UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

FACULTAD DE INGENIERÍA

ANÁLISIS DEL PROCESO DE COMUNICACIÓN DE CRM CON ARQUITECTURAS DE SOFTWARE EXTERNAS UTILIZANDO OPEN APIS

TESINA

QUE PARA OBTENER EL TÍTULO DE

INGENIERA EN COMPUTACIÓN

PRESENTA:

ALEJANDRA CARMONA FUENTES

ASESOR:

DR. MARCELO ROMERO HUERTAS

TOLUCA, EDO MÉXICO 2022

Resumen.

En la actualidad se ven constantes cambios tecnológicos, día con día aparecen nuevas tecnologías que permiten cambiar la forma de trabajar, comprar, vender y hasta la forma de relacionarse. Hablando específicamente del proceso de compraventa, ya no es necesario hacerse de forma presencial como en el pasado, ahora se usa un sistema de software que permite ofrecer productos y servicios a través de internet (FORBES, 11 de diciembre de 2020). Para ello existe software basado en la nube, el cual es alojado por proveedores externos poniendo sus servicios a disposición de los usuarios. Las empresas que adoptan una tecnología de este tipo lo hacen pensando en optimizar el soporte y mantenimiento de sus sistemas y programas, además de mejorar sus flujos de trabajo.

En los procesos de venta actuales, el cliente tiene más atención, puesto que, independientemente de los productos que se vendan, lo importante es convencerlo para que realice la compra. Si el objetivo principal de la empresa es el cliente, entonces, se necesita hacer uso de un *CRM* (*Customer Relationship Management*) basado en la nube para que de esta manera se tenga acceso a los productos y servicios a través de internet en cualquier momento.

Un CRM, específicamente *Salesforce*, permite la integración de toda la empresa, tecnologías trabajando conjuntamente como son bases de datos, sitio web, intranet-extranet, sistema de apoyo telefónico, contabilidad, marketing, ventas y producción, para permitir la comunicación entre las distintas partes de la organización y así dar un mejor servicio al cliente. Para potencializar el funcionamiento del CRM se puede hacer una integración de éste con aplicaciones de terceros con el propósito de ahorrar tiempo y dinero evitando se hagan constantemente cambios entre sistemas. Una integración permitirá tener toda la información de la organización centralizada y disponible en cualquier momento a través de internet. Usar una tecnología basada en la nube no implica necesariamente la desconexión de infraestructuras heredadas o externas, por el contrario, se pueden integrar ambas tecnologías reduciendo costos de mantenimiento, debido a que muchas veces la mayoría de las infraestructuras heredadas poseen un complejo conjunto de dependencias que no están documentadas. La integración o comunicación de varias tecnologías o sistemas es la columna vertebral para cualquier transformación digital.

Por ejemplo, si se tiene un sistema de cotización, facturación o pedidos de clientes este se puede integrar al CRM sin que tengan que hacerse grandes desarrollos de software. En caso de no tenerse, se podría contratar con un proveedor de servicios e integrarse. Para realizar la integración con estos sistemas externos se debe buscar un modelo de comunicación entre los sistemas y el CRM, este modelo debe indicar los estándares a seguir para la comunicación de los mismos pueda llevarse a cabo y sea entendible, sin importar el lenguaje de programación en el que estén desarrollados.

En esta tesina se busca explicar cómo realizar la comunicación del *CRM Salesforce* con infraestructuras de software externas o independientes utilizando como modelo de comunicación el marco de trabajo de *TM Forum Open APIs*, el cual define las especificaciones del modelo de datos compartido, diseñado para permitir la integración de datos de los diferentes sistemas. Algunos ejemplos de modelos enfocados principalmente a procesos comerciales son: Catálogo de productos, gestión de pedidos, facturación, cotizaciones o un *ERP* (*Enterprise Resource Planning*).

Para lograr una comunicación exitosa entre sistemas con diferentes arquitecturas es necesario contar con estándares o modelos que indiquen cómo lograr la comunicación, es por ello, que *TM Forum* se ha dedicado a desarrollar un marco de trabajo para definir dichos estándares. *TM Forum* es una alianza de empresas a nivel mundial que ha definido más de 50 *Open APIs* basadas en *REST*, las cuales permiten una integración rápida y flexible de sistemas.

Contenido

Introducción.

La comunicación en la actualidad ha tenido grandes avances y cambios, gracias a toda la tecnología que día con día emerge en nuestro entorno, la cual permite agilizar u optimizar algunas de las actividades que realizamos a diario. Nos encontramos ante un mundo global digitalizado que ha permitido cambiar la forma en la que interactuamos y trabajamos. Otro cambio muy notorio es la forma en la que se compran o se venden productos. Las empresas necesitan sumarse a este proceso de transformación digital para no quedarse obsoletas, si saben aprovechar bien estas tecnologías pueden generar nuevas oportunidades de negocios y abrirse paso en nuevos mercados. Es por ello, que cada vez son más las organizaciones que buscan nuevas soluciones para poder acercarse a los clientes y tener un mejor proceso de ventas, el *CRM (Customer Relationship Management)* es una de estas tecnologías que permite a las empresas transformarse e innovar dentro del sector IT/software.

Un *CRM* es de gran utilidad para el proceso de ventas en una organización y si se utiliza uno basado en *SaaS* (*Software as a Service*) es un plus extra, debido a que permite usar software por medio de internet como un servicio. Con este modelo, la empresa no necesita instalar, mantener y actualizar software o hardware, tendrá acceso a la información sólo con tener acceso a internet. No obstante, las tendencias están demostrando que tener el *CRM* en la nube tiene todavía más ventajas debido a que se puede integrar información de sistemas locales y sistemas externos en la nube y tenerla disponible en cualquier lugar y en cualquier momento.

Aunque existen en el mercado varias opciones de *CRM* este trabajo se enfoca en *Salesforce*, debido a que se posiciona como uno de los mejores CRM en el mercado que hace uso de tecnología de la nube. Además, de acuerdo a Garnet el 70% de las empresas consideran a *Salesforce Service Cloud* como su primera opción al momento de implementar una herramienta de este tipo. Se explica cómo al ya tener implementado *Salesforce* se puede interactuar o comunicarse con infraestructuras de software externas para poder integrar toda la información en la nube y de esta manera tenerla disponible desde cualquier lugar y en cualquier momento haciendo uso de internet.

Para poder realizar la comunicación entre sistemas externos y *Salesforce* se hace uso del marco de trabajo *SID (Shared Informational and Data)* de *TM Forum*, el cual plantea un conjunto de especificaciones técnicas para el mundo de los proveedores de servicios de telecomunicaciones que abarca elementos como procesos, datos, aplicaciones e integración. Dentro de la integración se define los estándares y buenas prácticas que se deben seguir para tener una buena comunicación entre sistemas independientemente del proveedor o lenguaje de programación que estos utilicen, esto se logra mediante el uso de *TM Forum Open API*.

La comunicación de sistemas se realiza mediante servicios web y puede ser de *Salesforce* hacia el sistema externo, en este caso, el *CRM* estará consumiendo el servicio del sistema legado o externo. En caso contrario, es decir, si el sistema externo se comunica con *Salesforce*, el *CRM* estará exponiendo un servicio. Tanto en la exposición como en el consumo de servicios se pueden realizar operaciones de actualización, creación, consulta y borrado de información.

Específicamente en este trabajo se hará uso de las *Open APIs PartyManagement* y *Geographic Address Management* para demostrar cómo se realiza la integración o comunicación de diferentes sistemas con Salesforce.

Objetivo General

Documentar el proceso de integración del *CRM* (*Customer Relationship Management*) *Salesforce* con arquitecturas de software externas utilizando el marco de trabajo de *TM Forum de Open APIs* para definir la estructura *json* que permite enviar y recibir información entre diferentes sistemas.

Alcances y limitaciones

Este trabajo se centra en *Salesforce Sales Cloud*, aplicación *CRM* principal, debido a que es el servicio que permite agilizar ventas, administrar clientes, cotizaciones, facturas y pagos, es por ello, que es el más utilizado por las empresas. Así mismo, el modelo de *TM Forum* cuenta con más de 65 *Open APIs* que pueden usarse en la integración de sistemas, sin embargo, se usa la *API PartyManagement y la API Geographic Address Management* para analizar y ejemplificar cómo se logra la comunicación entre diferentes estructuras de software, tanto para enviar como para recibir información a través de *Open APIs*.

La *API PartyManagement* se usa para enviar y recibir información de clientes y contactos, los cuales son objetos esenciales dentro del *CRM* para poder dar un mejor seguimiento a las ventas. Es decir, dentro de Salesforce se centralizará toda la información de clientes y contactos que se encuentre almacenada en otros sistemas, de esta manera se tendrá siempre disponible en la nube para realizar cualquier operación.

La *API Geographic Address Management* se usa para enviar y recibir información de direcciones, en este caso, direcciones relacionadas a los clientes.

Organización del documento

El presente trabajo se organiza de la siguiente manera:

Capítulo I *CRM* y *Open APIs*, en este capítulo se describe de manera general qué es un *CRM*, características y ejemplos. Se define el concepto de *API* y *Open APIs*, así como la interpretación de las especificaciones de un *Open API*.

Capítulo II *Salesforce* y *TM Forum Open APIs*, es este capítulo se describe *Salesforce* como un *CRM* basado en la nube, el cual para poder comunicarse con otros sistemas hace uso de servicios web usando el protocolo *HTTP*. Además, se describe el marco de trabajo de *TM Forum* que proporciona los estándares para que pueda realizarse dicha comunicación a través de *TM Forum Open APIs*.

Capítulo III Comunicación de *Salesforce* con Sistemas Externos, se describe cómo se realiza la comunicación de *Salesforce Sales Cloud* con sistemas externos, utilizando la *API PartyManagement* para integrar la información de clientes y contactos, así como la *API Geographic Address Management* para integrar información de direcciones. La comunicación de los diferentes sistemas con *Salesforce* se realiza mediante el uso de servicios web *REST*, que permiten realizar la consulta, actualización o creación de información en dichos objetos.

Capítulo I. CRM Y Open APIs.

En este capítulo se describen las tecnologías *CRM* y *Open APIs* las cuales son esenciales para el desarrollo de este trabajo.

1.1.- Definición de CRM.

Existen varias definiciones de *CRM*, para empezar por sus siglas es *Customer Relationship Management*, es decir, Manejo de Relaciones con el Cliente. A continuación, se presentan varias definiciones de acuerdo a los diferentes vendedores que existen en el Mercado.

- **Salesforce.**

CRM es una tecnología para gestionar todas las relaciones o interacciones de la empresa con el cliente y clientes potenciales. El objetivo de Salesforce es mejorar las relaciones comerciales para hacer crecer un negocio. Ayuda a las empresas a mantenerse conectadas con los clientes, optimizar procesos y mejorar la rentabilidad (Salesforce, 2021).

- **SAP.**

CRM anteriormente se utilizaba para referirse a la gestión de clientes. Actualmente se habla de *Customer Experience* o Experiencia del Cliente, debido a que no sólo se enfoca en procesos de *Front Office,* sino en procesos de punta a punta, es decir, desde el *front* hasta el *Back Office*. A los tradicionales procesos de ventas y servicio del *CRM* hay que agregarle todos los procesos de gestión de canales digitales, además, la posibilidad de vender productos y servicios por medios digitales y por supuesto la captura de la voz del cliente al momento de interactuar con el CRM para ofrecer una mejor experiencia en su compra. *CRM* es una parte de *CX* o *Customer Experience* (SAP, 2021).

- **Oracle.**

CRM es un sistema de software que gestiona las relaciones con los clientes. Para gestionar, analizar y mejorar las relaciones con los clientes de manera eficaz se necesita un conjunto integral de soluciones en la nube de acuerdo a las necesidades del cliente. Dicha solución debe incluir ventas, servicios, comercio y marketing, así como una plataforma de datos de clientes (*CDP*) habilitada para trabajar en conjunto con Inteligencia Artificial (*IA*) (Oracle, 2022).

- **Microsoft Dynamics.**

CRM es una solución de software que abarca distintos sistemas de gestión de clientes (ventas, servicio y marketing) y les permite funcionar en conjunto. Agiliza los procesos empresariales y conecta los datos de los clientes para ayudarle a crear

relaciones, aumentar la productividad y mejorar la interacción con ellos (Microsoft, 2022).

Existen otras definiciones de acuerdo a:

- **Forrester.**

Un *CRM* es un conjunto de procesos de negocio y tecnologías de apoyo que respaldan las actividades clave de focalización, adquisición, retención, comprensión y colaboración con los clientes (FORRESTER, 2018).

- **Gartner.**

Un *CRM* es una estrategia de negocios diseñada para optimizar la rentabilidad, las utilidades y la satisfacción de los clientes (Glender, 2018).

Tomando en cuenta las definiciones anteriores y desde un punto de vista personal, un *CRM* es un software basado en estrategias de negocio para cubrir las necesidades del cliente, utilizando un conjunto de tecnologías que permite mejorar interacción del cliente con la organización, de esta manera mejorar los procesos de ventas, marketing y servicios, con adaptabilidad a las diferentes arquitecturas de datos e interactuando con varios canales, lo que permite a las empresas aumentar sus ventas.

1.2.- Características de un *CRM*.

Existen varias opciones de *CRM* en el mercado. Todos ellos con la filosofía de proporcionar a las organizaciones soluciones para reforzar vínculos con sus clientes o posibles clientes. Una de las principales características a tomar en cuenta al momento de hacer uso de estas herramientas, es que se debe adaptar a las necesidades y procesos de cada organización. A continuación, se describen las características más importantes a considerar en un *CRM*.

1.- **Interfaz fácil de aprender a usar**: Un *CRM* debe ser una herramienta intuitiva, fácil de utilizar por los usuarios y fácil de implementar.

2.- **Automatización de procesos:** La herramienta debe ser capaz de automatizar tareas repetitivas para ahorrar tiempo a los usuarios y poder automatizar tareas administrativas de seguimiento a prospectos, por ejemplo, envío de e-mails.

3.- **Herramientas de informes**: Debe contar con las herramientas necesarias para poder realizar informes de análisis, predicciones y envío de notificaciones sobre los procesos de venta para ajustar las estrategias de marketing.

4.- **Gestión de clientes y contactos:** debe permitir crear, actualizar y buscar clientes y contactos en una interfaz intuitiva y fácil de usar. Debe poder integrarse con el correo electrónico y llamadas telefónicas.

5.- **Historial de interacción:** debe generar registros de las interacciones entre el cliente y la organización de manera automática para poder conocer el comportamiento de compra de los clientes.

6.- **Integración:** un buen *CRM* debe poder integrarse con otros sistemas que ya existan en la organización y se estén utilizando.

7.- **Servicio de Soporte:** debe contar con un servicio de soporte por parte del proveedor seleccionado que pueda apoyar a los colaboradores con respecto al uso del sistema y apoyarlos durante el proceso de implementación.

8.- **Omnicanalidad:** debe permitir integrar los diferentes canales de comunicación con el cliente, por ejemplo, el email, el chat, tienda online o tienda física. De este modo se puede dar un mejor servicio al cliente.

9.- **Acceso Móvil:** debe ser una herramienta que permita el acceso móvil desde cualquier dispositivo (PC, portátil, Tablet, celular) utilizando cualquier sistema operativo. Esto permite aumentar la productividad en las ventas.

1.3.- Tipos de *CRM*.

Conocer los diferentes tipos de *CRM* que existen es indispensable para poder elegir el más adecuado para cada organización. Si la organización elige el *CRM* correcto podrá tener control de todos sus procesos (ventas, marketing, gestión comercial, contabilidad o finanzas).

No existe un *CRM* estándar, por ello, es importante conocer las opciones que existen en el mercado. Existen tres tipos principales, aunque también se pueden establecer otras clasificaciones.

De acuerdo a sus características y funciones se pueden clasificar en *CRM* según su gestión y según su acceso. De acuerdo a su gestión se pueden clasificar en operativos, analíticos y colaborativos. De acuerdo a su acceso en *On Premiss* y On Demand.

1.- *CMR* según su gestión.
- *CRM* Operativos.
- *CRM* Analíticos.
- *CRM* Colaborativos.

2.- *CRM* según su acceso.
- *CRM On Premiss*.
- *CRM On Demand*.

1.3.1- *CRM* según su gestión.

Los *CRM* de gestión se dividen en tres grandes grupos, que pueden o no, colaborar entre sí; operativos, analíticos y colaborativos.

