por el contrario, los canales que los servicios financieros utilizan con menos frecuencia son (correo electrónico, mensajes de texto, notificaciones emergentes y chats) y son estos los canales más favorecidos por los clientes de hoy, ya que producen los mejores resultados. La morosidad está creciendo y las empresas necesitan las capacidades suficientes para dirigir las nuevas demandas de sus servicios, es probable que los recursos de los departamentos de cobranzas sean insuficientes, sus actuales métodos de cobranzas se han vuelto menos útiles, por lo que hay que considerar que del 89% al 92% de las tasas de pago pueden ser obtenidas usando canales digitales – banca en línea o mobile. Esto genera la oportunidad de implementar soluciones utilizando los canales digitales en la gestión de cobro debido a que sus modelos de negocio son cambiantes y por ello necesitan estar a la vanguardia de las tecnologías emergentes. Otra de las tecnologías emergentes son los microservicios, una tendencia que nos ofrece una forma diferente de enfocar el desarrollo de aplicaciones. La idea detrás de las arquitecturas de microservicios es el “divide y vencerás”, esto plantea dividir una aplicación en pequeñas funcionalidades o servicios, se estructura en un conjunto de servicios independientes entre sí; cada uno de ellos debe de cubrir una funcionalidad clara de negocio. Por lo tanto, el estudio plantea una arquitectura de software basada en microservicios para la aplicación web de cobranzas digitales.

Para la realización del estudio se utilizó una metodología analítica-sintética que permitió generar un diseño de la arquitectura basada en microservicios, sus componentes e interacciones para luego utilizarlos en la implementación de la aplicación web de cobranza digital a partir de la información obtenida. Del mismo modo, se utilizó las historias de usuario para caracterizar los requisitos como son percibidos, con una breve descripción y criterios de validación. también se utilizó el método ATAM (del inglés Architecture Trade-off Analysis Method) para evaluar los atributos de calidad arquitectónica y escenarios realizados por el equipo de desarrollo.

En el primer capítulo se describen los aspectos generales de la empresa Financial Systems Company SAC.

En el segundo capítulo se brindan los antecedentes de investigación del estudio y se consideran tesis de grado, maestría y doctorado, así como libros tecnológicos que servirán

como fundamento para la elaboración de la arquitectura basada en microservicios; se describen los aspectos generales y los objetivos del estudio.

En el tercer capítulo se dan a conocer las definiciones clave para el diseño de la arquitectura de software basada en microservicios para la implementación en la aplicación web de cobranza digital.

En el cuarto capítulo se describen las actividades realizadas como profesional de ingeniería de sistemas, también las metodologías considerando los métodos generales y específicos, el alcance de la investigación teniendo en cuenta el tipo y nivel del mismo. Adicionalmente se describen las técnicas, instrumentos utilizados para la recopilación de información y materiales para el desarrollo de las actividades.

En el quinto capítulo se muestran los resultados de la implementación de la arquitectura software basada en microservicios en el aplicativo web de cobranza digital cumpliendo los atributos de calidad, siendo escalable y soportando envío masivo de mensajes por canal sin impactar en el rendimiento.

CAPÍTULO I

ASPECTOS GENERALES DE LA EMPRESA Y/O INSTITUCIÓN

1.1. DATOS GENERALES DE LA EMPRESA

RUC: 20546907466

Razón Social: Financial Systems Company Perú SAC

Tipo Empresa: Sociedad Anónima Cerrada

Condición: Activo

Fecha Inicio de actividades: 22/03/2012

Gerente General: Luis Antúnez Olortegui

Gerente de Operaciones: Alain Zoghbi Dib

Ubicación: Av. República de Panamá Nro. 3655, Oficina 501 – San Isidro – Lima – Perú.

Oficina Principal: Av. Pradilla 900 Este, Centro Empresarial Centro Chía, Oficina 301, Chía – Cundinamarca – Colombia.

Actividad: Servicios informáticos – Financieros.

Teléfono: (01) 6419133

1.2. ACTIVIDADES PRINCIPALES DE LA INSTITUCIÓN Y/O EMPRESA

Financial Systems Company brinda servicios de Implementación, Consultoría, Mantenimiento y Soporte.

Implementación:

Equipo completo de especialistas para la implementación de los distintos proyectos, los cuales con atención personalizada y dedicada garantizarán la correcta operación e integración de ICS (Internet Collection System) con sus sistemas contribuyendo a la mejora los procesos de cobranzas actuales.

Consultoría:

Servicios de consultoría enfocados al mejoramiento de los indicadores de gestión y recuperación de cartera, a través de la evaluación de procesos de cobranza, análisis de datos y diseño de estrategias que involucren las mejores prácticas de cobro buscando facilitar la toma de decisiones estratégicas, además de apoyar y optimizar procesos operativos.

Mantenimiento y Soporte:

Ofrecemos diferentes esquemas de servicio según las necesidades de cada cliente, para asegurar el óptimo funcionamiento de nuestra plataforma. Dependiendo del nivel contratado, nuestros clientes se benefician de diversos servicios.

1.3. RESEÑA HISTORICA DE LA INSTITUCIÓN Y/O EMPRESA

Somos una multinacional latinoamericana líder en automatización de cobranzas, fundada en 1994 en Bogotá-Colombia y desde hace 25 años nos hemos destacado por brindar y desarrollar soluciones tecnológicas innovadoras para el proceso de cobranzas de nuestros clientes.

Desde nuestros inicios nuestro objetivo principal ha sido ofrecer la excelencia a través de cada uno de nuestros productos y servicios. Después de un robusto análisis de necesidades y requerimientos del mercado lanzamos la herramienta Internet Collection System (iCS), en su versión inicial conocido como CSCS, la más completa, avanzada y revolucionaria herramienta en gestión de cobranzas y recuperación de cartera, con sus modalidades de producto: Enterprise, Express y SaaS

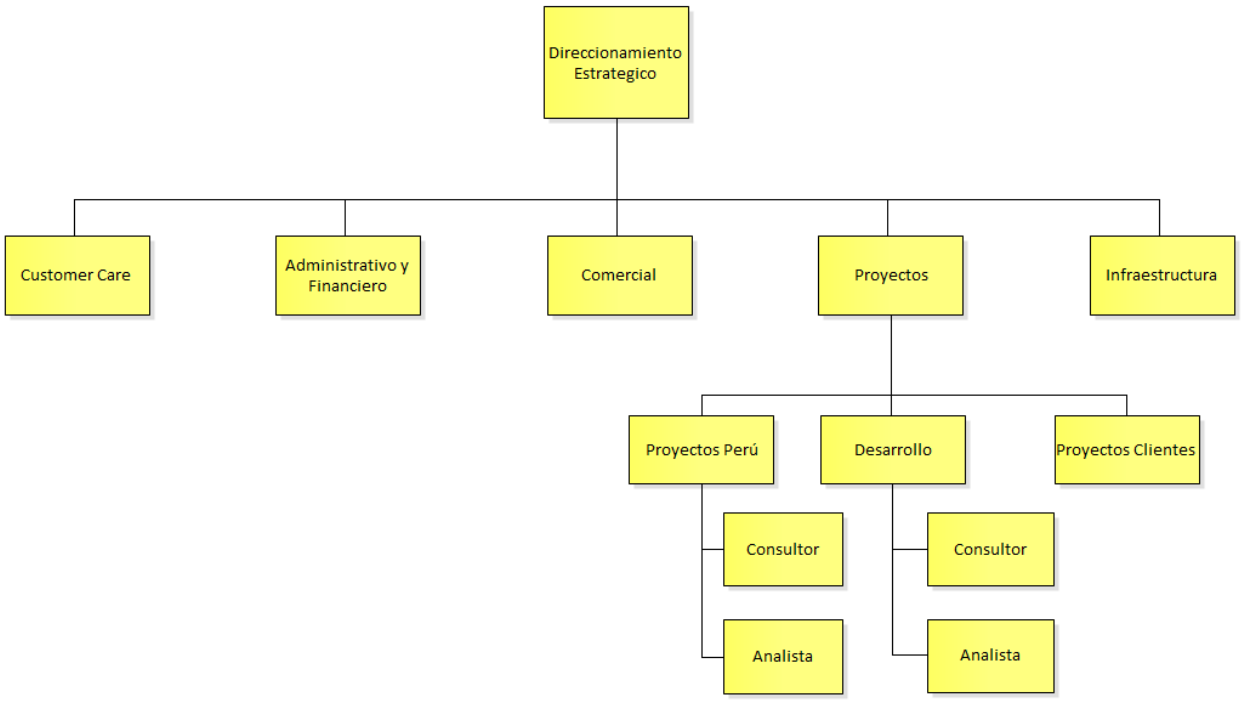
Actualmente tenemos presencia en: Colombia (Casa Matriz), Perú, Chile, USA, México, Ecuador, Panamá, Argentina y Venezuela. Contamos con clientes de diferentes sectores como el Financiero, Retail, Telecomunicaciones, Servicios públicos, Cajas de compensación, Microempresa y Empresas de cobranza.

Actualmente somos el mayor proveedor de soluciones de cobranzas en Latinoamérica.

Nuestro compromiso es la satisfacción de los Clientes y el mejoramiento continuo en nuestros procesos. Es por esto que implementamos y certificamos nuestro Sistema de Gestión de Calidad y Seguridad (SGCS) bajo los estándares ISO 9001:2015 (Sistema de Gestión de Calidad) e ISO 27001:2013 (Sistema de Gestión de Seguridad de la Información) para brindar un mejor servicio y productos de la más alta calidad y seguridad del mercado. Seguiremos trabajando para poner a su disposición soluciones especializadas para las organizaciones que requieren optimizar sus esquemas de cobranzas y recuperación de cartera, basados en actualidad, innovación y servicio cada día con el fin de aumentar la rentabilidad de nuestros usuarios.

1.4. ORGANIGRAMA DE LA INSTITUCIÓN Y/O EMPRESA

Figura 1 - Organigrama Financial Systems Company



Fuente: Elaboración propia

1.5. VISIÓN Y MISIÓN

Visión

Ser aliados de nuestros clientes desarrollando un ecosistema de cobranzas que resuelva todas sus necesidades tecnológicas de una forma segura, inteligente y fácil.

Misión

Crear tecnología que facilite la recuperación de los créditos logrando que nuestros clientes puedan ampliar la cobertura de préstamos en los diferentes países.

1.6. BASES LEGALES O DOCUMENTOS ADMINISTRATIVOS

Ley N° 26887 – Ley General de Sociedades

La legislación de la empresa Financial Systems Company Perú S.A.C, se basa en la ley N° 26887 - Ley General de Sociedades, la cual posee las siguientes características (Figura 2):

- En base a su naturaleza jurídica la S.A.C, son personas jurídicas, conformadas por personas naturales o jurídicas que se asocian para desarrollar una actividad comercial.
- En base al número de socios, la S.A.C, se constituyen con la participación de no menos de dos socios y no pueden superar al máximo de veinte socios.
- En base a la responsabilidad de los socios, siendo personas jurídicas distintas a los socios que las conforman, las obligaciones que asumen son cubiertas con su patrimonio, pero les otorgan a sus socios el beneficio de la responsabilidad limitada.
- En base a las acciones y participaciones, el capital social está representado en accionistas según su aporte al capital.

Figura 2 - Bases Legales - Financial Systems Company Perú SAC.

CONSULTA RUC: 20546907466 - FINANCIAL SYSTEMS COMPANY PERU SOCIEDAD ANONIMA CERRADA - FSC PERU S.A.C.			
Número de RUC:	20546907466 - FINANCIAL SYSTEMS COMPANY PERU SOCIEDAD ANONIMA CERRADA - FSC PERU S.A.C.		
Tipo Contribuyente:	SOCIEDAD ANONIMA CERRADA		
Nombre Comercial:	FSC PERU S.A.C.		
Fecha de Inscripción:	22/02/2012	**Fecha Inicio de Actividades:**	22/03/2012
Estado del Contribuyente:	ACTIVO		
Condición del Contribuyente:	HABIDO		
Dirección del Domicilio Fiscal:	AV. REP DE PANAMA NRO. 3655 DPTO. 501 LIMA - LIMA - SAN ISIDRO		
Sistema de Emisión de Comprobante:	MANUAL	**Actividad de Comercio Exterior:**	SIN ACTIVIDAD
Sistema de Contabilidad:	COMPUTARIZADO		
Actividad(es) Económica(s):	Principal - 6201 - PROGRAMACIÓN INFORMÁTICA		
Comprobantes de Pago c/aut. de impresión (F. 806 u 816):	FACTURA NOTA DE CREDITO NOTA DE DEBITO		
Sistema de Emisión Electrónica:	FACTURA PORTAL DESDE 28/01/2019		
Afiliado al PLE desde:	01/01/2015		
Padrones :	NINGUNO		

Fuente: Sunat

1.7. DESCRIPCION DEL ÁREA DONDE REALIZA SUS ACTIVIDADES PROFESIONALES

El área de proyectos de la empresa Financial Systems Company Perú SAC, es el área encargada de ejecución de los proyectos a clientes externos, cubre todos los proyectos de: consultoría, actualización, instalación, desarrollo, liberación de Release o producto, capacitación, collection review (evaluación de procesos de cobranza) y data mining (minería de datos) cuya duración sea mayor a 40 horas planeadas.

Para proyectos internos o liberación de producto, el alcance es dado por direccionamiento estratégico mediante una reunión interna y queda consignado en un acta o un correo entre las áreas de Software y el proceso de corrección de errores.

Definiciones y condiciones generales:

- Todo proyecto mayor o igual a 40 horas debe estar en la herramienta de manejo de cronogramas de Proyectos, junto con un cronograma y al menos una tarea independiente de quien lo ejecuta para tener visibilidad de las asignaciones de los recursos.
- Todo proyecto menor a 40 horas no tiene que ser entregado a proyectos ya que pueden ser manejados por Customer Care (Atención al cliente), Chile o Perú

directamente. Solo se hará entrega a la gerencia de proyectos los trabajos que contemplen 40 horas o más en la gestión.

- Para el seguimiento de los proyectos externos se puede utilizar alguna herramienta tipo Project Tracker (Seguimiento de proyectos) en donde se lleve el resumen del avance del proyecto. Para proyectos de liberación de Reléase o producto no es necesario este tipo de herramientas.
- La instalación o actualización de los ambientes para iCS debe realizarlo Infraestructura, salvo que Direccionamiento Estratégico tome otra decisión o dirección.
- Los entregables debe estar claramente definidos por las personas asignadas que hicieron la propuesta y levantaron información, permitiendo completar la información en su totalidad por la gerencia de proyectos.
- El cronograma inicia cuando se asigna el recurso, y se inicia cualquier actividad del mismo. Debe estar activo y con línea base.
- Los cronogramas buscan que las tareas estén paralelizadas a fin de acabar más rápido y tramitar facturación. El objetivo siempre debe estar orientado a la ejecución del proyecto lo más pronto posible para poder entregar resultados, y que el cliente tenga el beneficio, evacuar los recursos para poder facturar manteniendo el flujo de la empresa.
- Evitar las mezclas de texto y tipos de letra en el documento.
- Cuando se ajuste el documento debe hacerse en un solo conjunto y a la vez y medir para poder controlar su impacto.
- Al plantear los objetivos debe justificarlos de acuerdo al análisis y validación del alcance del proyecto. Y ser escritos de tal forma que satisfagan la necesidad del cliente.
- Los alcances deben estar relacionado con los objetivos alineándolo con el resumen ejecutivo del proyecto. Deben tener la justificación del cliente del por qué lo requiere para que de esta manera el recurso ejecutor del proyecto oriente su trabajo en satisfacer la necesidad del cliente.
- Los entregables deben ser medibles no solo para documentos.

- La entrega de los proyectos por parte del área comercial debe ser en reuniones de tal forma que el ingeniero asignado al proyecto entienda claramente lo que se vendió y la necesidad del cliente. Para proyectos internos se debe hacer en reunión con DE en la cual también se define el alcance o tickets a incluir si es un proyecto de liberación de producto.
- El proyecto puede ser devuelto al área comercial o a quien levantó la propuesta, si no coinciden las horas del cronograma con la propuesta, y con el tiempo que hasta ese momento se haya registrado en el caso. Las tareas comerciales las debe contemplar el cronograma también o si no se debe sacar de los indicadores del proyecto al momento de medirlos.
- Identificar y revisar los riesgos que apliquen al proyecto.
- Todos los conceptos diligenciados en el documento deben ser consistentes y alineados a la propuesta con lo que realmente requiere el cliente.

1.8. DESCRIPCIÓN DEL CARGO Y DE LAS RESPONSABILIDADES DEL BACHILLER EN LA INSTITUCIÓN Y/O EMPRESA

Cargo: Analista

Responsabilidades. Son funciones y responsabilidades del cargo de Analista, los siguientes:

a) Funciones:

1. Analizar y solucionar los casos asignados que han sido reportados por los clientes, cumpliendo con las políticas y estándares definidos en el SGCS de FSC PERU.
2. Contribuir a la armonía del equipo de trabajo que se consolide en FSC PERU.
3. Cumplir con las metas personales propuestas para un determinado periodo y mantener al día la herramienta diseñada para su control.
4. Cumplir en forma rigurosa con todos los procedimientos y políticas establecidas en la empresa sobre los cuales el trabajador tenga injerencia, relacionada con el sistema de gestión de calidad y seguridad.
5. En general, proponer, cuando lo considere pertinente, acciones que redunden en beneficios para la empresa.

6. Mantener informados a los directivos de FSC Perú de todas las situaciones relevantes para el buen funcionamiento de la empresa.
7. Notificar oportunamente los casos de mejora, incumplimiento u incidentes de seguridad de acuerdo al SGCS.
8. Participar activamente en el logro de los objetivos de FSC Perú y particularmente para alcanzar el éxito en la implementación de los proyectos con los clientes y/o acuerdos de niveles de servicio con los mismos.
9. Presentar dentro de los 3 días siguientes a la ocurrencia del hecho generador del gasto, una relación de gastos de acuerdo con el procedimiento y en el formato establecido por la empresa para este fin.
10. Velar por la optimización de los recursos a su disposición con el fin de minimizar los gastos de la empresa.
11. Analizar y administrar los riesgos de los proyectos a cargo de la empresa.
12. Asegurar los recursos físicos, financieros y humanos para los proyectos y actividades de la empresa y asignar las respectivas tareas teniendo en cuenta competencias y disponibilidad del personal.

b) Mantener la relación de contacto permanente con los clientes para determinar las necesidades del mercado y participar en la planificación estratégica de FSC Perú.

c) Aplicar correcta y responsablemente la política de personal referida a la cultura de trabajo donde la calidad y rentabilidad estén íntimamente integradas en toda decisión.

d) Cumplir las indicaciones de la gerencia del área de proyectos de FSC Perú conforme a los lineamientos del área de proyectos de FSC SAS.

e) Atender el soporte de primera línea de los clientes en Perú, conforme a los lineamientos del área de Customer Care de FSC SAS. Entendiéndose como soporte de primera línea el diagnóstico y solución de casos tipo A y diagnóstico de casos y/o errores difíciles de detectar desde el exterior.

CAPÍTULO II
ASPECTOS GENERALES DE LAS ACTIVIDADES PROFESIONALES

2.1. ANTECEDENTES O DIAGNÓSTICO SITUACIONAL

2.1.1. ANTECEDENTES

(Bolívar y Calla, 2018), en su investigación titulada: “Implementación de microservicios para el intercambio de datos en el canal de plataforma digital del Banco de Crédito del Perú”. En la Universidad de San Martín de Porres, concluye que: “El uso de microservicios mejoró el intercambio de datos entre las aplicaciones y servicios lo que generó un impacto positivo en los beneficiarios al cumplir sus expectativas en el modelo de negocio” (1).

(Ruedas, 2017), en su investigación titulada: “Modelo de composición de microservicios para la implementación de una aplicación web de comercio electrónico utilizando Kubernetes”. En la Universidad Nacional del Altiplano, llegó a las siguientes conclusiones (2):

- El modelo de composición de microservicios, basado en la arquitectura de microservicios, para la implementación de una aplicación Web de comercio electrónico utilizando Kubernetes funciona significativamente en comparación con un modelo monolítico en un 104% con respecto a los indicadores de rendimiento, disponibilidad y tiempo de respuesta.
- La evaluación del modelo de composición de microservicios para una aplicación Web de comercio electrónico a través de pruebas de carga

automatizada, permitió validar el comportamiento del modelo de composición de microservicios propuesto al ser comparado con un modelo monolítico.

(López, 2017) con la investigación titulada: “Arquitectura de software basada en microservicios para desarrollo de aplicaciones web de la asamblea nacional”. En la Universidad Técnica del Norte – Ibarra – Ecuador, llegó a las siguientes conclusiones (3):

- La arquitectura propuesta permitió realizar un desarrollo enfocado en la funcionalidad del sistema al dividirlo en pequeñas partes manejables en equipos independientes (trabajo en paralelo).
- Este tipo de diseños de arquitectura bajo microservicios no solo trae consigo una serie de beneficios, sino que también plantea retos en desarrollo de software, operaciones y cultura organizacional ya que plantea una nueva forma de construcción y soporte de sistemas de información, estos retos no solo son de tipo tecnológico sino organizacionales.
- A nivel de infraestructura y operaciones, se ha evidenciado que la aplicación del diseño propuesto plantea una nueva organización de la misma, esto implica la organización tanto de la infraestructura física, como lógica y de comunicaciones para dar soporte a la solución bajo esta arquitectura, todo esto por el uso de tecnologías como la contenerización, balanceo de carga y monitoreo, los cuales difieren del soporte necesario para un sistema monolítico.

(Cols y Salcedo, 2017) en la investigación titulada: “Arquitectura de Microservicios con RESTful”. En la Universidad Politécnica de Madrid – España, llegó a las siguientes conclusiones (4):

- Entre las ventajas observadas podemos mencionar que existen fronteras más firmes entre cada módulo por el hecho de estar en aplicaciones separadas; se pueden realizar despliegues independientes por cada microservicio; cada microservicio, al ser dueño de su propio dominio, puede ser implementada de la manera más conveniente para satisfacer sus necesidades.
- Cada uno tiene total control sobre las decisiones de implementación propias (como en qué lenguaje o framework (marco de trabajo) será implementado o que manejador de base de datos se adapta mejor a sus necesidades); así

como existe mayor facilidad para escalar tanto horizontal como verticalmente cada microservicio.

(Salazar, 2018) en su investigación titulada: “Implementación de arquitectura de microservicios utilizando virtualización por sistema operativo”. En la Universidad de San Carlos de Guatemala, llegó a las siguientes conclusiones (5):

- El implementar una arquitectura de microservicios utilizando virtualización por sistema operativo es el punto de unión para el desarrollo ágil, la entrega continua y la utilización de metodologías como DevOps, ya que, al utilizar contenedores, el tiempo de desarrollo y de despliegue se reduce a tal modo que queda perfecto para ser utilizado con metodologías de desarrollo ágil, como por ejemplo Scrum o Extreme Programming (XP – Programación Extrema)
- La mejor opción para implementar el patrón de arquitectura de microservicios es con el uso de contenedores, ya que estos eliminan las dificultades que conlleva el implementar microservicios.
- Al utilizar contenedores, el tiempo de implementación se mejora considerablemente, tanto al utilizar una arquitectura monolítica como una de microservicios.
- Se debe tener en cuenta los errores que presenta Docker, aunque es una tendencia, no es una tecnología tan madura y presenta problemas a tomar muy en cuenta a la hora de implementarlo en un ambiente productivo.

2.1.2. DIAGNÓSTICO SITUACIONAL

La empresa Financial Systems Company brinda soluciones en software de Cobranzas, con el Producto iCS el cual permite utilizar todos los canales de cobro, incluidos el Call center, Sucursales, Empresas de cobranzas, Gestores de terreno, Comunicaciones escritas (SMS, Cartas - Emails), Estudios jurídicos y Back office en una única plataforma (6), cuenta con las modalidades:

- ICS Enterprise: cuenta con un modelo de licenciamiento tradicional para brindarle a su compañía autonomía en el manejo de datos, infraestructura y la versatilidad necesaria para adaptarse a sus necesidades particulares.

- ICS Express: cuenta con un modelo de arrendamiento de la licencia tipo SaaS (Software as a Service), lo que le permite contar con una amplia flexibilidad, grandes volúmenes de cartera y con número de usuarios ilimitados.

La figura 3 muestra las soluciones los módulos de las soluciones iCS Enterprise/Express: Administración y gestión, Jurídico, Negociaciones, Visitas, Comisiones, Gestión de Bienes y Garantías, SelfService.

Figura 3 - Módulos iCS Entreprise/Express

Fuente: Financial Systems Company, 2018

Servicios adicionales:

Implementación: Equipo completo de especialistas para la implementación de los distintos proyectos, los cuales con atención personalizada y dedicada garantizarán la correcta operación e integración de iCS con sus sistemas contribuyendo a la mejora los procesos de cobranzas actual:

- TIME TO MARKET (Rápido) - En marcha y funcionando en pocos meses
- Entrenamiento a todos los canales y actores involucrados en el proceso de cobranzas.
- Pruebas en conjunto para asegurar la solución implementada.

Consultoría:

Servicios de consultoría enfocados al mejoramiento de los indicadores de gestión y recuperación de cartera, a través de la evaluación de procesos de cobranza, análisis de datos y diseño de estrategias que involucren las mejores prácticas de cobro buscando facilitar la toma de decisiones estratégicas, además de apoyar y optimizar procesos operativos.

- Realizamos diagnósticos completos para identificar las mejoras operativas y/o estratégicas que mejoren la efectividad y productividad de su gestión de cobranzas.
- Realizamos un completo benchmarking (evaluación comparativa) de su proceso actual de cobranza con respecto a las mejores prácticas del sector, para brindarle una orientación especializada de las últimas técnicas avanzadas de un proceso de cobranza exitoso.

2.2. IDENTIFICACIÓN DE OPORTUNIDAD O NECESIDAD EN EL ÁREA DE ACTIVIDAD PROFESIONAL

Los servicios financieros prestados a través de medios digitales están generando una verdadera revolución disruptiva.

Según como lo indica la Investigación realizada por Anif (Asociación Nacional de Instituciones Financieras) para FELABAN (Federación Latinoamericana de Bancos)– CAF (Banco de Desarrollo de América Latina) sobre el Panorama Global del Fintech y resultados de América Latina en octubre de 2018. (7)

El estudio indica que la revolución tecnológica ya ha generado grandes innovaciones con una característica transversal la cual ha radicado en el uso mucho más eficiente de los insumos de datos masivos y de su potencial expansión hacia nueva clientela.

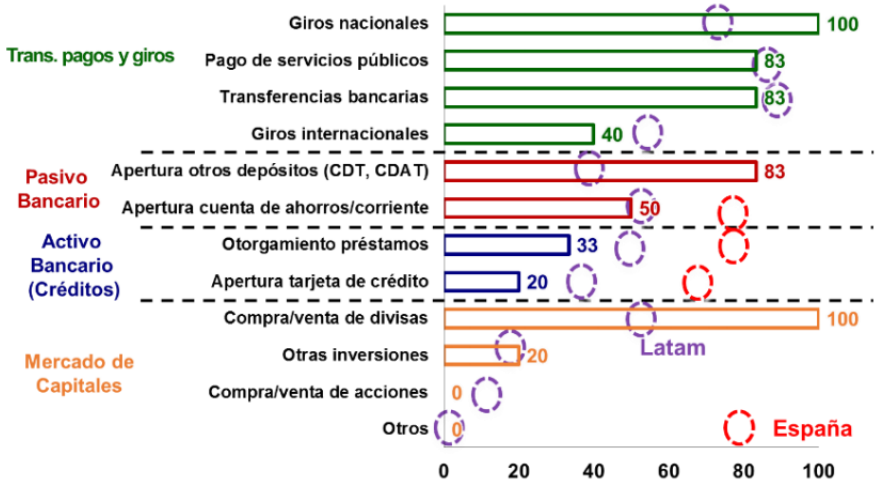
En esto juegan un papel primordial los avances de enganche tecnológico a través de plataformas informativas con algoritmos de Inteligencia Artificial que hoy tienen la capacidad de diseñar “máquinas que aprenden” (ejemplo Deep-Mind de Google y Watson-IBM) (8). Allí también se tienen las ventajas de las redes en línea y la capilaridad de la telefonía móvil, con el despliegue de teléfonos inteligentes de alta capacidad de procesamiento (cada vez a menores costos), convirtiéndose en verdaderas “agencias virtuales”.

En paralelo, los sistemas transaccionales masivos, de pagos P2P y las denominadas “wallets” (Paypal, Wechat-Tencent y hasta Whatsapp) están creciendo como importantes alternativas dentro del contexto del mercado financiero, representando una fuente de tensión para los negocios bancarios tradicionales, así el grado de disrupción actual sea “tan solo” del 28% (según la encuesta PwC).

Elementos de la oferta

En el frente de pagos y transferencias (Figura 4), se encontraron rezagos en la provisión de giros internacionales (40% en Perú vs. 53% del promedio de América Latina) y, en menor medida, en transferencias bancarias (83% vs. 88%) y el pago de servicios públicos (83% vs. 87%). Perú solo sobrepasó al promedio de América Latina en la provisión de giros nacionales (100% vs. 73%).

Figura 4 - Factores de oferta: Servicios que se gestionan totalmente por canales digitales



Fuente: Anif con base en “Encuestas: Los servicios financieros digitales en América Latina” (2018) Anif-FELABAN-CAF y KPMG (2017)

De acuerdo a las investigaciones y crecimiento de las Fintech, FSC encontró la oportunidad de integrar los canales digitales en la gestión de cobranza, tales como:

- SMS
- Whatsapp
- Email
- Chatbot

La oportunidad plantea la creación del producto de cobranza digital (IUCollect) utilizando la arquitectura de microservicios.

Mediante los cuales se pretenden realizar recordatorios de pago por los canales de SMS/Email y Whatsapp.

Adicionalmente se propone la utilización de chatbots (robots de chat) para elaborar un flujo conversacional utilizando Machine Learning (Aprendizaje máquina) de Google e interactuando con los medios SMS y Whatsapp.

Análisis Costo Beneficio:

De acuerdo a la estimación del flujo de caja proyectado (figura 5), es requerida una inversión inicial de 58,500 soles (duración del proyecto 2 meses); durante la proyección a 12 meses, se obtuvo una Tasa interna de retorno del 35% (aceptable), definiéndose como estrategia la implementación en al menos 2 clientes por mes y con una renta de tarifa básica (figura 6).

Figura 5 - Flujo de Caja Proyectado expresado en Soles

	2018					2019							Total
	0	SET	OCT	NOV	DIC	ENE	FEB	MAR	ABR	MAY	JUN	JUL	
Ingresos (tarifa básica)													
Implementación	0	1,998	1,998	1,998	1,998	1,998	1,998	1,998	1,998	1,998	1,998	1,998	21,978
Servicio de sms	0	5,000	10,000	15,000	20,000	25,000	30,000	35,000	40,000	45,000	50,000	55,000	330,000
Servicio de whatsapp	0	9,800	19,600	29,400	39,200	49,000	58,800	68,600	78,400	88,200	98,000	107,800	646,800
Servicio de email	0	360	720	1,080	1,440	1,800	2,160	2,520	2,880	3,240	3,600	3,960	23,760
Total Ingresos	0	17,158	32,318	47,478	62,638	77,798	92,958	108,118	123,278	138,438	153,598	168,758	1,022,538
Costos													
Personal	54,000	0	0	0	0	0	0	0	0	0	0	0	54,000
Tecnología	2,000	0	0	0	0	0	0	0	0	0	0	0	2,000
Instalación	1,500	0	0	0	0	0	0	0	0	0	0	0	1,500
Varios	1,000	0	0	0	0	0	0	0	0	0	0	0	1,000
Mantenimiento plataforma	0	900	900	900	900	900	900	900	900	900	900	900	9,900
Servicio sms	0	2,800	5,600	8,400	11,200	14,000	16,800	19,600	22,400	25,200	28,000	30,800	184,800
Servicio whatsapp	0	6,200	12,400	18,600	24,800	31,000	37,200	43,400	49,600	55,800	62,000	68,200	409,200
Servicio email	0	180	360	540	720	900	1,080	1,260	1,440	1,620	1,800	1,980	11,880
Pago de publicidad	0	250	250	250	250	250	250	250	250	250	250	250	2,750
Total Egresos	58,500	10,330	19,510	28,690	37,870	47,050	56,230	65,410	74,590	83,770	92,950	102,130	677,030
Flujo de caja económico	-58,500	6,828	12,808	18,788	24,768	30,748	36,728	42,708	48,688	54,668	60,648	66,628	
VAN	143,698												
TASA REF	10%												
TIR	35%												

Fuente: Elaboración propia

Figura 6 - Tarifa básica (Renta y Costo) expresada en soles

Renta tarifa básica	Unitario	Total
Implementación	999.00	999.00
Email (1-100,000 mensajes)	180.00	180.00
Whatsapp (1-50,000)	0.10	4,900.00
SMS (1-50,000)	0.05	2,500.00
Total Renta tarifa básica		8579.00

Costo Tarifa básica	Unitario	Total
Email (1-100,000 mensajes)	90	90.00
Whatsapp (1-50,000)	0.062	3,100.00
SMS (1-50,000)	0.028	1,400.00
Total Costo tarifa básica		4,590.00

Fuente: Elaboración propia

2.3. OBJETIVOS DE LA ACTIVIDAD PROFESIONAL

2.3.1. OBJETIVO GENERAL:

Diseñar la arquitectura de software basada en microservicios para la implementación en la aplicación web de cobranza digital en Financial Systems Company Perú SAC.

2.3.2. OBJETIVOS ESPECÍFICOS:

a. Identificar los requerimientos para implementar la arquitectura de software basada en microservicios para la aplicación web de cobranza digital en Financial Systems Company Perú SAC.

b. Diseñar la arquitectura de microservicios para una adecuada integración con los proveedores externos de servicios sms, email y whatsapp.

c. Implementar la arquitectura de software basada en microservicios en la aplicación web de cobranza digital.

d. Evaluar la arquitectura de software basada en microservicios implementada en la aplicación web de cobranza digital.

2.4. JUSTIFICACIÓN DE LA ACTIVIDAD PROFESIONAL

2.4.1. JUSTIFICACIÓN TEÓRICA

El presente trabajo de investigaciones tiene como propósito aportar al conocimiento existente sobre el uso de la arquitectura de microservicios brindando antecedentes de su aplicación y servirá de apoyo para futuras investigaciones dentro de esta área.

2.4.2. JUSTIFICACIÓN PRÁCTICA

Con el desarrollo de la presente investigación se podrá enfocar la aplicación de la arquitectura de microservicios acelerando el ciclo de entrega continua y operación, debido a la naturaleza ágil que se requiere en entidades el tipo tecnológico.

La utilización de la arquitectura basada en microservicios es un enfoque relativamente nuevo que permite la autonomía de cada servicio, siendo de este modo independientes de la plataforma, lenguaje de programación y tecnología a ser utilizada.

2.5. RESULTADOS ESPERADOS

Respecto a los resultados que se muestran líneas abajo, estos fueron definidos por Direccionamiento Estratégico en el año 2018 con la implementación del servicio de IUCollect para cobranza digital:

- Arquitectura de microservicios que faciliten la integración de nuevos canales.
- Integrar las tecnologías de contacto a la gestión de cobro como: chatbots, sms, whatsapp, email.
- Encadenamiento de diferentes canales de contacto en una línea de tiempo.
- Automatizar respuestas a clientes integrando Inteligencia Artificial para interpretación semántica

CAPÍTULO III
MARCO TEÓRICO

3.1. BASES TEÓRICAS DE LAS METODOLOGÍAS O ACTIVIDADES REALIZADAS

3.1.1. MICROSERVICIOS:

(Fugaro y Vocale, 2019), lo definen de la siguiente manera: “La arquitectura de microservicios está basada en el concepto de que una aplicación puede contener una colección de servicios de bajo acoplamiento, independientes y atómicas; los cuales implementan capacidades del negocio.

Utilizando este enfoque, es fácil crear software que divida una aplicación empresarial grande, también conocida como monolítica, con contextos más pequeños y consistentes, conocidos como microservicios.” (9)

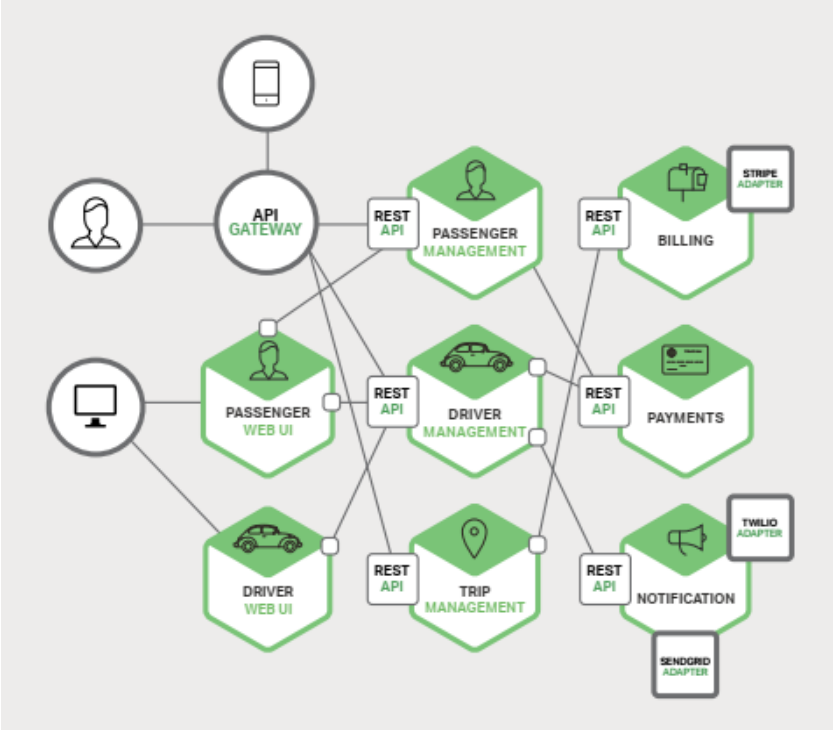
Para, (Indrasiri, 2018, p. 5) “La base de la arquitectura de microservicios consiste en desarrollar una aplicación única como un conjunto de servicios pequeños e independientes que se ejecutan en sus propios procesos, desarrollados y desplegados de forma independiente”. (10)

(Richardson, 2016, p. 4) considera que “Normalmente, un servicio implementa un conjunto de características o funciones distintas, como la gestión de pedidos, la gestión de clientes, etc. Cada microservicio es una mini-aplicación que tiene su propia arquitectura hexagonal que consiste en lógica empresarial junto con varios adaptadores.

Algunos microservicios expondrían una API consumida por otros microservicios o por los clientes de la aplicación. Otros microservicios podrían implementar una interfaz de usuario web. En el tiempo de ejecución, cada instancia es a menudo una máquina virtual en la nube (VM) o un contenedor de Docker.” (11)

La figura 7 muestra una representación de una aplicación monolítica de Taxis descompuesta en microservicios para su consumo a través de celulares y navegadores web.

Figura 7 - Aplicación monolítica descompuesta en microservicios



Fuente: Richardson, 2016

3.1.1.1. Beneficios:

(Indrasiri, 2018, p. 14-15), los principales beneficios de la arquitectura de microservicios (10)son:

- **Despliegue rápido y ágil de funcionalidades de negocio:** La arquitectura de microservicios favorece el desarrollo de servicios autónomos, el cual ayuda con el despliegue rápido y ágil. En una arquitectura convencional, convertir funcionalidades del negocio a

software listo para Producción toma muchos ciclos, principalmente por el tamaño del sistema, codificación y dependencias. Con el desarrollo de servicios autónomos, solo necesita enfocarse en la interface y la funcionalidad del servicio (no la funcionalidad del sistema completo), así como los servicios con los que se comunicará a través de llamadas de red o interfaces de servicios.

- **Remplazable:** debido a la naturaleza autónoma, los microservicios son también reemplazables. Desde que se construyen servicios como entidades independientes, el cual se comunica vía llamadas de red o APIs definidas, podemos fácilmente reemplazar la funcionalidad con otra implementación mejor.

- **Aislamiento de fallos y previsibilidad:** la capacidad de reemplazo también nos ayuda a lograr aislamiento y previsibilidad de fallos. Una aplicación basada en microservicios no puede explotar como una aplicación monolítica convencional debido a la falla de cualquier componente o servicio dado. Tener características de observación adecuadas también nos ayuda a identificar y predecir fallas potenciales.

- **Despliegue ágil y escalabilidad:** es la más importante propuesta de los microservicios. Con las infraestructuras nativas modernas basadas en cloud, la habilidad de desplegar servicios y escalarlos dinámicamente está haciéndose trivial. Cuando construimos servicios autónomos fácilmente podemos contenerlos en tecnologías nativas.

- **Alineamiento con la estructura organizacional:** los microservicios son orientadas a capacidades del negocio, por lo tanto, el dueño de cada servicio puede trabajar directamente con el dueño de la funcionalidad del negocio, de este modo constituir equipos pequeños para ser propietarios del microservicio.

3.1.1.2. Razones para utilizar Microservicios:

(Wolff, 2017, p. 55-67), menciona las razones para utilizar la arquitectura basada en microservicios (12):

Razones Técnicas:

- Reemplazar Microservicios. Trabajar con software antiguo, implica un reto significativo por el hecho de tener código de poca calidad y dificultoso de leer; siendo este el principal motivo de riesgo para reemplazar este software. Los microservicios pueden reemplazar individualmente y separar los procesos del negocio en pequeños procesos de negocio.
- Desarrollo de software sostenible. Con el tiempo, la arquitectura del software puede erosionar y el desarrollo puede tornarse difícil y complejo, los microservicios previenen la erosión arquitectural porque puede ser reemplazado en cualquier momento y ser sostenible.
- Manejo heredado. Reemplazar microservicios solo es posible en sistemas que están implementados basados en microservicios. Cambiar aplicativos heredados se realiza interceptando llamadas de procesamiento, por peticiones http o rest. De este modo se gestiona los sistemas heredados.
- Entrega continua. Permite que el software esté en producción regularmente gracias a un proceso simple y reproducible. Esto es logrado gracias al ciclo de entrega continua. El ciclo es rápido entregando pequeños microservicios que deben ser testeados y puestos en producción; el riesgo del despliegue disminuye porque las unidades son pequeñas y puede hacerse una reversa rápida (rollback).
- Escalamiento. Los microservicios son transmitidos a través de interfaces de red, los cuales pueden ser accedidas por instancias http, cada microservicio puede ejecutarse en algunos servidores y la carga puede ser distribuida a través de diferentes servidores, cada microservicio puede implementar su propio escalamiento.

- Robustez. Una arquitectura de microservicios correctamente diseñada tiene los mecanismos para prevenir fallas a través de la red, en caso de fallas no debe propagarse y manejar respuestas por defecto.
- Libre elección de tecnología. Se puede adoptar un tipo de tecnología mientras se desarrolla el microservicio y en caso no responda acorde a nuestras expectativas, solo debemos escribir nuestro microservicio en otra tecnología.
- Independencia. Debido a la libre elección de tecnología, robustez y que cada microservicio añade su mecanismo de prevención de fallos, logramos mencionar que se obtiene la independencia porque es necesaria poca coordinación entre microservicios.

Razón Organizacional.

Proyectos cortos, los microservicios permiten dividir un proyecto grande en equipos pequeños y que pueden ser independientes el uno del otro, se reduce la necesidad de centralizar la coordinación. Los proyectos largos fallan más frecuentemente que los proyectos cortos, tener alcances cortos en proyectos individuales permite estimaciones más precisas.

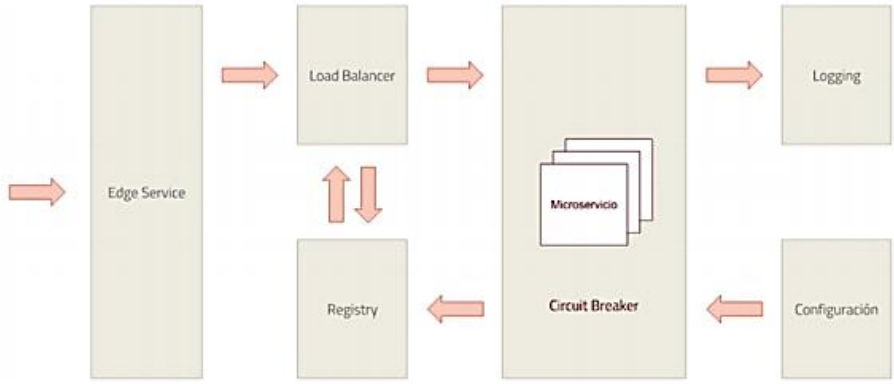
Razón desde la Perspectiva del negocio.

Trabajar historias en paralelo, la distribución en microservicios permite trabajar en diferentes historias en paralelo, cada equipo trabaja en su historia y no afecta a otros microservicios. Cuando un equipo trabaja lento o encuentra obstáculos no afecta negativamente a los otros equipos, por tanto, los riesgos asociados del proyecto son reducidos.

3.1.2. COMPONENTES DE LA ARQUITECTURA.

(Indrasiri, 2018, p. 154), en la figura 8 se muestra la interacción entre los microservicios del negocio y los componentes que la unen (10).

Figura 8 - Componentes básicos en una arquitectura de microservicios



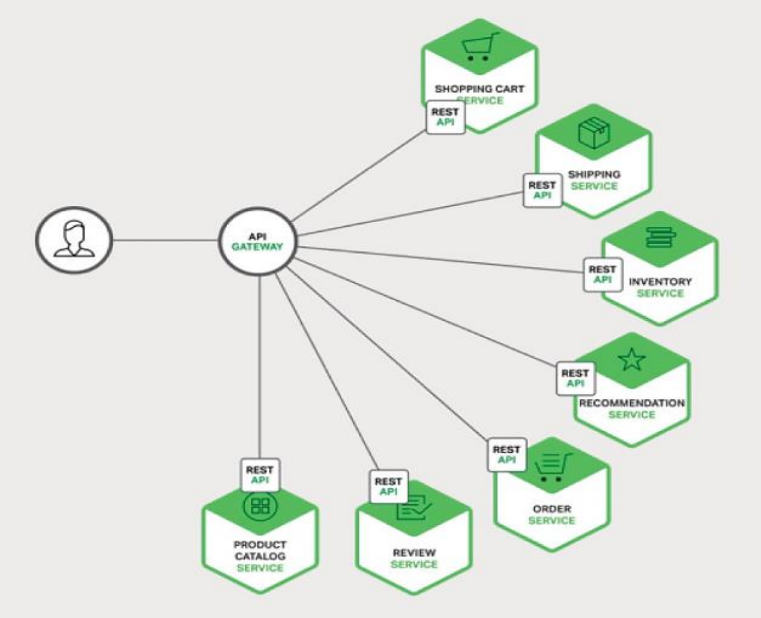
Fuente: Indrasiri, 2018

3.1.2.1. Edge Service/Api Gateway:

(Richardson, 2016, p. 12), define Api Gateway como “servicio de enrutamiento, permite a los clientes de una aplicación de microservicios acceder a servicios individuales por un único punto de entrada”.

En la figura 9 se muestra como un servicio de enrutamiento redirecciona las peticiones de un cliente hacia un determinado servicio.

Figura 9- Utilizando un servicio de enrutamiento



Fuente Richardson,

Para, (Fugaro y Vocale, 2019), “Este servicio es importante para abstraer a todos los microservicios del tráfico externo; mapear las URL de cada microservicio y permite redirigir peticiones a servicios específicos.

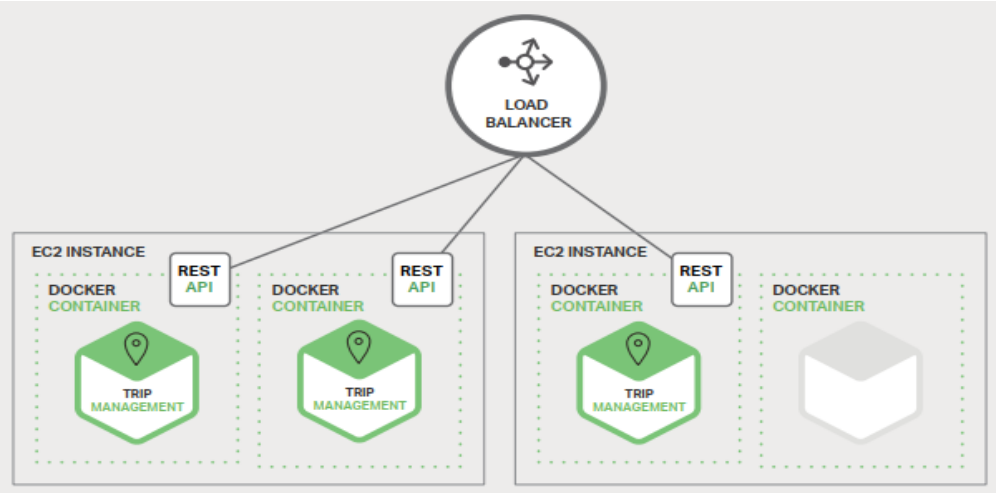
El servicio de enrutamiento encapsula la lógica para llamar correctamente a los servicios backend. Como este tiene un solo punto de entrada, puede añadir otras capacidades como servicios de seguridad, transformar cargas útiles (payloads), monitoreo, caché, puntuar las peticiones por servicio; es también responsable de exponer los servicios con diferentes protocolos”. (9)

3.1.2.2. Balanceador de carga

(Wolff, 2017, p. 144-148), menciona: “Esta es una de las ventajas de los microservicios que cada servicio puede ser escalado individualmente, para distribuir la carga entre las múltiples instancias, el balanceador de carga devuelve información sobre las instancias de los servicios para determinar la carga de cada instancia. Adicionalmente puede eliminar un servicio del balanceador de carga cuando el nodo no reacciona a la petición”. (11)

La figura 10, muestra un ejemplo de balanceador de carga utilizando contenedores Docker y servicios REST.

Figura 10- Balanceador de carga utilizando Docker



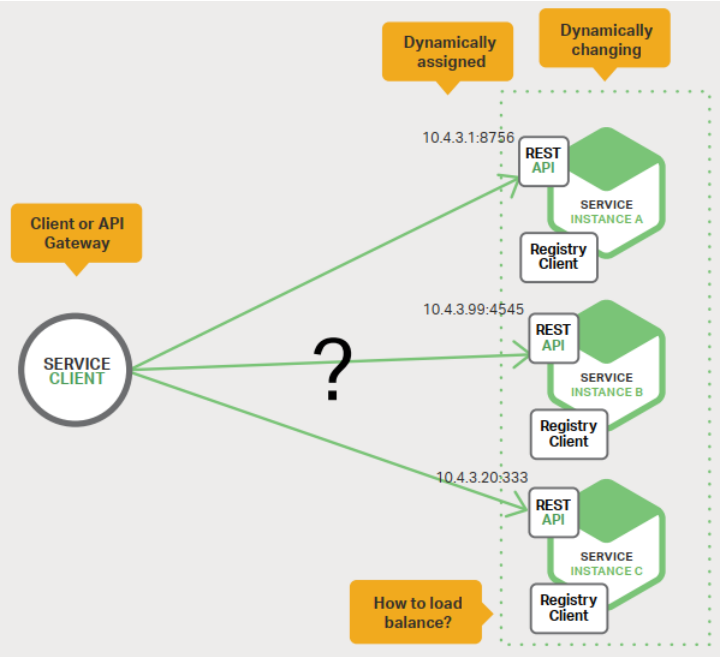
Fuente: Richardson 2016

3.1.2.3. Registro de servicios

(Newman, 2015, p. 236), menciona: “Una vez que se tiene varios microservicios activos, es inevitable conocer donde están ubicados. Por ejemplo, conocer qué microservicio está ejecutándose en un servidor específico, sabiendo eso puede monitorearse; adicionalmente debe proveerse de algún mecanismo para que una instancia pueda registrarse a sí misma, diciendo “estoy aquí” y proveer la forma para encontrarlo una vez sea registrado. Es importante para cuando se tengan microservicios que vayan destruyéndose cada cierto tiempo y creándose nuevas instancias de los servicios.” (13)

La figura 11, muestra la necesidad del registro de servicios, los cuales pueden cambiar dinámicamente por escalamiento o por fallas eventuales.

Figura 11 -Necesidad de Registro de Servicios



Fuente: Richardson, 2016

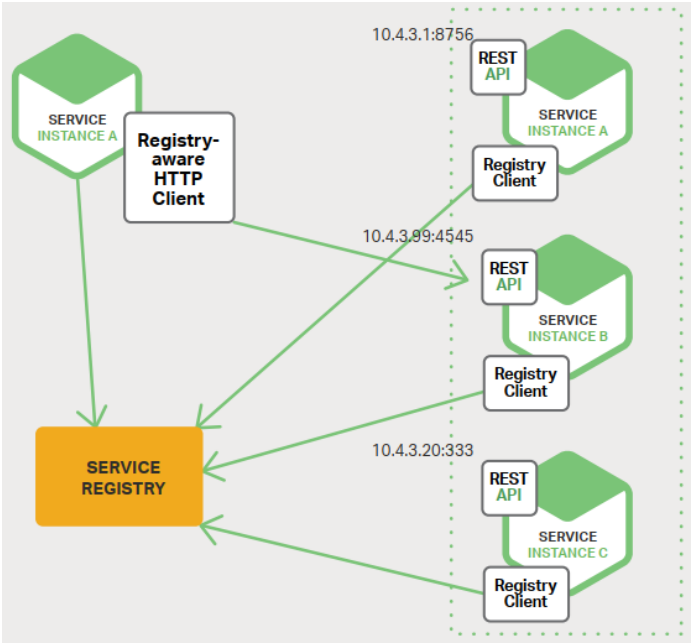
Descubrimiento del lado del cliente:

(Indrasiri, 2018, p. 157), indica "*El cliente es responsable de determinar tanto la ubicación de red de las instancias de los servicios disponibles, así como del balance de carga entre ellos.*

Un registro de servicios es consultado por el cliente, el cual es una base de datos de instancias de servicios disponibles, luego el cliente utiliza un algoritmo de balanceo de carga para seleccionar una instancia de servicio disponible y así proceder con la petición." (10)

La figura 12, muestra el modelo de descubrimiento de servicios por el lado del cliente, el cual consume un servicio de registros disponibles, de ese modo decide cual utilizar.

Figura 12 - Descubriendo servicios por el lado del cliente



Fuente: Richardson, 2016

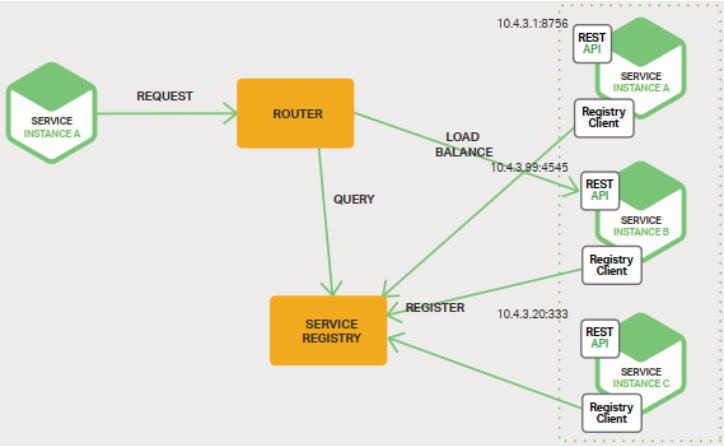
Descubrimiento del lado del servidor

(Indrasiri, 2018, p. 158), también menciona: "*Otro patrón para detección de servicios es encargar la tarea de detección de servicios a un componente intermedio, como un balanceador de carga.*

En este caso, el cliente invoca el servicio con una url predefinida y el balanceador de carga uso esto como una clave para resolver la url real del servicio, de este modo el cliente desconoce la existencia de un registro de servicios". (10)

La figura 13, muestra el modelo de descubrimiento de servicios de lado del servidor, el cual consulta un balanceador de carga para descubrir los servicios disponibles.

Figura 13 - Descubriendo servicios por el lado del servidor



Fuente: Richardson, 2016

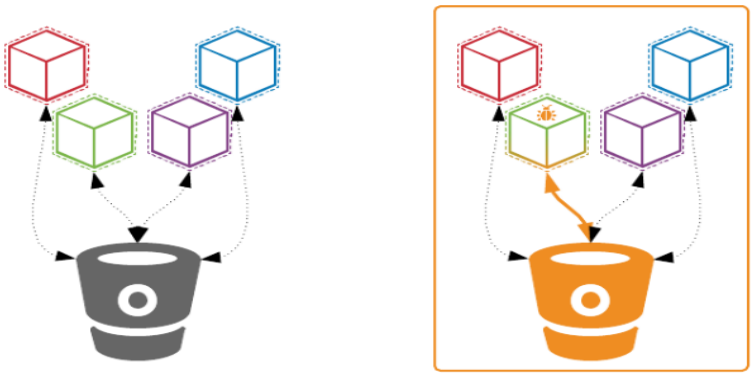
3.1.2.4. Circuit Breaker

(Fugaro y Vocale, 2019), definen Circuit Breaker como “Un circuito cerrado que sirve para degradar funcionalidad cuando la llamada a un método falla. Esto permite al microservicio continuar operando cuando un servicio relacionado falla, previniendo el fallo en cascada y dando tiempo de recuperarse al servicio que falló.

Una vez que el servicio fallido está operativo el circuito cerrado reseteará el flujo para mapearlo nuevamente”. (9)

La figura 14, muestra el aislamiento que tienen los servicios fallidos, impidiendo la propagación a otros servicios.

Figura 14 - Circuit Breaker aislando servicio fallido



Fuente: Fugaro y Vocale, 2019

3.1.2.5. Logging

Para (Wolff, 2017, p. 241-242), Con el log (registro) “una aplicación puede proveer información fácilmente acerca de los eventos ocurridos. Estos pueden ser errores y también puede ser eventos interesantes estadísticamente. Finalmente, la información de registro puede ayudar a los desarrolladores a localizar errores almacenando información detallada”.

Para los microservicios escribir y analizar los archivos de registro es altamente considerado, por lo siguiente:

- Muchas solicitudes solo pueden ser manejadas por la interacción de múltiples microservicios. En ese caso, el archivo de registro de un solo microservicio no es suficiente para comprender la secuencia completa de eventos.
- La carga a menudo se distribuye en varias instancias de un microservicio. Por lo tanto, la información contenida en el archivo de registro de una instancia individual no es muy útil.
- Finalmente, debido a una mayor carga, nuevas versiones o bloqueos, las nuevas instancias de un microservicio se inician constantemente. Los datos de un archivo de registro pueden perderse cuando una máquina virtual se apaga y su disco duro se elimina posteriormente.

Por lo tanto, los archivos de registro deben escribirse en el servidor centralizado de registros, de este modo cada microservicio no utilizará almacenamiento local. (12)

3.1.2.6. Configuración centralizada

(Wolff, 2017, p. 139), indica “En un sistema distribuido se necesita disponer de un componente de configuración centralizada. Esto permite asegurar que los ficheros de configuración sean únicos para todas las instancias de los microservicios”. (12)

3.1.3. DESPLIEGUE Y OPERACIÓN

(Wolff, 2017, p. 254), indica “Los principales objetivos de una arquitectura de microservicios es la velocidad para poner en producción y la capacidad evolutiva de la aplicación. A diferencia de una aplicación monolítica, el desarrollo de microservicios incluye muchos desarrollos individuales.

Cuando múltiples microservicios se ejecutan en una máquina virtual, el despliegue de un microservicio puede influenciar en otro microservicio. El despliegue puede generar una carga alta o introducir cambios en la máquina virtual.

Por lo tanto, los microservicios deben estar aislados cada uno, para lograr una gran estabilidad, de este modo, cuando un microservicio falle debe estar limitado al mismo y no afectar a otros microservicios”. (12)

3.1.3.1. Contenedores Docker

(Wolff, 2017, p. 261) menciona: “El primer objetivo de un contenedor Docker es empaquetar software en unidades estandarizadas que incluyan todo lo necesario para su ejecución, incluidas bibliotecas, herramientas del sistema, código y tiempo de ejecución”. (12)

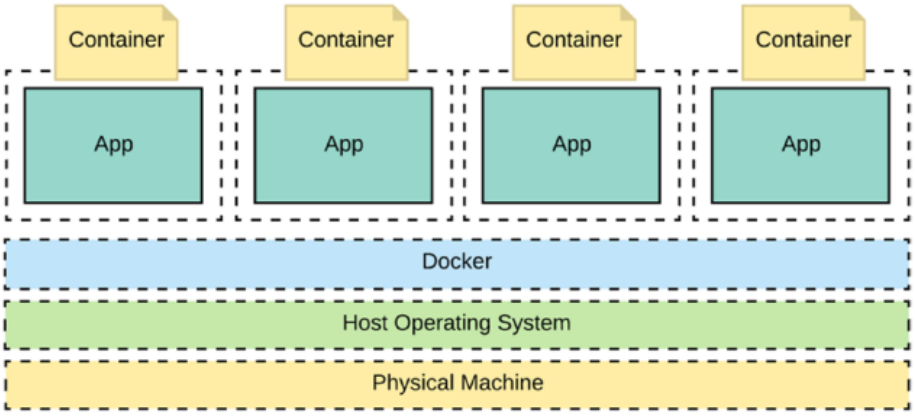
Docker ofrece una virtualización ligera, utilizando diferente tecnología:

- En lugar de una virtualización completa, Docker utiliza contenedores Linux. Esto habilita una alternativa ligera a las máquinas virtuales, todos los contenedores utilizan el mismo kernel. Por lo que se tiene solo una instancia del kernel en memoria. Procesos, conectividad, información del sistema y usuarios son separados de cada uno. En comparación con máquinas virtuales con su propio kernel y también muchos servicios del sistema operativo reduce enormemente la sobrecarga. Esto hace posible tener cientos de contenedores Linux en una simple laptop. Además, un contenedor se ejecuta más rápido que una máquina virtual con su propio kernel y sistema operativo. El contenedor no tiene que bootear el sistema operativo entero, este solo inicia un proceso. El contenedor por sí mismo no añade sobrecarga, solo necesita una

configuración personalizada para los recursos del sistema operativo.

- El sistema de archivos es optimizado, puedes ser utilizado el sistema de archivos básico en modo lectura, al mismo tiempo pueden añadirse archivos del sistema al contenedor, permitiendo la escritura. (12)
- La figura 15, muestra cómo pueden ejecutarse múltiples contenedores en un mismo sistema operativo gracias al modo de empaquetamiento que ofrece Docker.

Figura 15 - Múltiples contenedores ejecutándose en el mismo sistema operativo



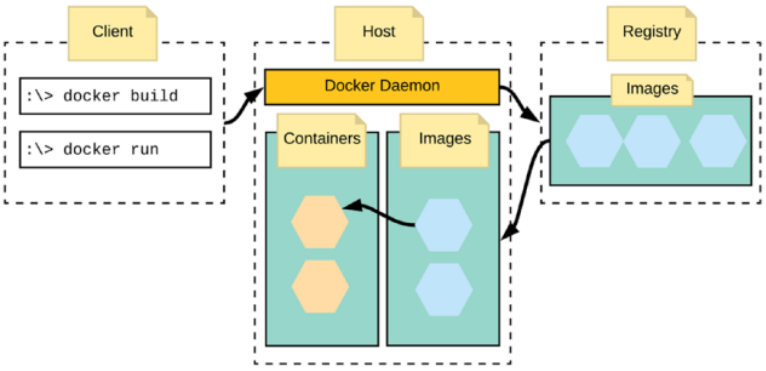
Fuente: Indrasiri, 2018

Arquitectura Docker

(Indrasiri, 2018, p. 224), con respecto a la arquitectura Docker, menciona: "*La siguiente figura muestra la arquitectura Docker en alto nivel, donde tenemos un cliente Docker, host Docker o el motor y el registro. El cliente Docker es la línea de comando que nos permite interactuar. El Docker daemon ejecutándose en el Docker Host escucha las peticiones del cliente y atiende las mismas. El cliente puede comunicarse con el Daemon utilizando REST Api sobre sockets Linux o una interface de red".* (10)

La figura 16, muestra los componentes de alto nivel por el que está compuesto Docker, a nivel cliente, host y registro.