1.3.1.1- CRM Operativo

Los *CRM* operativos también llamados *CRM* de *Front Office*, debido a que se enfocan en las funciones cara a cara con el cliente, son el ejemplo más completo dentro de este tipo. Su objetivo principal es que sus clientes se conviertan en contactos para posteriormente realizar ventas e incrementar ganancias, centrándose en tres áreas claves de la organización que son: ventas, marketing y servicios.

La **automatización de ventas** permite adquirir nuevos clientes y gestionar los ya existentes para incrementar o proyectar de mejor manera los procesos comerciales de ventas.

La **automatización de marketing** permite a la organización seleccionar cuáles son los canales efectivos para realizar el proceso de ventas, por ejemplo, por medio de llamadas telefónicas, emails, anuncios o redes sociales.

La **automatización de servicios** se enfoca en retener los clientes que ya existen en la organización implementando servicios de calidad. Un ejemplo de esto es la implementación de chats en vivo para dar seguimiento a las necesidades de cada cliente.

Todas las organizaciones que deseen mejorar la productividad en sus ventas, ya sea pymes o grandes empresas se pueden beneficiar de este tipo de *CRM*. Algunos ejemplos son ***Hubspot***, ***Microsoft Dinamycs*** y ***Salesforce***.

1.3.1.2. CRM Analítico.

Los *CRM* analíticos, también conocidos con el nombre de Inteligencia de Negocio (*Business Intelligence)*, permiten realizar análisis sobre la información disponible del cliente y las ventas. El término analítico es porque se enfoca al procesamiento y tratamiento de datos, se centra en el análisis de información con el objetivo de acelerar las ventas de la organización.

De manera general los componentes principales de un *CRM* analítico son el almacenamiento de datos, la minería de datos y las herramientas de procesamiento analítico en línea (*OLAP*). El almacén de datos (*Data Warehouse*) ayuda a

almacenar toda la información, actual e histórica, referente al cliente para su posterior análisis. La minería de datos (*Data Mining*) permite descubrir tendencias y patrones en el ciclo de vida del cliente, por ejemplo, se puede conocer el patrón de compra del cliente, lo que permitirá detectar áreas de oportunidad en el proceso de ventas. Las Herramientas *OLAP* (*OnLine Analytical Processing*) usando la información de la minería de datos, que permite conocer cuándo, cómo y dónde interactúan los clientes con la empresa, son útiles para crear mejores estrategias de ventas, marketing y servicio.

Un *CRM* analítico es más caro que un operativo, su curva de aprendizaje es mayor, es decir, son más difíciles de aprender a usar, es por ello, que se recomienda para organizaciones que cuentan con una gran cantidad de información histórica sobre clientes, ventas y campañas. Un ejemplo de este tipo es ***Zoho*** y ***SugarCRM***.

1.3.1.3. CRM Colaborativo*.*

Los *CMR* colaborativos, también llamados sociales, son necesarios para las organizaciones que cuentan con múltiples canales de comunicación, especialmente si son digitales. Su objetivo es monitorear lo que el cliente opina a través de distintas fuentes como son:

- Redes sociales y comunidades
- Chats, correos, teléfono.
- Foros, comentarios, noticias y páginas web.

Se recomiendan para las organizaciones que desean que sus diferentes departamentos tengan comunicación y de esta manera mejorar sus procesos. Si una empresa no desea compartir información del cliente dentro de toda la organización este tipo de *CRM* no es para ella. También lo pueden utilizar organizaciones con múltiples ubicaciones que requieran sincronizar la información del negocio. Ejemplos de este tipo son ***Cooper***, ***HootSuite*** y ***Salesforce*.**

1.3.2- *CRM* según su acceso.

Hasta el momento se tiene identificada la primera clasificación de un *CRM*, sin embargo, se puede considerar otra clasificación que es *CRM On Premiss* y *CRM On Demand*, los cuales a su vez pueden ser operativos, analíticos o colaborativos.

1.3.2.1.- CRM On Premiss.

Es un *CRM* local que se desarrolla de acuerdo a las necesidades específicas de cada organización, se almacena y administra directamente desde un servidor de la empresa y el costo suele ser elevado. Es un software a medida de cada organización que habitualmente se construye a través de módulos de *SAP*. La empresa tiene control total de datos e información y cada una de las actualizaciones o módulos a incluir tienen un costo extra.

1.3.2.2.- CRM On Demand.

También conocidos como *CRM* online o en la nube, se paga por su uso (*SaaS*, *Software as Service*), debido a ello su costo de implementación es bajo. Su acceso es inmediato a través de internet y desde cualquier dispositivo. Es ideal para organizaciones que no cuentan con infraestructuras grandes debido a que todo está almacenado en la nube. Se tiene como ejemplo ***Salesforce*** y ***Zoho***.

Cabe mencionar, aunque no se considera como una clasificación relevante, que también pueden dividirse según su tipo de código en *CRM* privado y *CRM Open Source*. Los *CRM* privados son un software cuyo código pertenece a una empresa privada que no permite realizar cambios al código sin el pago de una licencia. Los *CRM Open Source* su código es libre y compartido con toda la comunidad, su gran ventaja es el costo debido a que es gratis, cada organización puede modificar el código y adaptarlo de acuerdo con sus necesidades, dos ejemplos muy utilizados son ***SugarCRM*** y ***Vtiger***.

1.4.- Concepto de *API*.

Hoy en día, las *APIs* son parte fundamental de cualquier sistema debido a que permiten la comunicación entre diferentes aplicaciones simplificando la forma en la que los desarrolladores integran éstas con la arquitectura actual. Juegan un papel muy importante en la organización principalmente en el área de IT y el área comercial. Las necesidades comerciales de cualquier organización pueden cambiar constantemente de acuerdo a cómo se comportan los mercados digitales. Para que la organización pueda seguir siendo competitiva en necesario que implementen o desarrollen servicios innovadores y esto lo logrará a través del uso de las *APIs*, las cuales permiten conectar la infraestructura de la organización a través del desarrollo de aplicaciones nativas de la nube, pero también le permiten compartir sus datos con clientes y otros usuarios externos.

Quizá muchos de nosotros hemos visto o estamos familiarizados con el término *API*. De acuerdo a sus siglas, *Application Programming Interface*, traducido Interfaz de Programación de Aplicaciones. A continuación, se mencionan algunos conceptos o definiciones de *API*.

Una *API* es un conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar el software de las aplicaciones. Las *APIs* permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero. En ocasiones, las *APIs* se consideran como contratos, con documentación que representa un acuerdo entre las partes: si una de las partes envía una solicitud remota con cierta estructura en particular, esa misma estructura determinará cómo responderá el software de la otra parte (RedHat, 2020). En la Figura 1 se muestra cómo las *APIs* pueden conectar una infraestructura a través del desarrollo de diferentes aplicaciones. Como se puede observar el funcionamiento

de las *APIs* se encuentra en backend de todo sistema, por lo tanto, solo el desarrollador sabe de su existencia.

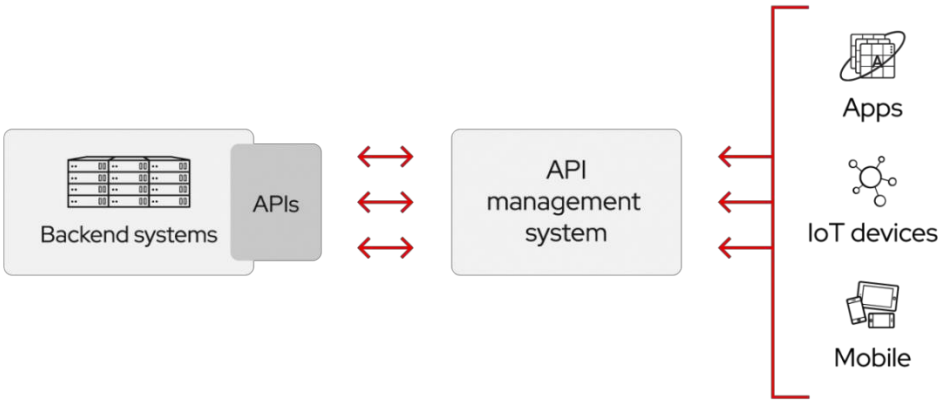

Figura 1. Conexión de infraestructura a través del desarrollo de diferentes aplicaciones utilizando *API* (RedHat, 2020).

Se puede entender por *API* como un código que indica a las aplicaciones cómo pueden mantener una comunicación entre sí. Estas reglas permiten que los distintos programas mantengan interacciones. Una *API* sirve para establecer una comunicación con una base de datos, un sistema operativo o un protocolo de comunicaciones. Una de las claves del funcionamiento es la facilidad de integración. Estas herramientas tienen que resultar simples de integrar a otro software para que las comunicaciones puedan desarrollarse con éxito (Definición DE, 2021).

Se podrían mencionar más definiciones, sin embargo, de manera particular y para fines de este trabajo son las definiciones más acertadas debido a que incluyen las palabras clave: comunicación entre sistemas e integración de sistemas, que es el tema central de este trabajo.

1.4.1- Tipos de *APIs*.

***APIs* de servicios web:** son las interfaces de desarrollo de aplicaciones que permiten el intercambio de información entre un servicio web (*software* que da acceso a un servicio concreto a través de una *URL*) y una aplicación. Normalmente ese intercambio se produce a través de peticiones *HTTP*. Tanto en la petición como en la respuesta del servicio web, el intercambio de información se realiza mediante los formatos *XML* o *JSON (*BBVA*,* 2016*).*

Existen cuatro tipos de *API* de servicios web que son *SOAP, XML-RP, JSON-RPC* y *REST. **SOAP*** es un protocolo estándar de intercambio de información y datos en *XML* entre dos objetos. ***XML-RPC*** es un protocolo de llamada a procedimiento remoto que usa *XML* como formato de datos y llamadas *HTTP* como sistema de comunicación. ***JSON-RPC***, es un protocolo de llamada a procedimiento remoto, pero en formato *JSON*. ***REST***, arquitectura de software para sistemas hipermedia en la *World Wide* Web; una *API REST* usa el protocolo *HTTP* (BBVA, 2016).

REST es la *API* más relevante dentro de este trabajo debido a que son las que se utilizarán para el desarrollo, *API REST*.

***APIs* basadas en bibliotecas:** este tipo de *APIs* son las que permiten que una aplicación importe una biblioteca de otro *software* para hacer el intercambio de información. Gran parte de las bibliotecas que dan acceso a productos y servicios están diseñadas en *JavaScript*. Un ejemplo de estas *API* son las que se utilizan dentro del mercado de la cartografía web (*Google Maps*, *Leaflet*, *ArcGIS*, *CartoDB*, *MapBox* o *D3.js*) (BBVA , 2016).

***APIs* basadas en clases:** este tipo de interfaces de desarrollo de aplicaciones permite la conexión con los datos en torno a las clases, son muy comunes en la programación orientada a objetos con *Java*. La *API* de *Java* usa clases abstractas para la creación de aplicaciones igual que cualquier programa desarrollado en este lenguaje. La interfaz de desarrollo de *Java* se organiza en paquetes y cada uno de estos paquetes contiene a su vez un conjunto de clases relacionadas entre sí (BBVA, 2016).

***APIs* de funciones en sistemas operativos:** los programas de *software* están continuamente interactuando con los sistemas operativos. En muchos casos, la forma en la que lo hacen es a través de *APIs*. Sistemas operativos como *Windows* disponen de *APIs* que permiten esa comunicación entre programas y el sistema operativo, como las *APIs*: interfaz de usuario, acceso y almacenamiento de datos, mensajería, gráficos y multimedia o diagnóstico de errores (BBVA, 2016).

Por otra parte, existen tres enfoques respecto a las políticas de las versiones de las *APIs*. Las *APIs* pueden ser privadas, es decir, únicamente para uso interno de la organización; compartidas, lo que significa, que las desarrollan un conjunto de socios empresariales específicos y las comparten entre sí, y públicas, todos tienen acceso a ellas y las pueden utilizar de acuerdo a sus necesidades.

De acuerdo a la clasificación de las *APIs* y las versiones que existen, para este trabajo se utilizarán las *APIs* de servicios web desarrolladas por socios.

1.5.- *Open APIs*.

Al momento de desarrollar o hacer uso de una *API* se necesitan estándares que indiquen cómo crearla o utilizarla, para ello existen las especificaciones *Open API*.

La especificación *Open API* en un inicio fue nombrada especificación *swagger* creada por Tomy Tam en 2010. Posteriormente, en 2015, fue adquirida por SmartBear, una empresa dedicada al desarrollo de software, al poco tiempo la nombró Iniciativa *Open API* en colaboración con la Fundación Linux. Otras empresas como 3Scale, APIgee, CAPItal One, Google, IBM, Intuit, Microsoft,

PayPal y Restlet se sumaron a este proyecto de código abierto y de ahí surgió con el nombre Especificación *Open API* (*OAS*).

Una especificación *Open API* define los estándares para que se pueda lograr una comunicación a nivel máquina, sin que intervengan lenguajes de programación específicos, y para que los desarrolladores pueden entender como enviar y recibir la información. En ocasiones la especificación suele llamarse contrato de *API* o *swagger*.

La especificación *Open API* debe estar en formato *JSON* (*JavaScript Object Notation*) con la extensión de archivo .json o en formato *YAML* (*Yet Another Markup Language*) con la extensión de archivo .yaml o .yml. El objetivo de este documento es describir las capacidades u operaciones que se pueden realizar con la *API*.

Se podría confundir especificación *Open API* con *swagger,* debido a que la especificación *Open API* nació con ese nombre. Es importante tener claro la diferencia de estos términos:

- *Open API* es una especificación.
- *Swagger* es una herramienta que usa la visualización de la especificación *Open API*. Por ejemplo*, OpenAPIGenerator* y *SwaggerU*I.

Las *Open APIs* están basadas en la arquitectura *REST* debido a esto también son conocidas como *APIs REST* o *APIs Web,* por lo tanto, se deben ajustar a la estructura *REST* que se muestra en la Tabla 1.

Tabla 1. Elementos de datos REST (Chakray,2021).

Elemento	Descripción
Recurso	Objetivo conceptual de una referencia hipertextual. Ej: Cliente/orden.
Identificador de recurso	Un localizador de recurso uniforme (*Uniform resource locator, URL*) o nombre de recurso uniforme (*Uniform resource name, URN*) identificando un recurso específico. Ej: http://my.rest.com/cliente/3435 / http://my.rest.com/order503
Metadato de recurso	Información describiendo el recurso. Ej. Etiqueta, autor, enlace de fuente, localización alterna, apodo.
Representación	El contenido del recurso puede ser un mensaje *JSON*, documento *HTML*, imagen JPEG.
Metadato de representación	Información describiendo cómo procesar la representación. Ej. Tipo de medio, hora de la última modificación
Dato de control	Información describiendo cómo optimizar el procesamiento de respuesta. Ej. If-modified-since, cache-control

Un recurso es una entidad de información lógica y cualquier entidad web que puede ser referenciada, por ejemplo, una *API*, un servicio, una aplicación, una localización, una acción, un vídeo o una imagen; la cual se identifica por una *URI*.

Un recurso es un identificador y es manipulado a través de una representación. Una representación puede ser un mensaje *JSON* describiendo a un cliente, un archivo de vídeo o un archivo de configuración *XML*. Por ejemplo, una representación puede ser devuelta en *JSON*, *XML*, o *CSV* basado en las capacidades de procesamiento del cliente. Las capacidades de procesamiento del cliente son comunicadas a través del encabezado Accept (Chakray, 2021).

1.5.1- Interpretación de una Especificación *Open API*.

En esta sección se describe cómo analizar una *API* mediante el formato *swagger* utilizando las especificaciones *Open API*. Es importante mencionar que una entidad o recurso de un *API* no debe representar una tabla en el modelo entidad-relación que se esté usando en conjunto. En los contratos de *API* no se debe proporcionar información sensible, como, por ejemplo, passwords. Una *API* representa recursos o entidades y operaciones sobre éstas. Las operaciones de una *API* no deben tener estado, es decir, las llamadas se deben realizar de forma independiente incluyendo los datos esenciales para que éstas se realicen de manera satisfactoria y se deben utilizar encabezados para validar la información.

Las secciones o apartados que se deben de considerar al momento de interpretar o documentar una *API* en formato *swagger* son información básica, operaciones, parámetros, códigos de respuesta y recursos.