Figura 16 - Arquitectura Docker de alto nivel



Fuente: Indrasiri, 2018

Imágenes Docker

(Indrasiri, 2018, p. 224-225), con respecto a las imágenes Docker menciona: “Una imagen Docker es un paquete que incluye el microservicio realizado con todas las dependencias. La aplicación solo vera las dependencias empaquetadas en la imagen Docker. Se define el sistema de archivos para la imagen y no tendrá acceso al sistema de archivos del host. Una imagen Docker es creada utilizando un Dockerfile. El Dockerfile define todas las dependencias para construir una imagen Docker”. (10)

Registro Docker

(Indrasiri, 2018, p. 225-226), con respecto al registro Docker, menciona “Un registro Docker es un repositorio de imágenes Docker, operando en diferentes niveles. El Docker Hub es un registro Docker público conde cualquiera puede colocar imágenes. Estar centralizado el registro de todas las imágenes Docker le permite compartir y reutilizar”. (10)

Contenedores

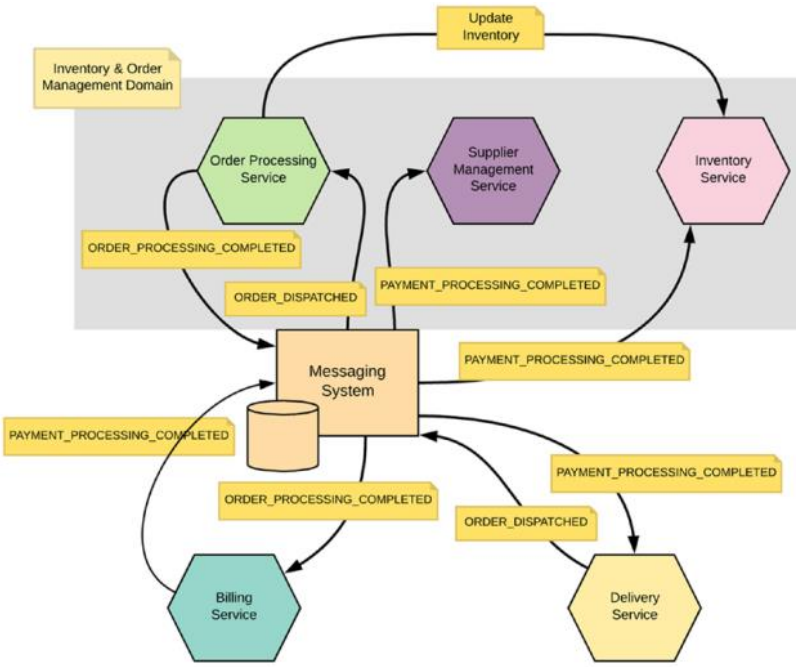
(Indrasiri, 2018, p.226), con respecto a los contenedores Docker, menciona “Un contenedor es una instancia en ejecución de una imagen Docker, las imágenes se convierten en contendores cuando se ejecutan en un motor Docker (Docker engine). Puede tener muchas imágenes en la máquina local, pero no todas ellas ejecutarse al mismo tiempo. Un contendor es un contenedor Linux regular definido por el namespace y los grupos de control”. (10)

Docker Compose

(Indrasiri, 2018, p.230), con respecto a Docker compose, menciona “En la práctica, en el despliegue de un microservicio tenemos más de un servicio, donde cada servicio tiene un contenedor. Puede darse el caso donde un microservicio dependa de otro servicio como una base de datos; la base de datos estará en otro contenedor, pero es parte de la misma aplicación. Docker compose ayuda a definir y gestionar entornos de múltiples contenedores”. (10)

La figura 17, muestra un ejemplo del despliegue de microservicios en múltiples contenedores, de este modo cada microservicio mantiene su independencia.

Figura 17 - Despliegue de microservicios en múltiples contenedores



Fuente: Indrasiri, 2018

3.1.3.2. Orquestación de contendores con Kubernetes

(Indrasiri, 2018, p. 235), menciona “Los contenedores y Docker alivian gran parte de la carga cuando se trabaja con microservicios. Resumiendo, Docker abstrae la máquina (o el computador), mientras que Kubernetes abstrae la red. Google hizo público Kubernetes en 2014 como

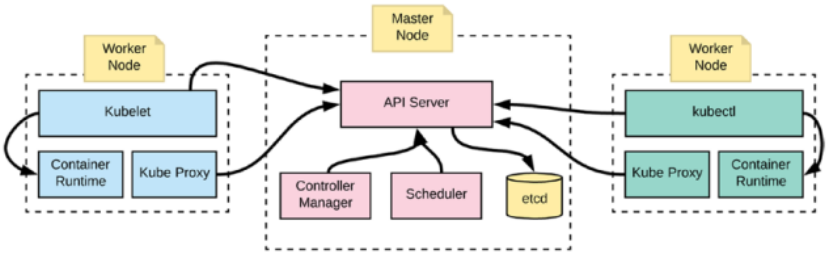
un proyecto de código abierto. Kubernetes le permite implementar y escalar aplicaciones de cualquier tipo, ejecutándose en un contenedor, a miles de nodos sin esfuerzo. Por ejemplo, en un descriptor de despliegue puede especificarse cuantas instancias se necesita desplegar. (10)

Arquitectura Kubernetes

Kubernetes sigue la arquitectura basada en cliente – servidor. Un Kubernetes master y un conjunto de nodos de trabajo conectados a este. El nodo maestro es también conocido como el Plano de control Kubernetes, el cual controla el clúster completo de Kubernetes, considerando estos 4 componentes principales: Api Server, scheduler, controller manager y etcd. (10)

La figura 18, muestra la arquitectura de Kubernetes que está compuesta por el nodo maestro y nodos que pueden activarse de acuerdo a la necesidad.

Figura 18 - Arquitectura de Alto Nivel Kubernetes



Fuente: Indrasiri, 2018

El gestor de controladores es responsable de manejar y trazar todos los nodos en el despliegue Kubernetes, asegurándose que los componentes son replicados y auto escalados correctamente. El etcd es un almacén de datos de alta disponibilidad. Un nodo de trabajo (worker node) consiste en 3 componentes -un kubelet, un contenedor de ejecución y un kube-proxy. (10)

Pod

En Kubernetes se llama pod al grupo de contenedores. Un pod es la unidad mínima de despliegue, no se puede desplegar solo contenedores. Primero necesitamos agrupar uno o más contenedores en un pod. (10)

ReplicaSet

Cuando se despliega un pod en el entorno Kubernetes, puede especificarse cuantas instancias del pod desean mantenerse en ejecución, esto es responsabilidad de ReplicaSet. (10)

Servicio

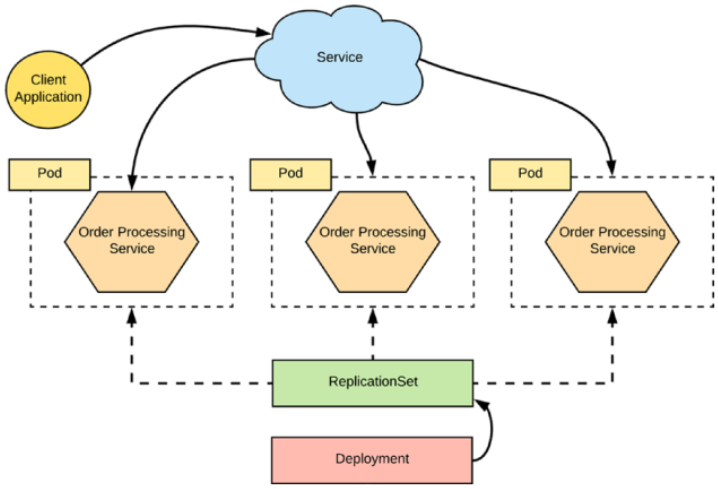
En resumen, un servicio en Kubernetes es un agrupamiento de pods que proveen la misma funcionalidad. (10)

Despliegue

Es un constructor de alto nivel que puede ser utilizado para desplegar y gestionar aplicaciones en un ambiente Kubernetes. (10)

La figura 19, muestra la interacción de los pod, servicios, replicaset y como pueden ser definidos en el despliegue.

Figura 19 - Pod, Servicios, ReplicaSet y Despliegues;



Fuente: Indrasiri 2018

3.1.3.3. Patrones de despliegue de microservicios

(Indrasiri, 2018, p. 253), Basado en los requerimientos del negocio, se describen a continuación los pros y contras de cada patrón de despliegue (10):

Múltiples servicios por host

Este modelo es utilizado cuando se tienen varios microservicios y no se espera que cada microservicio este aislado del otro; el host puede ser la máquina física o máquina virtual. Este patrón no escala si se tiene múltiples microservicios y además no ayuda a lograr los beneficios de una arquitectura de microservicios.

Servicio por host

Con este modelo, cada microservicio es aislado por la máquina física. Este patrón no escala si tiene varios microservicios. Se produce un desgaste de recursos.

Servicio por máquina virtual

Con este modelo, la máquina virtual aísla cada microservicio, este modelo es mejor que el anterior, sin embargo, no puede escalarse cuando se tenga muchos microservicios. Necesitaremos hardware de altas prestaciones para ejecutar varias máquinas virtuales en un host físico.

Servicio por contenedor

Este modelo es el más común y recomendado como modelo de despliegue. Cada microservicio es desplegado en su propio contenedor. Esto hace al microservicio más portable y escalable.

3.1.4. APLICACIÓN WEB

(Wolff, 2017, p. 167-175), indica “El término aplicación web hace referencia a cualquier tipo de aplicación que se visualiza a través de un navegador, utilizando una conexión a Internet para comunicarse con otras aplicaciones, tales como servidores u otros similares. Las aplicaciones Web se distinguen de los sitios web por la interacción avanzada que ofrecen ya que permiten al usuario interactuar y cambiar el contenido, haciendo que este sea dinámico”.

La definición de aplicación web también está relacionada con el modelo de negocio de Software como Servicio (del inglés Software as a Service - SaaS), ya que las aplicaciones SaaS a menudo se entregan como aplicaciones in-browser a los clientes. El denominador común de los sitios web dinámicos que usan viewports (espacio útil de la página donde se desarrolla las actividades del usuario) basados en páginas web, como aplicaciones de comercio electrónico, también pueden ser

vistos como aplicaciones web. Una diferencia importante entre una aplicación web simple y uno avanzado puede determinarse por el uso de tecnologías de comunicación asíncronas. Estas tecnologías permiten a la aplicación web comunicarse con el servidor de forma asíncrona o "en segundo plano", sin limitarse a la cadena de respuesta de acción de los sitios web típicos. Una implementación de comunicación asíncrona utiliza la funcionalidad Javascript del navegador para comunicarse con la aplicación de servidor a través de métodos HTTP y JavaScript Object Notation (JSON) o Extensible Markup Language (XML) para pasar los datos. (12)

La interfaz gráfica de usuario también puede diferir de los sitios web típicos, a menudo una aplicación web imita la interfaz de usuario de aplicaciones de escritorio. En este sentido, el enfoque más directo se toma en que la aplicación se ve gráficamente y es reconocible a la tarea que está realizando, por ejemplo, el editor de documentos de Google o el reproductor de música web de Spotify imitan a sus respectivas versiones de escritorio. (12)

3.1.5. SCRUM:

(Schwaber y Jeff, 2017), en el documento la guía de Scrum, definen: “Scrum es un marco de trabajo para desarrollar, entregar y mantener productos complejos. Esta definición consiste en los roles, eventos y artefactos de Scrum y las reglas que los relacionan”. (14)

3.1.5.1. Teoría de Scrum

Scrum se basa en la teoría de control de procesos empírica o empirismo. El empirismo asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Scrum emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo. (14)

Tres pilares soportan toda la implementación del control de procesos empírico: transparencia, inspección y adaptación.

Transparencia

Los aspectos significativos del proceso deben ser visibles para aquellos que son responsables del resultado. La transparencia requiere que dichos aspectos sean definidos por un estándar común,

de tal modo que los observadores compartan un entendimiento común de lo que se están viendo. (14)

Inspección

Los usuarios de Scrum deben inspeccionar frecuentemente los artefactos de Scrum y el progreso hacia un objetivo para detectar variaciones indeseadas. Su inspección no debe ser tan frecuente como para que interfiera en el trabajo. Las inspecciones son más beneficiosas cuando se realizan de forma diligente por inspectores expertos en el mismo lugar de trabajo. (14)

Adaptación

Si un inspector determina que uno o más aspectos de un proceso se desvían de límites aceptables y que el producto resultante será inaceptable, el proceso o el material que está siendo procesado deben ajustarse. Dicho ajuste debe realizarse cuanto antes para minimizar desviaciones mayores. (14)

Scrum prescribe cuatro eventos formales, contenidos dentro del Sprint, para la inspección y adaptación, tal y como se describen en la sección Eventos de Scrum del presente documento.

- Planificación del Sprint (Sprint Planning)
- Scrum Diario (Daily Scrum)
- Revisión del Sprint (Sprint Review)
- Retrospectiva del Sprint (Sprint Retrospective)

3.1.5.2. Equipo de Scrum

El equipo Scrum consiste en un Dueño de Producto (Product Owner), el Equipo de Desarrollo (Development Team) y un Scrum Master. Los Equipos Scrum son autoorganizados y multifuncionales. Los equipos autoorganizados eligen la mejor forma de llevar a cabo su trabajo y no

son dirigidos por personas externas al equipo. Los equipos multifuncionales tienen todas las competencias necesarias para llevar a cabo el trabajo sin depender de otras personas que no son parte del equipo. (14)

Dueño del Producto

El Dueño de Producto es el responsable de maximizar el valor del producto resultante del trabajo del Equipo de Desarrollo. El cómo se lleva a cabo esto podría variar ampliamente entre distintas organizaciones, Equipos Scrum e individuos.

El Dueño de Producto es la única persona responsable de gestionar la Lista del Producto (Product Backlog). La gestión de la Lista del Producto incluye:

- Expresar claramente los elementos de la Lista del Producto.
- Ordenar los elementos en la Lista del Producto para alcanzar los objetivos y misiones de la mejor manera posible.
- Optimizar el valor del trabajo que el Equipo de Desarrollo realiza.
- Asegurar que la Lista del Producto es visible, transparente y clara para todos y que muestra aquello en lo que el equipo trabajará a continuación.
- Asegurar que el Equipo de Desarrollo entiende los elementos de la Lista del Producto al nivel necesario. (14)

Equipo de Desarrollo

El Equipo de Desarrollo consiste en los profesionales que realizan el trabajo de entregar un Incremento de producto “Terminado” que potencialmente se pueda poner en producción al final de cada Sprint. Un Incremento “Terminado” es obligatorio en la Revisión del Sprint. Solo los miembros del Equipo de Desarrollo participan en la creación del Incremento. (14)

Los Equipos de Desarrollo tienen las siguientes características:

- Son autoorganizados. Nadie (ni siquiera el Scrum Master) indica al Equipo de Desarrollo cómo convertir elementos de la Lista del

Producto en Incrementos de funcionalidad potencialmente desplegables.

- Los Equipos de Desarrollo son multifuncionales, esto es, como equipo cuentan con todas las habilidades necesarias para crear un Incremento de producto.
- Scrum no reconoce títulos para los miembros de un Equipo de Desarrollo independientemente del trabajo que realice cada persona.
- Scrum no reconoce subequipos en los equipos de desarrollo, no importan los dominios que requieran tenerse en cuenta, como pruebas, arquitectura, operaciones o análisis de negocio.
- Los Miembros individuales del Equipo de Desarrollo pueden tener habilidades especializadas y áreas en las que estén más enfocados, pero la responsabilidad recae en el Equipo de Desarrollo como un todo. (14)

Scrum Master

El Scrum Master es responsable de promover y apoyar Scrum como se define en la Guía de Scrum. Los Scrum Masters hacen esto ayudando a todos a entender la teoría, prácticas, reglas y valores de Scrum. (14)

El Scrum Master es un líder que está al servicio del Equipo Scrum. El Scrum Master ayuda a las personas externas al Equipo Scrum a entender qué interacciones con el Equipo Scrum pueden ser útiles y cuáles no. El Scrum Master ayuda a todos a modificar estas interacciones para maximizar el valor creado por el Equipo Scrum.

3.1.5.3. Eventos Scrum

Sprint

El corazón de Scrum es el Sprint, es un bloque de tiempo (time-box) de un mes o menos durante el cual se crea un incremento de producto "Terminado" utilizable y potencialmente desplegable.
Es más conveniente si la duración de los Sprints es consistente a lo largo del esfuerzo de desarrollo. Cada nuevo Sprint comienza inmediatamente

después de la finalización del Sprint anterior. (14)

Los Sprints contienen y consisten en la Planificación del Sprint (Sprint Planning), los Scrums Diarios (Daily Scrums), el trabajo de desarrollo, la Revisión del Sprint (Sprint Review), y la Retrospectiva del Sprint (Sprint Retrospective).

Planificación de Sprint

El trabajo a realizar durante el Sprint se planifica en la Planificación de Sprint. Este plan se crea mediante el trabajo colaborativo del Equipo Scrum completo.

La Planificación de Sprint tiene un máximo de duración de ocho horas para un Sprint de un mes. Para Sprints más cortos el evento es usualmente más corto. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito. El Scrum Master enseña al Equipo Scrum a mantenerse dentro del bloque de tiempo. (14)

Scrum Diario

El Scrum Diario es una reunión con un bloque de tiempo de 15 minutos para el Equipo de Desarrollo. El Scrum Diario se lleva a cabo cada día del sprint. En él, el Equipo de Desarrollo planea el trabajo para las siguientes 24 horas. Esto optimiza la colaboración y el desempeño del equipo inspeccionando el trabajo avanzado desde el último Scrum Diario y haciendo una proyección del trabajo del Sprint a realizar a continuación. El Scrum Diario se realiza a la misma hora y en el mismo lugar todos los días para reducir la complejidad.

El Equipo de Desarrollo usa el Scrum Diario para evaluar el progreso hacia el Objetivo del Sprint y para evaluar qué tendencia sigue este progreso hacia la finalización del trabajo contenido en la Lista de Pendientes del Sprint. El Scrum Diario optimiza las posibilidades de que el Equipo de Desarrollo cumpla el Objetivo del Sprint. Cada día, el Equipo de Desarrollo debería entender cómo intenta trabajar en conjunto como un equipo autoorganizado para lograr el Objetivo del Sprint y crear el Incremento esperado hacia el final del Sprint. (14)

Revisión de Sprint

Al final del Sprint se lleva a cabo una Revisión de Sprint para inspeccionar el Incremento y adaptar la Lista de Producto si fuese necesario. Durante la Revisión de Sprint, el Equipo Scrum y los interesados colaboran acerca de lo que se hizo durante el Sprint. Basándose en esto y en cualquier cambio a la Lista de Producto durante el Sprint, los asistentes colaboran para determinar las siguientes cosas que podrían hacerse para optimizar el valor. Se trata de una reunión informal, no una reunión de seguimiento, y la presentación del Incremento tiene como objetivo facilitar la retroalimentación de información y fomentar la colaboración.

Se trata de una reunión de, a lo sumo, cuatro horas para Sprints de un mes. Para Sprints más cortos, el evento usualmente más corto. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito. El Scrum Master enseña a todos a mantener el evento dentro del bloque de tiempo fijado. (14)

Retrospectiva del Spring

La Retrospectiva de Sprint es una oportunidad para el Equipo Scrum de inspeccionarse a sí mismo y de crear un plan de mejoras que sean abordadas durante el siguiente Sprint.

La Retrospectiva de Sprint tiene lugar después de la Revisión de Sprint y antes de la siguiente Planificación de Sprint. Se trata de una reunión de, a lo sumo, tres horas para Sprints de un mes. Para Sprints más cortos el evento es usualmente más corto. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito.

El Scrum Master se asegura de que la reunión sea positiva y productiva. El Scrum Master enseña a todos a mantener el evento dentro del bloque de tiempo fijado. El Scrum Master participa en la reunión como un miembro del equipo ya que la responsabilidad del proceso Scrum recae sobre él. (14)

3.1.5.4. Artefactos Scrum

Lista de Producto

La Lista de Producto es una lista ordenada de todo lo que se conoce que es necesario en el producto. Es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El Dueño de Producto

(Product Owner) es el responsable de la Lista de Producto, incluyendo su contenido, disponibilidad y ordenación.

Una Lista de Producto nunca está completa. El desarrollo más temprano de la misma solo refleja los requisitos conocidos y mejor entendidos al principio. La Lista de Producto evoluciona a medida que el producto y el entorno en el que se usará también lo hacen. La Lista de Producto es dinámica; cambia constantemente para identificar lo que el producto necesita para ser adecuado, competitivo y útil. Si un producto existe, su Lista de Producto también existe. (14)

La Lista de Producto enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios a realizarse sobre el producto para entregas futuras. Los elementos de la Lista de Producto tienen como atributos la descripción, el orden, la estimación y el valor. Los elementos de La Lista de Producto muchas veces incluyen descripciones de las pruebas que demostrarán la completitud de tales elementos cuando estén “Terminados”.

Lista de pendientes del sprint

La Lista de Pendientes del Sprint es el conjunto de elementos de la Lista de Producto seleccionados para el Sprint, más un plan para entregar el Incremento de producto y conseguir el Objetivo del Sprint. La Lista de Pendientes del Sprint es una predicción hecha por el Equipo de Desarrollo acerca de qué funcionalidad formará parte del próximo Incremento y del trabajo necesario para entregar esa funcionalidad en un Incremento “Terminado”. (14)

La Lista de Pendientes del Sprint hace visible todo el trabajo que el Equipo de Desarrollo identifica como necesario para alcanzar el Objetivo del Sprint. Para asegurar el mejoramiento continuo, la Lista de Pendientes del Sprint incluye al menos una mejora de procesos de alta prioridad identificada en la Retrospectiva inmediatamente anterior.

Historias de usuario

Las Historias de Usuario son un elemento básico para aplicar metodologías Ágiles y especialmente para poder aplicar SCRUM. (14)

Su simpleza hace de esta técnica una gran herramienta para poder tratar casi todos los aspectos necesarios para la creación de productos, especialmente los de software. Y todo se basa en una regla de palabras muy curiosa:

Así, el <rol> que escojamos que va a utilizar la aplicación software, requiere de una <Acción> que ocurra, porque desea cubrir una <funcionalidad>. Corto y conciso. Directo. Claro. (14)

Cuando empezamos a trabajar con Historias de Usuario, siendo el equipo desarrollador o el cliente, lo más fácil es entender cuál es el PARA. puesto que lo que siempre queremos cubrir es la <funcionalidad>. El resto es una forma de responder al PARA.

¡Pero cuidado! Si estas empezando a utilizar el concepto de Historia de Usuario, cuando las escribes se puede confundir el QUIERO y el PARA por la sencilla razón de que el lenguaje te lleva a error. (14).

CAPÍTULO IV
DESCRIPCIÓN DE LAS ACTIVIDADES PROFESIONALES

4.1. DESCRIPCIÓN DE ACTIVIDADES PROFESIONALES

Dentro de las actividades que realizaba dentro del área de proyectos y desarrollo fueron:

- Diseñar la arquitectura de microservicios para el módulo de cobranza digital.
- Gerenciar proyectos de implementación de cobranza digital en clientes locales.
- Desarrollar y documentar el flujo conversacional (DialogFlow) para mensajes de doble vía.
- Desarrollo de interacción de contenedores Docker con codigos QR generados por whatsapp para apertura de líneas.
- Realizar lógica de peticiones sms de una vía y email para interacción con proveedores de envío Amazon.
- Desarrollo de modelo lógico de BD e implementación en Google Cloud Platform.

4.1.1. ENFOQUE DE LAS ACTIVIDADES PROFESIONALES

Las labores fueron realizadas en el área de Proyectos (Project & Development)

Las labores en el área de proyectos correspondían a la principal actividad dentro de la empresa, teniendo participación en los siguientes proyectos más destacados:

- **Cobranza Digital en Financiera Oh - Perú**

Objetivo: Implementar cobranza a través de canales digitales con la aplicación web de cobranza digital (IUCollect).

Resultado: cobranza digital en canales sms y whatsapp, cumpliendo los requisitos de calidad en la integración con la plataforma de cobranza digital.

Satisfacción del cliente: durante la evaluación de satisfacción realizada al líder del proyecto por parte del cliente, se obtuvo la puntuación 4, indicando que las pruebas deben extenderse cuando sea nuevo cliente.

Rol: Líder técnico

Responsabilidades: Elaboración de plantillas de comunicación por canal sms y whatsapp, elaboración de interfaces de entrada y actualización de información.

- **Arquitectura de microservicios para la implementación del módulo de cobranza digital, FSC.**

  Objetivo: Definir la arquitectura basada en microservicios e implementar la aplicación web de cobranza digital.

  Resultado: Fase 1 producto mínimo viable completo, los requisitos definidos en las historias de usuario fueron cumplidos en relación a calidad y tiempo definidos, se cumplieron los criterios de validación definidos en las historias de usuario y durante la implementación se presentaron los entregables requeridos con el grado de calidad y aceptación definidos por la empresa.

  Satisfacción del cliente: Al ser producto interno (de la empresa), el Product Owner entregó la calificación 5.

  Rol: Desarrollador

  Responsabilidades: definición de arquitectura, ajuste de línea de tiempo, creación de lógica para múltiples contenedores Docker, transferencia de lógica de respuestas sms al bot.

En la figura 20, se muestra la vista general de las fases del proyecto, que contiene una fase de visión, construcción y despliegue, la fase de administración y control es para el seguimiento del proyecto.

Figura 20 - Vista general de las fases del proyecto

	Ta	Task Name	Trabajo previsto
✓	XL	CobranzasDigitales_Fase1_FSCColombia_20180517	746 hrs
✓		Fase Visión	15 hrs
✓		Administracion y Control	53 hrs
✓		Fase Construcción	670 hrs
✓	M	Sprint 1 - Modificar base de datos	38 hrs
✓	M	Sprint 2 - Modificar editor de texto	39.5 hrs
✓	M	Sprint 3 - Ajustar linea de tiempo	39.5 hrs
✓	M	Sprint 4 - Modificar proceso de programacion de cartas	39.5 hrs
✓	M	Sprint 5 - Realizar proceso pasar de letter_transaction a sent_letter con estado pendiente	39.5 hrs
✓	M	Sprint 6 - Realizar logica decompilador de sms o whatsapp o email	39.5 hrs
✓	M	Sprint 7 - Realizar script manual paso de informacion de IUCollect a Remoto	39.5 hrs
✓	M	Sprint 8 - Crear logica para desplegar multiples contenedores de Docker	39.5 hrs
✓	M	Sprint 9 - Realizar lógica para extraer el QR	39.5 hrs
✓	M	Sprint 10 - Realizar logica que consume servicios REST remotos	39.5 hrs
✓	M	Sprint 11 - Realizar logica de peticiones de SMS una vía y Email	39.5 hrs
✓	M	Sprint 12 - Realizar proceso encargado de tomar el trabajo pendiente y enrutarlo al canal correspondiente	39.5 hrs
✓	M	Sprint 13 - Realizar lógica de consumo a dialogflow	39.5 hrs
✓	M	Sprint 14 - Realizar logica para exponer Webhook que recibe parámetros desde DialogFlow	39.5 hrs
✓	M	Sprint 15 - Realizar logica para whatsapp o sms en doble via	39.5 hrs
✓	M	Sprint 16 - Realizar lógica para manejo de conversaciones en el bot	39.5 hrs
✓	M	Sprint 17 - Realizar logica para que las lineas se hagan ping entre ellas	39.5 hrs
✓		Fase Despliegue	8 hrs

Fuente: Elaboración propia.

4.1.2. ALCANCE DE LAS ACTIVIDADES PROFESIONALES

Como fue detallado líneas arriba, la función dentro del proyecto de Arquitectura de microservicios e implementación de la aplicación web de cobranza digital fueron realizados los siguientes sprints:

Fase de Visión:

Obtención de requerimientos de usuario (historias de usuario) y Diseño de Arquitectura de Software basada en microservicios.

Construcción de Arquitectura de software basada en microservicios: Realizar la lógica de múltiples contenedores Docker, realizar lógica que consume servicios REST remotos, realizar lógica de envío de sms, email de una sola vía, realizar proceso encargado de tomar el trabajo pendiente y enrutarlo.

Ajustes de la aplicación web de cobranza digital para implementar la arquitectura propuesta: Ajuste de línea de tiempo, modificar proceso de programación de cartas, carga de registros Oracle-mysql, realizar procedimiento que carga trabajo pendiente y lo asigna a la línea activa, definir dialogflow conversacional para cobranza con chatbot.

Fase de despliegue:

Cierre del proyecto y presentación de informe de entrega (validación) interna del proyecto.

4.1.3. ENTREGABLES DEL PROYECTO DE INVESTIGACIÓN

De acuerdo al alcance y los objetivos, se detallan a continuación los entregables que serán validados y aceptados.

Entregables conforme al Objetivo Específico 1:

- Historias de usuario.

Entregables conforme al Objetivo Específico 2:

- Diagrama de Arquitectura de Microservicios

Entregables conforme al Objetivo Específico 3:

- Diagrama de BD Remota.
- DialogFlow para cobranzas con chatbot.

- Ejecución de proceso de estrategias para almacenar lo definido en la línea de tiempo.
- Documentación para enviar mensajes programados por canales digitales.
- Manual de uso del módulo de cobranza digital.
- Documentación de Kubernetes.
- Documentación extraer QR y consumir servicios Rest Remotos.
- Realizar lógica de peticiones de SMS una vía y Email.
- Documentación proceso encargado de tomar trabajo pendiente y enrutarlo al canal correspondiente.
- Documentación de webhook que recibe parámetros desde DialogFlow..

Entregables conforme al Objetivo Especifico 4:

- Informe y estadísticas de uso de la aplicación web de cobranza digital.
- Matriz de utilidad.

4.2. ASPECTOS TÉCNICOS DE LA ACTIVIDAD PROFESIONAL

4.2.1. METODOLOGÍAS

4.2.1.1. Método de investigación

Para la elaboración de la arquitectura basada en microservicios, se hizo uso de métodos generales y específicos que permitieron diseñar la arquitectura basada en microservicios de este modo se logró implementar en la aplicación web de cobranza digital.

a. Método general o teórico de la investigación

Para la investigación se utilizaron los siguientes métodos:

- Método analítico-sintético: se utilizó este método para determinar el diseño de la arquitectura basada en microservicios, sus componentes e interacciones para luego utilizarlos en la implementación de la aplicación web de cobranza digital en la empresa Financial Systems Company SAC.

- Método bibliográfico: se utilizó este método para recopilar literatura existente de investigaciones anteriores y libros relacionados a la arquitectura de microservicios y sus aplicaciones, para luego utilizarlas en la elaboración del diseño de la arquitectura basada en microservicios.

b. Método específico de la investigación

Para la investigación se utilizaron los siguientes métodos:

- Historias de Usuario, este método (ágil) ayudó a especificar los requisitos como lo percibe el usuario, definir una breve descripción de la funcionalidad y criterios de validación (aceptación), aplicables al equipo de desarrollo.
- ATAM (del inglés Architecture Trade-off Analysis Method), este método ayudó a la evaluación del diseño de la arquitectura de software basada en microservicios, permitiendo evaluar los atributos de calidad arquitectónica y los escenarios realizados por el equipo de desarrollo.

4.2.1.2. Alcance de la investigación

- **Tipo de Investigación**

  Es de tipo **tecnológico** porque buscó implementar una oportunidad encontrada en Financial Systems Company y se utilizó tecnología existente y conocimiento teórico para implementar dicha oportunidad.

- **Nivel de investigación**

  Es **aplicada** porque permitió diseñar la arquitectura basada en microservicios e implementarla en la aplicación web de cobranza digital en la empresa Financial Systems Company SAC.

4.2.1.3. Diseño de la investigación

- **Tipo de diseño de la investigación:**

  (Hernández, Fernández y Baptista, 2014, p. 127), menciona "*Los diseños de investigación pueden ser de tipo experimentales o no experimentales*". (15)

  El presente informe obedece a un diseño no experimental, porque se diseñó la arquitectura de microservicios en base a la literatura existente, analizando sus componentes e interacciones para la implementación en la aplicación web de cobranza digital de Financial Systems Company SAC.

- **Tipo de diseño no experimental:**

  (Hernández, Fernández y Baptista, 2014, p. 158), indica "*los diseños no experimentales se pueden clasificar en transaccionales y longitudinales"*. (15)

  La investigación tecnológica tuvo un tipo de diseño transversal o transaccional debido a que diseñar la arquitectura de microservicios permitió la implementación en la aplicación web de cobranza digital.

4.2.2. TÉCNICAS

De acuerdo a la naturaleza de la investigación tecnológica y para diseñar la arquitectura de microservicios, fue necesario contar con técnicas necesarias para la recolección de información. Para esto se hizo uso de fuentes primarias y secundarias, esta información fue necesaria para conocer los componentes e interacciones de la arquitectura de microservicios, analizarlos y seleccionarlas de modo que permita la adecuada implementación en la aplicación web de cobranza digital.

Técnicas utilizadas en la recolección de información:

- Grupos focales, se utilizaron los grupos focales en:

  Fase de Visión, con el objetivo de obtención de requerimientos (historias de usuario), participaron: El Product Owner (CEO), Scrum Master (Gerencia de Desarrollo) y Development Team (02 Analistas).

Sprint reviews (revisión de cada ciclo) con el objetivo de validar avances de desarrollo y cumplimiento de las historias de usuario, así como la elaboración del diseño e implementación de la arquitectura de microservicios, siendo validadas los participantes: El Product Owner (CEO), Scrum Master (Gerencia de Desarrollo) y Development Team (02 Analistas).

Cierre y entrega, con el objetivo de evaluar la arquitectura implementada en la aplicación web de cobranza digital, así como revisar el informe de uso, participaron: El Product Owner (CEO), Scrum Master (Gerencia de Desarrollo) y Development Team (02 Analistas).

- Checklist:

Con el objetivo de evaluar el diseño de la arquitectura de software basada en microservicios con relación a atributos de calidad arquitectónica de acuerdo a los escenarios desarrollados, se utilizó un checklist donde por medio de preguntas se especificaba el cumplimiento de los atributos de calidad a partir de método ATAM, participaron El Product Owner (CEO), Scrum Master (Gerencia de Desarrollo) y Development Team (02 Analistas).

4.2.3. INSTRUMENTOS

Los instrumentos primordiales utilizados son:

- Las historias de usuario que permitieron recopilar los requerimientos tal y como los percibe el usuario y transmitido por el Product Owner (CEO), así como los criterios de validación. Durante la fase de visión, este instrumento formalizó 05 historias de usuario, enunciando las historias, las cuales describen el Rol (permisos de perfil), la característica (funcionalidad) y la razón (resultado); también definiéndose 10 escenarios y criterios de aceptación, los que permitieron la realización de los objetivos específicos 1,2, 3 y 4. Las plantillas de historias de usuario contienen la estructura de cómo ser utilizados y se encuentran en los anexos 8 y 9.

- Checklist de utilidad, cuyo informe corresponde a los resultados obtenidos de la implementación de la arquitectura de software basada en microservicios,

contando con 10 atributos de calidad (Interoperabilidad, mantenibilidad, modificabilidad, instalación, reemplazo, tolerancia a fallos, recuperabilidad, reusabilidad, disponibilidad y seguridad), también consta del medio de cumplimiento (componente arquitectural), la tecnología utilizada y la validación del cumplimiento, este checklist fue aplicado al Development Team (02 analistas), Scrum Master (Gerencia de Desarrollo) y validado por el Product Owner (CEO). Esto fue esencial para el cumplimiento del objetivo general. El modelo utilizado se encuentra en el anexo 10 con los atributos de calidad que deben ser cumplidos.

4.2.4. EQUIPOS Y MATERIALES UTILIZADOS EN EL DESARROLLO DE LAS ACTIVIDADES

Docker: Es un proyecto de código abierto capaz de automatizar el despliegue de aplicaciones dentro de contenedores de software, proporcionándonos así una capa adicional de abstracción y automatización en el nivel de virtualización de sistema operativo sobre Linux.

Utilización: la contenerización mediante Docker fue utilizada para almacenar las aplicaciones de whatsapp y replicarlos (escalar) conforme sea necesario (ver anexo 2)

Kubernetes (referido en inglés comúnmente como “K8s”) es un sistema de código libre para la automatización del despliegue, ajuste de escala y manejo de aplicaciones en contenedores que fue originalmente diseñado por Google y donado a la Cloud Native Computing Foundation (parte de la Linux Foundation). Soporta diferentes ambientes para la ejecución contenedores, incluido Docker.

Utilización: kubernetes fue utilizado mediante los pods, se consideraron dos pods para el nodo principal y el nodo whatsapp (ver anexo 2).

AI Cloud Google - Tensorflow: TensorFlow es una biblioteca de código abierto para aprendizaje automático a través de un rango de tareas, y desarrollado por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos. Actualmente es utilizado tanto en la investigación como en los productos de Google.

Utilización: TensorFlow fue utilizado para establecer los flujos conversacionales durante la interacción de mensajería doble vía sms o whatsapp (ver anexo 3)

Mysql es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

Utilización: mysql fue utilizado como gestor de base de datos del nodo principal, como base de datos remota que permitió almacenar las tablas de las relaciones, servicios rest, trabajo pendiente (ver anexo 2 y 4).

Amazon Web Services (AWS abreviado) es una colección de servicios de computación en la nube pública (también llamados servicios web) que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de Internet por Amazon.com. Es usado en aplicaciones populares como Dropbox, Foursquare, HootSuite. Es una de las ofertas internacionales más importantes de la computación en la nube y compite directamente contra servicios como Microsoft Azure y Google Cloud Platform.

Utilización: los servicios de amazon fueron utilizados para el envío de email, como proveedores de envío (ver anexo 2).

Google Cloud Platform (Nube de Google), es una plataforma que ha reunido todas las aplicaciones de desarrollo web que Google estaba ofreciendo por separado. Es utilizada para crear ciertos tipos de soluciones a través de la tecnología almacenada en la nube y permite por ejemplo destacar la rapidez y la escalabilidad de su infraestructura en las aplicaciones del buscador. Google Cloud se refiere al espacio virtual a través del cual se puede realizar una serie de tareas que antes requerían de hardware o software y que ahora utilizan la nube de Google como única forma de acceso, almacenamiento y gestión de datos.

Utilización: como plataforma principal, que permitió almacenar la base de datos remota, los pods de kubernetes y contenedores Docker de la aplicación, así como los logs de seguimiento (ver anexo 2).

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación ECMAScript,

asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

Utilización: la tecnología de Node.js permitió realizar la interacción con whatsapp y los métodos javascript (ver anexo 2).

PM2 es un administrador de procesos de producción para aplicaciones Node.js con un equilibrador de carga incorporado. Le permite mantener las aplicaciones vivas para siempre, recargarlas sin tiempo de inactividad y facilitar las tareas comunes de administración del sistema.

Utilización: como administrador de procesos de los contenedores whatsapp y la interacción con node.js en los pods de kubernetes (ver anexo 2).

Jakarta EE (Java Platform, Enterprise Edition) es una plataforma de programación para desarrollar y ejecutar software empresarial usando el lenguaje de programación Java.

Utilización**:** el lenguaje de desarrollo de la aplicación web de cobranza digital.

4.3. EJECUCIÓN DE LAS ACTIVIDADES PROFESIONALES

El proyecto tuvo una duración de 02 meses, con la participación del Product Owner (CEO), Scrum Master (Gerente de Desarrollo) y Development Team (02 analistas), las diferentes fases, así como los Sprints (con el enfoque ágil) fueron cumplidos teniendo una variación de 0.98% de SPI (permitido de acuerdo los procedimientos de la empresa) y de acuerdo al cronograma de actividades como se muestra en la tabla 1

El cronograma de actividades del proyecto fue segmentado en 3 fases representativas (Tabla 1):

Fase de Visión: actividades de obtención de requerimientos y elaboración del diseño arquitectónico del software basado en microservicios.

Fase de construcción: actividades de implementación del diseño arquitectónico basado en microservicios y ajustes en aplicativo web de cobranzas digitales para soportar la arquitectura, estas actividades se realizaron utilizando el marco de trabajo Scrum, definiéndose 17 sprints.

Teniendo en cuenta que el envío de SMS o Email puede realizarse a través del proveedor (Web Service REST) seleccionado por el cliente. Lo relacionado con WhatsApp y Chat Bot es suministrado por FSC.

La carga de información del servidor de iUCollect al servidor Remoto se hará por medio de scripts, al igual que reportar la información del estado del envío de los mensajes del Remoto a iUCollect. Con esto, no se pretende que siempre se opere manualmente, sino agilizar la Fase 1, y en las próximas fases se estará incorporando esto de forma automática.

El envío de la información de iUCollect al Servidor del Cliente, está a cargo del área de desarrollo. Sin embargo, aún está por definir, debido a que no está claro la información relacionada a que tablas se va a enviar y que mecanismo vamos a utilizar para esta fase.

La Fase 1 contempla el envío de emails solo texto, es decir que subreportes e imágenes será en otra fase; cabe aclarar que el envío de información del servidor del cliente a iUCollect está a cargo del área de proyectos, por lo tanto, se debe definir claramente con el cliente estrategias, plantillas por canal, y acciones y efectos a utilizar.

Fase de Despliegue: actividades de cierre del proyecto y presentación de informe de entrega (validación) interna del proyecto.

Tabla 1 Fases del proyecto Cobranzas Digitales - Detallado

Actividades	Trabajo	Comienzo	Fin
CobranzasDigitales_Fase1	**746 hrs**	**mar 15/05/18**	**mié 11/07/18**
Fase Visión - Obtención de requerimientos y Diseño de arquitectura de microservicios	**15 hrs**	**mar 15/05/18**	**mié 16/05/18**
Administración y Control	**53 hrs**	**mié 16/05/18**	**mié 4/07/18**
Fase Construcción	**670 hrs**	**jue 17/05/18**	**mié 11/07/18**
Sprint 1 - Modificar base de datos	**38 hrs**	**jue 17/05/18**	**mar 22/05/18**
Sprint 1 - Sprint Planning	2 hrs	jue 17/05/18	jue 17/05/18
Sprint 1 - Daily Scrum	2 hrs	jue 17/05/18	jue 17/05/18
Sprint Development	**28 hrs**	**jue 17/05/18**	**mar 22/05/18**
Sprint 1 - Comité Técnico	2 hrs	jue 17/05/18	jue 17/05/18
Sprint 1 - Desarrollo	20 hrs	jue 17/05/18	lun 21/05/18
Sprint 1 - Pruebas	4 hrs	lun 21/05/18	mar 22/05/18
Sprint 1 - Merge	2 hrs	mar 22/05/18	mar 22/05/18
Sprint 1 - Certificación	4 hrs	mar 22/05/18	mar 22/05/18
Sprint 1 - Sprint Retrospective	2 hrs	mar 22/05/18	mar 22/05/18
Sprint 2 - Modificar editor de texto	**39.5 hrs**	**mar 22/05/18**	**lun 28/05/18**
Sprint 2 - Sprint Planning	1 hr	mar 22/05/18	mar 22/05/18
Sprint 2 - Daily Scrum	1 hr	mar 22/05/18	mar 22/05/18
Sprint Development	**31 hrs**	**mar 22/05/18**	**lun 28/05/18**
Sprint 2 - Comité Técnico	2 hrs	mar 22/05/18	mar 22/05/18
Sprint 2 - Desarrollo	23 hrs	mar 22/05/18	vie 25/05/18
Sprint 2 - Pruebas	4 hrs	vie 25/05/18	lun 28/05/18

Sprint 2 - Merge	2 hrs	lun 28/05/18	lun 28/05/18
Sprint 2 - Certificación	4 hrs	lun 28/05/18	lun 28/05/18
Sprint 2 - Sprint Review	1.5 hrs	lun 28/05/18	lun 28/05/18
Sprint 2 - Sprint Retrospective	1 hr	lun 28/05/18	lun 28/05/18
Sprint 3 - Ajustar línea de tiempo	**39.5 hrs**	**lun 28/05/18**	**vie 1/06/18**
Sprint 3 - Sprint Planning	1 hr	lun 28/05/18	lun 28/05/18
Sprint 3 - Daily Scrum	2 hrs	lun 28/05/18	lun 28/05/18
Sprint Development	**30 hrs**	**lun 28/05/18**	**vie 1/06/18**
Sprint 3 - Comité Técnico	2 hrs	lun 28/05/18	lun 28/05/18
Sprint 3 - Desarrollo	22 hrs	mar 29/05/18	jue 31/05/18
Sprint 3 - Pruebas	4 hrs	jue 31/05/18	vie 1/06/18
Sprint 3 - Merge	2 hrs	vie 1/06/18	vie 1/06/18
Sprint 3 - Certificación	4 hrs	vie 1/06/18	vie 1/06/18
Sprint 3 - Sprint Review	1.5 hrs	vie 1/06/18	vie 1/06/18
Sprint 3 - Sprint Retrospective	1 hr	vie 1/06/18	vie 1/06/18
Sprint 4 - Modificar proceso de programación de cartas	**39.5 hrs**	**vie 1/06/18**	**vie 8/06/18**
Sprint 4 - Sprint Planning	1 hr	vie 1/06/18	vie 1/06/18
Sprint 4 - Daily Scrum	2 hrs	vie 1/06/18	vie 1/06/18
Sprint Development	**30 hrs**	**vie 1/06/18**	**vie 8/06/18**
Sprint 4 - Comité Técnico	2 hrs	vie 1/06/18	vie 1/06/18
Sprint 4 - Desarrollo	22 hrs	vie 1/06/18	jue 7/06/18
Sprint 4 - Pruebas	4 hrs	jue 7/06/18	vie 8/06/18
Sprint 4 - Merge	2 hrs	vie 8/06/18	vie 8/06/18
Sprint 4 - Certificación	4 hrs	vie 8/06/18	vie 8/06/18
Sprint 4 - Sprint Review	1.5 hrs	vie 8/06/18	vie 8/06/18
Sprint 4 - Sprint Retrospective	1 hr	vie 8/06/18	vie 8/06/18
Sprint 5 - Realizar proceso pasar de letter_transaction a sent_letter con estado pendiente	**39.5 hrs**	**vie 8/06/18**	**vie 15/06/18**
Sprint 5 - Sprint Planning	1 hr	vie 8/06/18	vie 8/06/18
Sprint 5 - Daily Scrum	2 hrs	vie 8/06/18	vie 8/06/18
Sprint Development	**30 hrs**	**vie 8/06/18**	**vie 15/06/18**
Sprint 5 - Comité Técnico	2 hrs	vie 8/06/18	vie 8/06/18
Sprint 5 - Desarrollo	22 hrs	vie 8/06/18	jue 14/06/18
Sprint 5 - Pruebas	4 hrs	jue 14/06/18	vie 15/06/18
Sprint 5 - Merge	2 hrs	vie 15/06/18	vie 15/06/18
Sprint 5 - Certificación	4 hrs	vie 15/06/18	vie 15/06/18
Sprint 5 - Sprint Review	1.5 hrs	vie 15/06/18	vie 15/06/18
Sprint 5 - Sprint Retrospective	1 hr	vie 15/06/18	vie 15/06/18
Sprint 6 - Realizar lógica de compilador de sms o whatsapp o email	**39.5 hrs**	**vie 15/06/18**	**jue 21/06/18**
Sprint 6 - Sprint Planning	1 hr	vie 15/06/18	vie 15/06/18
Sprint 6 - Daily Scrum	2 hrs	vie 15/06/18	vie 15/06/18

Sprint Development	**30 hrs**	**vie 15/06/18**	**jue 21/06/18**
Sprint 6 - Comité Técnico	2 hrs	vie 15/06/18	vie 15/06/18
Sprint 6 - Desarrollo	22 hrs	vie 15/06/18	mié 20/06/18
Sprint 6 - Pruebas	4 hrs	mié 20/06/18	jue 21/06/18
Sprint 6 - Merge	2 hrs	jue 21/06/18	jue 21/06/18
Sprint 6 - Certificación	4 hrs	jue 21/06/18	jue 21/06/18
Sprint 6 - Sprint Review	1.5 hrs	jue 21/06/18	jue 21/06/18
Sprint 6 - Sprint Retrospective	1 hr	jue 21/06/18	jue 21/06/18
Sprint 7 - Realizar script manual paso de información de IUCollect a Remoto	**39.5 hrs**	**jue 21/06/18**	**mié 27/06/18**
Sprint 7 - Sprint Planning	1 hr	jue 21/06/18	jue 21/06/18
Sprint 7 - Daily Scrum	2 hrs	jue 21/06/18	jue 21/06/18
Sprint Development	**34 hrs**	**jue 21/06/18**	**mié 27/06/18**
Sprint 7 - Comité Técnico	2 hrs	jue 21/06/18	jue 21/06/18
Sprint 7 - Desarrollo	28 hrs	jue 21/06/18	mié 27/06/18
Sprint 7 - Pruebas	4 hrs	mié 27/06/18	mié 27/06/18
Sprint 7 - Sprint Review	1.5 hrs	mié 27/06/18	mié 27/06/18
Sprint 7 - Sprint Retrospective	1 hr	mié 27/06/18	mié 27/06/18
Sprint 8 - Crear lógica para desplegar múltiples contenedores de Docker	**39.5 hrs**	**jue 17/05/18**	**mié 23/05/18**
Sprint 8 - Sprint Planning	1 hr	jue 17/05/18	jue 17/05/18
Sprint 8 - Daily Scrum	2 hrs	jue 17/05/18	jue 17/05/18
Sprint Development	**34 hrs**	**jue 17/05/18**	**mié 23/05/18**
Sprint 8 - Comité Técnico	2 hrs	jue 17/05/18	jue 17/05/18
Sprint 8 - Desarrollo	28 hrs	jue 17/05/18	mar 22/05/18
Sprint 8 - Pruebas	4 hrs	mar 22/05/18	mié 23/05/18
Sprint 8 - Sprint Review	1.5 hrs	mié 23/05/18	mié 23/05/18
Sprint 8 - Sprint Retrospective	1 hr	mié 23/05/18	mié 23/05/18
Sprint 9 - Realizar lógica para extraer el QR	**39.5 hrs**	**mié 23/05/18**	**mar 29/05/18**
Sprint 9 - Sprint Planning	1 hr	mié 23/05/18	mié 23/05/18
Sprint 9 - Daily Scrum	2 hrs	mié 23/05/18	mié 23/05/18
Sprint Development	**34 hrs**	**mié 23/05/18**	**mar 29/05/18**
Sprint 9 - Comité Técnico	2 hrs	mié 23/05/18	mié 23/05/18
Sprint 9 - Desarrollo	28 hrs	mié 23/05/18	lun 28/05/18
Sprint 9 - Pruebas	4 hrs	lun 28/05/18	mar 29/05/18
Sprint 9 - Sprint Review	1.5 hrs	mar 29/05/18	mar 29/05/18
Sprint 9 - Sprint Retrospective	1 hr	mar 29/05/18	mar 29/05/18
Sprint 10 - Realizar lógica que consume servicios REST remotos	**39.5 hrs**	**mar 29/05/18**	**mar 5/06/18**
Sprint 10 - Sprint Planning	1 hr	mar 29/05/18	mar 29/05/18
Sprint 10 - Daily Scrum	2 hrs	mar 29/05/18	mar 29/05/18
Sprint Development	**34 hrs**	**mar 29/05/18**	**mar 5/06/18**
Sprint 10 - Comité Técnico	2 hrs	mar 29/05/18	mar 29/05/18
Sprint 10 - Desarrollo	28 hrs	mar 29/05/18	mar 5/06/18

Sprint 10 - Pruebas	4 hrs	mar 5/06/18	mar 5/06/18
Sprint 10 - Sprint Review	1.5 hrs	mar 5/06/18	mar 5/06/18
Sprint 10 - Sprint Retrospective	1 hr	mar 5/06/18	mar 5/06/18
Sprint 11 - Realizar lógica de peticiones de SMS una vía y Email	**39.5 hrs**	**mar 5/06/18**	**mié 13/06/18**
Sprint 11 - Sprint Planning	1 hr	mar 5/06/18	mar 5/06/18
Sprint 11 - Daily Scrum	2 hrs	mar 5/06/18	mar 5/06/18
Sprint Development	**34 hrs**	**mar 5/06/18**	**mar 12/06/18**
Sprint 11 - Comité Técnico	2 hrs	mar 5/06/18	mar 5/06/18
Sprint 11 - Desarrollo	28 hrs	mar 5/06/18	mar 12/06/18
Sprint 11 - Pruebas	4 hrs	mar 12/06/18	mar 12/06/18
Sprint 11 - Sprint Review	1.5 hrs	mar 12/06/18	mar 12/06/18
Sprint 11 - Sprint Retrospective	1 hr	mié 13/06/18	mié 13/06/18
Sprint 12 - Realizar proceso encargado de tomar el trabajo pendiente y enrutarlo al canal correspondiente	**39.5 hrs**	**mié 13/06/18**	**mar 19/06/18**
Sprint 12 - Sprint Planning	1 hr	mié 13/06/18	mié 13/06/18
Sprint 12 - Daily Scrum	2 hrs	mié 13/06/18	mié 13/06/18
Sprint Development	**34 hrs**	**mié 13/06/18**	**mar 19/06/18**
Sprint 12 - Comité Técnico	2 hrs	mié 13/06/18	mié 13/06/18
Sprint 12- Desarrollo	28 hrs	mié 13/06/18	lun 18/06/18
Sprint 12 - Pruebas	4 hrs	lun 18/06/18	mar 19/06/18
Sprint 12 - Sprint Review	1.5 hrs	mar 19/06/18	mar 19/06/18
Sprint 12 - Sprint Retrospective	1 hr	mar 19/06/18	mar 19/06/18
Sprint 13 - Realizar lógica de consumo a dialogflow	**39.5 hrs**	**mar 19/06/18**	**lun 25/06/18**
Sprint 13 - Sprint Planning	1 hr	mar 19/06/18	mar 19/06/18
Sprint 13 - Daily Scrum	2 hrs	mar 19/06/18	mar 19/06/18
Sprint Development	**34 hrs**	**mar 19/06/18**	**lun 25/06/18**
Sprint 13 - Comité Técnico	2 hrs	mar 19/06/18	mar 19/06/18
Sprint 13- Desarrollo	28 hrs	mar 19/06/18	vie 22/06/18
Sprint 13 - Pruebas	4 hrs	vie 22/06/18	lun 25/06/18
Sprint 13 - Sprint Review	1.5 hrs	lun 25/06/18	lun 25/06/18
Sprint 13 - Sprint Retrospective	1 hr	lun 25/06/18	lun 25/06/18
Sprint 14 - Realizar lógica para exponer Webhook que recibe parámetros desde DialogFlow	**39.5 hrs**	**lun 25/06/18**	**vie 29/06/18**
Sprint 14 - Sprint Planning	1 hr	lun 25/06/18	lun 25/06/18
Sprint 14 - Daily Scrum	2 hrs	lun 25/06/18	lun 25/06/18
Sprint Development	**34 hrs**	**lun 25/06/18**	**vie 29/06/18**
Sprint 14 - Comité Técnico	2 hrs	lun 25/06/18	lun 25/06/18
Sprint 14- Desarrollo	28 hrs	lun 25/06/18	vie 29/06/18
Sprint 14 - Pruebas	4 hrs	vie 29/06/18	vie 29/06/18
Sprint 14 - Sprint Review	1.5 hrs	vie 29/06/18	vie 29/06/18
Sprint 14 - Sprint Retrospective	1 hr	vie 29/06/18	vie 29/06/18

Sprint 15 - Realizar lógica para whatsapp o sms en doble vía	**39.5 hrs**	**mié 27/06/18**	**jue 5/07/18**
Sprint 15 - Sprint Planning	1 hr	mié 27/06/18	mié 27/06/18
Sprint 15 - Daily Scrum	2 hrs	mié 27/06/18	mié 27/06/18
Sprint Development	**34 hrs**	**mié 27/06/18**	**mié 4/07/18**
Sprint 15 - Comité Tecnico	2 hrs	mié 27/06/18	mié 27/06/18
Sprint 15 - Desarrollo	28 hrs	jue 28/06/18	mié 4/07/18
Sprint 15 - Pruebas	4 hrs	mié 4/07/18	mié 4/07/18
Sprint 15 - Sprint Review	1.5 hrs	jue 5/07/18	jue 5/07/18
Sprint 15 - Sprint Retrospective	1 hr	jue 5/07/18	jue 5/07/18
Sprint 16 - Realizar lógica para manejo de conversaciones en el bot	**39.5 hrs**	**jue 5/07/18**	**mié 11/07/18**
Sprint 16 - Sprint Planning	1 hr	jue 5/07/18	jue 5/07/18
Sprint 16 - Daily Scrum	2 hrs	jue 5/07/18	jue 5/07/18
Sprint Development	**34 hrs**	**jue 5/07/18**	**mié 11/07/18**
Sprint 16 - Comité Técnico	2 hrs	jue 5/07/18	jue 5/07/18
Sprint 16 - Desarrollo	28 hrs	jue 5/07/18	mar 10/07/18
Sprint 16 - Pruebas	4 hrs	mar 10/07/18	mié 11/07/18
Sprint 16 - Sprint Review	1.5 hrs	mié 11/07/18	mié 11/07/18
Sprint 16 - Sprint Retrospective	1 hr	mié 11/07/18	mié 11/07/18
Sprint 17 - Realizar lógica para que las líneas se hagan ping entre ellas	**39.5 hrs**	**vie 29/06/18**	**lun 9/07/18**
Sprint 17 - Sprint Planning	1 hr	vie 29/06/18	vie 29/06/18
Sprint 17 - Daily Scrum	2 hrs	vie 29/06/18	vie 29/06/18
Sprint Development	**34 hrs**	**vie 29/06/18**	**vie 6/07/18**
Sprint 17 - Comité Técnico	2 hrs	vie 29/06/18	vie 29/06/18
Sprint 17 - Desarrollo	28 hrs	vie 29/06/18	vie 6/07/18
Sprint 17 - Pruebas	4 hrs	vie 6/07/18	vie 6/07/18
Sprint 17 - Sprint Review	1.5 hrs	vie 6/07/18	vie 6/07/18
Sprint 17 - Sprint Retrospective	1 hr	lun 9/07/18	lun 9/07/18
Fase Despliegue	**8 hrs**	**mié 11/07/18**	**mié 11/07/18**
Realizar Documento Final del Proyecto	2 hrs	mié 11/07/18	mié 11/07/18
Entrega interna del proyecto	2 hrs	mié 11/07/18	mié 11/07/18
Reunión de cierre	4 hrs	mié 11/07/18	mié 11/07/18

Fuente: Elaboración propia.

4.3.1. CRONOGRAMA DE ACTIVIDADES REALIZADAS

La figura 21, muestra el cronograma de actividades para el proyecto de cobranza digital, fechas de comienzo y fin, así como el trabajo previsto en horas.

Figura 21- Cronograma de Actividades Realizadas (General)

Task Name	Trabajo previsto	Comienzo	Fin
CobranzasDigitales_Fase1_FSCColombia_20180517	746 hrs	mar 15/05/18	mié 11/07/18
Fase Visión	15 hrs	mar 15/05/18	mié 16/05/18
Administracion y Control	53 hrs	mié 16/05/18	mié 4/07/18
Fase Construcción	670 hrs	jue 17/05/18	mié 11/07/18
Sprint 1 - Modificar base de datos	38 hrs	jue 17/05/18	mar 22/05/18
Sprint 2 - Modificar editor de texto	39.5 hrs	mar 22/05/18	lun 28/05/18
Sprint 3 - Ajustar linea de tiempo	39.5 hrs	lun 28/05/18	vie 1/06/18
Sprint 4 - Modificar proceso de programacion de cartas	39.5 hrs	vie 1/06/18	vie 8/06/18
Sprint 5 - Realizar proceso pasar de letter_transaction a sent_letter con estado pendiente	39.5 hrs	vie 8/06/18	vie 15/06/18
Sprint 6 - Realizar logica decompilador de sms o whatsapp o email	39.5 hrs	vie 15/06/18	jue 21/06/18
Sprint 7 - Realizar script manual paso de informacion de IUCollect a Remoto	39.5 hrs	jue 21/06/18	mié 27/06/18
Sprint 8 - Crear logica para desplegar multiples contenedores de Docker	39.5 hrs	jue 17/05/18	mié 23/05/18
Sprint 9 - Realizar lógica para extraer el QR	39.5 hrs	mié 23/05/18	mar 29/05/18
Sprint 10 - Realizar logica que consume servicios REST remotos	39.5 hrs	mar 29/05/18	mar 5/06/18
Sprint 11 - Realizar logica de peticiones de SMS una vía y Email	39.5 hrs	mar 5/06/18	mié 13/06/18
Sprint 12 - Realizar proceso encargado de tomar el trabajo pendiente y enrutarlo al canal correspondiente	39.5 hrs	mié 13/06/18	mar 19/06/18
Sprint 13 - Realizar lógica de consumo a dialogflow	39.5 hrs	mar 19/06/18	lun 25/06/18
Sprint 14 - Realizar logica para exponer Webhook que recibe parámetros desde DialogFlow	39.5 hrs	lun 25/06/18	vie 29/06/18
Sprint 15 - Realizar logica para whatsapp o sms en doble via	39.5 hrs	mié 27/06/18	jue 5/07/18
Sprint 16 - Realizar lógica para manejo de conversaciones en el bot	39.5 hrs	jue 5/07/18	mié 11/07/18
Sprint 17 - Realizar logica para que las lineas se hagan ping entre ellas	39.5 hrs	vie 29/06/18	lun 9/07/18
Fase Despliegue	8 hrs	mié 11/07/18	mié 11/07/18

Fuente: Elaboración propia

4.3.2. PROCESO Y SECUENCIA OPERATIVA DE LAS ACTIVIDADES PROFESIONALES

Fase de visión:

Los requerimientos de implementación de la arquitectura basada en microservicios para el aplicativo web de cobranza digital se consolidaron en historias de usuario (tabla 2) con enunciados de la historia y los criterios de aceptación

Tabla 2 - Historias de Usuario

	Enunciado de la historia			Criterio de aceptación	
Identificador (ID) de la historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario	Resultado / Comportamiento esperado
1	Como un Técnico Administrador o Administrador	Necesito crear plantillas en el editor de texto para los canales WhatsApp, WhatsApp con chat Bot, SMS con chat Bot	Con la finalidad de realizar las plantillas que van a ser utilizadas como primer mensaje al enviar información por estos canales	1	Crear plantillas para los tipos WhatsApp, WhatsApp Chat Bot y SMS Chat Bot, con el manejo de variables actual del editor de texto.
				2	Serán evaluadas las mismas condiciones actuales de un SMS, es decir solo texto y hasta 150 caracteres.
2	Como un Técnico Administrador o Administrador	Necesito crear actividades para las estrategias con canales como WhatsApp, WhatsApp con Chat Bot, SMS con Chat Bot	Con la finalidad de definir en la línea de tiempo que día se debe programar el envío de mensajes a través de canales digitales	1	Crear actividades en la línea del tiempo para una estrategia, la cual pueda usar canales digitales como email, SMS, SMS chat Bot, WhatsApp y WhatsApp chat Bot, teniendo en cuenta las plantillas definidas para estos tipos
				2	Las plantillas que utilice cada canal serán las previamente definidas en el editor de texto
3	Como un usuario que ejecuta procesos batch	Necesito que al ejecutar el proceso de estrategias me quede programado lo	Con la finalidad de realizar cobranza digital enfocada a canales como	1	Programar el envío de mensajes por medio de los canales SMS, SMS Chat Bot, WhatsApp, WhatsApp Chat Bot, Email

		definido en la línea del tiempo para los canales digitales	SMS, SMS Chat Bot, WhatsApp, WhatsApp Chat Bot, Email	2	Estos deben ser alojados en la tabla destinada para esto (letter_transaction)
4	Como un aplicativo que presta servicios digitales	Necesito enviar los mensajes programados por canales digitales para el día pactado	Con la finalidad de realizar una cobranza digital y de esta forma lograr una mayor recuperación de cartera	1	Enviar los SMS / Emails programados para el día a los proveedores indicados por el cliente y entregar un resultado del envío
				2	Enviar los WhatsApp programados para el día haciendo uso de la implementación interna de FSC
				3	Cuando se requiere el uso de Chat Bot se debe validar que este sea en lenguaje natural. Se aclara que este también es haciendo uso de la implementación interna de FSC
5	Como un aplicativo que presta servicios digitales	Necesito obtener el resultado del envío de mensajes por canales digitales	Con la finalidad de obtener una estadística de lo que se cobra por canal digital	1	Almacenar el estado del mensaje enviado por canal digital

Fuente: Elaboración propia.

También fue elaborado el diseño de la arquitectura basada en microservicios, con los componentes e interacciones, se elaboró un borrador de la arquitectura deseada (ver anexo 1), luego de las revisiones con el equipo de desarrollo quedó definido la arquitectura basada en microservicios para producción (ver anexo 2)

Fase de Construcción:

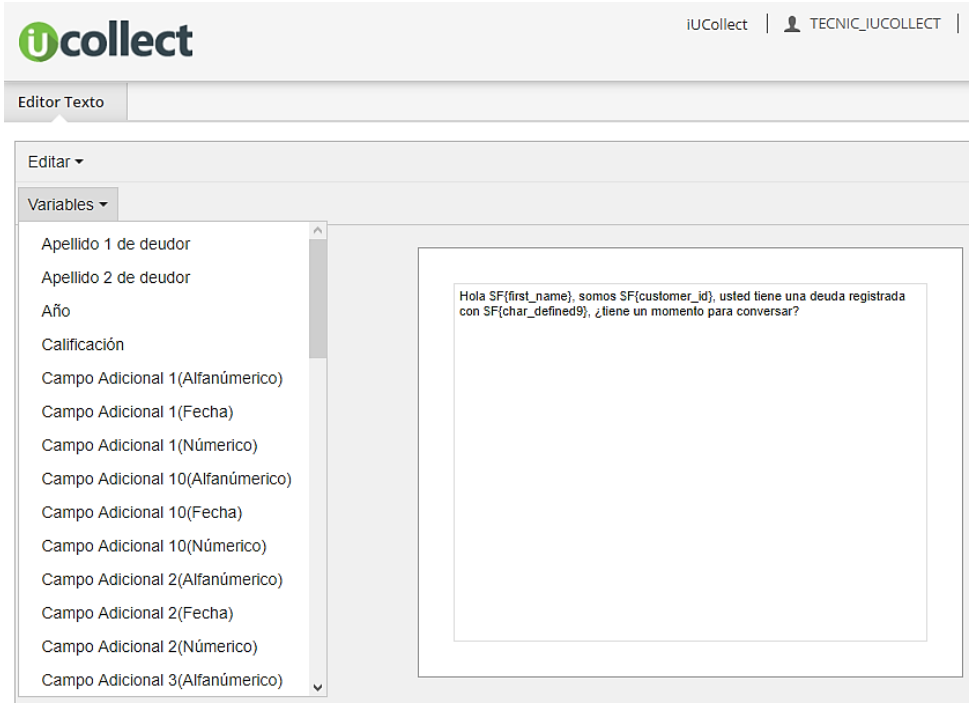
(Sprint 1) Se creó el diagrama de base de datos remota, el cual interactuará con los servicios web, soportará la interacción con los canales sms, email y whatsapp (ver anexo 4). Considerando 18 tablas:

- Bot_control_messages: encargado del control de mensajes, almacenando el teléfono de interacción y el tipo de mensaje a enviar (sms, email, whatsapp)
- Bot_interaction: almacena las interacciones a través de mensajes realizados por el bot (sms o whatsapp)
- Customer: Entidades
- Data_sql: contiene consultas personalizadas para mejorar la interacción por los canales digitales.
- Global_parameters: parámetros de tiempo de sesión y timeouts.
- Parameters_wscollect: parámetros para comunicación con el webservices de iCS.
- Promise: almacena los compromisos de pago obtenidos por la interacción de mensajes.
- Rest_relations: determinación de los servicios rest asociados a los tipos de canal digital.
- Rest_services: web services con la configuración, métodos de invocación, intentos permitidos.
- Status_iucollect_eq: tabla de estados
- Tail_process_bot_messages: seguimiento al hilo conversacional.
- Tmp_workload: flujo de trabajo temporal
- Transaction_log: seguimiento de acciones
- Validation_requests: peticiones de validación

- Whatsapp_relations: asociación de lineas telefónicas y códigos de país para la comunicación via whatsapp.
- Whatsapp_telephones: líneas telefónicas activas para la comunicación por este canal.
- Workload: carga de trabajo
- Workdone: seguimiento trabajo realizado

(Sprint 2) Modificar editor de texto, se realizó la modificación para la utilización de variables que pueden ser incluidos en los sms y whatsapp, de acuerdo a la Figura 22.