Información básica: en este apartado se define la versión del lenguaje, información general de la interfaz, tipo de licencia, nombre del host, protocolos de transporte válidos y seguridad. Esta sección se considera meramente informativa y no es crítica al momento de hacer uso de una *API*. El Listado 1 se muestra un ejemplo de cómo se encuentra documentada esta sección, en el cual muestra la versión de la *API*, el título, la descripción, el recurso, el protocolo y los formatos soportados.

```
1   swagger: '2.0'
2   info:
3     version: 1.3.2
4     title: API Party
5     description: |-
6       ## TMF API Reference : TMF 632 - Party
7
8       ### Release : 19.0
9
10     The party API provides standardized mechanism for party management.
11    contact: {}
12   host: serverRoot
13   basePath: /tmf-api/party/v1/
14   schemes:
```

```
15      - https
16    consumes:
17      - application/json
18    produces:
19      - application/json
```

Listado 1. Información básica de un *Open API*.

Operaciones: las operaciones de la *API* representan cómo va a interactuar el cliente con las diferentes entidades que se comunican o comparten información. Pueden usarse operaciones seguras, las cuales no realizan acciones sobre el servidor, únicamente se usan para obtener información de los recursos (*GET*). Pueden ser operaciones de modificación, las cuales realizan acciones sobre el servidor (*PUT, POST, PATCH, DELETE*). La Figura 2 representa las operaciones cinco operaciones HTTP utilizados en una *API REST*.

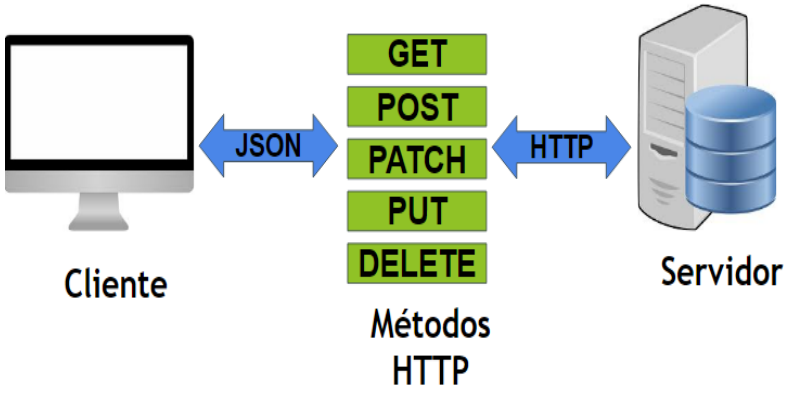

Figura 2. Operaciones básicas de un *Open API*.

Parámetros: las operaciones que se describen en un contrato pueden necesitar información adicional para poder acceder a la información del servidor, para ello, se usan parámetros, los cuales pueden ser de navegación o filtrado. En esta sección se puede documentar tipo de objeto, tamaño máximo, obligatoriedad, formato, lista de valores válidos, esquema de objeto y formato de mensaje. El Listado 2 muestra cómo se pueden configurar los parámetros, por ejemplo, Authorization, correlatorId, messageId y requestDate.

```
20   paths:
21     /individual:
22       get:
23         description: This operation list or find Individual entities
24         summary: listIndividual
25        tags:
26           - individual
27         operationId: listIndividual
28         deprecated: false
29         produces:
30           - application/json
31         parameters:
32           - name: Authorization
```

```
33            in: header
34              required: false
35              type: string
36              description: >-
37                A secure token chain that indicates that it is an authorized
38                request.
39            - name: correlatorId
40              in: header
41              required: false
42              type: string
43              description: >-
44                Identifier used to match the request with its corresponding
45                asynchronous notification
46            - name: messageId
47              in: header
48              required: false
49              type: string
50              description: Transaction identity for each request
51            - name: requestDate
52              in: header
53              required: false
54              type: string
55              format: date-time
56              description: Time and date the request is sent
```

Listado 2. Configuración de parámetros Authorization, correlatorId, messageId y requestDate en un *Open API*.

Códigos de respuesta: en un *API* los códigos de respuesta deben estar basados en el protocolo HTTP permitiendo identificar la respuesta de cada operación o petición que se realiza hacia el servidor. Los códigos de respuesta pueden variar de acuerdo a cada operación. El Listado 3 representa la manera de implementar los códigos dentro de la *API*.

```
354       responses:
355         '201':
356           description: Created
357           schema:
358             $ref: '#/definitions/Individual'
359           headers:
360             country:
361               type: string
362             originSystem:
363               type: string
364             messageId:
365               type: string
366             correlatorId:
367               type: string
368             responseDate:
369               type: string
370             targetSystem:
371               type: string
372         '400':
373           description: Bad Request
374           headers:
375             country:
376               type: string
377             originSystem:
378               type: string
379             messageId:
```

```
380              type: string
381            correlatorId:
382              type: string
383            responseDate:
384              type: string
385            targetSystem:
386              type: string
387          schema:
388            $ref: '#/definitions/GenericFault'
```

Listado 3. Configuración de códigos de respuesta 201 y 400 en un *Open API*.

La Figura 3 muestra de manera general los códigos de respuesta que se pueden encontrar dentro del contrato de un *Open API*.

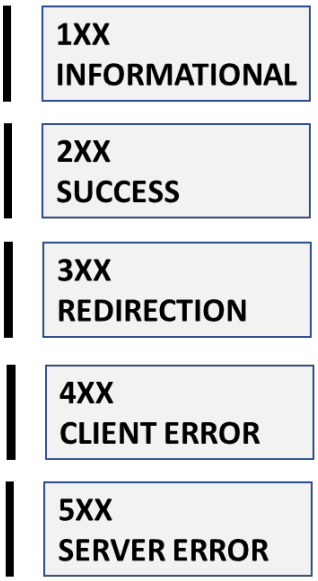

Figura 3. Códigos generales *HTTP*.

Recursos: los recursos son la información a la que se requiere acceder para consultar o modificar, independientemente de su formato. Dentro de la *API* se definen estos recursos y las operaciones que se pueden realizar sobre ellos. La Figura 4 muestra la configuración de los recursos en una *API, en este caso, la API* tiene dos recursos *individual y organization*. Para cada recurso se debe definir una URL, que permite identificar de forma única al recurso, localizarlo, poder acceder a él y compartir su ubicación. La *URL* está compuesta por el dominio o dirección *IP* del servidor más el *path* del recurso definido en el contrato.

Por ejemplo:
https://APIservicedesa.clarochile.cl:7006/PartyManagement/tmf-API/party/v1/individual

Donde:

***IP* del servidor**: https://APIservicedesa.clarochile.cl:7006
Path: /PartyManagement/tmf-API/party/v1/individual.

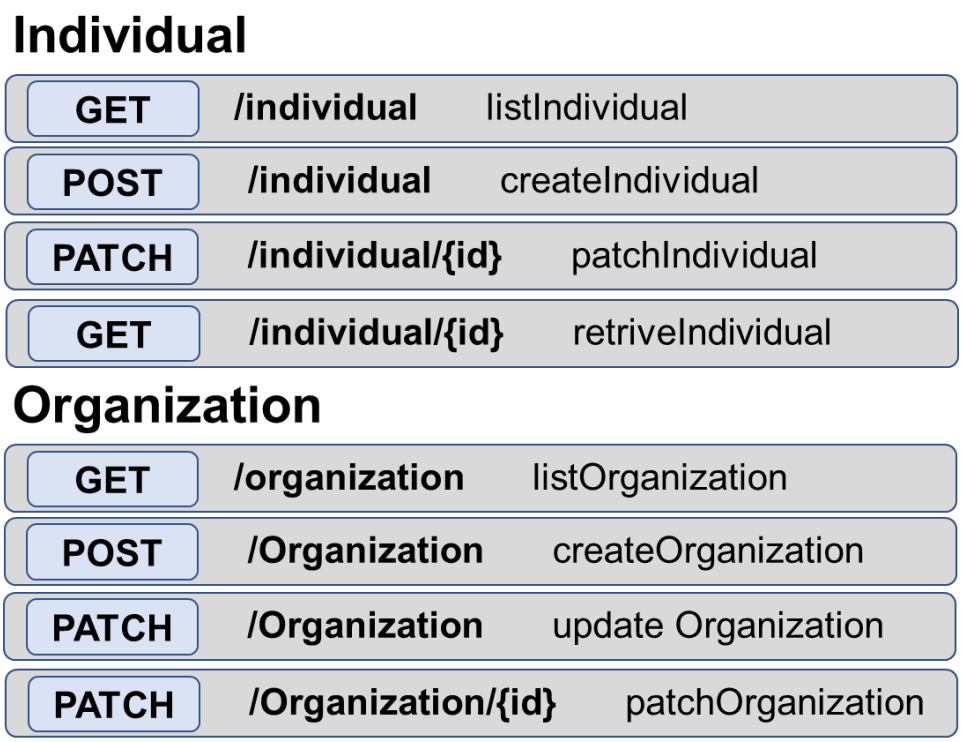

Figura 4. Definición de recursos y operaciones para cada recurso.

Dentro del contrato de la *API* se especifican los atributos que tiene cada recurso, así como el tipo de datos de cada uno. Los atributos pueden ser obligatorios o no. Si un campo es obligatorio viene marcado con *. En el Listado 4 se muestra un ejemplo de los tipos de datos que se pueden utilizar de acuerdo a la especificación de *Open API*.

```
Individual{
            description:
                        Individual represents a single human being (a
                        man, woman or child). The individual can be
                        a customer, an employee or any other person
                        that the organization needs to store
                        information about.

            id*                 string
                                Unique identifier of the individual

            href                string
                                Hyperlink to access the individual

            familyName          string
                                Contains the non-chosen or inherited
                                name. Also known as last name in the
                                Western context

            givenName           string
                                First name of the individual
```

```
                secondLastName    string
                                  Second Last Name
}
```

Listado 4. Tipos de datos *Open API*.

La Tabla 2 muestra los tipos de datos a utilizar en las especificaciones *Open API*.

Tabla 2. Tipos de datos *Open API*.

Tipo de dato	Formato
Integer	int32
Integer	int64
Number	Float
Number	Double
String	
String	Byte
String	Binary
Boolean	
String	Date
String	date-time
String	Password

En cada petición que se realiza hacia el servidor, este envía un código de respuesta, pero además envía información en estructura *json*, la cual puede indicar un error o estar correcta de acuerdo a la petición. El Listado 5 muestra la estructura *json*, de acuerdo a la especificación *Open API*, la cual se obtiene cuando se logra hacer una comunicación exitosa y se regresa el *status code* o código de respuesta 200.

```
[
  {
    "id": "string",
    "href": "string",
    "accountType": "string",
    "description": "string",
    "lastModified": "2021-11-19T20:06:26.711Z",
    "name": "string",
    "state": "string",
    "accountBalance": [
     {
       "balanceType": "string",
        "@baseType": "string",
        "@schemaLocation": "string",
        "@type": "string",
        "amount": [
          {
```

```
            "unit": "string",
            "value": 0,
            "description": "string"
          }
        ],
        "type": "string",
        "validFor": {
          "endDateTime": "2021-11-19T20:06:26.711Z",
          "startDateTime": "2021-11-19T20:06:26.711Z"
        },
        "characteristic": [
          {
            "id": "string",
            "name": "string",
            "valueType": "string",
            "value": "string"
          }
        ]
      }
    ]
]
```


Listado 5. Estructura *json* correcta *Open API* con *status code* 200.

El Listado 6 muestra la estructura *json*, de acuerdo a la especificación *Open API*, la cual se obtiene cuando se logra hacer tener comunicación, pero por algún motivo existe algún error. La estructura contiene el mensaje del error y un *status code* o código de respuesta 400.


```
{
    "messageId": "string",
    "responseDate": "2021-11-19T20:06:26.733Z",
    "latency": 0,
    "errorList": {
      "error": [
        {
          "code": "string",
          "severityLevel": "string",
          "description": "string",
          "actor": "string",
          "businessMeaning": "string"
        }
      ]
 }
```


Listado 6. Estructura *json* de mensaje de error con status code 400.

Dentro de la estructura *json* todos los nombres de campo de la especificación distinguen entre mayúsculas y minúsculas.

Un contrato puede comenzar y terminar con { } o [ ]. Puede contener varias claves (*Key*)/valores(*values*) dentro separados por una coma. Si empieza con [ ], significa que representa un arreglo y si empieza con { } significa que representa un solo objeto. Así mismo, cada clave es seguida de dos puntos para separarla del

valor. A continuación, se muestra un ejemplo que contiene dos claves (ciudad, país) y dos valores (Monterrey, México).

El objeto se representa como se muestra a continuación:

```
{
        "ciudad”: “Monterrey",
        "pais”: “México"
}
```

El arreglo se representa de la siguiente manera:

```
[
    {
        "ciudad":"Monterrey",
        "pais":"México"
    },
    {
        "ciudad”: “Guadalajara",
        "país”: “México"
    }
]
```

Capítulo II. *Salesforce* y *TM Forum Open APIs*.

En este capítulo se describen el *CRM Salesforce* y las *Open APIs* de *TM Forum* las cuales son utilizadas para la comunicación de los sistemas.

2.1.- *Salesforce*.

Salesforce es una plataforma *CRM* basada en la nube, es decir, no se necesita tener ningún software instalado internamente para hacer uso de éste. Las compañías pueden acceder a sus datos a través de la web.

Con la implementación de *Salesforce* dentro de una empresa se pretende dinamizar el equipo de ventas, crear u optimizar un servicio de atención a clientes, crear campañas de marketing y analítica, de tal manera, que se tenga una recopilación de información que permita tomar acciones o decisiones para que la compañía mejore sus procesos y crezca de forma ascendente.

La empresa fue fundada por Marc Benioff en 1999, un ex-ejecutivo de Oracle, en un tiempo en el que tener un CRM en la nube estaba lejos de convertirse en lo que es hoy en día.

Salesforce como CRM se basa en la filosofía Customer 360 o vista 360°, la cual puede considerarse una recopilación de información de las interacciones que tiene una organización con sus clientes. Es una visión extremo a extremo de la interacción del cliente a través de varios canales y puntos de contacto que permite conocer cuáles son sus intereses de compra. Por ello, es importante conectar o incluir varias herramientas dentro de un CRM que permitan conocer de mejor manera las preferencias de cada cliente. La Figura 5 muestra los servicios con los que cuenta Salesforce, como son: sales(ventas), service(atención al cliente), marketing(campañas de mercadotecnia), commerce(comercio digital), engagement(compromiso con usuarios), platform(plataforma), integration(integración), analytics(analitica de datos), industries(industrias), communities(comunidades), enablement (capacitación) y collaboration(colaboración).

Como se puede observar en la Figura 5 *Salesforce* es una plataforma que ofrece varias posibilidades para mejorar los procesos de cada organización, de acuerdo los servicios y productos que ésta ofrezca. Entre más aplicaciones o módulos se implementen mejores serán los beneficios o cosas que se puedan realizar con este *CRM.*

Figura 5. Vista *Customer* 360° *Salesforce* (Salesforce, 2021).

Algunas razones para utilizar Salesforce:

1.-Solución Cloud: servicios basados en la nube que se actualizan automáticamente.

2.-Confiable y personalizable: se adapta a las necesidades o manera de trabajar de cada organización.

3.- Soluciones Móviles: los datos son accesibles desde cualquier dispositivo móvil sin desarrollos adicionales.

4.- Robusto y seguro: al ser una herramienta utilizada por muchas empresas, esto hace que sea robusta y esté en constante actualización. Cuenta con millones de usuarios, por lo tanto, es difícil que deje de actualizarse y dar soporte.

5.- Liderazgo: es uno de los líderes en el mercado, es un *CRM* personalizable y adaptable a cualquier tipo de organización, sea grande, mediana o pequeña. En el momento de decidir qué *CRM* contratar, es muy importante tener presente el número de usuarios que se tienen.

6.-Innovador: es una de las herramientas que utiliza metadatos. Con esto se garantizan que las personalizaciones y las actualizaciones se realicen a tiempo y sean únicas para cada organización. Por otro lado, se basa en la filosofía de mínimos desarrollos, lo que también le hace innovador frente a sus competidores.

Salesforce está integrado por varios módulos o nubes interconectados de tal manera que permiten interactuar a los diferentes departamentos de la empresa para que puedan compartir información, así agilizar sus ventas y tener una mejor captación de clientes.

1.-*Sales Cloud*: permite vender más rápido y desde cualquier lugar. Crea previsiones de venta y detecta oportunidades de negocio.

2.-*Service Cloud*: permite brindar una mejor atención al cliente, supervisando su actividad desde diferentes canales.

3.-*Marketing Cloud*: Plataforma con distintas herramientas de difusión, venta de productos y desarrollo de campañas multicanal para los clientes.

4.-*Community Cloud*: plataforma que conecta a empleados, socios, clientes, proveedores y distribuidores.

5.-*Apps Cloud*: permite crear, diseñar y desarrollar aplicaciones personalizadas en un mismo sistema.

6.-*Analytics Cloud*: es la nube de Salesforce que permite recabar, analizar y distribuir los datos del negocio.