Figura 22 - Editor de texto

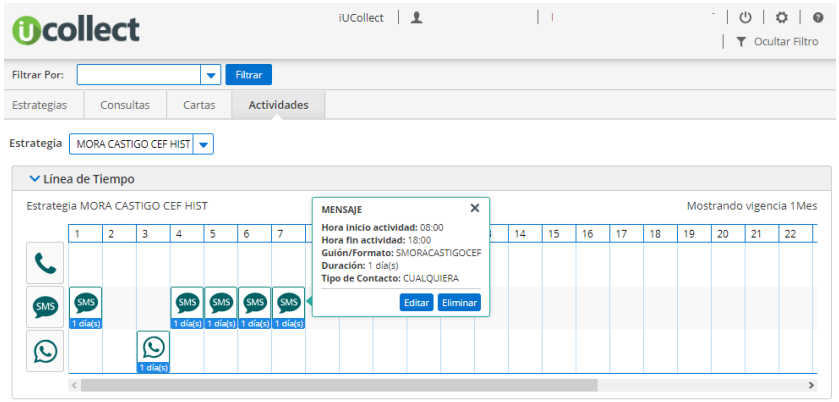


(Sprint 3) Ajustar línea de tiempo, se realizó el ajuste que permite la visualización de las estrategias asociadas a la línea de tiempo y su canal digital, como se visualiza en la Figura 23.

Figura 23 - Línea de tiempo

(Sprint 4) Modificar proceso de programación de cartas, fue modificado para permitir la programación de la carta desde pantalla, como se visualiza en la figura 24.

Figura 24 - Programación de cartas



(Sprint 5) Realizar proceso de pasar de letter_transaction a sent_letter con estado pendiente. Al ejecutar el proceso de estrategias, se almacene lo definido en la línea de tiempo; este proceso fue realizado por scripts en la base de datos (paquetes). (ver anexo 5)

(Sprint 6) Realizar lógica de compilador de sms, whatsapp o email, esta lógica está cubierta por las tablas rest_relations y rest_services.

(Sprint 7) Realizar Manual de paso de información de IUCollect a Remoto, manual adjunto en el anexo 11.

(Sprint 8) Documentación de Kubernetes, documento del sprint adjunto en el anexo 12.

(Sprint 9, 10) Extracción de código QR y lógica de consumo de web services REST, utilizando la tabla whatsapp_telephones y workload. Ver anexo 13.

(Sprint 11) Realizar lógica de peticiones de SMS una vía y Email, visualizar el anexo 14, donde se exponen los métodos de consumos para SMS con el proveedor Intico, estos son los que fueron utilizados y parametrizados en las tablas rest_services y rest_relations.

(Sprint 12) Realizar proceso encargado de tomar el trabajo pendiente y enrutarlo al canal correspondiente, durante la prueba se puede evidenciar por logs del servidor el despliegue de los workers correspondientes, el tiempo de ejecución y la diferencia de datos en las tablas WORK_DONE y WORK_LOAD, documentación en el anexo 15.

(Sprint 13, 14, 15, 16) Fue elaborado el flujo conversacional “DialogFlow” para TensorFlow en el cual se establecen las posibles interacciones considerando validaciones semánticas, esta interacción aplica a los canales de whatsapp y sms (ver anexo 3), asimismo se expusieron los webhook para recibir los estados conversacionales y almacenarlos en la base de datos remota. (ver anexo 16)

Fase de Despliegue:

La aplicación web de cobranza digital que está implementada bajo la arquitectura de microservicios, funciona con estrategias y línea de tiempo como lo indica la figura 25; al definir estrategias se definen las reglas del negocio (periodicidad de uso, días de morosidad, días de ejecución) y en la línea de tiempo se establecen los canales digitales (sms, whatsapp, whatsapp bot, email, sms bot) que quieren ser utilizados durante la gestión de cobranza.

Figura 25 - Cobranzas Digitales

Fuente: Empresa en estudio

Informe (Checklist) de Utilidad, Validación de la arquitectura por ATAM, cumpliendo los criterios de evaluación planteados, así como su implementación tecnológica conforme a las tecnologías seleccionadas. Cabe recalcar que todos los elementos arquitecturales y sus respectivas tecnologías forman parte de una implementación del estilo de arquitectura de microservicios, por lo que hace disponible los beneficios descritos en la sección 3.1.1.1 del marco teórico.

En consecuencia, la aplicación del diseño de arquitectura bajo microservicios es adecuada y cumple a cabalidad los requerimientos de la funcionales y tecnológicos de Financial Systems Company en comparación con el diseño monolítico utilizado en el producto iCS

Durante la fase de despliegue se realizó el **informe de estadísticas de uso** (Anexo 7) de la aplicación web de cobranza digital. Como se visualiza en la figura 26, la gestión de cobro del cliente Financiera Uno utiliza los canales sms, sms bot, whatsapp bot que son los canales digitales por los que se obtiene más contactabilidad con el cliente. La aplicación web de cobranza digital puede gestionar hasta 50,000 mensajes por día y en caso se requiera mayo capacidad, puede escalarse con los pods de kubernetes.

Figura 26 - Reporte Consolidado de cobranza digital

FECHA	CANAL	ENVIADOS	INTERACCIONES	TOTAL MENSAJES
06/09/2018	SMS	5.872	0	5.872
06/09/2018	SMSBOT	2.186	8	2.194
12/09/2018	SMS	2.013	0	2.013
12/09/2018	SMSBOT	606	66	672
13/09/2018	SMS	525	0	525
13/09/2018	WHATSAPPBOT	128	22	150
14/09/2018	SMSBOT	2.581	308	2.889
14/09/2018	WHATSAPPBOT	210	62	272
17/09/2018	WHATSAPPBOT	316	72	388
24/09/2018	SMS	76	0	76
24/09/2018	SMSBOT	561	58	619
27/09/2018	WHATSAPPBOT	86	46	132
TOTAL MENSAJES		**15.160**	**642**	**15.802**

CAPÍTULO V
RESULTADOS

5.1. RESULTADOS FINALES DE LAS ACTIVIDADES REALIZADAS

En este punto, se presentan los resultados de forma general y serán analizados en el punto 5.5.

5.1.1. RESULTADOS CONFORME AL OBJETIVO GENERAL

El diseño de arquitectura basado en microservicios para la implementación en la aplicación web de cobranza digital, según la matriz de Utilidad (tabla 3) cumple los atributos según el objetivo general.

Tabla 3 - Matriz de Utilidad

Atributo	Medio de cumplimiento (componente arquitectural)	Tecnología usada	Cumple
Interoperabilidad	Microservicios, Integración (REST)	Node.js y Java	SÍ
Mantenibilidad	Microservicios	Node.js y Java	SÍ
Modificabilidad	Microservicios, Integración continua, Despliegue continuo	Node.js, Java, M2, Gradle, Jenkins	SÍ
Instalación	Contenerización de aplicaciones	Docker	SÍ
Reemplazo	Replicación de Microservicios	Docker	SÍ

Tolerancia a fallos	Patrón circuit breaker	Kubernetes	SÍ
Recuperabilidad	Replicación de Microservicios, patrón circuit breaker	Kubernetes	SÍ
Reusabilidad	Microservicios	Docker	SÍ
Disponibilidad	Replicación de Microservicios	Docker, Kubernetes	SÍ
Seguridad	Aseguramiento de servicios y canal de comunicación	SSL	SÍ

5.1.2. RESULTADOS CONFORME AL OBJETIVO ESPECÍFICO 1

Identificación de requerimientos (tabla 4) para implementar la arquitectura de software basada en microservicios.

Tabla 4 - Historias de Usuario

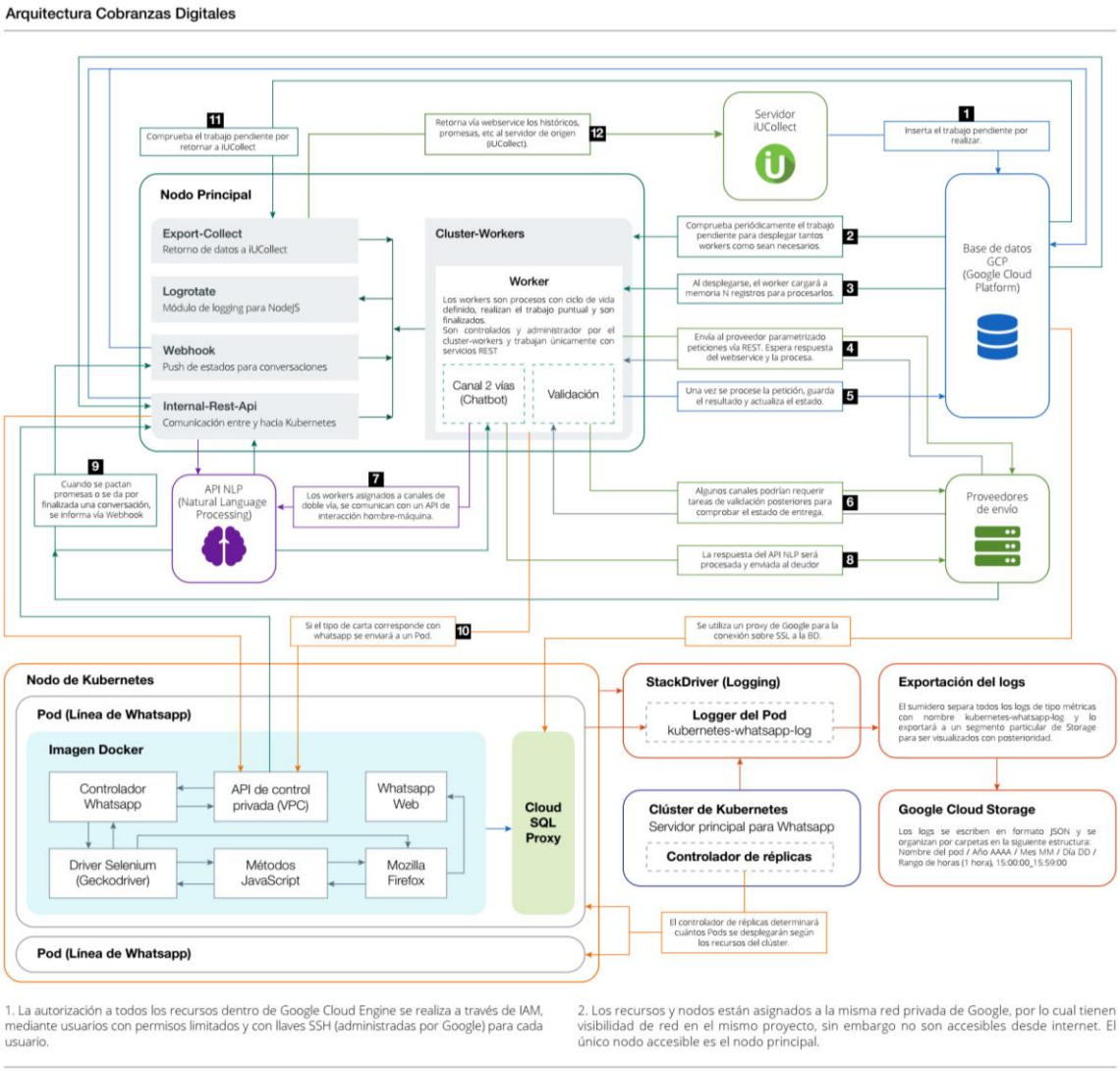
Identificador (ID) de la historia	Enunciado de la historia			Criterio de aceptación	
	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario	Resultado / Comportamiento esperado
1	Como un Técnico Administrador o Administrador	Necesito crear plantillas en el editor de texto para los canales WhatsApp, WhatsApp con chat Bot, SMS con chat Bot	Con la finalidad de realizar las plantillas que van a ser utilizadas como primer mensaje al enviar información por estos canales	1	Crear plantillas para los tipos WhatsApp, WhatsApp Chat Bot y SMS Chat Bot, con el manejo de variables actual del editor de texto.
				2	Serán evaluadas las mismas condiciones actuales de un SMS, es decir solo texto y hasta 150 caracteres.
2	Como un Técnico Administrador o Administrador	Necesito crear actividades para las estrategias con canales como WhatsApp, WhatsApp con Chat Bot, SMS con Chat Bot	Con la finalidad de definir en la línea de tiempo que día se debe programar el envío de mensajes a través de canales digitales	1	Crear actividades en la línea del tiempo para una estrategia, la cual pueda usar canales digitales como email, SMS, SMS chat Bot, WhatsApp y WhatsApp chat Bot, teniendo en cuenta las plantillas definidas para estos tipos
				2	Las plantillas que utilice cada canal serán las previamente definidas en el editor de texto
3	Como un usuario que ejecuta procesos batch	Necesito que al ejecutar el proceso de estrategias me quede programado lo definido en la línea del tiempo para los canales digitales	Con la finalidad de realizar cobranza digital enfocada a canales como SMS, SMS Chat Bot, WhatsApp, WhatsApp Chat Bot, Email	1	Programar el envío de mensajes por medio de los canales SMS, SMS Chat Bot, WhatsApp, WhatsApp Chat Bot, Email
				2	Estos deben ser alojados en la tabla destinada para esto (letter_transaction)

4	Como un aplicativo que presta servicios digitales	Necesito enviar los mensajes programados por canales digitales para el día pactado	Con la finalidad de realizar una cobranza digital y de esta forma lograr una mayor recuperación de cartera	1	Enviar los SMS / Emails programados para el día a los proveedores indicados por el cliente y entregar un resultado del envío
				2	Enviar los WhatsApp programados para el día haciendo uso de la implementación interna de FSC
				3	Cuando se requiere el uso de Chat Bot se debe validar que este sea en lenguaje natural. Se aclara que este también es haciendo uso de la implementación interna de FSC
5	Como un aplicativo que presta servicios digitales	Necesito obtener el resultado del envío de mensajes por canales digitales	Con la finalidad de obtener una estadística de lo que se cobra por canal digital	1	Almacenar el estado del mensaje enviado por canal digital

5.1.3. RESULTADOS CONFORME AL OBJETIVO ESPECÍFICO 2

Diseño de Arquitectura de software para la aplicación web de cobranza digital (Figura 27).

Figura 27 - Arquitectura de Microservicios Cobranza Digital



5.1.4. RESULTADOS CONFORME AL OBJETIVO ESPECÍFICO 3

Criterios de aceptación (tabla 5) implementados con la arquitectura de software basada en microservicios en la aplicación web de cobranza digital

Tabla 5 - Aceptación de la implementación

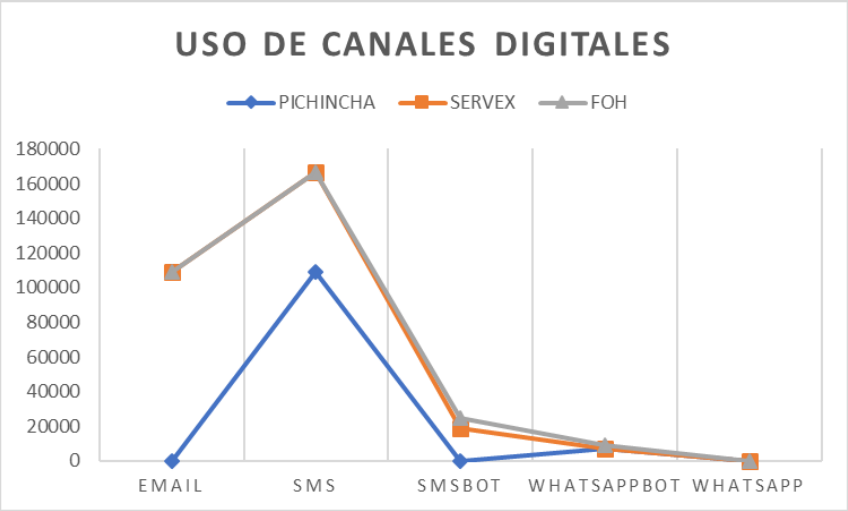
	Enunciado de la historia			Criterio de aceptación		
Identificador (ID) de la historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario	Resultado / Comportamiento esperado	Aceptado
1	Como un Técnico Administrador o Administrador	Necesito crear plantillas en el editor de texto para los canales WhatsApp, WhatsApp con chat Bot, SMS con chat Bot	Con la finalidad de realizar las plantillas que van a ser utilizadas como primer mensaje al enviar información por estos canales	1	Crear plantillas para los tipos WhatsApp, WhatsApp Chat Bot y SMS Chat Bot, con el manejo de variables actual del editor de texto.	SÍ
				2	Serán evaluadas las mismas condiciones actuales de un SMS, es decir solo texto y hasta 150 caracteres.	SÍ
2	Como un Técnico Administrador o Administrador	Necesito crear actividades para las estrategias con canales como WhatsApp, WhatsApp con Chat Bot, SMS con Chat Bot	Con la finalidad de definir en la línea de tiempo que día se debe programar el envío de mensajes a través de canales digitales	1	Crear actividades en la línea del tiempo para una estrategia, la cual pueda usar canales digitales como email, SMS, SMS chat Bot, WhatsApp y WhatsApp chat Bot, teniendo en cuenta las plantillas definidas para estos tipos	SÍ
				2	Las plantillas que utilice cada canal serán las previamente definidas en el editor de texto	SÍ
3		Necesito que al ejecutar el proceso de	Con la finalidad de realizar cobranza	1	Programar el envío de mensajes por medio de los canales SMS, SMS Chat	SÍ

	Como un usuario que ejecuta procesos batch	estrategias me quede programado lo definido en la línea del tiempo para los canales digitales	digital enfocada a canales como SMS, SMS Chat Bot, WhatsApp, WhatsApp Chat Bot, Email		Bot, WhatsApp, WhatsApp Chat Bot, Email	
				2	Estos deben ser alojados en la tabla destinada para esto (letter_transaction)	SÍ
4	Como un aplicativo que presta servicios digitales	Necesito enviar los mensajes programados por canales digitales para el día pactado	Con la finalidad de realizar una cobranza digital y de esta forma lograr una mayor recuperación de cartera	1	Enviar los SMS / Emails programados para el día a los proveedores indicados por el cliente y entregar un resultado del envío	SÍ
				2	Enviar los WhatsApp programados para el día haciendo uso de la implementación interna de FSC	SÍ
				3	Cuando se requiere el uso de Chat Bot se debe validar que este sea en lenguaje natural. Se aclara que este también es haciendo uso de la implementación interna de FSC	SÍ
5	Como un aplicativo que presta servicios digitales	Necesito obtener el resultado del envío de mensajes por canales digitales	Con la finalidad de obtener una estadística de lo que se cobra por canal digital	1	Almacenar el estado del mensaje enviado por canal digital	SÍ

5.1.5. RESULTADOS CONFORME AL OBJETIVO ESPECÍFICO 4

Evaluación de uso de la arquitectura de software basada en microservicios implementada en la aplicación web de cobranza digital (figura 28)

Figura 28 - Uso de canales digitales



5.2. LOGROS ALCANZADOS

Dentro de los logros alcanzados están:

- Arquitectura de microservicios en la implementación de la aplicación web de cobranza digital exitoso y puesto en producción.
- Integración de tecnologías a través de los canales digitales de sms, email y whatsapp.
- Uso masivo de canales digitales.

5.3. DIFICULTADES ENCONTRADAS

Las dificultades encontradas durante la implementación:

a. Aplicar gestión digital por IVR y llamada, no es tan aceptada la comunicación por este medio.

b. Múltiples servidores de servicios por el manejo de varias instancias de los clientes.

c. Definir hora específica para la ejecución de los Jobs de trabajo pendiente en el canal digital y pueda ser procesado correctamente.

d. Demora en el aprendizaje del robot para la interacción basada en condiciones semánticas.

e. Los contextos tienen una vida de sesión de 10 minutos, esto se convierte en una dificultad teniendo en cuenta que una persona puede responder el mensaje 2 horas después de recibir el mensaje.

f. Algunos intents, especialmente los de fallback pueden quedarse sin parámetros, por lo que se mostrarían cosas como “Lo siento #conversationParameters.debtorName, en este…”

g. Requerimiento de almacenar las promesas y se tengan clientes de distintas zonas horarias.

h. Limitación de 155 caracteres a mensajes de whatsapp y sms.

i. La lectura del código QR para ser utilizado por whatsapp web se realiza de forma manual.

j. Las líneas de whatsapp se banean rápidamente

k. El tiempo de aprobación de una conversación por whatsapp tarda aproximadamente 3 meses.

l. Uso de variables limitadas solo a información del deudor.

5.4. PLANTEAMIENTO DE MEJORAS

En el desarrollo de la implementación fueron surgiendo inconvenientes, los cuales se propusieron las siguientes mejoras:

- Debido a que la activación de la línea de whatsapp se realiza de forma manual, se recomendó contratar a un proveedor dedicado, concretamente a Twilio API for WhatsApp, por lo tanto, debería facilitarse los webservices y estructura para el envío de información.
- Recortar el texto cuando excedan los 155 caracteres cuando se definan mensajes de tipo whatsapp y sms, por lo tanto, no debería limitarse desde la pantalla.
- Habilitar las variables del deudor que impliquen valores personalizados, de este modo se adapta a las necesidades del cliente.

5.4.1. METODOLOGÍA PROPUESTA

- Al tener dificultades durante la activación de la línea de whatsapp y realizándose de forma manual, se propone la integración con el proveedor Twilio, quienes son especialistas en la comunicación vía whatsapp
- Suprimir el límite de 155 caracteres desde la pantalla de gestión de cobranza digital (IUCollect), lo cual implica realizar un parche, certificarlo y aplicarlo en producción.
- Existen variables del deudor que no están siendo mostrados al realizar las plantillas de SMS, whatsapp y email, es importante brindarle al cliente la facilidad de utilizar la mayor cantidad de campos posibles que ayuden a la gestión de cobro, mostrar todos los valores posibles implica realizar un parche, certificarlos y aplicarlo en producción.

5.4.2. DESCRIPCIÓN DE LA IMPLEMENTACIÓN

- Proveedor Twilio para la interacción de mensajes whatsapp, cuya especificación es indicada en las figuras 29 y 30.

Figura 29 - Twilio API para Whatsapp

Figura 30 - Twilio Api Rest Ejemplo

NODE.JS C# PHP RUBY PYTHON JAVA CURL 7.X

```java
// Install the Java helper library from twilio.com/docs/java/install

import com.twilio.Twilio;
import com.twilio.rest.api.v2010.account.Message;
import com.twilio.type.PhoneNumber;

public class Example {
    // Find your Account Sid and Token at twilio.com/console
    // DANGER! This is insecure. See http://twil.io/secure
    public static final String ACCOUNT_SID = "ACXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
    public static final String AUTH_TOKEN = "your_auth_token";

    public static void main(String[] args) {
        Twilio.init(ACCOUNT_SID, AUTH_TOKEN);
        Message message = Message.creator(
                new com.twilio.type.PhoneNumber("whatsapp:+15005550006"),
                new com.twilio.type.PhoneNumber("whatsapp:+14155238886"),
                "Hello there!")
            .create();

        System.out.println(message.getSid());
    }
}
```

OUTPUT

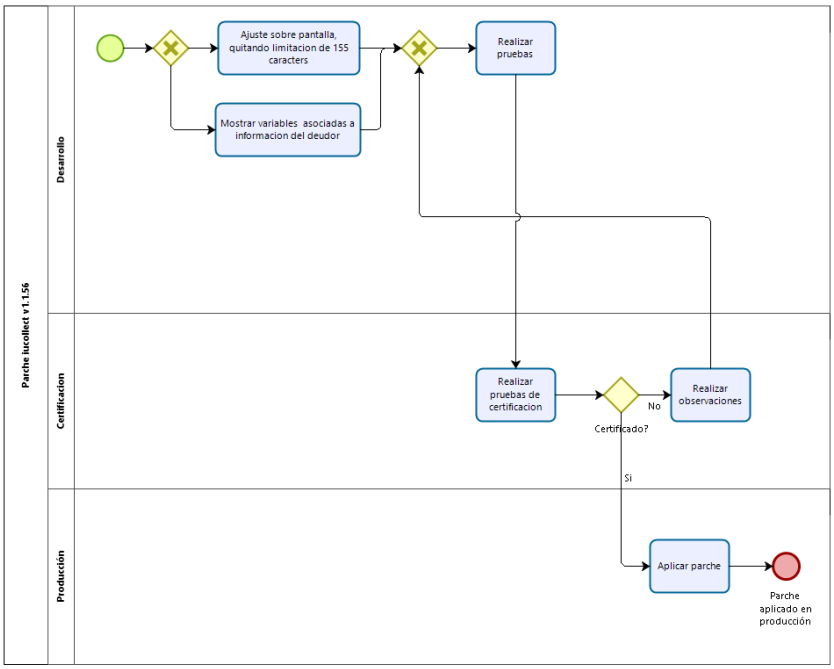

```
{
  "account_sid": "ACXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  "api_version": "2010-04-01",
  "body": "Hello there!",
  "date_created": "Thu, 30 Jul 2015 20:12:31 +0000",
  "date_sent": "Thu, 30 Jul 2015 20:12:33 +0000",
  "date_updated": "Thu, 30 Jul 2015 20:12:33 +0000",
  "direction": "outbound-api",
  "error_code": null,
  "error_message": null,
  "from": "whatsapp:+14155238886",
  "messaging_service_sid": "MGXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  "num_media": "0",
  "num_segments": "1",
  "price": -0.00750,
  "price_unit": "USD",
  "sid": "MMXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  "status": "sent",
```

- Parche para el módulo de gestión digital que permita solventar, de acuerdo al detalle de la figura 31:

  Quitar la limitación de 155 caracteres asociados a un mensaje de tipo whatsapp y sms.

  Mostrar todos los campos desde la información del deudor que puedan ser utilizados en la gestión de cobro.

Figura 31 - Parche para aplicación web de cobranza digital



5.5. ANÁLISIS

5.5.1. ANÁLISIS CONFORME AL OBJETIVO GENERAL

El diseño de arquitectura basado en microservicios cumple los atributos de calidad de acuerdo a las especificaciones de la matriz de utilidad (tabla 6), debido a que cada atributo cuenta con un medio cumplimiento y tecnología usada que fue aceptada por el Product Owner (CEO), en la fase de despliegue, durante la reunión de entrega del proyecto.

Esta matriz de utilidad también fue validada por el Scrum Master (Gerencia de Desarrollo), por lo cual se toma por aceptado y que el objetivo general fue cumplido en su totalidad.

Tabla 6 - Matriz de Utilidad

Atributo	Medio de cumplimiento (componente arquitectural)	Tecnología usada	Cumple
Interoperabilidad	Microservicios, Integración (REST)	Node.js y Java	SÍ
Mantenibilidad	Microservicios	Node.js y Java	SÍ
Modificabilidad	Microservicios, Integración continua, Despliegue continuo	Node.js, Java, M2, Gradle, Jenkins	SÍ
Instalación	Contenerización de aplicaciones	Docker	SÍ
Reemplazo	Replicación de Microservicios	Docker	SÍ
Tolerancia a fallos	Patrón circuit breaker	Kubernetes	SÍ
Recuperabilidad	Replicación de Microservicios, patrón circuit breaker	Kubernetes	SÍ
Reusabilidad	Microservicios	Docker	SÍ
Disponibilidad	Replicación de Microservicios	Docker, Kubernetes	SÍ
Seguridad	Aseguramiento de servicios y canal de comunicación	SSL	SÍ

5.5.2. ANÁLISIS CONFORME AL OBJETIVO ESPECÍFICO 1

Identificación de requerimientos para implementar la arquitectura de software basada en microservicios, los cuales fueron detallados durante la fase de Visión (tabla 7) y transmitidas de forma clara por parte del Product Owner (CEO) hacia el Equipo de Desarrollo, estas historias de usuario reflejan la necesidad del usuario final frente a la oportunidad del uso de canales digitales. Se formalizaron 5 historias de usuario las cuales se utilizaron durante el proceso de diseño de la arquitectura e implementación de la aplicación web de cobranza digital. Por lo cual se dio por cumplido el objetivo específico 1.

Tabla 7 - Historias de Usuario

	Enunciado de la historia			Criterio de aceptación	
Identificador (ID) de la historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario	Resultado / Comportamiento esperado
1	Como un Técnico Administrador o Administrador	Necesito crear plantillas en el editor de texto para los canales WhatsApp, WhatsApp con chat Bot, SMS con chat Bot	Con la finalidad de realizar las plantillas que van a ser utilizadas como primer mensaje al enviar información por estos canales	1	Crear plantillas para los tipos WhatsApp, WhatsApp Chat Bot y SMS Chat Bot, con el manejo de variables actual del editor de texto.
				2	Serán evaluadas las mismas condiciones actuales de un SMS, es decir solo texto y hasta 150 caracteres.
2	Como un Técnico Administrador o Administrador	Necesito crear actividades para las estrategias con canales como WhatsApp, WhatsApp con Chat Bot, SMS con Chat Bot	Con la finalidad de definir en la línea de tiempo que día se debe programar el envío de mensajes a través de canales digitales	1	Crear actividades en la línea del tiempo para una estrategia, la cual pueda usar canales digitales como email, SMS, SMS chat Bot, WhatsApp y WhatsApp chat Bot, teniendo en cuenta las plantillas definidas para estos tipos
				2	Las plantillas que utilice cada canal serán las previamente definidas en el editor de texto

3	Como un usuario que ejecuta procesos batch	Necesito que al ejecutar el proceso de estrategias me quede programado lo definido en la línea del tiempo para los canales digitales	Con la finalidad de realizar cobranza digital enfocada a canales como SMS, SMS Chat Bot, WhatsApp, WhatsApp Chat Bot, Email	1 2	Programar el envío de mensajes por medio de los canales SMS, SMS Chat Bot, WhatsApp, WhatsApp Chat Bot, Email Estos deben ser alojados en la tabla destinada para esto (letter_transaction)
4	Como un aplicativo que presta servicios digitales	Necesito enviar los mensajes programados por canales digitales para el día pactado	Con la finalidad de realizar una cobranza digital y de esta forma lograr una mayor recuperación de cartera	1 2 3	Enviar los SMS / Emails programados para el día a los proveedores indicados por el cliente y entregar un resultado del envío Enviar los WhatsApp programados para el día haciendo uso de la implementación interna de FSC Cuando se requiere el uso de Chat Bot se debe validar que este sea en lenguaje natural. Se aclara que este también es haciendo uso de la implementación interna de FSC
5	Como un aplicativo que presta servicios digitales	Necesito obtener el resultado del envío de mensajes por canales digitales	Con la finalidad de obtener una estadística de lo que se cobra por canal digital	1	Almacenar el estado del mensaje enviado por canal digital

5.5.3. ANÁLISIS CONFORME AL OBJETIVO ESPECÍFICO 2

Fue presentado y aceptado el documento de diseño arquitectónico del software de cobranza digital, el cual tiene las siguientes características:

Base de datos remota almacenada en Google Cloud Platform que tiene comunicación con el servidor de la aplicación web de cobranzas digitales (IUCollect), la mencionada base de datos permite almacenar el trabajo pendiente (mensajes por canal digital).

Los cluster workers son procesos que realizan trabajos puntuales y trabajan únicamente con servicios rest, valida el trabajo pendiente almacenado en la base datos, los carga en memoria y determina el canal por el cual debe ser transmitido (sms, email, whatsapp, bot), finalmente envía los mensajes al canal parametrizado (con servicios rest).

Los workers asignados a canales de doble vía se comunican con el API NLP (Natural Language Processing) y realizan las actividades definidas en TensorFlow (Anexo 3), mediante estas interacciones brinda una respuesta al canal de doble vía y establece una conversación, al pactarse una promesa o finalizar la conversación se transmite el resultado a través del webhook habilitado (Anexo 16)

Los workers asignados a canales whatsapp permiten inicializar un pod en kubernetes por cada línea de whatsapp (este modo de trabajo fue sugerido para mejora, utilizando el api Twilio)

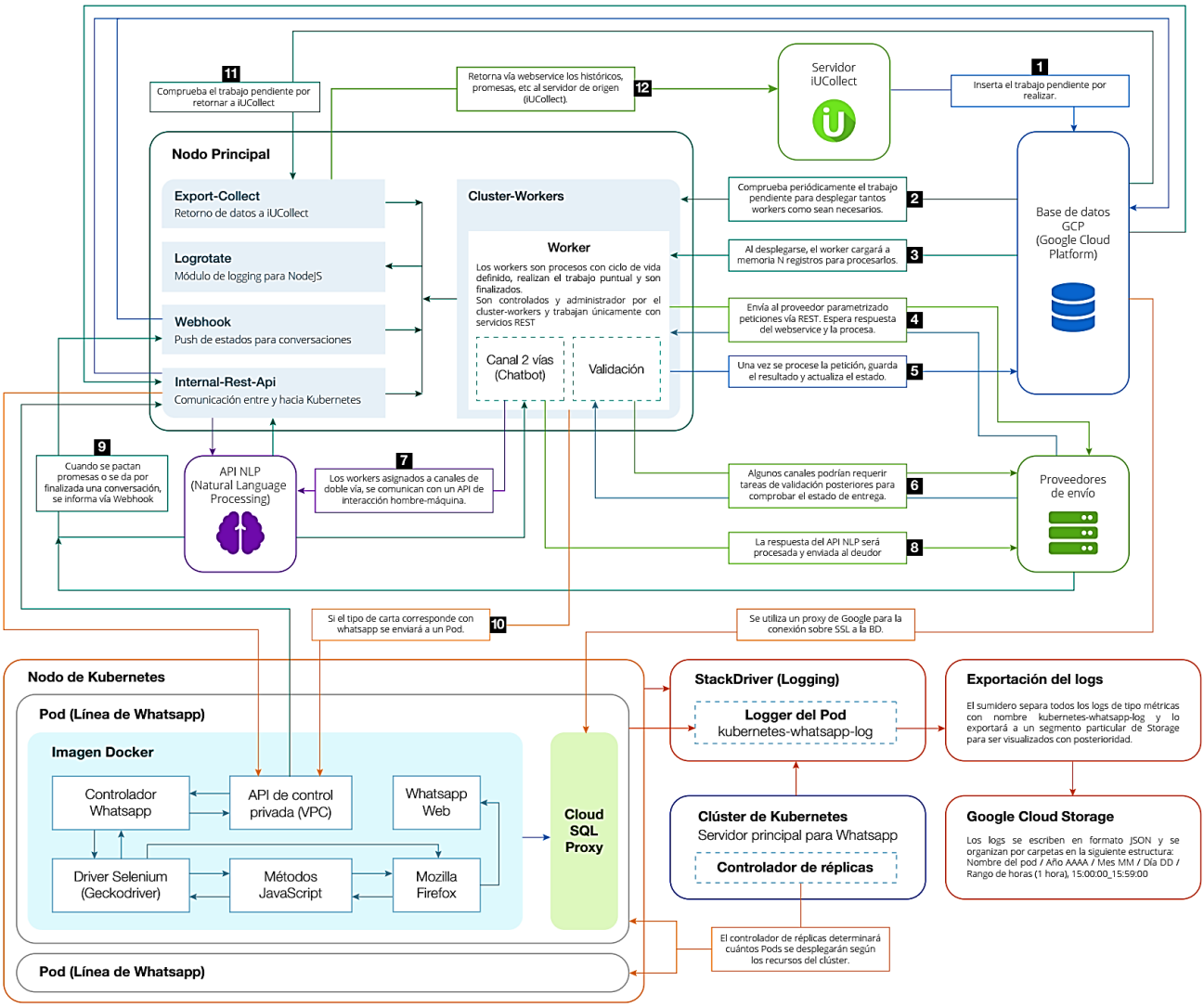
Los workers asignados a canales email se comunican con el API de Amazon y realizan el procesamiento de forma encapsulada (Anexo 13), al finalizar la tarea se recibe la respuesta de envío a través del webhook.

Los workers asignados a canales sms se comunican con el API de Intico (Anexo 14), por lo cual se construye el mensaje validando que no exceda los 155 caracteres permitidos por la plataforma.

Se mantienen logs de seguimiento para posibles anomalías en los pods de Kubernetes y los contenedores Docker, los cuales están alojados en Google Cloud Platform, se visualiza en la figura 32.

Con estas especificaciones se cumplió el objetivo específico 2.

Figura 32 - Arquitectura de Microservicios Cobranza Digital



1. La autorización a todos los recursos dentro de Google Cloud Engine se realiza a través de IAM, mediante usuarios con permisos limitados y con llaves SSH (administradas por Google) para cada usuario.

2. Los recursos y nodos están asignados a la misma red privada de Google, por lo cual tienen visibilidad de red en el mismo proyecto, sin embargo no son accesibles desde internet. El único nodo accesible es el nodo principal.

5.5.4. ANÁLISIS CONFORME AL OBJETIVO ESPECÍFICO 3

Es importante indicar que estos enunciados fueron optimistas, debido a que se presentaron inconvenientes en las historias de usuario durante la implementación, detalladas a continuación:

ID 1: funcionalidad: Necesito crear plantillas en el editor de texto para los canales WhatsApp, WhatsApp con chat Bot, SMS con chat Bot, dificultad: por limitaciones de la aplicación web de cobranza digital, se utilizaron las variables del deudor como

tamaños fijos, fue aplicado un parche que recortó el tamaño del campo cuando el mensaje está construido.

ID 4: Como un aplicativo que presta servicios digitales; dificultad: los mensajes de doble vía no pueden tardar más de 10 minutos, la plataforma TensorFlow lo acepta, sin embargo, responde con valores nulos, esto no fue especificado, por lo que se incluyó como una restricción durante la implementación.

Durante la implementación además se presentaron los siguientes entregables:

- Diagrama de BD Remota (Anexo 4).
- DialogFlow para cobranzas con chatbot (Anexo 3).
- Ejecución de proceso de estrategias para almacenar lo definido en la línea de tiempo (Anexo 5).
- Documentación para enviar mensajes programados por canales digitales (Anexo 6).
- Manual de uso del módulo de cobranza digital (Anexo 11).
- Documentación de Kubernetes (Anexo 12).
- Documentación extraer QR y consumir servicios REST Remotos (Anexo 13).
- Realizar lógica de peticiones de SMS una vía y Email (Anexo 14).
- Documentación proceso encargado de tomar trabajo pendiente y enrutarlo al canal correspondiente (Anexo 15).
- Documentación de Webhook que recibe parámetros desde DialogFlow (Anexo 16).

Con respecto al objetivo de implementar la arquitectura de software basada en microservicios en el aplicativo web de cobranza digital se da por cumplido, porque fueron aceptados los entregables y todos los criterios de aceptación evidenciados en la tabla 8.

Tabla 8 - Aceptación de la implementación

Identificador (ID) de la historia	Enunciado de la historia			Criterio de aceptación		
	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario	Resultado / Comportamiento esperado	Aceptado
1	Como un Técnico Administrador o Administrador	Necesito crear plantillas en el editor de texto para los canales WhatsApp, WhatsApp con chat Bot, SMS con chat Bot	Con la finalidad de realizar las plantillas que van a ser utilizadas como primer mensaje al enviar información por estos canales	1	Crear plantillas para los tipos WhatsApp, WhatsApp Chat Bot y SMS Chat Bot, con el manejo de variables actual del editor de texto.	SÍ
				2	Serán evaluadas las mismas condiciones actuales de un SMS, es decir solo texto y hasta 150 caracteres.	SÍ
2	Como un Técnico Administrador o Administrador	Necesito crear actividades para las estrategias con canales como WhatsApp, WhatsApp con Chat Bot, SMS con Chat Bot	Con la finalidad de definir en la línea de tiempo que día se debe programar el envío de mensajes a través de canales digitales	1	Crear actividades en la línea del tiempo para una estrategia, la cual pueda usar canales digitales como email, SMS, SMS chat Bot, WhatsApp y WhatsApp chat Bot, teniendo en cuenta las plantillas definidas para estos tipos	SÍ
				2	Las plantillas que utilice cada canal serán las previamente definidas en el editor de texto	SÍ
3	Como un usuario que ejecuta procesos batch	Necesito que al ejecutar el proceso de estrategias me quede programado lo definido en la línea del tiempo para los canales digitales	Con la finalidad de realizar cobranza digital enfocada a canales como SMS, SMS Chat Bot, WhatsApp, WhatsApp Chat Bot, Email	1	Programar el envío de mensajes por medio de los canales SMS, SMS Chat Bot, WhatsApp, WhatsApp Chat Bot, Email	SÍ
				2	Estos deben ser alojados en la tabla destinada para esto (letter_transaction)	SÍ

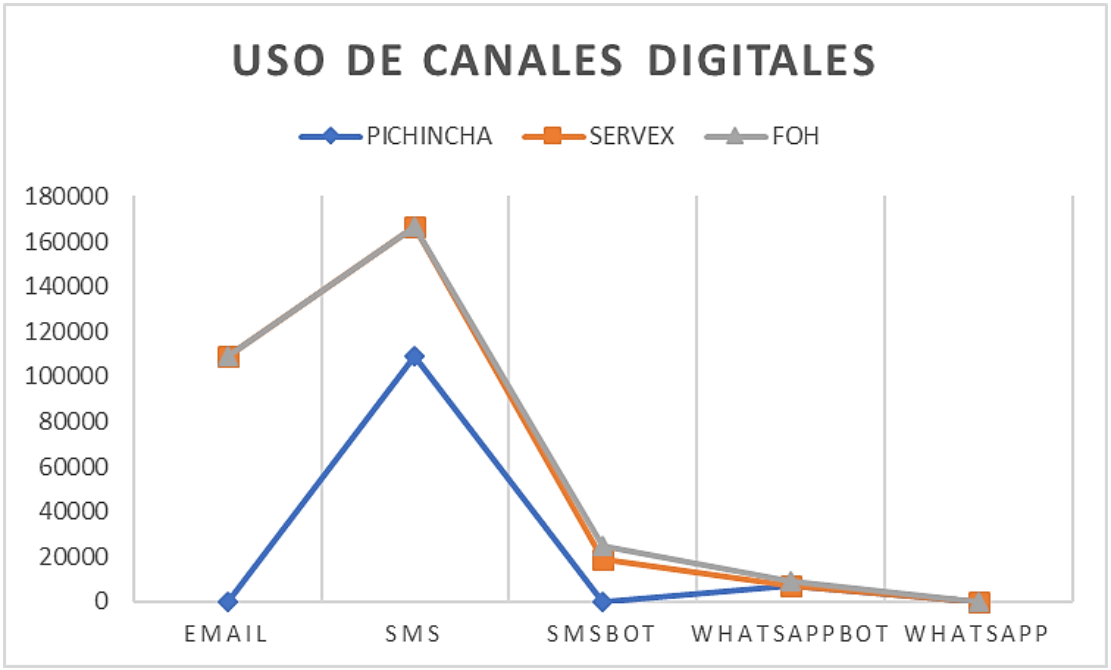
4	Como un aplicativo que presta servicios digitales	Necesito enviar los mensajes programados por canales digitales para el día pactado	Con la finalidad de realizar una cobranza digital y de esta forma lograr una mayor recuperación de cartera	1	Enviar los SMS / Emails programados para el día a los proveedores indicados por el cliente y entregar un resultado del envío	SÍ
				2	Enviar los WhatsApp programados para el día haciendo uso de la implementación interna de FSC	SÍ
				3	Cuando se requiere el uso de Chat Bot se debe validar que este sea en lenguaje natural. Se aclara que este también es haciendo uso de la implementación interna de FSC	SÍ
5	Como un aplicativo que presta servicios digitales	Necesito obtener el resultado del envío de mensajes por canales digitales	Con la finalidad de obtener una estadística de lo que se cobra por canal digital	1	Almacenar el estado del mensaje enviado por canal digital	SÍ

5.5.5. ANÁLISIS CONFORME AL OBJETIVO ESPECÍFICO 4

Evaluación de uso de la arquitectura de software basada en microservicios implementada en la aplicación web de cobranza digital, consolidando la información mostrada en la Figura 33 y que utiliza como insumos las figuras 34, 35 y 36 (entregable resultado del envío de mensajes por canales digitales - Anexo 7); se obtiene que al implementarse la aplicación web de cobranza digital, fueron mayormente utilizados los canales tipo email, sms y smsbot. La tendencia se ve marcada porque es un servicio nuevo que está ofreciendo Financial Systems Company y por lo tanto se inicia con volúmenes bajos, sin embargo, la respuesta que tuvo la arquitectura basada en microservicios en esta implementación permitió utilizar los canales digitales sin interrupción permitiendo así su uso continuo.

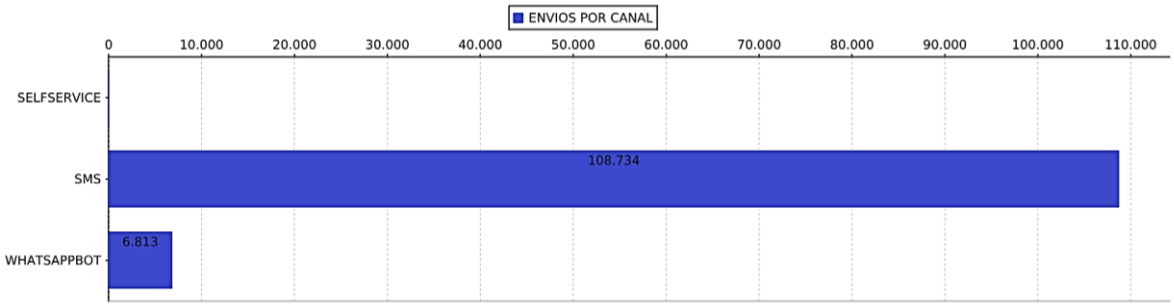
Con base a estos resultados se da por cumplido y aceptado el objetivo específico 4.

Figura 33 – Uso de canales digitales (Cantidad de mensajes por canal)



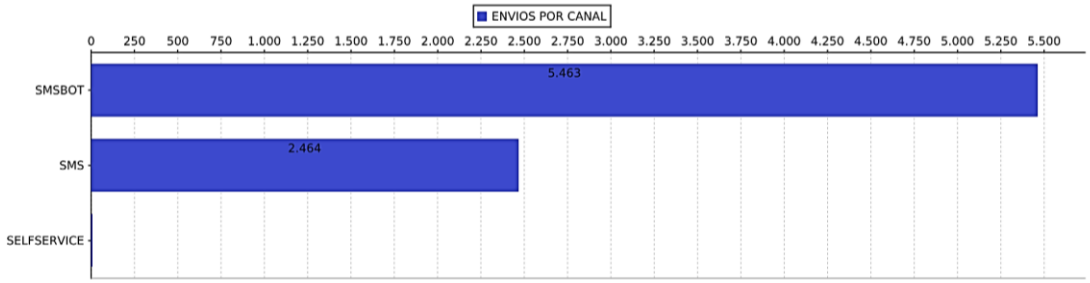
Envíos por canal – Pichincha

Figura 34 - Envíos por canal - Pichincha



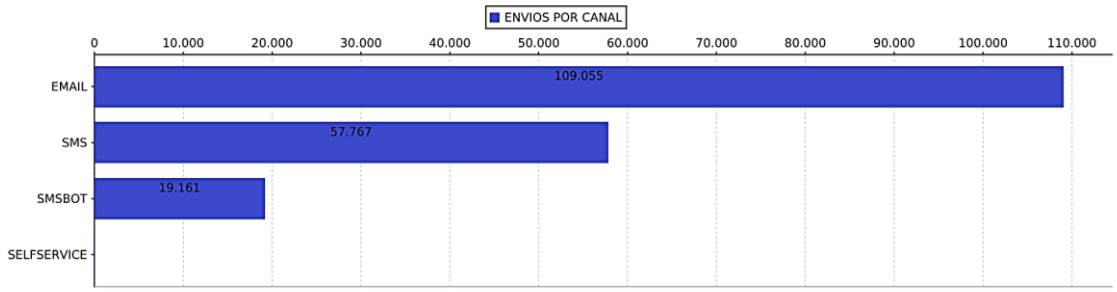
Envíos por canal – FOH (Financiera Oh)

Figura 35 - Envíos por canal - FOH



Envíos por canal – Servex

Figura 36 – Envíos por canal - Servex



5.6. APORTE DEL BACHILLER EN LA EMPRESA

Como profesional de la carrera de ingeniería de sistemas e informática, el aporte realizado a la empresa Financial Systems Company SAC, fue promover el uso de tecnologías emergentes y su adopción conforme a los resultados, a continuación, detallaré los aportes realizados:

- Realizar y refinar la arquitectura de microservicios en la implementación del módulo de cobranza digital.
- Desarrollo de scripts a nivel del módulo de cobranza digital para cumplir lo solicitado en las historias de usuario.
- Promover la utilización del estándar BPMN, actualmente se trabaja con diagramas de flujo de datos.
- Promover la utilización de herramientas que aporten durante la ejecución de proyectos agiles, herramientas como Trello que nos permite la colaboración basada en la metodología Scrum.
- Gerenciar los proyectos de implementación de forma exitosa, cumpliendo los criterios de aceptación y garantizando la satisfacción del usuario.
- Participar activamente en las reuniones de Scrum semanal y mensual.

CONCLUSIONES

1. Se implementó satisfactoriamente la arquitectura de software basada en microservicios con la aplicación web de cobranza digital en Financial Sytems Company SAC.
2. La identificación de requisitos a través de historias de usuario para implementar la arquitectura de software basada en microservicios, fue clara y permitió su adecuación en las fases de construcción y despliegue.
3. El diseño de la arquitectura de software basada en microservicios incorpora servicios de terceros como son Google, Amazon e Intico asegurando la alta disponibilidad y transmitiendo la responsabilidad a terceros, siendo de este modo independientes por cada canal digital.
4. La implementación de arquitectura de software basada en microservicios en la cobranza digital puede ser implementado en otras empresas añadiendo el factor diferenciador.
5. De acuerdo a la evaluación de uso de canales digitales, la arquitectura de microservicios puede soportar envíos masivos de mensajes sin impactar en el rendimiento y manteniendo la alta disponibilidad.

RECOMENDACIONES

1. Se recomienda el uso de la arquitectura de software basado en microservicios, porque permite desarrollar cada servicio de forma independiente y reduce los tiempos de desarrollo.
2. Se recomienda identificar y definir adecuadamente los requisitos de forma clara incluyendo los criterios de aceptación (utilizar enfoque ágil), esto brindará un panorama global que permitirá continuar con las actividades de desarrollo y adaptarse a cambios.
3. Se recomienda hacer una revisión arquitectónica de servicios brindados por terceros, integración y limitantes antes de ser incluidos en el diseño de la arquitectura basada en microservicios.
4. Se recomienda utilizar estándares de la industria durante la implementación, con soluciones probadas y que permitan escalabilidad, éstos conocen el modo de operación y tienen mejores prestaciones, tales como: Google y Amazon, que fueron los principales componentes integradores utilizados en la presente investigación.
5. Se recomienda realizar el monitoreo de los microservicios durante su uso, para proyectar la escalabilidad y brindar alta disponibilidad.

REFERENCIAS BIBLIOGRÁFICAS

1. **Bolivar, Mario Cesar y Calla, Carlos Alberto.** Implementación de microservicios para el intercambio de datos en el canal de plataforma digital del Banco de Crédito del Perú. Lima, Perú : Universidad San Martin de Porres, 2018. Tesis de grado.

2. **Ruedas, Donia Alizandra.** Modelo de Composición de microservicios para la implementación de una aplicación web de comercio electrónico utilizando Kubernetes. Puno, Perú : Universidad Nacional del Altiplano, 2017. Tesis doctoral.

3. **Lopez, Jose Daniel.** Arquitectura de Software basada en microservicios para desarrollo de aplicaciones web de la asamblea nacional. Ibarra, Ecuador : Universidad Técnica del Norte, 2017. Tesis master.

4. **Cols, Alberto y Salcedo, Mariana.** Arquitectura de Microservicios con RESTful. Madrid, España : Universidad Politécnica de Madrid, 2017. Tesis master.

5. **Salazar, William.** Implementacion de arquitectura de microservicios utilizando virtualizacion por sistema operativo. Guatemala : Universidad de San Carlos de Guatemala, 2017. Tesis de grado.

6. **Financial Systems Company.** Brochure iCS Enterprise. [En línea] [Citado el: 18 de Junio de 2018.] https://www.fsc-int.com/images/PDF/Brochure%iCS.pdf.

7. **ANIF.** Los Servicios Financieros Digitales en América Latina Investigación realizada para FELABAN - CAF. [En línea] [Citado el: 04 de Julio de 2018.] http://felaban.s3-website-us-west-2.amazonaws.com/documentos_interes/archivo20181204163600PM.pdf.

8. **Ajay, Joshua Gans.** Prediction Machines: The Simple Economics of Artificial Intelligence. Primera ed. Boston Massachusetts : Harvard Business Review Press, 2018. pág. 272. ISBN: 978-1-63369-567-2.

9. **Fugaro, Luigi y Vocale, Mauro.** Hands-On Cloud-Native Microservices with Jakarta EE. Primera ed. Birmingham : Packt Publishing, 2019. pág. 352. ISBN: 978-1-78883-786-6.

10. **Indrasiri, Kasun.** Microservices for the Enterprise. Primera ed. San Jose, USA : Apress, 2018. pág. 434. ISBN-13: 978-1-4842-3858-5.

11. **Richardson, Chris.** Microservices form Design to Development. Primera ed. USA : Nginx, 2016. pág. 80.

12. **Wolff, Eberhard.** Microservices: Flexible software Architecture. Primera ed. USA : Addison-Wesley, 2017. pág. 432. ISBN: 978-0-134-60241-7.

13. **Newman, Sam.** Building Microservices: Designing fine-grained Systems. Primera ed. USA : O'Reilly Media, 2015. pág. 280. ISBN: 978-1-491-95035-7.

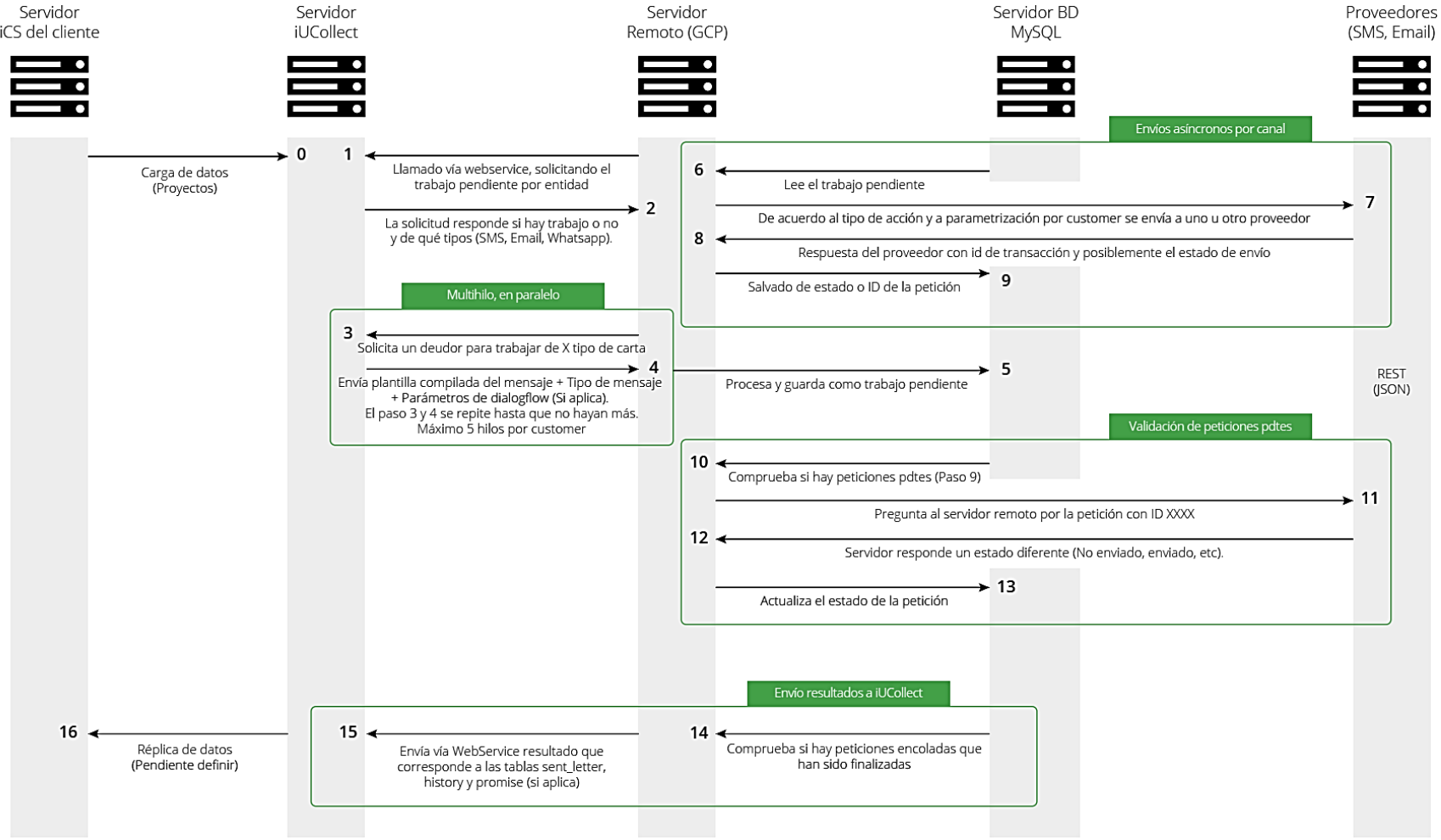
14. **Schwaber, Ken y Jeff, Sutherland.** La Guía de Scrum. [En línea] 2017. [Citado el: 9 de Julio de 2018.] https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Spanish-SouthAmerican.pdf.

15. **Hernandez, Roberto, Fernandez, Carlos y Baptista, Pilar.** Metodología de la Investigación. Sexta ed. México : Mc Graw Hill Education, 2014. pág. 634. ISBN: 978-1-4562-2396-0.

16. **Piscoya, L.** Investigación científica y educacional. Lima, Perú: Amaru Editores, 1987.

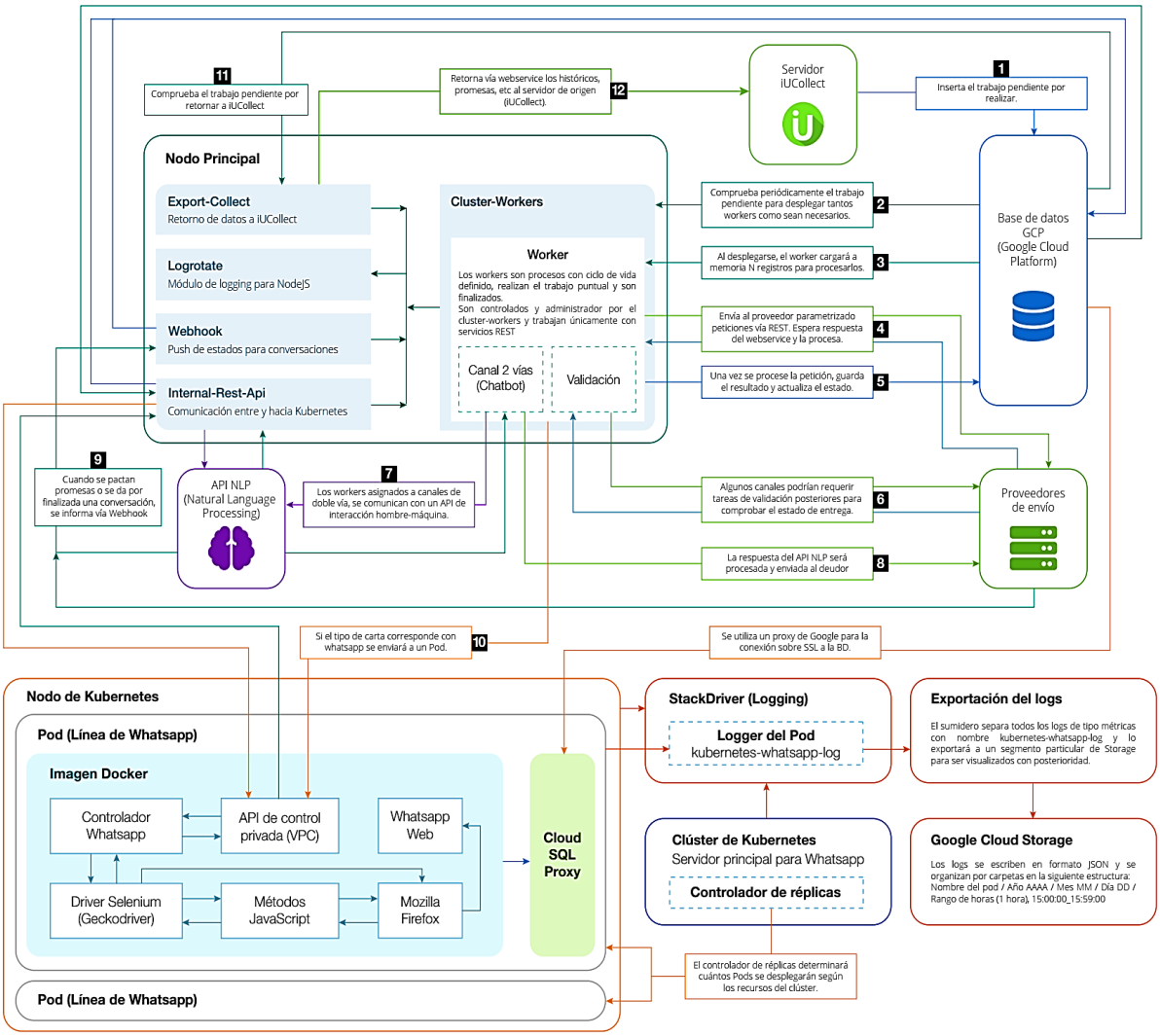
ANEXOS

ANEXO 1 - ARQUITECTURA DE MICROSERVICIOS INICIAL



ANEXO 2 - ARQUITECTURA DE MICROSERVICIOS PRODUCCIÓN

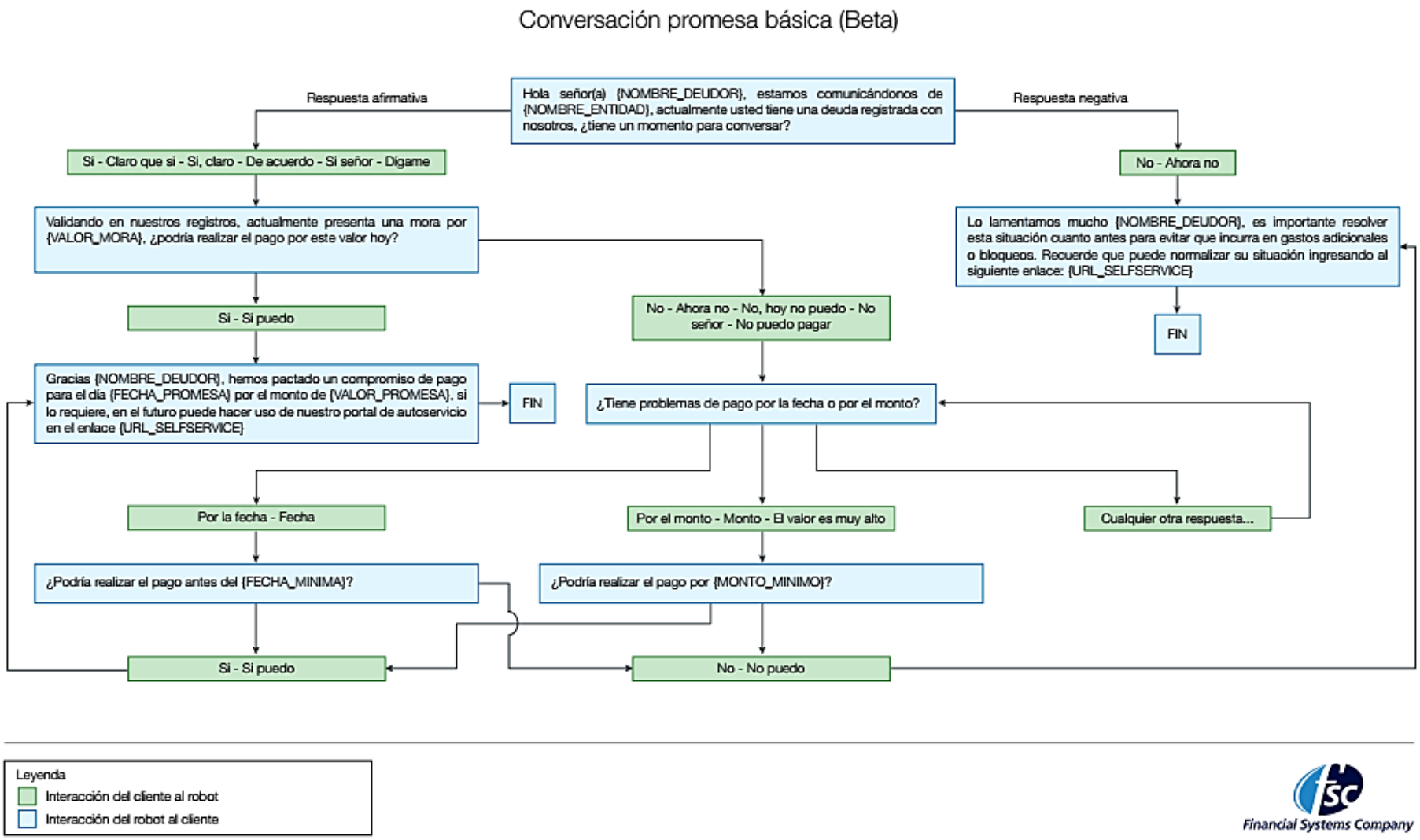
Arquitectura Cobranzas Digitales



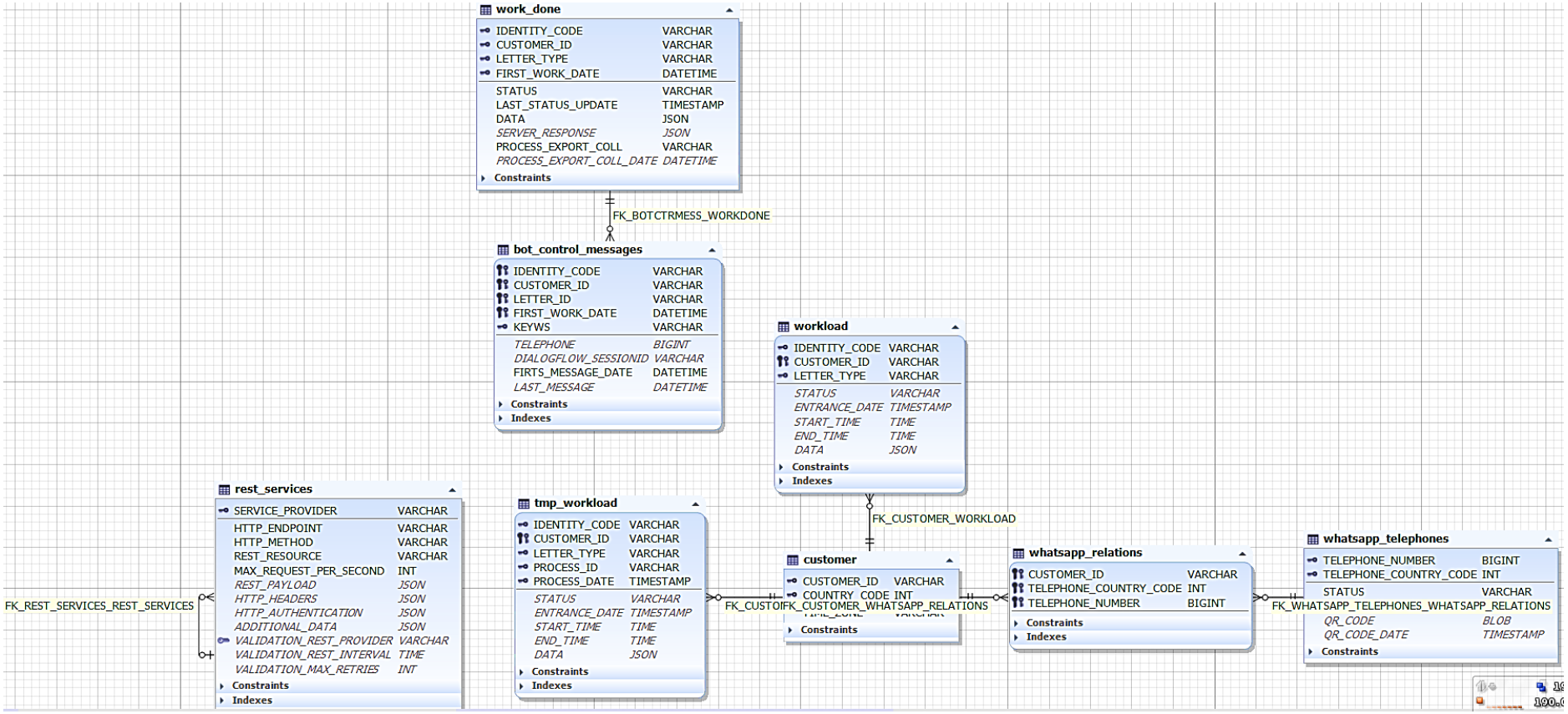
1. La autorización a todos los recursos dentro de Google Cloud Engine se realiza a través de IAM, mediante usuarios con permisos limitados y con llaves SSH (administradas por Google) para cada usuario.