7.-*IoT Cloud*: conecta el internet de las cosas con el internet del cliente al asociar los dispositivos de los mismos con los procesos corporativos.

8.-*Commerce Cloud*: permite contactar con compradores a través de diferentes canales digitales y administrar pedidos.

El fuerte posicionamiento de *Salesforce* en el mercado frente a sus principales competidores se debe a que nació en la nube, mientras que sus principales rivales como Oracle, Microsoft y *SAP* tuvieron que adaptarse a esta tendencia porque sólo ofrecían servicios como *CRM* locales. La Figura 6 muestra como *Salesforce* se ha posicionado en el mercado dejando muy atrás a sus principales competidores.

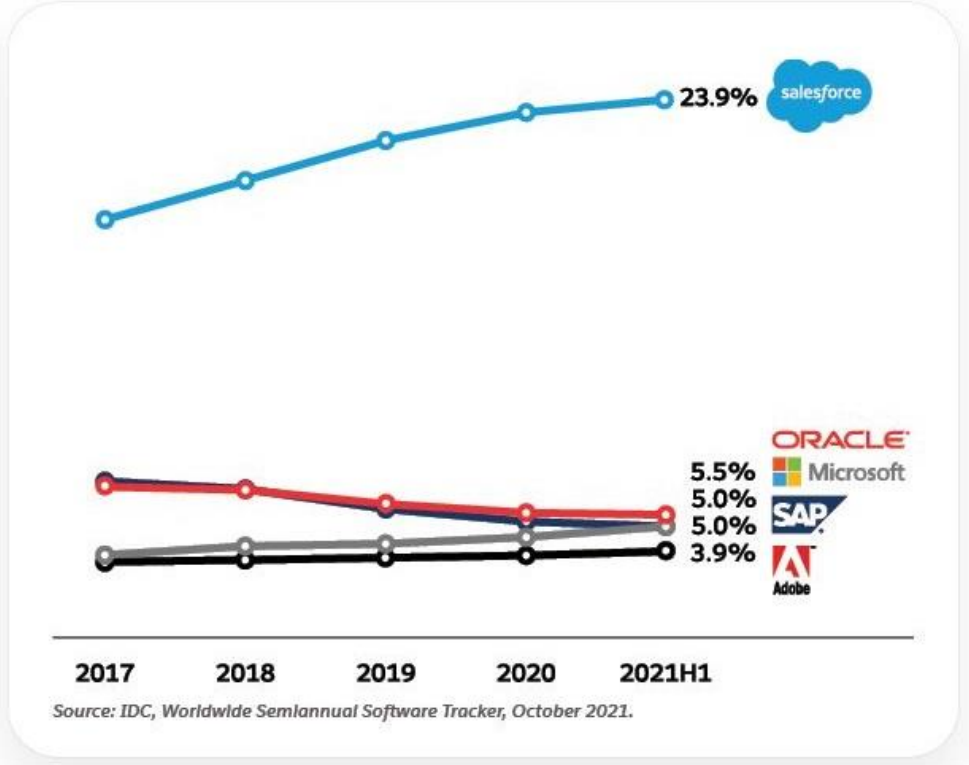

Figura 6. Posicionamiento de *Salesforce* en el mercado (IDC, 2022).

Por otra parte, las adquisiciones que ha hecho *Salesforce* son parte fundamental de la estrategia de su crecimiento, entre 2016 y 2019 ha adquirido alrededor de 60 empresas distintas, modificando sus soluciones para integrarlas y poder ofrecer más servicios. Se trata de empresas que cuentan con alguna estrategia innovadora que *Salesforce* no había explorado y le interesaba hacer uso de ella. Por ejemplo, Mulesoft, Integration Cloud, Datorama, Tableau y Click Software. Su más reciente adquisición es la empresa Slack, comprada en el año 2020.

Salesforce está diseñado o pensado tanto para pequeñas como grandes empresas. Si se trata de una pequeña empresa puede hacer uso de sus módulos básicos *Sales Cloud* o *Service Cloud*. Para grandes empresas se pueden elegir varios módulos dependiendo de las necesidades y objetivos de éstas.

Para empezar a hacer uso de *Salesforce* se puede utilizar la versión gratuita que es accesible para todos y tiene una vigencia de 30 días. Al término de este periodo de prueba ya se tendrá una visión m ás clara del funcionamiento del *CRM* y de las licencias que se deben adquirir de acuerdo a las necesidades de la empresa. Existen cuatro tipos de licencias, *Salesforce Essential*, *Salesforce Professional*, *Salesforce Entreprise* y *Salesforce Unlimited*.

La edición de *SalesForce* más básica y barata es *Essential*, cuesta $25 dólares por usuario y por mes, está limitada a 10 usuarios. Esta sería la mejor opción para un inicio pequeño donde solo unos pocos usuarios accederán al *CRM*.

La edición *Professional* cuesta $75 dólares por usuario y por mes, permite dar seguimiento a prospectos de ventas, oportunidades y casos de clientes. Se pueden administrar campañas de marketing, contratos y pedidos. Se puede obtener un mejor entendimiento del negocio en tiempo real mediante pronósticos de ventas acertadas, reportes y tableros personalizables.

La edición *Enterprise* cuesta $150 dólares por usuario y por mes, permite automatizar procesos empresariales mediante flujos de trabajo. Se pueden utilizar tipos de registros personalizados y realizar integraciones con cualquier sistema mediante las *APIs* de servicios web. Así mismo, permite gestionar territorios de venta complejos y ver cómo los acuerdos de venta evolucionan.

La edición *Unlimited* cuesta $300 dólares por usuario y por mes, da acceso a capacitación en línea ilimitada, más de 100 servicios de administración, apoyo telefónico 24 horas al día los 7 días a la semana. Se pueden crear un número ilimitado de aplicaciones personalizadas, pestañas y objetos. Los administradores tendrán acceso a múltiples *sandboxs*(servidores) para el desarrollo y la evaluación.

2.1.1. *Salesforce* basado en la nube.

Cuando *Salesforce* fue creado se convirtió en el primer software que ofrecía servicios en un sitio web derivado de esto se le llamó computación en la nube o *cloud computing*.

De una manera simple, la computación en la nube es una tecnología que permite acceso remoto a softwares, almacenamiento de archivos y procesamiento de datos por medio de Internet, siendo así, una alternativa a la ejecución en una computadora personal o servidor local. En el modelo de nube, no hay necesidad de instalar aplicaciones localmente en computadoras.

La computación en la nube (*Cloud Computing*), no es más que un medio de comunicación donde las empresas, organizaciones y negocios en general, ven una solución de a sus problemas de tecnología, tanto de infraestructura tecnológica, como la prestación del servicio para lograr ser económicamente rentable (Orozco, 2016).

A finales de junio 2010, la consultora Gartner con servicios a nivel mundial, publicó un informe donde ratificaba el crecimiento vertiginoso de la computación en la nube. Desde el punto de vista como proveedores de servicios informáticos de hardware y software, la mayoría de las grandes empresas como: IBM, Microsoft, Oracle, Hewlett-Packard, Cisco, entre otros, han gestionado estrategias con la finalidad de proveer estos servicios. En este sentido, es por ello que las operadoras

de telecomunicaciones europeas (Telefónica, Vodafone, Telcel), y las americanas (AT&T, Verizon), se asocian a las empresas de Internet, que en su conjunto forman parte de la nube (Cloud) como lo son Amazon, Google, Yahoo u otras redes sociales como Facebook o Twitter. Las empresas que actualmente se mantienen en constante innovación, deben tomar ventaja de estos recursos y realizar nuevas propuestas dirigidas a su mercado, ya que, aquellas que ignoren estas ventajas se arriesgan a quedar desactualizadas y tal vez, fuera del negocio (Orozco, 2016).

La idea de Benioff era crear aplicaciones empresariales sobre un modelo de negocio conocido como *Software as a Service* (*SaaS*), cuyo objetivo era eliminar las grandes inversiones en licencias, hardware, implementaciones y actualizaciones del modelo tradicional que existía en esos momentos. En el año 2000 salió el primer prototipo del *CRM* (*Salesforce*.com), aunque no era nada comparado con lo que es hoy en día, y como aún muy era básico Benioff utilizó el slogan "*No Software*" para poder promocionarlo, lo que lo llevó a posicionarse fuertemente en el mercado ante sus competidores, tanto así, que hoy en día *Salesforce* cuenta con servicios *SaaS* y *PaaS*, siendo la plataforma *CRM* en la nube más utilizada. De acuerdo con la Figura 6 Salesforce posee mercado, con un 23.9 %, muy por delante de sus principales competidores, SAP y Oracle.

Salesforce cuenta con 4 partes fundamentales *Sales Cloud, Service Cloud, Community Cloud*, integrado estas la parte SaaS, y *Lightning Platform* (anteriormente conocida como Force.com) que constituye el componente *PaaS*.

2.1.2. *Sales Cloud* de *Salesforce*.

Sales Cloud es un *CRM* de ventas cuyo principal objetivo es ayudar a las organizaciones a vender en menos tiempo y de una manera inteligente, creando tareas simples de forma automatizada que permiten al equipo comercial enfocarse en las etapas más importantes del flujo de la venta.

.

Las principales funciones de *Sales Cloud* son:

- **Cerrar más acuerdos.**

Se cuenta con la gestión de contactos que permite tener una visión completa de sus clientes, incluyendo el historial de actividades, los contactos importantes y cómo comunicarse con sus clientes. Se puede obtener información de sitios de redes sociales populares como Facebook, Twitter, LinkedIn y YouTube para tener un mejor conocimiento del cliente.

Se puede realizar la gestión de oportunidades de ventas para conocer la etapa, productos, competencia y cotizaciones de la misma, lo que permite cerrar más rápido un acuerdo.

- **Obtener más clientes potenciales.**

Se pueden crear automatizaciones para implantar y gestionar campañas de marketing eficaces, uniendo el departamento de ventas con el de marketing para atraer, clasificar candidatos y reducir los ciclos de ventas.

- **Acelerar la productividad.**

Gracias a la funcionalidad *Lightning Voice* se puede llamar a los clientes con un simple clic y tomar notas viendo toda la información del contacto y recibiendo llamadas entrantes dentro de Salesforce.

La aplicación *Mobile* de *Salesforce* permite registrar las llamadas y responder a clientes potenciales, teniendo todos los tableros para supervisarlos.

Sales Cloud permite crear *workflows* para automatizar los procesos comerciales fácilmente.

Tiene integración con aplicaciones de correo electrónico para no cambiar la forma de trabajar del equipo comercial. Además, permite sincronizar archivos mediante un uso compartido para analizar y realizar cambios en tiempo real.

- **Toma de decisión sabías.**

Mediante *dashboards* que ofrecen un panorama general y en tiempo real con reportes detallados y fáciles de crear, se obtiene una visión de los pronósticos de ventas de su equipo comercial ayudándolos a tomar mejores decisiones al momento de realizar la venta.

2.1.3. Integración de *Salesforce* con sistemas externos usando *Apex* y *REST*.

Salesforce cuenta con su propio lenguaje de programación llamado *Apex*. Es un lenguaje de programación tipado, orientado a objetos, que se encuentra disponible en la plataforma *Salesforce*. Tiene una sintaxis muy parecida a *Java*, pero la forma en la cual es utilizado y sus estructuras difieren de los lenguajes tradicionales. Permite a los desarrolladores agregar lógica a la mayoría de los eventos del sistema, incluidos los clics en los botones, las actualizaciones de registros y las páginas de Visualforce. El código *Apex* puede iniciarse mediante solicitudes de servicios web y desde desencadenadores en objetos.

Para modificar una instancia u objeto de Salesforce, se debe utilizar Apex. El lenguaje se caracteriza por hacer uso de *DML*, *SOQL*, *SOSL* y estructuras *batch*.

- *Data Manipulation Language* (*DML*), permite la creación/modificación/borrado de objetos.

- *Salesforce Object Query Language* (*SOQL*) *y Salesforce Object Search Language* (*SOSL*), permiten escribir consultas embebidas en el código para retornar objetos.

- Cuenta con estructuras que brindan la posibilidad de procesar un conjunto de objetos en forma de *batch*. Esto ayuda a que no se bloquee la instancia (recordar que está en la nube, lo que significa que comparte recursos con otras instancias).

REST, por sus siglas ***RE****presentational* ***S****tate* ***T****ransfer,* significa Transferencia de Estado Representacional, es una arquitectura para conectar varios sistemas basados en el protocolo *HTTP*, permitiendo generar y obtener datos a través de operaciones los cuales se representan en formato *XML* o *JSON*.

Salesforce puede comunicarse con distintas aplicaciones o sistemas de software utilizando código Apex.

Los servicios web son muy utilizados, se pueden usar al comprar online, leer el periódico, reservar una mesa en un restaurante o ver alguna película, estas son interacciones que se producen a diario entre el usuario y la máquina. Además, y aunque pueda pasar desapercibido, estas interacciones también tienen lugar entre máquinas: el cliente y el servidor se están enviando continuamente solicitudes y respuestas, transmitiendo información de uno a otro gracias a los servicios web, de lo cual el usuario no se da cuenta.

A manera de resumen, un servicio web es un método de comunicación que permite la interoperabilidad de máquina a máquina a través de una red. Éste permite que una computadora o programa pueda solicitar y recibir información de otra computadora o programa. A quien solicita la información se le llama cliente y a quien envía la información se le llama servidor. Las aplicaciones escritas en varios lenguajes de programación que funcionan en plataformas diferentes pueden utilizar los servicios web para intercambiar información a través de una red.

La Figura 7 muestra cómo interactúa *Salesforce* con otras infraestructuras de software logrando una comunicación mediante el uso de *APIs* y servicios web, así mismo, se puede observar que para poder comunicarse con estas infraestructuras se hace uso de la especificación *Open API* que se abordó en el Capítulo I.

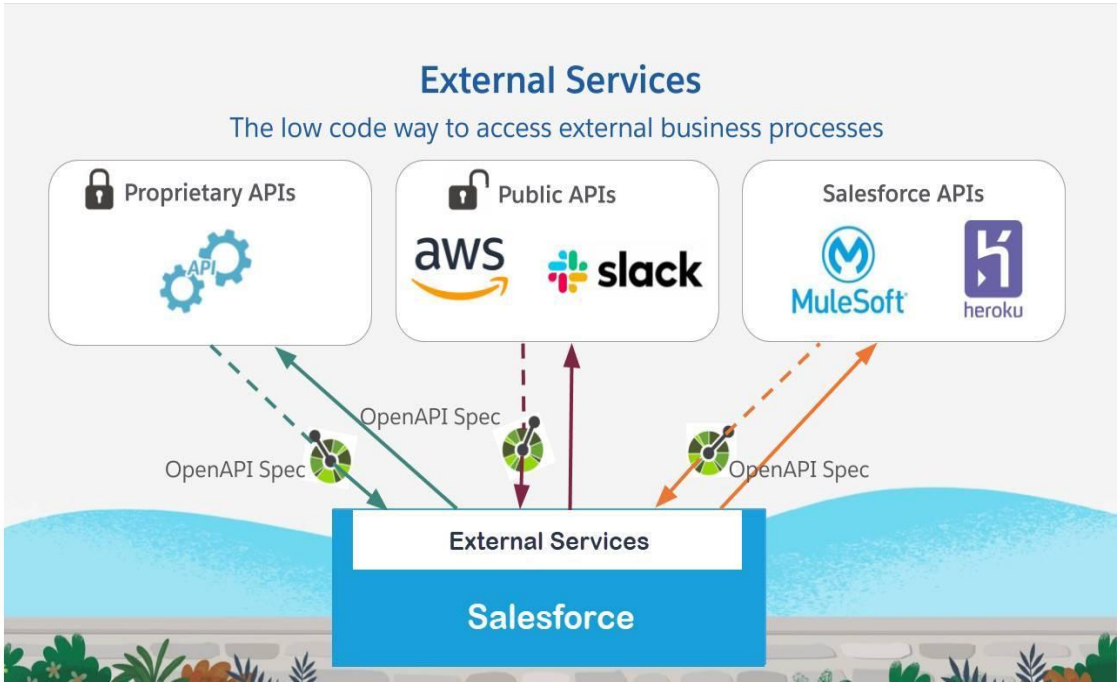

Figura 7. Interacción de Salesforce con servicios externos mediante *APIs* *(Salesforce, 2021).*

Cuando un sistema externo se comunica con *Salesforce*, se dice que este último expone sus servicios hacia el sistema externo y cuando *Salesforce* se comunica con el sistema externo se dice que éste consume los servicios del sistema externo.