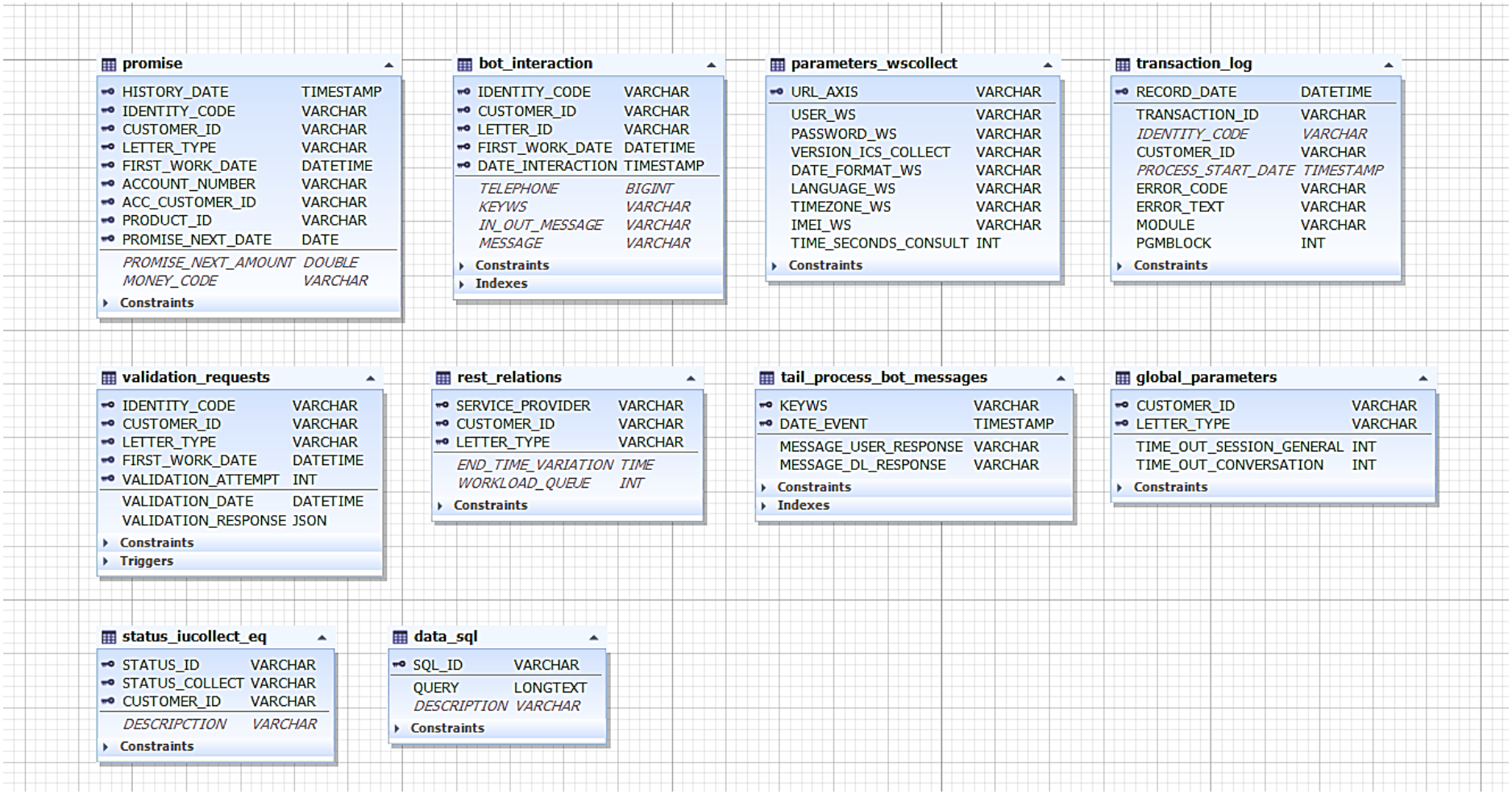
2. Los recursos y nodos están asignados a la misma red privada de Google, por lo cual tienen visibilidad de red en el mismo proyecto, sin embargo no son accesibles desde internet. El único nodo accesible es el nodo principal.

ANEXO 3 - DIALOG FLOW DEFINIDO PARA TENSORFLOW



ANEXO 4 - DIAGRAMA BD REMOTA




promise
HISTORY_DATE TIMESTAMP
IDENTITY_CODE VARCHAR
CUSTOMER_ID VARCHAR
LETTER_TYPE VARCHAR
FIRST_WORK_DATE DATETIME
ACCOUNT_NUMBER VARCHAR
ACC_CUSTOMER_ID VARCHAR
PRODUCT_ID VARCHAR
PROMISE_NEXT_DATE DATE
PROMISE_NEXT_AMOUNT DOUBLE
MONEY_CODE VARCHAR
Constraints
bot_interaction
IDENTITY_CODE VARCHAR
CUSTOMER_ID VARCHAR
LETTER_ID VARCHAR
FIRST_WORK_DATE DATETIME
DATE_INTERACTION TIMESTAMP
TELEPHONE BIGINT
KEYWS VARCHAR
IN_OUT_MESSAGE VARCHAR
MESSAGE VARCHAR
Constraints
Indexes
parameters_wscollect
URL_AXIS VARCHAR
USER_WS VARCHAR
PASSWORD_WS VARCHAR
VERSION_ICS_COLLECT VARCHAR
DATE_FORMAT_WS VARCHAR
LANGUAGE_WS VARCHAR
TIMEZONE_WS VARCHAR
IMEI_WS VARCHAR
TIME_SECONDS_CONSULT INT
Constraints
transaction_log
RECORD_DATE DATETIME
TRANSACTION_ID VARCHAR
IDENTITY_CODE VARCHAR
CUSTOMER_ID VARCHAR
PROCESS_START_DATE TIMESTAMP
ERROR_CODE VARCHAR
ERROR_TEXT VARCHAR
MODULE VARCHAR
PGMBLOCK INT
Constraints
validation_requests
IDENTITY_CODE VARCHAR
CUSTOMER_ID VARCHAR
LETTER_TYPE VARCHAR
FIRST_WORK_DATE DATETIME
VALIDATION_ATTEMPT INT
VALIDATION_DATE DATETIME
VALIDATION_RESPONSE JSON
Constraints
Triggers
rest_relations
SERVICE_PROVIDER VARCHAR
CUSTOMER_ID VARCHAR
LETTER_TYPE VARCHAR
END_TIME_VARIATION TIME
WORKLOAD_QUEUE INT
Constraints
tail_process_bot_messages
KEYWS VARCHAR
DATE_EVENT TIMESTAMP
MESSAGE_USER_RESPONSE VARCHAR
MESSAGE_DL_RESPONSE VARCHAR
Constraints
Indexes
global_parameters
CUSTOMER_ID VARCHAR
LETTER_TYPE VARCHAR
TIME_OUT_SESSION_GENERAL INT
TIME_OUT_CONVERSATION INT
Constraints
status_iucollect_eq
STATUS_ID VARCHAR
STATUS_COLLECT VARCHAR
CUSTOMER_ID VARCHAR
DESCRIPCTION VARCHAR
Constraints
data_sql
SQL_ID VARCHAR
QUERY LONGTEXT
DESCRIPTION VARCHAR
Constraints

ANEXO 5 - EJECUCIÓN DEL PROCESO DE ESTRATEGIAS QUE ALMACENA LO DEFINIDO EN LA LÍNEA DE TIEMPO

Source:

```
-- Case 56453 - dgaleano - 08/09/2017 - paquete para el envío de cartas estrategias con actividades
CREATE OR REPLACE PACKAGE BODY pkg_ics_letter_activities IS
 ----------------------------------------------------------------------------------------------------
 --   PROCEDURE p_ics_main  - procedimiento principal que se invoca
 ----------------------------------------------------------------------------------------------------
 PROCEDURE p_ics_main(avCustomerId IN cobra.strategy.customer_id%TYPE,
                      ad_date      IN DATE) IS
  lnPgmBlock cobra.transaction_log.pgmblock%TYPE;
  ldBegin    DATE := TRUNC(ad_date);
  ldEnd      DATE := TRUNC(ad_date) + (24 * 60 * 60 - 1) / (24 * 60 * 60);
  ldSysdate  DATE := ad_date;

  lobInsert  cobra.tp_ics_people_activity_table := cobra.tp_ics_people_activity_table();
  gvLanguage cobra.system_maintenance.official_language%TYPE := NULL;
  lvMessage  cobra.messages.msgtext%TYPE;

  CURSOR cursorGetLanguage IS
   SELECT sm.official_language
    FROM system_maintenance sm
    WHERE sm.customer_id = 'DEFAULT';

  CURSOR cursorGetPeople(acvCustomerId IN cobra.customer.customer_id%TYPE) IS
    SELECT CAST(MULTISET (SELECT identity_code, customer_id, strategy_id, script_letter, sequence, telephone_number, telephone_type, letter_type, common_id
      FROM (WITH p AS (SELECT p.identity_code,
                      p.customer_id,
                      sm.prefix_length || p.identity_code common_id,
                      pa.contact_method,
                      pa.strategy_id,
                      pa.script_letter,
                      NVL(s.dir_visit_letter, 'ANY') AS dir_visit_letter,
                      s.letter,
                      (CASE pa.contact_method
                        WHEN 'LETTER' THEN 'L'
                        WHEN 'SMS' THEN 'S'
                        WHEN 'IVR' THEN 'I'
                        WHEN 'EMAIL' THEN 'E'
                        WHEN 'SMSBOT' THEN 'M'
                        WHEN 'WHATSAPP' THEN 'W'
                        WHEN 'WHATSAPPBOT' THEN 'B'
                       END) AS letter_type
                  FROM people p
                  INNER JOIN strategy s
                    ON s.strategy_id = p.strategy
                    AND s.customer_id = p.customer_id
                    AND s.strategy_status = 'ACTIVE'
                  INNER JOIN people_activities pa
                    ON p.strategy = pa.strategy_id
                    AND p.identity_code = pa.identity_code
                    AND p.customer_id = pa.customer_id
                    AND p.state_id != 'INACTIVE'
                  INNER JOIN system_maintenance sm
                    ON sm.customer_id = pa.customer_id
                    AND p.customer_id = sm.customer_id
                  INNER JOIN strategy_letter sl
                    ON sl.letter_id = pa.script_letter
                    AND sl.customer_id = pa.customer_id
                    AND (sl.strategy_id = pa.strategy_id OR sl.strategy_id = '*****')
                  INNER JOIN letter l
                    ON l.letter_id = sl.letter_id
                    AND l.customer_id = sl.customer_id
                  INNER JOIN letter_resources lr
                    ON lr.letter_id = sl.letter_id
                    AND lr.customer_id = pa.customer_id
                    AND lr.main = 'Y'
                  WHERE pa.activity_expire_date BETWEEN ldBegin AND ldEnd
                    AND pa.contact_method IN ('LETTER', 'SMS', 'IVR', 'EMAIL', 'SMSBOT', 'WHATSAPP', 'WHATSAPPBOT')
                    AND p.customer_id = 'BANCOFSC'
                    AND f_ics_count_debtor_commitments(p.identity_code, p.customer_id) <= 0),
           addr AS ( SELECT a.sequence, a.identity_code, a.common_id, a.address_type, a.address_number, p.contact_method,
                       ROW_NUMBER() OVER(PARTITION BY a.common_id, p.contact_method ORDER BY a.contact_success DESC, a.update_date DESC) dorder
                    FROM address a, p
```

```
WHERE p.identity_code = a.identity_code
                    AND p.common_id = a.common_id
                    AND p.contact_method IN ('LETTER','EMAIL')
                    AND a.address_status = 'ACTIVE'
                    AND ((p.contact_method = 'LETTER' AND p.letter = '1' AND a.use_this = 'Y' AND p.dir_visit_letter <> 'EMAIL' AND
                      a.address_type = CASE
                           WHEN (p.dir_visit_letter = 'ANY' AND a.address_type <> 'EMAIL') THEN
                              a.address_type
                           WHEN p.dir_visit_letter = a.address_type THEN a.address_type END) OR
                             (p.contact_method = 'EMAIL' AND p.contact_method = a.address_type))),
          tel AS ( SELECT sequence, t.identity_code, t.common_id, t.telephone_type, TO_CHAR(t.telephone_number) AS telephone_number, p.contact_method,
                     ROW_NUMBER() OVER(PARTITION BY t.common_id, p.contact_method ORDER BY t.contact_success DESC, t.update_date DESC) dorder
                   FROM telephone t, p
                   WHERE p.identity_code = t.identity_code
                    AND p.common_id = t.common_id
                    AND t.line_state = 'OK'
                    AND (((p.contact_method = 'SMS' OR p.contact_method = 'SMSBOT' OR p.contact_method = 'WHATSAPP' OR p.contact_method = 'WHATSAPPBOT') AND
t.telephone_type = 'CEL') OR
                     (p.contact_method = 'IVR' AND t.telephone_type IN ('CEL', 'TEL'))))

          SELECT *
           FROM (SELECT p.identity_code,
                    p.customer_id,
                    p.strategy_id,
                    p.script_letter,
                    (CASE p.contact_method
                      WHEN 'LETTER' THEN
                       NVL(addr.sequence, 0)
                      WHEN 'EMAIL' THEN
                       NVL(addr.sequence, 0)
                      WHEN 'IVR' THEN
                       NVL(tel.sequence, 0)
                      WHEN 'SMS' THEN
                       NVL(tel.sequence, 0)
                      WHEN 'SMSBOT' THEN
                       NVL(tel.sequence, 0)
                      WHEN 'WHATSAPP' THEN
                       NVL(tel.sequence, 0)
                      WHEN 'WHATSAPPBOT' THEN
                       NVL(tel.sequence, 0)
                    END) AS sequence,
                    tel.telephone_number,
                    tel.telephone_type,
                    p.letter_type,
                    p.common_id
                  FROM p
                  LEFT JOIN addr
                    ON p.identity_code = addr.identity_code
                   AND p.common_id = addr.common_id
                   AND p.contact_method = NVL(addr.contact_method,p.contact_method)
                   AND addr.dorder = 1
                  LEFT JOIN tel
                    ON p.identity_code = tel.identity_code
                   AND p.common_id = tel.common_id
                   AND tel.dorder = 1
                   AND p.contact_method = NVL(tel.contact_method, p.contact_method)) i
          WHERE NOT EXISTS
          (SELECT 1
                FROM letter_transaction l
                WHERE l.identity_code = i.identity_code
                 AND l.customer_id = i.customer_id
                 AND l.letter_id = script_letter
                 AND l.history_date BETWEEN ldBegin AND ldEnd
                 AND l.letter_type = i.letter_type
                 AND l.destination_id = i.identity_code
                 AND l.sequence = i.sequence))
     ) AS cobra.tp_ics_people_activity_table)FROM dual;

 BEGIN

  OPEN cursorGetPeople(avCustomerId);
  FETCH cursorGetPeople
    INTO lobInsert;
  CLOSE cursorGetPeople;

  lnPgmBlock := 1;
```

```
  INSERT INTO letter_transaction(identity_code, customer_id, letter_id, history_date, group_id, user_id, destination_id, sequence, canceled,
                        letter_type, urgency, printing_status, telephone_number, querie_id, common_id)
  SELECT i.identity_code,
       i.customer_id,
       i.script_letter AS letter_id,
       ldSysdate AS history_date,
       'DEFAULT' AS group_id,
       'SYSTEM' AS user_id,
       i.identity_code,
       i.sequence,
       'N' AS canceled,
       i.letter_type,
       'URG' AS urgency,
       'PENDING' AS printing_status,
       i.telephone_number,
       i.telephone_type AS querie_id,
       i.common_id
  FROM TABLE(lobInsert) i
  WHERE NVL (i.sequence,0) <> 0;

  INSERT INTO history(identity_code, history_date, user_id, customer_id, action_id, group_id, sequence, response_id, strategy,
                process_type, response_weight, record_date)
  SELECT i.identity_code,ldSysdate AS history_date, 'SYSTEM' AS user_id, i.customer_id,
       (CASE
        WHEN i.letter_type = 'L' THEN  'SEND LETTER'
        WHEN ((i.letter_type = 'S') OR (i.letter_type = 'B') OR (i.letter_type = 'M') OR (i.letter_type = 'W') OR (i.letter_type = 'I') OR (i.letter_type = 'E')) THEN
         'SEND MESSAGE'
       END) AS action_id,
       'DEFAULT' AS group_id,
       i.sequence,
       (CASE i.letter_type
        WHEN 'S' THEN 'SMS'
        WHEN 'B' THEN 'WHATSAPPBOT'
        WHEN 'M' THEN 'SMSBOT'
        WHEN 'W' THEN 'WHATSAPP'
        WHEN 'E' THEN 'EMAIL'
        WHEN 'I' THEN 'VOICE'
        WHEN 'L' THEN 'DEBTOR'
       END) AS response_id,
       i.strategy_id AS strategy,
       'B' AS process_type,
       '0' AS response_weight,
       SYSDATE AS record_date
  FROM TABLE(lobInsert) i
  WHERE NVL (i.sequence,0) <> 0;
  lnPgmBlock := 2;
  COMMIT;
  OPEN cursorGetLanguage;
  FETCH cursorGetLanguage
   INTO gvLanguage;
  CLOSE cursorGetLanguage;

  lvMessage := ics_commoncode.errmsg('-20619', 'DEFAULT', gvLanguage);

  FOR c IN ((SELECT i.identity_code,
              i.customer_id,
              SYSDATE AS record_date,
              i.sequence
            FROM TABLE(lobInsert) i
           WHERE NVL(i.sequence,0) = '0')) LOOP
    END LOOP;
 EXCEPTION
  WHEN OTHERS THEN
    pkg_ics_exceptions.p_ics_log_error('pkg_ics_letter_activities',
                          'p_ics_main',
                          'pkg_ics_letter_activities',
                          SYSDATE,
                          USER,
                          'N',
                          SQLCODE,
                          SQLERRM(SQLCODE),
                          NULL,
                          NULL,
                          NULL,
                          NULL,
                          NULL,
                          NULL,
                          NULL,
                          NULL,
                          'p_ics_main',
                          'pkg_ics_letter_activities',
                          lnPgmblock,
                          0);
 END p_ics_main;
END pkg_ics_letter_activities;
```

Validamos:

```
select strategy, stra_entrance_date, stra_expire_date, next_activity_date
  from v_ics_people where customer_id = 'BANCOFSC'
 group by strategy, stra_expire_date, next_activity_date, stra_entrance_date;
```

Select people_activities | Select v_ics_people

	STRATEGY	STRA_ENTRANCE_DATE	STRA_EXPIRE_DATE	NEXT_ACTIVITY_DATE
1	DEFAULT	20/06/2018 5:30:11 PM	20/06/2018 5:30:11 PM	20/06/2018 5:30:11 PM

Se ejecuta el proceso 280, con fecha de hoy y se hará revisión de las tablas relacionadas:

```
--Estrategias
exec pkg_process.main('BANCOFSC',280,280,sysdate,'','',sysdate,false,false,false,false,1,0);
/
```

```
SQL>

PL/SQL procedure successfully completed


PL/SQL procedure successfully completed


SQL>
```

PL/SQL procedure successfully completed in 25.911 seconds

```
select action_id, history_date
  from history
 group by action_id, history_date
 order by history_date desc;

select * from history order by history_Date desc;
```

Select people_activities | Select v_ics_people | Select letter_transaction | Select history

	ACTION_ID	HISTORY_DATE
1	CHANGE STRATEGY	05/07/2018 6:19:31 PM
2	CHANGE STRATEGY	05/07/2018 6:19:21 PM
3	CHANGE STRATEGY	05/07/2018 6:19:10 PM
4	CHANGE STRATEGY	05/07/2018 6:18:59 PM
5	SEND MESSAGE	05/07/2018 12:00:03 AM
6	SEND MESSAGE	05/07/2018 12:00:02 AM
7	SEND MESSAGE	05/07/2018 12:00:01 AM

```
select * from letter_transaction order by identity_code ;

select action_id, history_date
  from history
 group by action_id, history_date
 order by history_date desc;

select * from history order by history_Date desc;
```

Select people_activities | Select v_ics_people | Select letter_transaction | Select history | Select history

	IDENTITY_CODE	CUSTOMER_ID	LETTER_ID	HISTORY_DATE	GROUP_ID	USER_ID	DESTINATION_ID	SEQUENCE	CA
1	100011705681	BANCOFSC	em	05/07/2018 12:00:02 AM	DEFAULT	SYSTEM	100011705681	2	N
2	100011705681	BANCOFSC	sms	05/07/2018 12:00:01 AM	DEFAULT	SYSTEM	100011705681	1	N
3	100011705681	BANCOFSC	ws	05/07/2018 12:00:03 AM	DEFAULT	SYSTEM	100011705681	1	N
4	100024051281	BANCOFSC	em	05/07/2018 12:00:01 AM	DEFAULT	SYSTEM	100024051281	2	N
5	100024051281	BANCOFSC	sms	05/07/2018 12:00:02 AM	DEFAULT	SYSTEM	100024051281	1	N
6	100024051281	BANCOFSC	ws	05/07/2018 12:00:03 AM	DEFAULT	SYSTEM	100024051281	1	N
7	100036396881	BANCOFSC	em	05/07/2018 12:00:01 AM	DEFAULT	SYSTEM	100036396881	2	N
8	100036396881	BANCOFSC	sms	05/07/2018 12:00:03 AM	DEFAULT	SYSTEM	100036396881	1	N
9	100036396881	BANCOFSC	ws	05/07/2018 12:00:02 AM	DEFAULT	SYSTEM	100036396881	1	N
10	100048742481	BANCOFSC	em	05/07/2018 12:00:02 AM	DEFAULT	SYSTEM	100048742481	2	N
11	100048742481	BANCOFSC	sms	05/07/2018 12:00:03 AM	DEFAULT	SYSTEM	100048742481	1	N
12	100048742481	BANCOFSC	ws	05/07/2018 12:00:01 AM	DEFAULT	SYSTEM	100048742481	1	N
13	100061088081	BANCOFSC	em	05/07/2018 12:00:02 AM	DEFAULT	SYSTEM	100061088081	2	N

```
select *
  from people_activities
 order by identity_code, stra_entrance_date, activity_entrance_date;
```

Select people_activities | Select v_ics_people | Select letter_transaction | Select history | Select history

	IDENTITY_CODE	STRATEGY_ID	CONTACT_METHOD	CUSTOMER_ID	STRA_ENTRANCE_DATE	STRA_EXPIRE_DATE	ACTIVITY_
1	100011705681	MEDIA	CALL	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	05/07/201
2	100011705681	MEDIA	EMAIL	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	05/07/201
3	100011705681	MEDIA	SMS	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	05/07/201
4	100011705681	MEDIA	WHATSAPP	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	05/07/201
5	100011705681	MEDIA	EMAIL	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	06/07/201
6	100011705681	MEDIA	WHATSAPP	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	06/07/201
7	100011705681	MEDIA	CALL	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	07/07/201
8	100011705681	MEDIA	EMAIL	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	07/07/201
9	100011705681	MEDIA	SMS	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	07/07/201
10	100024051281	AVANZADA	CALL	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	05/07/201
11	100024051281	AVANZADA	EMAIL	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	05/07/201
12	100024051281	AVANZADA	WHATSAPP	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	05/07/201
13	100024051281	AVANZADA	SMS	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	05/07/201
14	100024051281	AVANZADA	EMAIL	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	06/07/201
15	100024051281	AVANZADA	WHATSAPP	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	06/07/201
16	100024051281	AVANZADA	SMS	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	07/07/201
17	100024051281	AVANZADA	EMAIL	BANCOFSC	05/07/2018	08/07/2018 11:59:59 PM	07/07/201

ANEXO 6 - ENVIAR MENSAJES PROGRAMADOS POR CANALES DIGITALES

1. El proceso para construir los mensajes en la tabla sent_letter y cargar los registros en la base de datos remota se realizó mediante un jar que se deberá correr en línea de comandos java -jar y con dos argumentos, el primero la entidad a realizar el proceso y el segundo la opción a realizar bien sea 1 la construcción de mensajes o 2 la carga a la base de datos remota.

2. Para la ejecución del jar es necesario que exista un archivo de configuración donde se deben ingresar los parámetros necesarios para la conexión mysql y Oracle asi como las rutas de keystore y cacerts necesarios para la conexión ssl mysql. Estos son los parámetros

   urlMysql = 999.999.999.999
   portMysql = 9999
   dbnameMysql = nameDb
   usernameMysql = usernameDb
   passwordMysql = passwordDb
   keyStorePath = C:/Program Files/Apache Tomcat/.keystore
   keyStorePass = passKeystore
   trustStorePath = C:/Program Files/Java/jdk1.8.0_152/jre/lib/security/cacerts
   trustStorePass = passTrustStore
   urlOracle = 999.999.999.999
   portOracle = 9999
   dbnameOracle = nameDb
   usernameOracle = usernameDb
   passwordOracle = passwordDb

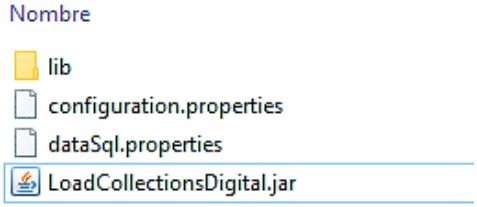
3. La conexión mysql se realiza por ssl por tanto es necesario realiza la configuración de seguridad con llave publica y llave privada para su utilización en java con un .keystore y un cacerts correspondiente, además en el archivo de configuración mencionado en el anterior punto, especificar la ruta de los archivos y su clave de acceso.

4. El módulo realizado en nodeJS para la comunicación expone un servicio REST el cual para su correcto funcionamiento deberá enviarse un texto de tipo json como primer mensaje similar al siguiente ejemplo:

   ```
   { "customerId": "CUSTOMER",
   "identityCode": "9999999" ,
   "firtsMessage" : {
       "nombreDeudor": "Nombre Deudor",
       "nombreEntidad": "Entidad",
       "fechaMinima": "01/01/2018",
       "fechaMaxima": "31/01/2018",
       "valorMinimo": "500000",
       "valorMora": "2000000",
       "urlSelfService": "https://demo.fsc-int.com:8443/iCSSelfService/"
   },
   "letterType":"B"
   }
   ```

   El servicio REST responde si es correcto enviando una cadena tipo JSON con el mensaje de respuesta por parte del agente DIALOGFLOW y un id de sesión el cual deberá encadenarse en las subsecuentes respuestas por parte del cliente que interactua con el agente DIALIGFLOW consumiendo nuevamente el servicio REST con un mensaje similar al siguiente ejemplo:

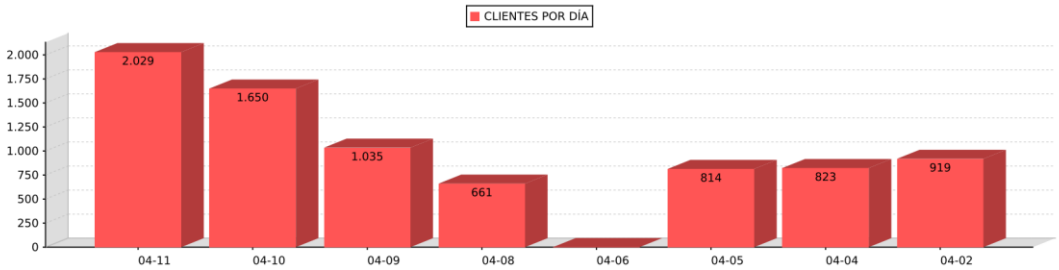
   ```
   { "customerId": "BANCOFSC",
   "identityCode": "80075012" ,
   "message" : "si",
   "uuid" : "94fb2090-7a23-11e8-bb35-4100a4bb2c15"
   }
   ```

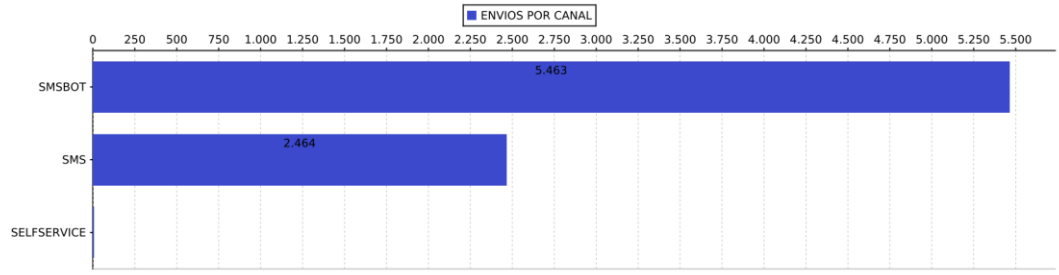


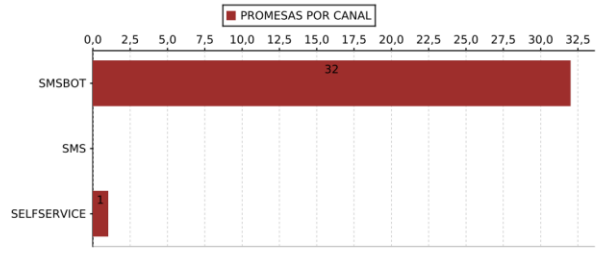
ANEXO 7 - OBTENER EL RESULTADO DEL ENVÍO DE MENSAJES POR CANALES DIGITALES

iUCollect

DASHBOARD







	NÚMERO	VALOR
TOTAL CLIENTES	7.932	$10.841.282,78
TOTAL PROMESAS	33	$9.513,57
TOTAL PAGADO	0	$13,30
INDICE PROMESAS	0,42%	0,09%
INDICE DE PAGO	0%	0%

Page 1 of 1

ANEXO 8 - PLANTILLA DE HISTORIAS DE USUARIO

	Enunciado de la historia			Criterio de aceptación	
Identificador (ID) de la historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario	Resultado / Comportamiento esperado

ANEXO 9 – PLANTILLA DE VALIDACIÓN DE HISTORIAS DE USUARIO

	Enunciado de la historia			Criterio de aceptación		
Identificador (ID) de la historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de escenario	Resultado / Comportamiento esperado	Aceptado

ANEXO 10 - MATRIZ DE UTILIDAD ATAM

Atributo	Medio de cumplimiento (componente arquitectural)	Tecnología usada	Cumple
Interoperabilidad			
Mantenibilidad			
Modificabilidad			
Instalación			
Reemplazo			
Tolerancia a fallos			
Recuperabilidad			
Reusabilidad			
Disponibilidad			
Seguridad			

ANEXO 11 - MANUAL DE PASO DE INFORMACIÓN DE IUCOLLECT A REMOTO

Proceso de Compilación de Reportes para Cobranzas Digitales.

Para que el proceso de generación de reportes para cobranzas digitales funcione correctamente es necesario que esté correctamente configurada la parametrización de las cartas asociadas a las actividades parametrizadas. También la parametrización de la estrategia asociada a las actividades de la línea de tiempo y se deberá ejecutar el proceso de Asignación de Estrategia a deudores (proceso Batch con identificación 280 por medio del pkg_process). Una vez hecho esto, según las actividades parametrizadas, se podrá verificar los registros correspondientes en la tabla PEOPLE_ACTIVITIES. El proceso de Asignación de Estrategias a deudores también verifica y crea los registros correspondientes registros en la tabla LETTER_TRANSACTION para el día que se ejecuta el proceso. Esta generación es diaria, por lo que no se puede tratar de generar los registros correspondientes para varios días a la vez.

Para la generación de los reportes se deberá ejecutar o programar una ejecución del aplicativo .jar `LoadCollectionsDigital.jar` y agregar dos argumentos a la ejecución. El primero corresponde a la entidad sobre la cual se generarán los reportes, y el segundo, el número "1", para indicar que se ejecutará el proceso de generación de reportes. Ejemplo:

```
java -jar LoadCollectionsDigital BANFOFSC 1
```

Para la ejecución de este proceso el módulo cuenta con dos archivos de propiedades, los cuales deben configurarse correctamente. A continuación, se describe el uso de estos dos archivos.

Archivo de propiedades de conexión `configuration.properties`

Este archivo tiene los parámetros necesarios para la conexión a la base de datos de iCS en Oracle y de la base de datos *remota* en MySQL estos son los valores:

urlMysql: url del servidor MySQL de la base de datos remota.
portMysql: puerto del servidor MySQL de la base de datos remota.
dbnameMysql: nombre de la base de datos remota MySQL.
usernameMysql: usuario de conexión a la base de datos remota MySQL.
passwordMysql: contraseña del usuario de conexión a la base de datos remota MySQL.
mysqlEnableSSL: indica si la conexión utilizará o no seguridad SSL para la conexión (Y/N)
keyStorePath: en caso de que se utilice seguridad SSL para la conexión, se deberá indicar el path del keystore correspondiente.
keyStorePass: en caso de que se utilice seguridad SSL para la conexión se deberá indicar la contraseña del keystore correspondiente.
trustStorePath: en caso de que se utilice seguridad SSL para la conexión se deberá indicar el path del cacerts correspondiente.
trustStorePass: en caso de que se utilice seguridad SSL para la conexión se deberá indicar la contraseña del cacerts correspondiente.
urlOracle: url del servidor Oracle de la base de datos de iCS.
portOracle: puerto del servidor Oracle de la base de datos de iCS.
dbnameOracle: nombre de la instancia de datos Oracle de iCS

usernameOracle: usuario de conexión a la base de datos iCS
passwordOracle: contraseña del usuario de conexión a la base de datos de iCS.
timeNotification: indica el número de registros a procesar para mostrar el tiempo transcurrido.

Archivo de propiedades de consultas `dataSql.properties`

Este archivo tiene las dos consultas utilizadas por el aplicativo .jar las cuales son:

QueryGetLetTranCollDig: Esta consulta selecciona los registros de la tabla LETTER_TRANSACTION cruzados contra la tabla LETTER_RESOURCES, de cierto tipo de carta, y que se hayan pre-compilado. Es decir que estén guardados como recursos en la base de datos iCS, para que el módulo los incluya en la tabla SENT_LETTER.

QueryLoadSentLetterReg: Esta consulta selecciona los registros de la tabla SENT_LETTER, cruzando además con tablas como PEOPLE_ACTIVITIES, SYSTEM_MAINTENANCE, LETTER, TELEPHONE, ADDRESS, PEOPLE, ACCOUNT_SUMMARY, RELATIONSHIP, PRODUCT, STRATEGY, RELATIONSHIP, y ACCOUNT, para que el módulo los inserte en la tabla WORKLOAD de la base de datos remota.

Proceso de carga de actividades a servidor remoto.

Este proceso toma los registros existentes en la tabla SENT_LETTER de la Base de datos iCS, para cargarlos en la base de datos remota MysSQL en la tabla WORKLOAD. El proceso se encarga por medio de una consulta de traer toda la información necesaria de las tablas PEOPLE_ACTIVITIES, SYSTEM_MAINTENANCE, LETTER, TELEPHONE, ADDRESS, PEOPLE, ACCOUNT_SUMMARY, RELATIONSHIP, PRODUCT, STRATEGY, RELATIONSHIP, ACCOUNT y según el tipo de mensaje, construye un objeto *json,* el cual se carga en la tabla remota.

Para la carga de actividades al servidor remoto se deberá ejecutar o programar una ejecución del aplicativo .jar `LoadCollectionsDigital.jar` y agregar dos argumentos a la ejecución. El primero corresponde a la entidad sobre la cual se cargarán las actividades, y el segundo, el número “2”, para indicar que se ejecutará el proceso de carga de actividades. Ejemplo:

```
java -jar LoadCollectionsDigital BANFOFSC 2
```

Este proceso también utiliza los dos archivos de propiedades de Configuración (.properties) que se describieron anteriormente.

ADVERTENCIAS
Las dos consultas del archivo `dataSql.properties` tienen ciertos campos con los cuales se procesa la información. No se debe realizar cambios sobre dichos campos o si se realiza un cambio esencial, el tipo de dato deberá ser igual y el alias de la columna

deberá mantenerse. De lo contrario, el módulo presentará errores. Lo recomendable es solo realizar cambios en las condiciones de las consultas, mas no, en los datos consultados.

Puesto que el texto de las consultas del archivo `dataSql.properties` es muy extenso se ha separado con saltos de línea para facilitar su lectura y manipulación, cuando hay un salto de línea se agrega una cadena para representarla (`\n\`). Si esta cadena no se respeta entre cada salto de línea, el módulo presentará errores.

Cada propiedad de los archivos de propiedades está denominada de cierta manera (ejemplo: urlMysql, QueryGetLetTranCollDig), que no podrán ser modificadas. De lo contrario, el módulo presentará fallos.

En caso de que la conexión a la base de datos MySQL sea por puerto seguro utilizando SSL, se deberá configurar un .keystore y un cacerts donde se importen los certificados suministrados por el servidor MySQL. De lo contrario el módulo presentará errores.

Proceso de Envió de mensajes SMS doble vía.

El proceso de envío de mensajes SMS doble vía está basado en un guión previamente establecido, donde se estructura un diálogo entre un robot y el cliente. Al robot se le denomina bot, y éste para el aplicativo realizado, se encuentra parametrizado en un proyecto de tipo Google Cloud *DialogFlow*. Este diálogo es único para cada cliente, por lo que su parametrización necesita la creación de un proyecto tipo *DialogFlow* por cada cliente en Google Cloud.

Para la conexión entre el aplicativo y el proyecto de DialogFlow se requiere parametrizar un json de conexión, el cual se encuentra en la base de datos MySQL en la tabla REST_SERVICES en el campo ADDITIONAL_DATA. La propiedad del json dialogFlowProperties es la que tiene las propiedades de conexión. Dentro de esta propiedad se encuentra la propiedad
credentials, que corresponde al objeto json suministrado por Google Cloud.
La propiedad projectProperties contiene dos propiedades: El languageCode: es el código de lenguaje con el cual trabaja el proyecto DialogFlow y el projectId del proyecto. En este json del campo ADDITIONAL_DATA también está la propiedad dialFlowParamConversation, la cual contiene los parámetros que se envían al inicio de la conversación, así como de que tabla y campo de la base de datos se puede obtener el valor de acuerdo a cada deudor y entidad. También tiene la propiedad serviceProviderHelper el cual es un json que indica el módulo y método que se ejecuta como ayudante para el consumo del servicio. Por último, el json también contiene la propiedad changeStatusHelper el cual es un json con el cual se indica que método del módulo se ejecuta como ayudante para determinar el cambio de estado para el registro de cada deudor y entidad en la tabla WORK_DONE.

Primero el proceso toma los registros de la tabla WORKLOAD de tipo sms_bot y envía el primer mensaje de la conversación, el cual está en el json del campo DATA, utilizando el

servicio web parametrizado para el tipo de actividad. El servicio web responde un mensaje indicando que el mensaje se ha enviado correctamente y envía un código único por cada mensaje enviado, que se almacena en la tabla BOT_CONTROL_MESSAGES. También se elimina el registro de la tabla WORKLOAD y lo adiciona a la tabla WORK_DONE con STATUS W, que indica que se está trabajado el registro. Si ocurre algún error en el envío del primer mensaje, se almacena en la tabla WORK_DONE con STATUS E. En la tabla BOT_ITERACTIONS se almacena las iteraciones bot – cliente cliente – bot que hubiese por cada registro, por tanto, en este primer mensaje se ingresa un registro en la tabla de tipo out.

Puesto que una vez iniciado una conversación en DialogFlow se sigue un flujo de conversación y se mantienen unos datos en memoria, cada sesión o conversación iniciada en DialogFlow tiene un tiempo de vida que a la fecha es de 10 minutos, por tanto, este control se realiza en el aplicativo. Puesto que el primer mensaje no necesita interactuar con DialogFlow este se envía directamente con el servicio web de mensajería y puede esperar por un tiempo indefinido.Sin embargo, para efectos de control se recomienda no pasar de 8 horas. Una vez el usuario contesta el primer mensaje (este si utiliza Dialogflow para saber que contestar), allí si se controlan los 10 minutos que dura la sesión.

Para hacer este control el aplicativo se basa en los campos TIME_OUT_SESSION_GENERAL y TIME_OUT_CONVERSATION de la tabla GLOBAL_PARAMETERS. EL campo TIME_OUT_SESSION_GENERAL almacena el tiempo de espera a partir del cual se envía el primer mensaje en minutos y en TIME_OUT_CONVERSATION almacena el tiempo de espera entre cada respuesta de parte del bot hacia el cliente, el cual no deberá ser superior a 10 minutos.

Si un usuario contesta el mensaje el proveedor del servicio web informará al aplicativo consumiendo un webhook, cada x tiempo informando en un json los mensajes que contestaron con el id único que proporcionan al enviar el mensaje por primera vez. Esta respuesta json se itera para cada cliente y se envía la respuesta del cliente a DialogFlow el cual retorna el mensaje que se deberá retornar al usuario, el cual se almacena en la tabla TAIL_PROCESS_BOT_MESSAGES.
El aplicativo está monitoreando qué mensajes se deben enviar de tipo sms_bot y nuevamente consume el servicio_web de mensajería. Esto se realiza entonces de manera cíclica, hasta que se llegue a un mensaje final, el cual indica que se termina la conversación, bien sea porque se pactó una promesa, o porque el usuario termina la conversación. Esto se determina también en el flujo de conversación del proyecto DialogFlow, pues se puede marcar una interacción de la conversación para que consuma un webhook, cuando es un fin de conversación que indica si se pactó o no una promesa, y se actualiza el estado del registro de la tabla WORKLOAD a completado.

Cuando se procesa el webhook para cada usuario también se controla si la sesión no ha expirado. En la tabla BOT_CONTROL_MESSAGES se guarda la fecha de envío del primer mensaje (FIRST_WORK_DATE) y la fecha de envío del siguiente mensaje (LAST_MESSAGE_DATE). Entonces si la columna LAST_MESSAGE_DATE es nula, indica que se ha enviado el primer mensaje hasta ahora, se debe validar si no se ha vencido la sesión con el tiempo global (TIME_OUT_SESSION_GENERAL). Si por el contrario, la columna no está nula, significa que ya el usuario contestó al menos una vez y se evalúa si la sesión no ha expirado con respecto al tiempo de sesión del DialogFlow

(TIME_OUT_CONVERSATION). Cada vez que el usuario contesta, es decir cuando se procesa un ítem del json, que envían al webhook, se actualiza el campo TIME_OUT_CONVERSATION.

Además del control de sesión cuando el usuario responde, existe un control que se hace cíclicamente sobre los registros de la tabla WORK_DONE, evaluando si la sesión está vencida, bien sea por la sesión global, o por la sesión de DialogFlow.

Si al registro de la tabla WORK_DONE, el sistema lo detecta como sesión vencida, se marca como completado, por sesión vencida y no debe gestionarse más.

Proceso de envío de datos del servidor remoto a iCS

Este proceso es cíclico, programado desde nodejs, y contínuamente está revisando la tabla WORK_DONE, si el registro está marcado como completado (C), erróneo (E), fuera de rango (O), sesión expirada (S), o no respondió (N). Para los registros de tipo SMS, WHATSAPP, SMS se contruye una cadena XML para guardar un registro en la tabla HISTORY y otra para actualizar el registro en la tabla SENT_LETTER. Si el registro es de tipo SMS_BOT o WHATSAPP_BOT adicionalmente se verifica si se pactó una promesa, para construir de igual manera una para guardar en la tabla PROMISE.

Para cada registro procesado se consume el servicio web para guardar una transacción y enviar el xml completo (actualizar sent_letter, insertar history e insertar promise si es el caso).

ANEXO 12 - LÓGICA PARA DESPLEGAR CONTENEDORES DOCKER

SPRINT 8

1. DEFINICION

Despliegue de aplicación whatsapp-bot en contenedores sobre Kubernetes

2. CONDICIONES

- Los contenedores deben desplegarse dinámicamente en GCP.
- Se debe usar docker como gestor de contenedores.
- Los contenedores deben ser capaces de subir automáticamente al incrementar el número de nodos del cluster.

3. PROCESO

Se ingresa a Google Cloud Platform, se valida que existan las imágenes correspondientes en Container Registry, Se evidencia la imagen con tag de versión 0.89 creada el 28/05/2018 a las 11:07 UTC-5.

Posteriormente se ingresa a Kubernetes, se explora el cluster contenedor y el nodo de despliegue principal. Dentro del nodo se evidencia que se ha desplegado previamente un nodo con id whatsappbot083-7777f7795-b7jvp. Dentro del nodo se valida que hay un contenedor del proyecto whatsappbot083. Se evidencian los logs del contenedor junto al archivo YAML.

Posteriormente, en el CLI del SDK de GCP se evidencia la descripción del despliegue y se contrasta con los datos existentes en el navegador web para el proyecto que se está trabajando.

Se validan los pods desplegados para whatsappbot083, existe solo 1 y se comprueba que los ids de los pods sean equivalentes, una vez se tiene la descripción se realiza el proceso de escalado iniciando una nueva replica, Posteriormente se visualizan nuevamente los pods del proyecto y se puede ver que ahora han iniciado dos con el mismo nombre del proyecto, pero con ids diferentes.

Pese a que en el navegador web no se visualiza, se puede ver en la línea de comandos el nuevo Pod con estado Pending y AGE de 6s.

Se recomienda seguir el proceso descrito con el video adjunto al sitio del proyecto en la herramienta Project Online

ANEXO 13 - EXTRAER QR Y CONSUMIR SERVICIOS REST REMOTOS

HTTP GCP (VPC interna del proyecto)

Cod HTTP	responseCode	Descripcion
200	1000	Peticion completada sin errores
200	1001	No hay numeros de celular disponibles para asignar
401	1002	Los datos de autenticacion son inexistentes

Mtodo HTTP	Endpoint	Parametros
POST	/getInitialParameters	{"specName" : [POD_SP

HTTP GCP (VPC interna del proyecto)

Respuesta	Descripcion
{ status : 'Peticion completada' , responseCode : 1000, responseData : { db : [DB_PROXY_KUBERNETES], phone : { phoneNumber : [ASSIGNED_PHONE_NUMBER], countryCode : [ASSIGNED_PHONE_COUNTRY_CODE] } } }	Endpoint de inicialización, validará que el specname que se envia como parametro exista dentro del cluster de Kubernetes y tenga un estado apropiado para trabajar (Por defecto, running). Posteriormente, buscará un número celular disponible y lo asignará al Pod, cambairá el estado y enviará datos de vuelta al Pod que inició la petición

Tabla WHATSAPP_TELEPHONES

Status	Descripcion
I	Inactivo, disponible para asignar
P	Pendiente, se ha asignado por petición a un Pod en Kubernetes y está esperando inicio de sesion de whatsapp
Q	Esperando QR, el pod inició correctamente, escribió el QR y está esperando para el escaneo
A	Activa, el contendor asignado ha iniciado sesion en whatsapp y está trabajando con el numero
E	Con errores, se ha desconectado el numero desde whatsapp
N	No disponible
F	Cola llena, la linea no está disponible para recibir más trabajo

STATUS

Letra	Abreviacion	Requiere Reintento
P	Pending	NO
A	Assigned	NO
V	Validation Required	SI
C	Completed	NO
E	Errored	SI
O	Out of range	NO
W	Waiting	NO
N	No Response	NO
S	Session Expired	NO
H	Half Way	NO

ANEXO 14 - REALIZAR LÓGICA DE PETICIONES DE SMS UNA VÍA Y EMAIL

IV.- Consumir WS a través de REST

Dirección WS : http://inticosms.com:8090/rest/webresources/envioSMS/sms
User :xxxxx
Password : xxxxx

Para realizar el envío de SMS, se debe invocar al método **sms**, entregándole como parámetro: **usuario, password, celular, mensaje, senderId** (en el mismo orden).
Por cada envío el método entregará como retorno el **codigo** que permitirá conocer el estado de entrega del SMS enviado.

Para realizar la solicitud de estados de entrega, se debe invocar al método **confirma** entregándole como parámetro: **usuario, password, codigo** (en el mismo orden).

Para realizar la solicitud de respuestas SMS asociadas a un número, se debe invocar al método **respuesta** entregándole como parámetro: **usuario, password, celular, fecha** (en el mismo orden).

V.- Retornos WS :

I) Envío de SMS :

- Cuando se realicé un envío de SMS exitoso el WS retornará en formato JSON :

{ estado:1 , codigo: OKabcdef }

Donde:

estado=1 : El sms llegó a la plataforma INTICO.

codigo= OKabcdef : El código que entrega el sistema asociado al SMS enviado.

- Cuando se produzca un error al momento de enviar un SMS el WS retornará:

  - **error: usuario o password incorrectos:** Se produce cuando la autentificación con el WS es incorrecta.
  - **error: parametro (tipo de parametro) vacio:** Se produce cuando algún parámetro esperado por el WS es enviado en blanco.
  - **error: parametro celular incorrecto:** Se produce cuando el numero del destinatario SMS no cumple con formato en la cantidad de números.
  - **error: IP no autorizada:** Se produce cuando la IP del cliente no está registrada en el sistema INTICO.

II) Estados de Entrega:

- Cuando se realice una solicitud de estado de entrega de SMS el WS retornará:

  - **{estado: 1, flag_entrega: (C, R, P), fecha_entrega:YYYYMMDD, hora_entrega: HH:MM:SS }:** Indica la información de entrega o rechazo de un SMS enviado.

    - **estado:** Indica si el sms ha llegado a la plataforma INTICO.
    - **flag_entrega:** estado de entrega del SMS.
    - **fecha_entrega:** Fecha de entrega del SMS.
    - **hora_entrega:** Hora de entrega del SMS.

  El flag_entrega puede tener 3 distintos estados:

    - P: SMS en espera de Confirmación o Rechazo Entrega del Operador
    - C: SMS fue Confirmado de entrega hacia la Red del Operador
    - R: SMS fue Rechazado de entrega desde la Red del Operador

- Cuando se produzca un error al momento de consultar el estado de entrega de un SMS el WS retornará:

- **error: usuario o password incorrectos:** Se produce cuando la autentificación con el WS es incorrecta.

- **error: parametro (tipo de parametro) vacio:** Se produce cuando algún parámetro esperado por el WS es enviado en blanco.

- **error: IP no autorizada:** Se produce cuando la IP del cliente no está registrada en el sistema INTICO.

- **error: El mensaje no existe:** Se produce cuando el código ingresado es incorrecto o no está asociado a un mensaje de la plataforma.

VI.- Seguridad

El WebService de INTICO requiere previamente autorizar la IP desde donde el cliente realizara las peticiones de uso del mismo.

VI.- Ejemplo y Script de Envío SMS

Para realizar los envíos por el WebService REST de INTICO, se deben envíar los parámetros mediante el método POST. La respuesta a los envíos retorna un formato JSON.

a) Ejemplo en JAVA.

```
package test;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import javax.xml.parsers.ParserConfigurationException;
import org.xml.sax.SAXException;
```

```
public class clienteREST {

    public static void main(String[] args) throws ClientProtocolException, IOException,
            ParserConfigurationException, SAXException {

        String respuesta = "";
        HttpClient client = new DefaultHttpClient();

          HttpPost post = new HttpPost("http://inticosms.com:8090/rest/webresources/envioSMS/sms
                  ");

List nameValuePairs = new ArrayList(1);

        nameValuePairs.add(new BasicNameValuePair("usuario", "usuario"));
        nameValuePairs.add(new BasicNameValuePair("password", "password"));
        nameValuePairs.add(new BasicNameValuePair("celular", "989216368"));
        nameValuePairs.add(new BasicNameValuePair("mensaje", "Mensaje de Prueba"));
        nameValuePairs.add(new BasicNameValuePair("senderId", "test"));

        post.setEntity(new UrlEncodedFormEntity(nameValuePairs));
        try {

            HttpResponse response = client.execute(post);
                    BufferedReader rd = new BufferedReader(new
                    InputStreamReader(response.getEntity().getContent()));
            String line;
            while ((line = rd.readLine()) != null) {
                respuesta = respuesta.trim() + line.trim();
            }

            System.out.println(respuesta);

        } catch (Exception e) {

            System.out.println(e.getMessage());
        }

    }

}
```

La respuesta en formato JSON que retornará será la siguiente:

{ estado: 1, codigo: OKabcdefg }

ANEXO 15 - PROCESAR TRABAJO PENDIENTE Y ENRUTARLO AL CANAL CORRESPONDIENTE

SPRINT 12

1. DEFINICION

Realizar la lógica correspondiente al proceso que toma el trabajo pendiente y lo enruta al canal correspondiente

2. CONDICIONES

- El proceso maestro debe ser capaz de determinar el trabajo pendiente y enrutarlo al canal que corresponda mediante el despliegue de procesos hijos.
- Una vez se inicie el proceso hijo (que trabajará por Customer y Letter Type), este debe realizar el envío de peticiones al servidor remoto tan rápido como se pueda de acuerdo con el parámetro MAX_REQUEST_PER_SECOND de la tabla REST_SERVICES.
- Al finalizar el trabajo pendiente, los procesos hijos deben informar al proceso maestro para que este los finalice apropiadamente, evitando dejar trabajo sin procesar. El proceso maestro a su vez debe ser capaz de determinar cuándo un proceso hijo finaliza por petición o por un error no controlado (como fallas de memoria en el servidor), en este último caso debe ser capaz de levantar nuevamente el proceso hijo
- Se usará NodeJs 8.11.2 LTS como entorno de desarrollo de la aplicación.
- Durante el proceso, se llamarán a los procesos hijos como Workers.
- Los workers deberán procesar únicamente el trabajo que se les asigne para el día en el que se están ejecutando (09/07/2018).

3. PROCESO

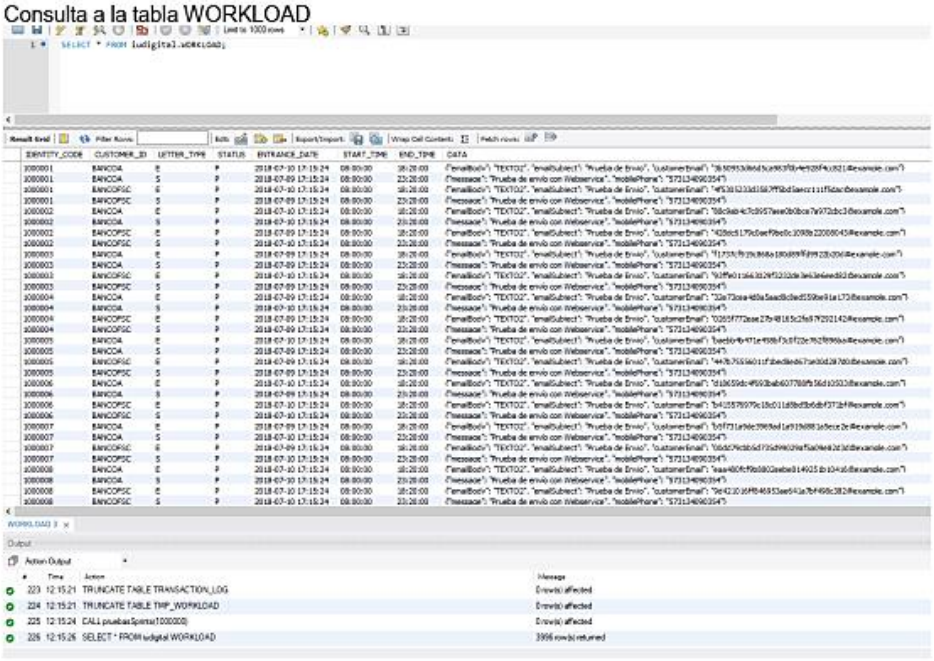
Se cargaron 3996 registros aleatorios, que corresponden a trabajo por realizar en 2 días (09/07/2018 y 10/07/2018) para 2 entidades (BANCOA y BANCOFSC) y para 2 tipos de carta (E: Email y S: SMS de 1 vía).

Se iniciará el proceso maestro, este deberá desplegar 4 workers, uno por cada entidad y tipo de carta, es decir:

1. Worker1 -> BANCOA : S
2. Worker2 -> BANCOA : E
3. Worker3 -> BANCOFSC : E
4. Worker4 -> BANCOFSC : S

Al desplegar cada worker, este procesará el trabajo que tiene asignado y empezará a realizar las peticiones correspondientes. Para efectos de pruebas se logueará en consola cuando se desplieguen y terminen los workers. Y se utilizará un webservice REST local que responde lo mismo que se le envíe en 2 endpoints (/ws1 y /ws2). A su vez, y para validar que sea el webservice quien response, se añadirá a la respuesta un parámetro que puede ser PONG1 o PONG2

Consulta a la tabla WORKLOAD



Las tablas WORK_DONE y TMP_WORKLOAD estarán vacías al empezar la prueba.

La prueba se puede visualizar en el archivo PruebaSprint12.tvs adjunto al proyecto.

Durante la prueba se puede evidenciar por logs del servidor el despliegue de los workers correspondientes, el tiempo de ejecución y la diferencia de datos en las tablas WORK_DONE y WORK_LOAD, como se evidencia, los workers desplegados únicamente realizaron el trabajo asignado para el día 09/07/2018. Dado que las actividades restantes están programadas para realizarse en otras fechas.

En total se procesaron 1962 registros a 2 webservices en 65.34 segundos, es decir, un ratio de envío de aprox 30.02 peticiones / s.

ANEXO 16: WEBHOOK QUE RECIBE PARÁMETROS DESDE DIALOGFLOW

DOCUMENTACION WEBHOOK

Se expondrá un webhook que tendrá como objetivo recibir los mensajes de texto de doble vía que se hayan respondido en un plazo de tiempo determinado por FSC.

Endpoint: https://webhook.iucollect.com/push-sms-status
Método: POST
Content-Type: application/json

Payload de la petición:

El cuerpo o carga útil de la petición debe contener un array JSON cuyos nodos hijos serán objetos con la siguiente estructura:

```
[
    {
        "send-code": "CODIGOENVIOEJEMPLO",
        "reception-code": "CODIGORECEPCION",
        "reception-date": "2018-06-15",
        "reception-time": "15:00:05",
        "country-code": "57",
        "phone-number": "3131234567",
        "message": "Mensaje de respuesta"
    },
    {
        "send-code": "CODIGOENVIOEJEMPLO2",
        "reception-code": "CODIGORECEPCION2",
        "reception-date": "2018-06-15",
        "reception-time": "15:05:33",
        "country-code": "57",
        "phone-number": "1234567890",
        "message": "Mensaje de respuesta 2"
    }
]
```

Headers mínimos requeridos:

La petición HTTP que se realice al webhook debe contener como mínimo 2 headers:

- Authorization: Ver la sección "Autenticación con el webhook"
- Content-Type: Determina la codificación del cuerpo de la petición, debe enviarse como "application/json"

Parámetros requeridos:

- ***send-code*** [String]: Código de respuesta otorgado por Proveedor de mensajería al enviar el primer SMS
- ***reception-code*** [String] : Código único de recepción
- ***reception-date*** [String]: Fecha de recepción en formato YYYY-MM-DD
- ***reception-time*** [String]: Hora de recepción en formato HH:mm:ss (24 Horas)
- ***country-code*** [Number | String]: Código de país
- ***phone-number*** [Number | String]: Código de área (Si aplica) + Número celular desde el cual se recibió el mensaje.
- ***message*** [String]: Cadena del mensaje (No debe tener palabras clave para identificar que es una respuesta)

Tenga en cuenta que todos los parámetros son requeridos y las llaves deben estar en minúsculas.

Autenticación con el webhook:

En la petición debe incluirse el header Authorization, este debe contener como valor un string que concatena los siguientes parámetros:

- ***Nombre de Usuario:*** Proveído por FSC, será un string que identifica al usuario que está intentando enviar la petición al Webhook, el parámetro debe enviarse en minúsculas.
- ***Hash HMAC:*** Para generar este parámetro, se debe tomar la carga útil de la petición (string) y aplicar cifrado SHA256 con una llave proveída por FSC. Posteriormente se debe convertir el resultado al equivalente hexadecimal. Este parámetro debe enviarse en minúsculas.

Ejemplo de construcción:

Visualice el siguiente ejemplo de construcción, este se muestra en pseudo código, por lo cual tendrá que encontrar la equivalencia en el lenguaje que esté usando.

Se construirá el header de autenticación para la siguiente carga útil:

```
[
    {
        "send-code": "CODIGOENVIOEJEMPLO",
        "reception-code" : "CODIGORECEPCION",
        "reception-date" : "2018-06-15",
        "reception-time" : "15:00:05",
        "country-code": "57",
        "phone-number": "3131234567",
        "message": "Mensaje de respuesta"
    }
]
```

Note que la carga útil corresponde al string del JSON sin espacios entre los dos puntos ni las comas, además, se han removido los saltos de línea y las tabulaciones. Normalmente los serializadores de JSON populares realizarán este trabajo.

```
var     carga_util     =     "[{"send-code":"CODIGOENVIOEJEMPLO","reception-
code":"CODIGORECEPCION","reception-date":"2018-06-15","reception-time":"15:
00: 05","country-code":"57","phone-number":"3131234567","message":"Mensaje de
respuesta"}]";

var usuario_webhook = "example";
var llave_firmado = "CC89F0B4CFCC48AD04D3";

var firma_hmac = createHMAC('sha256', llave_firmado, carga_util);
var firma_hex = firma_hmac.toHexadecimal();
var firma_final = firma_hex.toLowercase();

var header_autorizacion = usuario_webhook + firma_final;
```

Al recuperar el valor de la variable header_autorizacion se debe tener el siguiente valor:

```
example7c297c876598d646d877ec93d3402b39efac2fe4fda82a8b7a31373eac244c1a
```

Respuesta satisfactoria

Cuando se envíe una petición satisfactoria, el webhook responderá un JSON con la siguiente información:

```
{
  "status": "SUCCESS",
  "message": "Envío de datos completado con éxito",
  "utc-date": "2018-06-15T17:27:26.828Z"
}
```

El parámetro utc-date corresponde a la fecha UTC de acuerdo al estándar ISO-8601, en formato YYYY-MM-DDTHH:mm:ss.sssZ