Los servicios web *REST* y *API REST* que usa *Salesforce* para lograr una comunicación con el sistema externo se basan en recursos. Un recurso es una entidad, la cual se almacena principalmente en un servidor y el cliente solicita realizar alguna operación sobre ésta. Se crea una petición *HTTP* en el servidor conocida como *Request* con la información necesaria de la operación que se desea realizar, se envía al cliente el cual devuelve una respuesta llamada *Response* con la información necesaria de la operación. Se apoya en los métodos básicos de *HTTP*: *POST, GET, PUT, PATCH* y *DELETE*. Todos los objetos o recursos se manipulan mediante una *URI* o *URL*.

Las solicitudes *HTTP* constan de los elementos URI, métodos HTTP, headers, cuerpo de la solicitud, cuerpo de la respuesta y códigos de respuesta.

- ***URI.***

Uniform Resource Locator o Identificador Uniforme de Recursos, es la ruta de un recurso en *Salesforce* o en un sistema externo y debe ser un identificador único para cada objeto o *API*. Esta ruta cambia dependiendo del recurso, pero tiene una estructura básica que siempre es la misma.

Si se desea que *Salesforce* se comunique con el sistema externo para obtener información o para realizar alguna otra operación, la *uri* deber de tener la siguiente estructura:

https://negocio-dev.cla.pe/API/V1//party/individual ?

Cuando el sistema externo se comunica con *Salesforce* para que éste le devuelva información o que se realice alguna otra acción dentro del mismo, la *uri* debe tener la estructura:

https://transforma--transdev.my.salesforce.com/services/apexrest/party/individual/

Lo importante a destacar en estas estructuras es que al menos deben de tener el dominio al que se desea acceder y el nombre del *API* que se está utilizando. Adicional a esto, se puede complementar la *uri* con la versión del *API* o parámetros que se usen en algún método en específico.

- **Métodos *HTTP*.**

Los métodos estándar que admite el protocolo *HTTP* son: *HEAD, GET, POST, PATCH, PUT* y *DELETE.* Para fines de este trabajo se utilizan los métodos *POST, GET* y *PACHT* que son los que se utilizarán para realizar la comunicación con otros sistemas. La descripción de cada uno de los métodos se encuentra en la Tabla 3.

Tabla 3. Descripción de los métodos HTTP.

Método	Descripción
HEAD	El método *HEAD* pide una respuesta idéntica a la de una petición *GET*, pero sin el cuerpo de la respuesta.
POST	El método *POST* se utiliza para enviar una entidad a un recurso en específico, se pueden crear datos en el servidor.
GET	El método *GET* solicita una representación de un recurso específico. Las peticiones que usan el método *GET* sólo deben recuperar datos.
PATCH	El método *PATCH* es utilizado para aplicar modificaciones parciales a un recurso.
DELETE	El método *DELETE* borra un recurso en específico.
PUT	El modo *PUT* reemplaza todas las representaciones actuales del recurso de destino.

- ***Headers*(encabezados).**

Los encabezados permiten pasar parámetros para personalizar las peticiones que se realicen. Se pueden utilizar encabezados *HTTP* estándar, así como encabezados propios de *Salesforce*. Los parámetros más comunes a utilizar en un encabezado son:

HTTP Accept: indica el formato en el que se está enviando la información. Los valores posibles son *application/json* y *application/xml*. El valor predeterminado es *application/json*.

HTTP Content-type: indica el formato en el que se envía la información cuando se trata de un archivo adjunto. Los valores posibles son *application/json* y *application/xml*.

Autorización *HTTP*: proporciona el *token* de acceso *OAuth 2.0* para que se pueda acceder al recurso(*uri*).

Estos son los encabezados que no pueden faltar en una petición *https,* sin embargo, se pueden adicionar parámetros extras que sean necesarios para que se logre tener una buena comunicación entre los diferentes sistemas. El parámetro *HTTP Content-type* no está considerado dentro de las peticiones que se realizarán más adelante en este trabajo y es relevante mencionar que se usarán otros parámetros dentro de los encabezados.

- **Cuerpo de la solicitud (*Request Body*).**

El cuerpo de la solicitud es un texto que se envía en la petición y puede estar en formato *json* o *xml*. Dentro de este formato se envían los campos o atributos necesarios para poder crear, actualizar, consultar o eliminar registros

Cuando se realiza una solicitud *HTTP* se espera recibir una respuesta, que devuelve un *status code* y un cuerpo (*response*), el cual puede ser la información solicitada o un mensaje de error.

- **Cuerpo de la respuesta (*Response Body*).**

El cuerpo de la respuesta es un texto que se envía en cada petición y puede estar en formato *json* o *xml*. Dentro de este formato se envían los campos o atributos necesarios solicitados o en su defecto un mensaje de error que indique porque no se devuelve la información correcta.

- ***Status Code*.**

Existen varios códigos *html* de estatus para las peticiones realizadas mediante el protocolo *HTTP*. De acuerdo las especificaciones de las *Open API* que se utilizan en este trabajo sólo se consideran los códigos:

200: OK.

Código que indica que la petición que se realizó ha sido entendida, enviada y recibida. En otras palabras, todo está correcto en la petición.

201: Creado.

Es como un código 200 *OK*, sin embargo, un código 201 significa que una solicitud se procesó correctamente y creó un recurso. Se utiliza para las solicitudes *POST*.

400: Mala solicitud.

Código de error que significa que el servidor no puede procesar la solicitud debido a un problema del cliente, por ejemplo, por mala sintaxis.

401: No autorizado.

Código de error que indica que la solicitud no incluye las credenciales de autenticación adecuadas.

403: Prohibido.

Código de error que indica que se ha entendido la solicitud del cliente, pero el servidor no la autoriza, por lo que, el cliente no puede acceder a ella. Por ejemplo, cuando el servidor no tiene autorización de usar algún método.

405: Método no permitido.

Código de error que indica que un recurso específico solicitado por el cliente no es compatible con el servidor. Es decir, que el servidor no soporta el método del que hace uso el cliente.

409: Conflicto.

Código de error que indica un conflicto de solicitud con el recurso destino. Por ejemplo, puede obtener una respuesta 409 al cargar un archivo que es más antiguo que el que ya está en el servidor, lo que genera un conflicto de control de versiones.

500: Error Interno del Servidor.

Código de error que indica que el servidor encontró un problema y no puede procesar la solicitud. Es un error genérico que suele ser devuelto cuando ningún otro código es adecuado o cuando existen problemas de conexión con el servidor.

2.2. TM Forum.

TM Forum es una alianza de empresas mundiales que cuenta con alrededor de 850 empresas como miembros, sin fines de lucro. Trabajan juntas con el objetivo de romper las barreras tecnológicas y culturales que en la actualidad existen entre los proveedores de servicios digitales, los proveedores de tecnología, las consultorías y los integradores de sistemas, con el objetivo de estandarizar, facilitar la interoperabilidad y lograr la comunicación o integración de las distintas arquitecturas digitales abiertas (*TM FORUM*, 2021).

Se enfoca en simplificar los procesos y operaciones comerciales a través de la transferencia de conocimientos, la colaboración y los estándares. Ha definido más de 50 Open *APIs* basadas en el modelo *REST* y la especificación *Open API* que permiten una integración de diferentes sistemas, creando de esta manera un ecosistema de comunicaciones. Las *APIs* de *TM Forum* son escépticas a la tecnología y pueden ser utilizadas en cualquier escenario de servicios digitales. Por lo que, son perfectas para las organizaciones que desean integrar sus sistemas utilizando software libre.

2.2.1. Marco de trabajo *TM Forum*.

El marco de trabajo de *TM Forum* es un como conjunto de estándares y mejores prácticas, se compone de cuatro elementos principales: marco de procesos de negocio (*business process framework*), marco de información (*information framework*), marco de aplicación (*application framework*) y marco de integración (*integration framework).* La Figura 8 muestra cómo está integrado el *framework TM Forum*.

Se realiza una breve descripción del marco de trabajo de *TM Forum* debido a que mediante los estándares de este modelo y *Open API* se puede compartir información al comunicar o integrar diferentes sistemas.

- **Business Process Framework**: enhanced Telecomunication Operations Map (*eTOM*) proporciona una visión completa de los procesos comerciales clave que un proveedor de servicios requiere para administrar su negocio.

- **Information Framework**: Shared Informational and Data model (*SID*) proporciona una definición integral a las industrias para que la información fluya a través de la empresa y entre los proveedores de servicios y sus socios comerciales.

- **Application Framework**: Telecommunications Applications Map (*TAM*) proporciona un modelo para agrupar procesos y su información asociada en aplicaciones reconocibles. Define un lenguaje común y un sistema de

identificación entre el comprador y el proveedor para todas las áreas de aplicación.

- **Integration Framework:** marco de integración que define cómo se pueden automatizar los procesos y la información detrás de estos sistemas mediante la definición de interfaces basadas en arquitectura orientada a servicios para poder integrar información de diferentes arquitecturas.

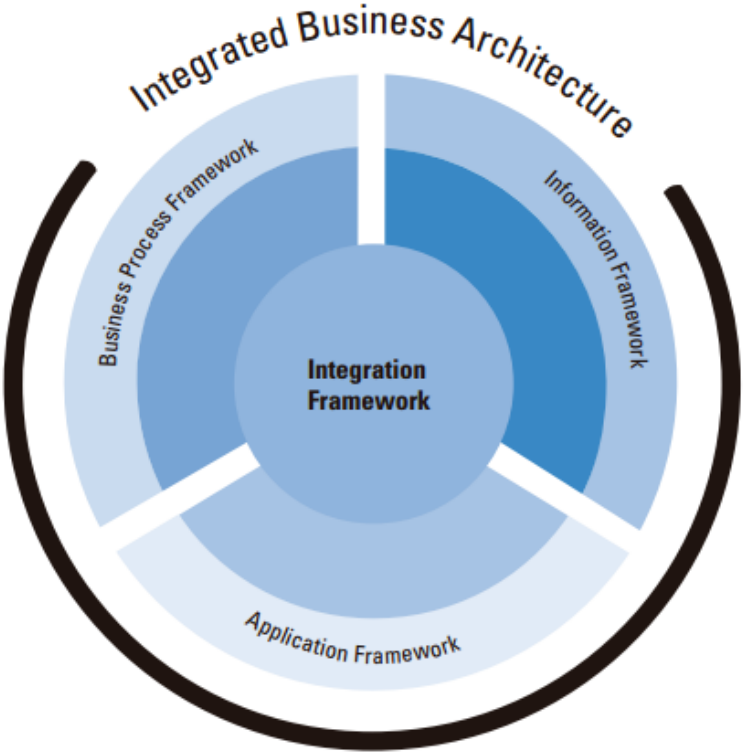

Figura 8. Marco de trabajo *Frameworx* del *TM Forum* (*Salesforce*, 2016).

El modelo brinda un "lenguaje común" para los proveedores de software e integradores de sistemas para ser usado a la hora de describir información de gestión, lo que permite una integración más fácil y efectiva entre las aplicaciones provenientes de diferentes fabricantes.

Uno de sus principales objetivos es lograr un alto grado de compatibilidad a la hora de definir una entidad. No basta con que los datos se almacenen utilizando la misma tecnología de bases de datos; si las entidades en cada sistema están definidas de distinta manera, resulta muy difícil lograr la integración entre estos sistemas. Es aquí donde el modelo entra a jugar su papel, definiendo entidades comunes, logrando así la estandarización de la información.

Por otra parte, *Salesforce* cuenta con un modelo de datos alineado al modelo de información compartida de *TM Forum*, con el objetivo de garantizar que el intercambio de datos entre diferentes proveedores se realice con éxito. En la Figura 9 se muestra el modelo de datos de *Salesforce*, el cual se compone en total de 43 entidades u objetos, 16 objetos estándar y 23 nuevos objetos. La integración de los nuevos objetos es con la finalidad de poder incorporar al modelo información sobre métodos de pago, servicios, facturación, inventario, recursos, y órdenes de trabajo referentes a los productos que se le pueden ofrecer a un cliente.

Como se mencionó anteriormente uno de los objetivos del *SID* era definir entidades compatibles, es por ello, que el modelo de datos de *Salesforce* incorpora nuevos objetos para que estén homologados.

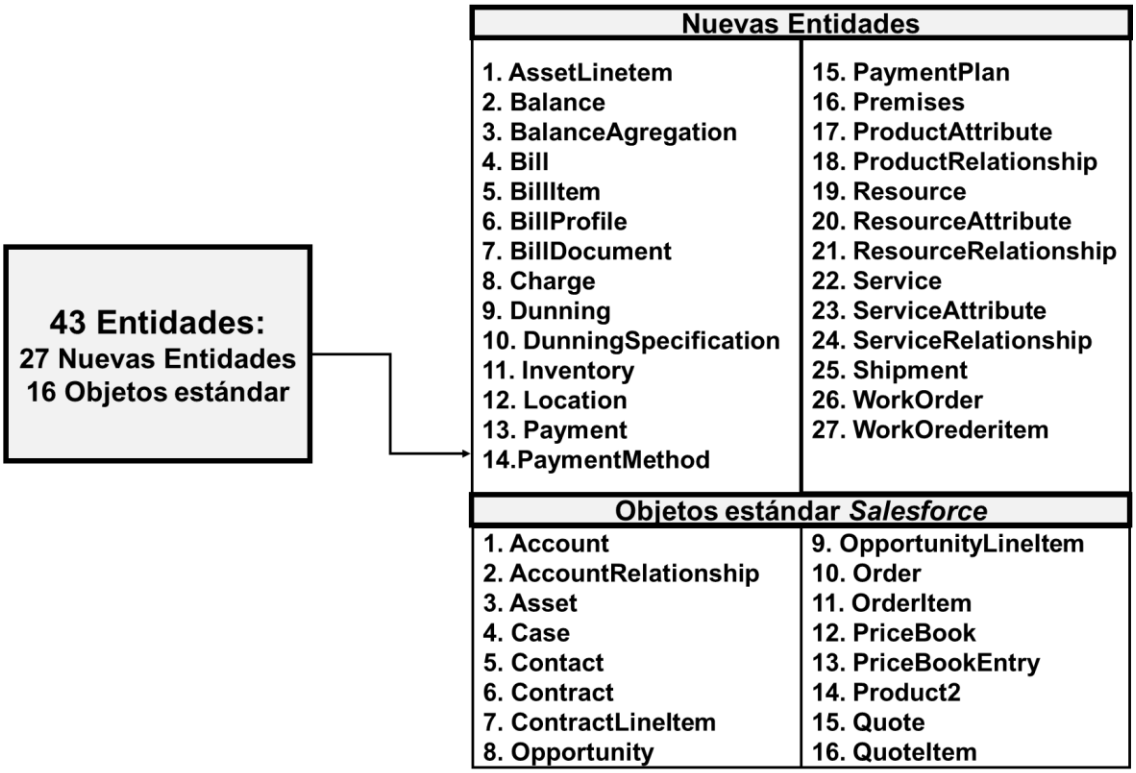

Figura 9. Modelo de datos *Salesforce* (*Salesforce*, 2016).

2.2.2. *Open APIs TM Forum*.

Actualmente existen 99 proveedores de servicios de comunicaciones (*CSP)* y participantes del ecosistema tecnológico han firmado el manifiesto de *APIs* abiertas para demostrar públicamente su respaldo a *TM Forum*.

Un grupo de 80 socios tecnológicos líderes se compromete a utilizar *Open APIs* de *TM Forum* en aplicaciones de productos relevantes y a continuar proporcionando comentarios y extensiones de las mismas para garantizar el crecimiento continuo de la comunidad en apoyo de éstas. Dentro de este grupo se

encuentra *Salesforce*, el cual ha colaborado continuamente con *TM Forum,* entre otras cosas, para probar cómo realizar los mapeos de las Open *APIs* demostrando que se pueden integrar en su plataforma sin ningún problema para lograr la comunicación con cualquier otro proveedor.

Una de las partes más importantes del programa de *APIs* de *TM Forum* es que están basadas en *REST* para permitir una integración rápida, repetible y flexible. Como se mencionó anteriormente *Salesforce* también cuenta con *API REST Apex*, por lo tanto, como ambos se basan en *REST* son totalmente compatibles.

Las *APIs* de *TM Forum* están diseñadas para su uso en cualquier servicio digital para la comunicación de dispositivos, incluidos *B2B*, Internet de las cosas (*IoT*), Salud digital, *Smart Grid, Big Data, NFV y Next Generation OSS/BSS*. Para ello se han desarrollado varias *APIs*, alrededor de 50, algunas ya terminadas al 100% y otras se encuentran en fase de pruebas para ser liberadas. Cuando la *API* es liberada se cuenta con una guía de especificaciones, de acuerdo al estándar especificación *Open API*, la cual contiene el nombre de la *API*, la versión, nombre de los recursos, los métodos *http*, los atributos de cada recurso, su tipo de datos y los códigos de respuesta para cada método.

El contexto *B2B* se refiere a la compra y venta de productos o servicios entre empresas. Desde la perspectiva digital, el B2B ha tomado un protagonismo importante con la creación de tiendas online con el objetivo de agrupar a vendedores y compradores.

El contexto *B2B2C* se refiere a las empresas que se asocian para fabricar un producto o prestar un servicio al consumidor final. Este es un proceso que implica una combinación de esfuerzos, pues en este modelo el proveedor establece una alianza comercial con distribuidores, minoristas o mayoristas con el fin de llegar a más clientes.

No es posible realizar la descripción de todas las *APIs* de *TM Forum* dentro de este trabajo, por lo tanto, se toma como ejemplo PartyManagement y *Geographic Management Address* para su desarrollo.

Capítulo III. Comunicación de *Salesforce* con sistemas externos.

En este capítulo se describe cómo se realiza la comunión de *Salesforce* con los diferentes sistemas externos.

3.1. Consumo y exposición de servicios desde *Salesforce*.

Dentro de *Salesforce* existen varias *APIs* que permiten que éste se comunique con diferentes sistemas o aplicaciones. La Tabla 4 muestra las opciones de *APIs* existentes para poder lograr esa comunicación.

Tabla 4. *APIs* de *Salesforce* para conectar dos o más aplicaciones (*Salesforce,* 2021).

Nombre de *API*	Protocolo	Formato de Datos	Comunicación
REST API	*REST*	*JSON, XML*	Síncrona
SOAP API	*SOAP (WSDL)*	*XML*	Síncrona
Chatter REST API	*REST*	*JSON, XML*	Síncrona (Las fotos se procesan de manera asíncrona).
Analytics REST API	*REST*	*JSON, XML*	Síncrona
Bulk API	*REST*	*CSV, JSON, XML*	Asíncrona. Carga de datos muy grandes, se hace por lotes.
Metadata *API*	*SOAP (WSDL)*	*XML*	Asíncrona. Recupera, implementa y modifica metadatos
Streaming API	*Bayeux*	*JSON*	Asíncrona.
Apex REST API	*REST*	*JSON, XML, personalizado.*	Síncrona
Apex SOAP API	*SOAP (WSDL)*	*XML*	Síncrona
Tooling API	Herramientas de desarrollo personalizadas para aplicaciones de la plataforma *Salesforce*		

En la Tabla 4 se pueden observar las diferentes opciones de *API* con las que cuenta *Salesforce*. Existe *REST API* y *Apex REST API*, es importante aclarar la diferencia entre ambas, ya que pudiera considerarse que sirven para lo mismo, sin embargo, aunque ambas están basadas en la arquitectura *REST*, existen algunas diferencias entre ellas. *REST API* es una *API* estándar que puede realizar operaciones *CRUD* sobre los registros, es decir, puede crear, leer, actualizar o borrar registros, pero únicamente sobre un objeto en específico de *Salesforce* a través de un servicio web. *Apex REST API* permite realizar operaciones personalizadas a través de métodos y clases de *Apex* dentro de *Salesforce*, las cuales puede usar un servicio web para realizar operaciones *CRUD* sobre varios

objetos, por ejemplo, se puede actualizar el objeto cuenta y el objeto contacto, en una misma clase de *Apex* utilizando dos métodos, uno para cada objeto. Esto último no se podría realizar con *REST API*.

Una vez aclarada la diferencia entre estas dos *API*, es importante mencionar que para este trabajo se usa *Apex REST API*, las demás *API* quedan fuera del alcance del mismo.

Como se mencionó en el capítulo anterior, *Salesforce* puede consumir o exponer servicios. La Figura 10 muestra el flujo general de cómo se expone un servicio desde Salesforce. Cualquier infraestructura externa puede enviar a *Salesforce* alguna petición, ya sea de consulta, creación, actualización o borrado de información, en formato *json*. Antes de que *Salesforce* atienda esta solicitud debe verificar que dicha aplicación tenga acceso a sus datos mediante la verificación del *token* que envía el sistema externo. Una vez verificada la autenticidad del sistema, *Salesforce* devuelve o responde la información solicitada en formato *json* y además envía un código de estatus que indica si la petición se realizó sin ningún error o si existió algún inconveniente al momento de procesar la petición.

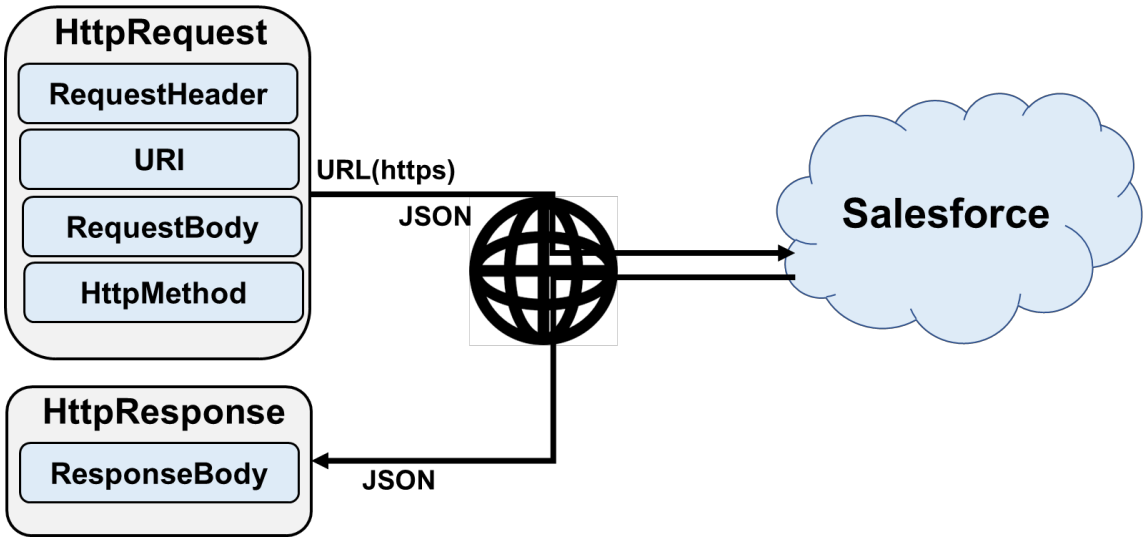

Figura 10. Flujo para exponer servicios desde *Salesforce*.

Para que una clase *Apex* esté disponible para un servicio web, se debe definir la clase como global y los métodos como estáticos globales. También se debe utilizar la anotación a nivel clase ***@RestResourse*** que permite exponer una clase como un recurso *REST*. Por ejemplo, ***@RestResource(urlMapping='/party/*')***, la clase que contiene esta anotación está exponiendo un servicio de la *API PartyManagement*. Los métodos que se exponen en una clase también utilizan ciertas anotaciones que nos permiten identificar qué operación se está realizando. La Tabla 5 muestra las anotaciones que se pueden utilizar dentro de estos métodos.

Tabla 5. Anotaciones usadas en métodos para exponer servicios.

Anotación	Acción	Detalles
@HttpGet	Leer	Se leen o recuperan registros.
@HttpPost	Crear	Se crean registros.
@HttpDelete	Eliminar	Se eliminan registros.
@HttpPut	Actualizar e insertar	Se suele usar para actualizar registros existentes o crear registros.
@HttpPatch	Actualizar	Se suele usar para actualizar campos de registros existentes.

Al exponer un servicio se hace uso de la clase ***RestContext*** que permite recuperar el *request* de la petición y enviar le *response* de la misma. Sus propiedades son ***RestRequest*** y ***RestResponse***.

RestRequest req = RestContext.request, por ejemplo, la variable *req* guarda la información de la solicitud que se envió en la petición del sistema externo hacia *Salesforce*.

RestRespose res = RestContext.response, aquí la variable *res* almacena la información que envía *Salesforce* a la solicitud del sistema externo.

Cuando se realiza una petición hacia S*alesforce* se debe contar con a la instancia con la se desea hacer la comunicación, en este caso, **https://transforma--transdev.my.salesforce.com/services/apexrest/**

Se puede decir, que cuando *Salesforce* consume un servicio, es el proceso inverso a cuando lo expone, envía una petición al sistema externo y éste le devuelve una respuesta. La Figura 11 muestra un ejemplo del flujo del proceso para consumir un servicio externo. Desde *Salesforce* se envía una petición o *request* en formato *json* hacia la infraestructura de software externa y está devuelve la respuesta o *response* además de un código de respuesta. Antes de responder la petición verifica que *Salesforce* tenga acceso a sus datos mediante el *token* que se envía. El código de respuesta indica si la petición fue procesada correctamente o no.

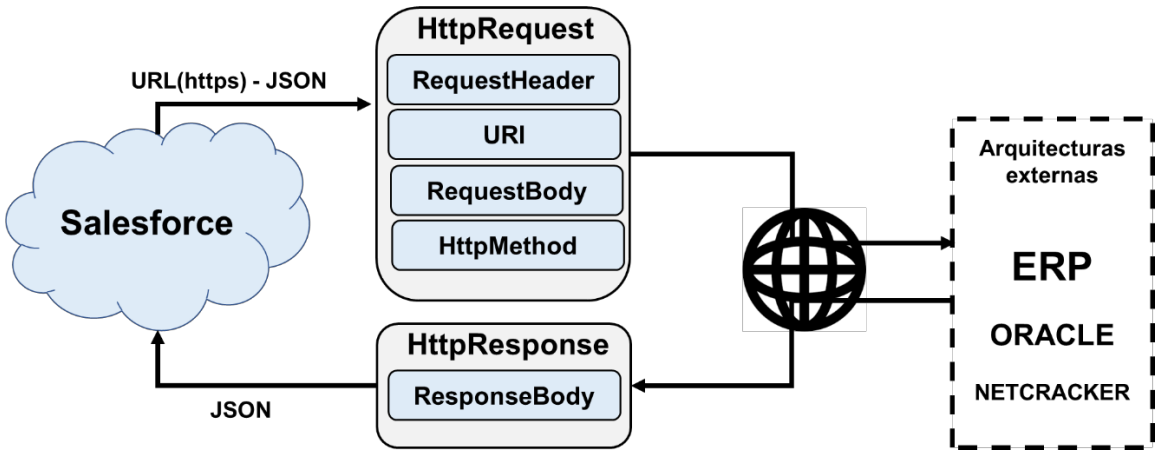

Figura 11. Flujo para consumir servicios desde *Salesforce*.

Para consumir el servicio hacia otro sistema se debe crear una clase pública. Los métodos de la clase pueden invocarse desde el *front* de *Salesforce* para que se active la petición. Para obtener los datos de un servicio se pueden usar los métodos *GET, POST, PUT, PATCH* y *DELETE*, anteriormente descritos. Existe la clase ***HttpRequest*** que permite realizar la comunicación, para ello, dentro de la clase se crea una nueva petición en la cual se indica el *endpoint* al cual se envía la petición y el método que se utiliza en dicha petición.

```
HttpRequest req = new HttpRequest();
req.setEndpoint('https://APIservicedesa.clarochile.cl:7006/PartyManagement/tmf-API/party/v1/individual/id=1234');
req.setMethod('GET');
```

Cabe aclarar, que cuando se hacen peticiones de consulta (*GET*) no es necesario enviar dentro de la petición una estructura, solamente se envían los parámetros necesarios en la *URL* para que se pueda realizar la consulta. La respuesta a dicha petición si deberá ser una estructura json, ya sea con la información correcta o con un mensaje de error.

3.2. OAuth 2.0 en Salesforce.

Salesforce utiliza el protocolo *OAuth* para proporcionar autorización a las *API* de forma estandarizada, el cual permite compartir información entre sitios sin tener que compartir la identidad. Dentro del marco de trabajo implementa *OAuth 2.0* para trabajar en conjunto con *API REST* y *Apex REST API* para establecer el envío continuo de credenciales entre cliente y servidor.Dentro de *OAuth 2.0* existen diferentes roles que van a participar en el proceso, como son dueño del recurso, cliente, servidor de recursos protegidos y servidor de autorización.

Dueño del recurso *(Owner)*: es el usuario que da autorización a una determinada aplicación para acceder a su cuenta y poder realizar operaciones en sus recursos, es decir, determina, si se puede actualizar, borrar o sólo consultar.

Cliente (*Client*): es la aplicación a la que desea acceder la cuenta de usuario, antes de que pueda hacerlo

Servidor de recursos protegidos (*Resource Server*): es el responsable de gestionar las peticiones de autorización. Verifica la identidad y genera un *token* de acceso a la aplicación del cliente.

Servidor de autorización (*Authorization Server*): es la *API* propiamente, es decir, es el servidor que aloja el recurso al que se hacen las peticiones.

La Figura 12 muestra el flujo genérico del protocolo *OAuth 2.0*. Una aplicación cliente quiere acceder a un recurso y para ello le pide la autorización al dueño del recurso por medio de un usuario y contraseña. Si todo sale bien, este le concede la autorización y pasa al servidor de autorización para obtener el *token* de acceso, una vez obtenido el *token* se conecta al servidor de recursos y se obtiene el acceso al recurso protegido.

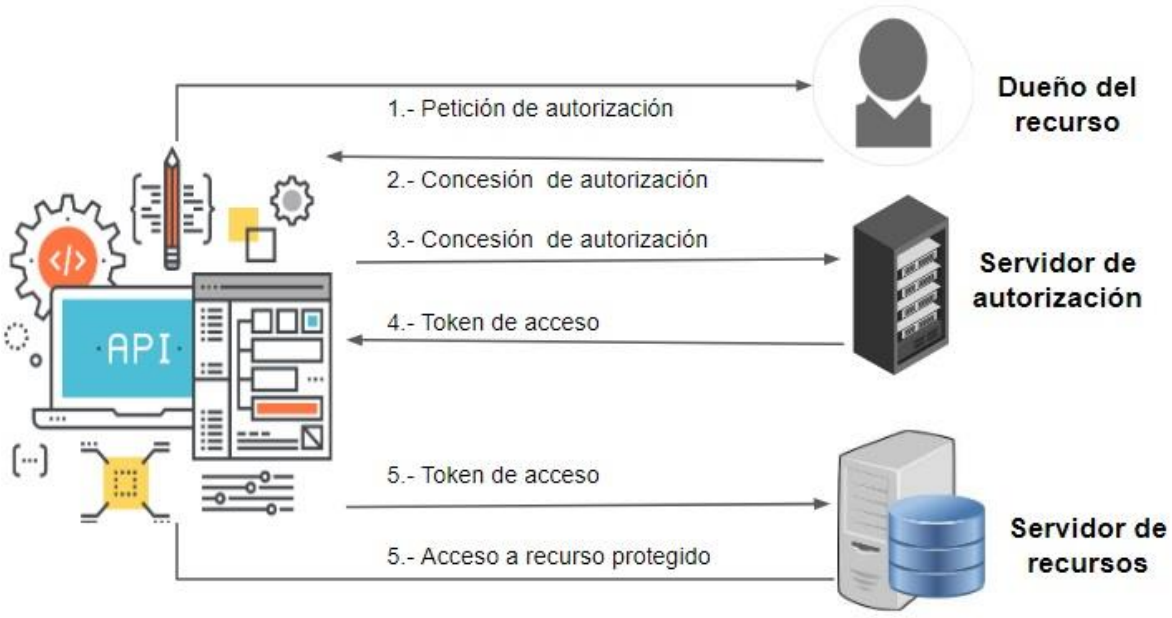

Figura 12. Flujo general OAuth 2.0.

Cuando *Salesforce* funge como servidor se debe dar de alta dentro del sistema el usuario con el cual el cliente va poder conectarse y realizar sus peticiones. Así mismo, es recomendable crear un perfil para asignarlo al usuario

y definir los permisos o accesos que tiene al sistema. Es recomendable crear un perfil descriptivo que permita identificar fácilmente que operaciones en el sistema se están realizando por peticiones externas.

Otra parte, se debe generar o crear una aplicación conectada en la configuración de *Salesforce* para obtener *client_id* y el *client_secret,* estos dos elementos permitirán generar el *token* de autenticación al sistema externo para tener acceso a *Salesforce*.

Una vez que se tiene la configuración, el flujo de autorización que se utiliza es *password* también conocido como *Resource Owner Password Credentials*, recomendado para entornos seguros, donde existe una relación de confianza entre el cliente y el servidor, se puede utilizar cuando un servicio del sistema operativo o una aplicación que requiera permisos elevados. Se puede hacer uso de la herramienta *Postman* para verificar que se obtiene correctamente el *token* con los parámetros indicados utilizando la *url* de autorización de *Salesforce* ***https://test.salesforce.com/services/oauth2/token***.

Una vez que se obtenga el *token* se debe enviar en el encabezado de la autorización *HTTP* en el siguiente formato ***Authorization: Bearer Access_Token.***

Dentro de *Postman* se configuran los parámetros que previamente se dieron de alta en *Salesforce*, para verificar la obtención del *token*. Los parámetros a configurar son:

- ***grant_type*:** es "*password*".
- ***username*:** es el nombre del usuario que se usa para identificarse en el servidor.
- ***password*:** es la contraseña del usuario que usa para identificarse en el servidor.
- ***client_id*:** es el identificador público de la aplicación. Una aplicación es el resultado de registrar un nuevo cliente en el servidor *OAuth*.
- ***client_secret*:** es la contraseña que se genera en el servidor de *OAuth* en relación con el cliente (la aplicación).

Obteniendo como resultado:

"**access_token**": "00D1g00000090Oq!ARAAQPZsrposrVzHu3730o_sSfmCJvLc1wNc2lqV2zL.GAhurMfJPbFMayJW7nW0P6J4kTX3CSHXVtjSMRh9v9hRveweS_pi"

Del lado del cliente, se deben de dar de alta las direcciones IP de *Salesforce* permitidas para evitar que el tráfico de Internet dirigido a Salesforce sea interceptado o redirigido a un sitio web deshonesto. El equipo de ingeniería de

Salesforce recomienda que los clientes incluyan en su lista de elementos permitidos las siguientes direcciones *IP* con máscara de subred que se incluyen en la Tabla 6.

Tabla 6. *Whitelist* direcciones *IP* permitidas por *Salesforce* (*Salesforce*, 2021).

Red IPv4	Intervalo IP IPv4
ARIN	
13.108.0.0/14	13.108.0.0 - 13.111.255.255
66.231.80.0/20	66.231.80.0 - 66.231.95.255
68.232.192.0/20	68.232.192.0 - 68.232.207.255
96.43.144.0/20	96.43.144.0 - 96.43.159.255
128.17.0.0/16	128.17.0.0 - 128.17.255.255
128.245.0.0/16	128.245.0.0 - 128.245.255.255
136.146.0.0/15	136.146.0.0 - 136.147.255.255
198.245.80.0/20	198.245.80.0 - 198.245.95.255
199.122.120.0/21	199.122.120.0 - 199.122.127.255
204.14.232.0/21	204.14.232.0 - 204.14.239.255
RIPE	
85.222.128.0/19	85.222.128.0 - 85.222.159.255
159.92.128.0/17	159.92.128.0 - 159.92.255.255
160.8.0.0/16	160.8.0.0 - 160.8.255.255
161.71.0.0/17	161.71.0.0 - 161.71.127.255
163.76.128.0/17	163.76.128.0 - 163.76.255.255
163.79.128.0/17	163.79.128.0 -163.79.255.255
185.79.140.0/22	185.79.140.0 - 185.79.143.255
APNIC	
101.53.160.0/19	101.53.160.0 - 101.53.191.255
104.161.128.0/17	104.161.128.0 - 104.161.255.255
161.32.64.0/18	161.32.64.0 - 161.32.127.255
161.32.128.0/17	161.32.128.0 - 161.32.255.255
161.71.128.0/17	161.71.128.0 - 161.71.255.255
182.50.76.0/22	182.50.76.0 - 182.50.79.255
202.129.242.0/23	202.129.242.0 - 202.129.243.255

Si la compañía tiene políticas para incluir únicamente dominios de *Salesforce*, se deben incluir los aquí mencionados para garantizar que recibe todos los contenidos procedentes de *Salesforce*:

force.com
salesforce.com
salesforceliveagent.com
visualforce.com
documentforce.com

lightning.com
salesforce-communities.com
forceusercontent.com

Salesforce puede actuar como cliente o como servidor para la comunicación mediante integraciones teniendo en cuenta la *Whitelist*.

***Salesforce* como cliente.**

- El servidor proporcionará un certificado de seguridad en formato *Java Keystore* (.jks) el cual se utilizará en cada llamada a los servicios externos a *Salesforce*.

***Salesforce* como servidor.**

- *Salesforce* generará un certificado de seguridad (.crt) el cuál utilizará el cliente para obtener una comunicación adecuada.

Salesforce puede mantener una comunicación de doble vía de autenticación mediante la cual ambas partes generarán sus respectivos certificados de seguridad.

Ahora, cuando *Salesforce* es cliente, de igual manera se debe obtener un *token* de acceso al servidor. Para ello se hace uso de metadatos personalizados para dar de alta toda la información de acceso. Los metadatos representan la información que describe la configuración de la organización de cada usuario. En este caso se creó el metadato ***INT_Credential***, para almacenar el usuario, *password*, *client_id* y *client_secret* del servidor al que se realizan las peticiones.

Una vez que se configuran los parámetros de autenticación es necesario crear un método dentro de una clase *Apex* en *Salesforce* para obtener el *token* y verificar las credenciales con las que se desea acceder al sistema externo.

El Listado 7 muestra el código del método para obtener *token* y credenciales.

```
/**
* @description Metodo que retorna el token por usuario y password
* @param strCountry Country of endpoint and resource to search
* @param integration Name of integration to search in custom metadatas
* @return String Token */

Public static String obtenerTokenUserPasss(String strCountry, String
integration){
      List<INT_Endpoint__mdt> endpoint = [SELECT URL__c
                                          FROM INT_Endpoint__mdt
                                          WHERE Country__c =: strCountry];
      List<INT_Resource__mdt> resource = [SELECT Resource__c
                                          FROM INT_Resource__mdt
                                          WHERE Country__c =: strCountry
                                         AND        Integration__c        =:
                              integration];
      String username = '';
      String password = '';
```

```
      List<INT_Credential__mdt>credentials=

INT_Builder.getCredentials(strCountry);
      for (INT_Credential__mdt credential: credentials) {
            if(credential.Key__c.equals('pass')){
                  password = credential.Value__c;
            }
            if(credential.Key__c.equals('user')){
                  username = credential.Value__c;
            }
      }
      System.debug('::::: user : '+ username + '    passs: '+password);
       Blob headerValue = Blob.valueOf(username + ':' + password);
      String       authorizationHeader       =       'Basic       '       +
      EncodingUtil.base64Encode(headerValue);
      Map<String,String> response = new Map<String,String>();
      response.put('Content-Type', 'application/x-www-form-urlencoded');
      response.put('Authorization', authorizationHeader);
      String url = endpoint[0].URL__c+':7000';
      System.debug('::::::: url '+url);
      INT_HTTP_Client request = new INT_HTTP_Client (url, 'POST');
      request.setBody(resource[0].resource__c);
      request.setHeaders(response);
      TokenAccess wrapObj =  null;
      try {
            HttpResponse res =  request.getResponse();
            System.debug('--> Endpoint : '+endpoint[0]);
            System.debug('--> Body : '+res.getBody());
            System.debug('--> Header : '+res.getHeader('Authorization'));
            System.debug('res:  '+res);
            String str = res.getBody();
            System.debug('Body:  '+str);
            wrapObj                                                    =
(TokenAccess)Json.deserialize(str,TokenAccess.class);
            System.debug('wrapObj.access_token : '+wrapObj.access_token);
      } catch (System.CalloutException e) {
            System.System.debug('Error : ' +e.getMessage());
      }
                  return wrapObj.access_token;
      }

/**
* @description Method to get credentials for login processes for each
country
* @param country, Country of credentials to search
* @return List<INT_Credential__mdt>  List of credentials*/
public static List<INT_Credential__mdt> getCredentials (String country) {
      return [SELECT Key__c, Value__c
              FROM INT_Credential__mdt
               WHERE Country__c =: country];
}
```

Listado 7. Método para obtener *token y credenciales de autenticación.*

Se crearon los metadatos ***INT_Endpoints*** e ***INT_Resources***. En el primero se almacenan los *enpoints* o dominios a los que se desea acceder y en el segundo

el complemento de la *uri* que corresponde al recurso al que se va a acceder. Estos metadatos permiten obtener la *url* a la cual se enviará la petición, por ejemplo,

https://APIservicedesa.clarochile.cl:7006/PartyManagement/tmf-API/party/v1/individual

En ***INT_Endpoints*** se guarda **https://APIservicedesa.clarochile.cl:7006,** que es el dominio al que se va realizar la petición y en ***INT_Resources*** se almacena **/PartyManagement/tmf-API/party/v1/individual**, nombre del recurso al que se desea acceder.

Para recuperar los datos de *enpoints* y recursos se genera el método ***getEndPoint***. El Listado 8 muestra el código de dicho método, el cual devuelve el *url* completo para poder acceder al recurso, es decir, **https://APIservicedesa.clarochile.cl:7006/PartyManagement/tmf-API/party/v1/individual**

```
/**
* @description Method to get endpoint for request through custom metadatas
      INT_Endpoint__mdt and INT_Resource__mdt like: https:endpoint.com/resource
* @param country Country of endpoint and resource to search
* @param integration Name of integration to search in custom metadatas
* @return String Endoint
*/
public static String getEndpoint (String country, String integration) {
      List<INT_Endpoint__mdt> endpoint = [SELECT URL__c
                                          FROM INT_Endpoint__mdt
                                          WHERE Country__c =: country];
      List<INT_Resource__mdt> resource = [SELECT Resource__c
                                          FROM INT_Resource__mdt
                                          WHERE Country__c =: country
                                        AND Integration__c =: integration];
      String result = '';
      if (!endpoint.isEmpty()) {
            if (!resource.isEmpty()) {
                  result = endpoint[0].URL__c + resource[0].resource__c;
            } else {
                  result = 'Resource not found in Metadata';
            }
      } else {
            result = 'Endpoint not found in Metadata';
      }
     system.debug('URL -------> ' + result);
      return result;
}
```

Listado 8. Método para obtener *URL* completo para realizar la petición al sistema externo.

Es necesario dar acceso a los *endpoints* o dominios dentro de *Salesforce* para poder establecer la comunicación, para ello, se configuran o se dan de alta dentro de Sitios Web Remotos. Si se omite este paso no podrá establecerse la comunicación entre ambos sistemas.

3.3. Encabezados o headers de recursos.

Como parte de la definición de los recursos se cuenta con parámetros que viajan en las cabeceras de las peticiones y en sus respectivas respuestas, para dichas cabeceras se cuenta con la estructura presentada en la Figura 13, en la cual se observa tanto la estructura del encabezado al momento de enviar una petición como la estructura del encabezado al momento recibir la respuesta.

Cabe destacar que esta estructura es empleada en los recursos del presente trabajo PartyManagement y *Geographic Management Address*, así como en las operaciones de éstos. Las cabeceras se encuentran divididas en dos: *Request Headers* y *Response Headers*, empleados en la petición y respuesta de las interfaces, respectivamente.

Los encabezados *HTTP* personalizados proporcionan información de contexto al momento de realizar las peticiones, como la región, los detalles de la organización o el rol de la persona que ve el objeto externo. La manera de definir los encabezados debe venir en el contrato de cada *API*. La Figura 13 muestra la definición de los encabezados tanto en el *request* como en el *response* de la petición.

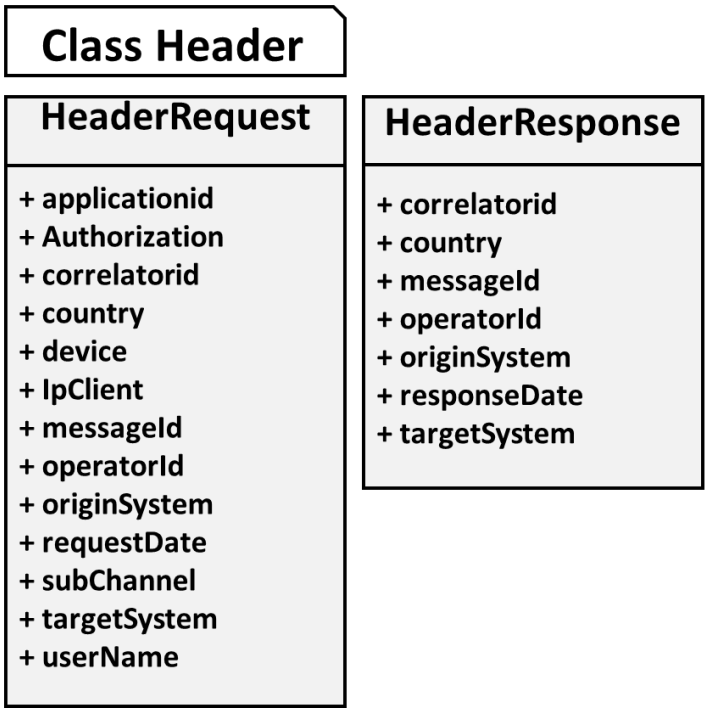

Figura 13. Definición de parámetros de encabezados.

La Tabla 7 muestra la definición de los parámetros para *HeaderRequest* y la Tabla 8 para los parámetros *HeaderResponse*.

Tabla 7. Parámetros de *Header Request*.

Elemento	Tipo	Multiplicidad	Descripción
messageID	String	Opcional	Identificador del mensaje.
originSystem	String	Opcional	Sistema origen de la transacción
requestDate	Date-time	Opcional	Fecha de la petición
targetSystem	String	Opcional	Sistema destino de la transacción
Country	String	Opcional	País a en-rutar. ISO 1366 - 3 Caracteres
correlatorId	String	Opcional	Identificador utilizado para hacer coincidir la solicitud con su correspondiente notificación asincrónica
Authorization	String	Opcional	Una cadena de tokens segura que indica que es una solicitud autorizada
ipClient	String	Opcional	IP del dispositivo desde que se originó la invocación.
channel	String	Opcional	Identifica el canal que invoca el servicio.
subChannel	String	Opcional	Identifica el sub-canal que invoca el servicio.
applicationId	String	Opcional	Identificador de la aplicación
Device	String	Opcional	Dispositivo desde donde se invoca el servicio.
userName	String	Opcional	Identificador del usuario
operatorId	String	Opcional	Identificador del operador

Tabla 8. Parámetros de Header Response.

Elemento	Tipo	Multiplicidad	Descripción
messageID	string	Opcional	Identificador del mensaje.
originSystem	string	Opcional	Sistema origen de la transacción
responseDate	Date-time	Opcional	Fecha de la petición
targetSystem	string	Opcional	Sistema destino de la transacción
Country	string	Opcional	País a en-rutar. ISO 1366 - 3 Caracteres
correlatorId	string	Opcional	Identificador utilizado para hacer coincidir la solicitud con su correspondiente notificación asincrónica

Como se mencionó en el apartado 3.2 cuando *Salesforce* realiza una petición a un sistema externo se debe de enviar el *token* que permite esta comunicación. El *token* se envía dentro del *header* de la petición, pero además se envían más parámetros con información personalizada. Se creó ***INT_headers*** dentro de los metadatos de *Salesforce* para dar de alta todos los parámetros que se requieren al momento de armar el encabezado. Algunos parámetros son fijos, mientras que otros se obtienen del sistema. Los parámetros que son fijos están almacenados dentro de los metadatos en ***INT_Headers***.

A través del método ***fillHeaders*** se obtienen los parámetros que se envían en una petición hacia el sistema externo. El Listado 9 muestra el código del método armar el *header* en cada petición.

```
/**
* @description Fill the request or response headers. This Headers get from
Custom * Metadata entity,
* @param token A token string for Authentication.
* @param country Country
* @param messageId A message that contains a identifier for the transaction
* @param isRequest      if the value is true is Request on the other hand
if is false * is a Response
* @return A map with headers in key-value format
*/
public static Map<String, String> fillHeaders (String token, String
country, String messageId, Bo
olean isRequest) {
      List<INT_Header__mdt> headers = new List<INT_Header__mdt> ();
      Map<String,String> response = new Map<String,String>();
      String integration = '';
      String fCountry = country;
      if (isRequest) {
            integration = 'RequestHeader';
      } else {
            integration = 'ResponseHeader';
      }
      System.debug('----------------------Headers----------------------');
      headers = [SELECT Key__c, Value__c FROM INT_Header__mdt
                  WHERE Integration__c =: integration];
      for (INT_Header__mdt header: headers) {
            if (header.Key__c == 'Authorization') {
                header.Value__c = token;
            } else if (header.Key__c == 'requestDate' ||
                        header.Key__c ==  'responseDate') {
//se actualiza la forma de mostrar las fechas con el formato 2020-12-
03T06:16:43Z
                  String strDateTime = String.valueOf(Datetime.now());
                  header.Value__c    =    strDateTime.substring(0,10)    +
'T00:00:00Z';
            } else if (header.Key__c == 'country') {
                  header.Value__c = country;
            } else if (header.Key__c == 'messageId') {
                  header.Value__c = messageId;
            } else if (header.Key__c == 'userName') {
                  header.Value__c = System.UserInfo.getUserName();
            } else if (header.Key__c == 'operatorId') {
                  header.Value__c = getOperatorId(fCountry);
            }
            response.put(header.Key__c, header.Value__c);
            System.debug(header.Key__c + ' = '+header.Value__c);
      }
      response.put('Content-Type','application/json;charset=utf-8');
      System.debug('----------------------Headers----------------------');
      return response;
}
```

Listado 9. Método para armar *header* de peticiones.

A manera de resumen, los valores del *header* obtenidos usando el método ***fillHeaders*** se muestran en la Tabla 9.

Tabla 9. Ejemplo de valores de *Header Request*.

Parámetro	Valor
Authorization	Bearer eyJ4NXQiOiJNell4TW1Ga09HWXdNV0kwWldObU5EY3hOR1l3WW1NNFpUQTNNV0kyTkRBelpHUXpOR00wWkdSbE5qSmtPREZrWkRSaU9URmtNV0ZoTXpVMlpHVmxOZyIsImtpZCI6Ik16WXhNbUZrT0dZd01XSTBaV05tTkRjeE5HWXdZbU00WlRBM01XSTJOREF6WkdRek5HTTBaR1JsTmpKa09ERmtaRFJpT1RGa01XRmhNelUyWkdWbE5nX1JTMjU2IiwiYWxnIjoiUlMyNTYifQ.eyJzdWIiOiJlYWd1aWxhckVycHNvbEBjYXJib24uc3VwZXIiLCJhdXQiOiJBUFBMSUNBVElPTiIsImF1ZCI6InVpa2RjbGZ2SV90aUpma01LVzExWU9Ga2FiQWEiLCJuYmYiOjE2Mzk0MjU2MjksImF6cCI6InVpa2RjbGZ2SV90aUpma01LVzExWU9Ga2FiQW
Content-Type	application/json
Device	WEB
ipClient	0.0.0.0
operatorId	client_credentials
subChannel	SUBCANAL
messageId	20211213-140029-11
applicationId	IDAPP
Country	CHI
Accept	application/json;charset=utf-8
requestDate	2021-12-13T00:00:00Z
Channel	CANAL
targetSystem	BUS
userName	carmonaa@globalhitss.com.transdev
originSystem	SFDC

Los valores calculados son ***Authorization***, que contiene el *token* que previamente se obtuvo para tener acceso al servidor al cual se está realizando la petición; ***messageId***, en el cual se envía la fecha y hora del sistema; ***requestDate***, el cual contiene la fecha del sistema en formato de *Salesforce* y ***userName***, en el que se envía el nombre del usuario logueado con el que se realiza la petición. Los demás valores son fijos y se encuentran definidos en los metadatos de *Salesforce*.

De la misma manera, cuando el sistema externo envía una petición hacia *Salesforce* debe de venir la información necesaria dentro de los encabezados para realizarse la conexión.

3.4. Mapeo de contratos de *API TM Forum* a objetos de *Salesforce*.

En el Capítulo I se mencionó que toda *API* que se crea mediante el estándar de especificaciones *Open API* tiene un contrato o *swagger* en el cual se define la estructura *json* que se utilizará para enviar y recibir información. El *API PartyManagement* de *TM Forum* fue creada bajo este estándar, por lo tanto, cuenta con su contrato para definir los recursos a los que se tiene acceso y métodos que soporta.

Con *PartyManagement* se puede acceder a los recursos *Organization, Individual y Checkcredit*, de los cuales utilizaremos los dos primeros. *Organization* se usa con el objeto *Account* o cliente y soporta los métodos *GET*, *PATCH* y *POST*. *Individual* se usa el objeto *Contact* o contacto y soporta los métodos anteriormente mencionados.

En *Salesforce* existen objetos estándar o personalizados. Un objeto estándar es aquel que existe por default y si es necesario se le pueden agregar más campos para complementar la funcionalidad, para efectos de este trabajo *Account*(clientes) y *Contact*(contacto) son objetos de este tipo. Por otro lado, un objeto personalizado es aquel que se crea desde cero dentro de *Salesforce*, sin embargo, no se utiliza ninguno en este alcance.

Una vez que se tiene definido el contrato de la *API* es necesario relacionar u homologar los campos del mismo con los campos de los objetos utilizados en *Salesforce*. Para ello, se realiza un mapeo de información para conocer de dónde se debe obtener la información que debe viajar en la estructura del *json* cuando se realiza una petición o para saber dónde almacenar la información proveniente de dicha estructura, esto permitirá lograr una buena comunicación entre ambos sistemas.

Salesforce puede o no almacenar la información proveniente del *json*, en algunas ocasiones sólo servirá para ser mostrada dentro de las pantallas del sistema sin que quede almacenada, sobre todo cuando se trata de peticiones de consulta.

En el Listado 10 se muestra la estructura de un *json* para crear un contacto, esta petición puede ser bidireccional, es decir, se puede consumir el servicio hacia el sistema externo, se manda crear el contacto al sistema externo; o se puede exponer el servicio, es decir, el sistema externo manda crear el contacto en *Salesforce*.

```
1 {
2   "title": "Catedrático",
3   "secondLastName": "Ariel",
4   "relatedParty": [
5     {
6       "role": "customer",
7       "identifierType": "customerId",
8       "id": "03368627",
9       "externalReference": [
10         {
11           "name": "0011g00000xY28pAAC",
12           "externalReferenceType": "LegacyCustomerId"
13         }
14       ]
15     }
16   ],
17   "partyProfileType": [
```

```
    {
      "name": "CL - Contactos",
       "id": "RegistroCode-001",
      "description": "Técnico",
      "category": "TipoRegistro"
    }
  ],
  "partyCharacteristic": [
    {
      "value": "CLP",
      "name": "Divisa"
    },
    {
       "value": "Comercial",
       "name": "Cargo"
     },
     {
       "value": "false",
       "name": "hasBiometrics"
     }
   ],
   "nationality": "Argentina",
   "lastUpdateDateTime": "2021-07-19T00:00:00Z",
   "languageAbility": [
     {
       "languageName": "Español"
     }
   ],
   "individualIdentification": [
     {
       "identificationType": "RUT",
       "identificationId": "11597762",
       "checkDigit": "8"
     }
   ],
   "id": "00339005",
   "givenName": "Pedro",
   "gender": "Masculino",
   "familyName": "Salgado",
   "externalReference": [
     {
       "externalReferenceType": "LegacyCustomerId"
     }
   ],
   "contactMedium": [
     {
       "preferred": true,
       "mediumType": "emailAddress",
       "characteristic": {
         "emailAddress": "antonio.salgado@gmail.com"
       }
     },
     {
      "preferred": true,
       "mediumType": "telephoneNumber",
       "characteristic": {
         "phoneNumber": "5567899001"
```

```
75        }
76      },
77      {
78        "preferred": true,
79        "mediumType": "mobileNumber",
80        "characteristic": {
81          "mobileNumber": "5567899020"
82        }
83      },
84      {
85        "mediumType": "AttentionTime",
86        "characteristic": {
87          "calendar": [
88            {
89              "hourPeriod": [
90                {
91                  "startHour": "15:00:00.000Z",
92                  "endHour": "19:00:00.000Z"
93                }
94              ],
95              "day": "Miércoles",
96              "comment": "Matutino"
97            }
98          ]
99        }
100     },
101     {
102       "preferred": true,
103      "mediumType": "contactAddress",
104       "address": {
105         "streetType": {
106           "name": "Calle"
107         },
108         "streetNr": "203",
109         "streetName": "Arco 2 BCD 034",
110         "stateOrProvince": {
111           "type": "state",
112           "name": "Queretaro",
113           "id": "100"
114         },
115         "postcode": "78100",
116         "locality": {
117           "type": "locality",
118           "name": "Obrera",
119           "id": "103"
120         },
121         "id": "12309s",
122         "geographicSubAddress": [
123           {
124             "type": "building",
125             "subUnitType": "particular",
126             "subUnitNumber": "88",
127             "levelType": "Floor",
128             "levelNumber": "2",
129             "buildingName": "2"
130           },
131           {
```

```
132             "type": "block",
133             "name": "Empresalia",
134             "id": "104"
135           }
136         ],
137         "geographicLocation": {
138           "geometry": [
139             {
140               "y": "34.000000",
141               "x": "21.000000"
142             }
143           ]
144         },
145         "city": {
146           "type": "city",
147           "id": "102"
148         }
149       }
150     }
151   ],
152   "birthDate": "1980-04-23T00:00:00"
153 }
```

Listado 10. Estructura *json* para crear un Contacto.

Tomando como ejemplo la estructura del *json* del Listado 10, la línea 3, 54 y 56 representan el nombre del contacto y se mapean de acuerdo a la Tabla 10.

Tabla 10. Mapeo del nombre del Contacto.

Entidad Salesforce	Campos SFDC	Parámetro	Tipo de dato y longitud	Obligatorio Integración	Valores	Parámetro TMF Open API
Contacto (Contact)	FirstName	Nombre	Texto(40)	Si		givenName
	LastName	Apellido paterno	Texto(40)	Si		familyName
	LastName	Apellido materno	Texto(40)	Si		secondLastName

Otro ejemplo de mapeo se representa en la Tabla 11. Tomando como ejemplo de la línea 46 a la 52 del Listado 10. Estas líneas representan los datos fiscales del contacto.

Tabla 11. Mapeo de datos fiscales del Contacto.

Entidad Salesforce	Campos SFDC	Parámetro	Tipo de dato y longitud	Obligatorio Integración	Valores	Parámetro TMF Open API
Contacto (Contact)	TipoDocumento__c	Tipo de documento	Lista de selección	Si	RUT S/D	individualIdentification[] .identificationType
	IdentificadorTributario__c	Número de documento	Texto(20)	Si		individualIdentification[]. identificationId
	DigitoVerificacion__c	Digito verificador	Lista de selección	Si	0 1 2 3 4 5 6 7 8 9 K	individualIdentification[]. checkDigit

En la estructura *json* del Listado 10 no sólo viene información del contacto, además, contiene información del cliente al que está ligado el contacto y la dirección del mismo. Por esto, es importante tener mapeada la información de cada *API* para tener claro en qué objeto de *Salesforce* se debe almacenar o de qué objeto se debe extraer.

Los contratos de cada *API* se deben de convertir a una clase de *Apex* para que se pueda leer la estructura de cada *json* y utilizar los campos dentro de otras clases, generalmente, esta clase se conoce como *wrapper* y debe existir al menos uno por *API*. Es importante considerar que esta clase se podrá usar para consumir o exponer servicios.

Se usa el término *wrapper* para indicar que existe compatibilidad o interoperabilidad entre diferentes estructuras de software o para poder representar algo a nivel visual. El programa principal se comunica con el *wrapper* y este se encarga de transmitir o devolver los comandos necesarios para una correcta interpretación de información entre dos sistemas distintos.

Por ejemplo, si dentro del contrato se tiene definido el parámetro ContactMedium como se muestra en la Tabla 12.

Tabla 12. Definición de parámetro ContactMedium.

Elemento	Tipo	Mandatorio/ Opcional	Descripción
mediumType	string	Opcional	Tipo de medio de contacto, ejemplo: Número telefónico, Número celular, dirección postal, etc.
Preferred	boolean	Opcional	Si es verdadero indica que es el medio de contacto preferido
Characteristic	MediumCharacteristic	Opcional	Describe las características del medio de contacto que pueden ser usadas para contactar al party

Address	Geographic AddressRef	Opcional	Referencia a un recurso de dirección.
validFor	TimePeriod	Opcional	Periodo de validez para el medio de contacto

En la clase *wrapper* se debe definir el código del parámetro ContactMedium. El Listado 11 muestra el código con el que se define este parámetro dentro de la clase *wrapper*. Esta parte de código permite llenar los datos que se representan en el Listado10 de la línea 62 a la 151.

```
1 public List<ContactMedium> contactMedium {get;set;}

2 public class ContactMedium {
3     public String mediumType {get;set;}
4     public Boolean preferred {get;set;}
5     public Characteristic characteristic {get;set;}
6     public Address address {get;set;}
7     public ValidFor validFor {get;set;}

8      public ContactMedium() {
9       }
10    public ContactMedium(JSONParser parser) {
11          while (parser.nextToken() != System.JSONToken.END_OBJECT) {
12                if (parser.getCurrentToken() ==
System.JSONToken.FIELD_NAME) {
13                      String text = parser.getText();
14                      if (parser.nextToken() !=
System.JSONToken.VALUE_NULL) {
15                           if (text == 'mediumType') {
16                                mediumType = parser.getText();
17                           } else if (text == 'preferred') {
18                                preferred = parser.getBooleanValue();
19                           } else if (text == 'characteristic') {
20                                characteristic = new
Characteristic(parser);
21                           } else if (text == 'address') {
22                                address = new Address(parser);
23                                 else if (text == 'validFor') {
24                                validFor = new ValidFor(parser);
25                           } else {
                                  System.debug(LoggingLevel.WARN,
                                  'ContactMedium consuming unrecognized
                                  property: '+text);
26                                consumeObject(parser);
27                           }
28                      }
29                }
30          }
31    }
32 }
```

Listado 11. Código para definir el parámetro ContactMedium en la clase *wrapper*.

El Listado 11 muestra sólo un ejemplo del código completo que compone la clase *wrapper*, así como se define el parámetro ContactMedium se pueden definir

los demás parámetros que existen en la *API*. En la línea 8 y 9 se define un constructor del parámetro ContactMedium para que se utilice cuando se expone un servicio desde *Salesforce*. De la línea 10 a la línea 31 se define el código del parámetro que permite se consuman servicios hacia Salesforce.

3.5. Especificaciones de *API TM Forum*.

3.5.1 Especificación PartyManagement.

En las especificaciones de la *API* se incluye el modelo de relación entre recursos y subrecursos, los tipos de datos que se usan, así como las operaciones que se pueden realizar. En el caso de *PartyManagement* se puede consultar, crear o actualizar información de cada recurso. La *API* se encuentra dentro de *TM Forum* con el número de documento *TMF632* en su última versión que es la 19.0.1.

La *API* de *PartyMAnagement* permite gestionar información de contactos o individuos y de clientes o cuentas de las cuales la organización necesite almacenar información. La *API* maneja los recursos *Individual* y *Organization*:

Individual: representa a un individuo, ser humano (un hombre, una mujer o un niño). El individuo puede ser un cliente, un empleado o cualquier otra persona sobre la que la organización necesite almacenar información.

Organization*:* representa a un grupo de personas identificadas por intereses o propósitos compartidos, es decir, una empresa de la cual se necesita almacenar información.

Las operaciones que se pueden realizar en ambos recursos son consulta, actualización, creación y eliminación de información.

- Recuperación de información de un individuo o una organización
- Actualización parcial de un individuo o una organización
- Creación de un individuo o una organización
- Eliminación de un individuo o una organización

La Figura 14 muestra de manera general y a modo de diagrama las operaciones que tiene *Open API PartyManagement* para la convivencia con otros componentes dentro de la arquitectura de integración o bien con otros sistemas y/o plataformas.

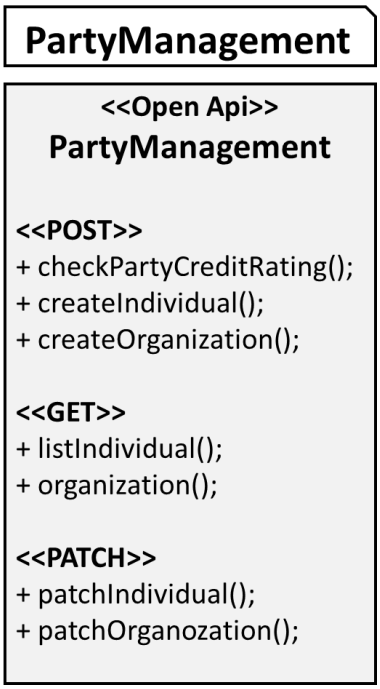

Figura 14. Operaciones del *API PartyManagement*.

3.5.1.1 Recurso Individual.

La Figura 15 muestra el modelo de datos que se utiliza para el recurso *Individual*. En Anexo 1 se encuentra la estructura completa del *json* que representa el modelo, con todos los atributos y tipos de datos que se deben utilizar. En este diagrama únicamente se muestran los nombres de los subrecursos del recurso *individua*l y su relación entre sí.

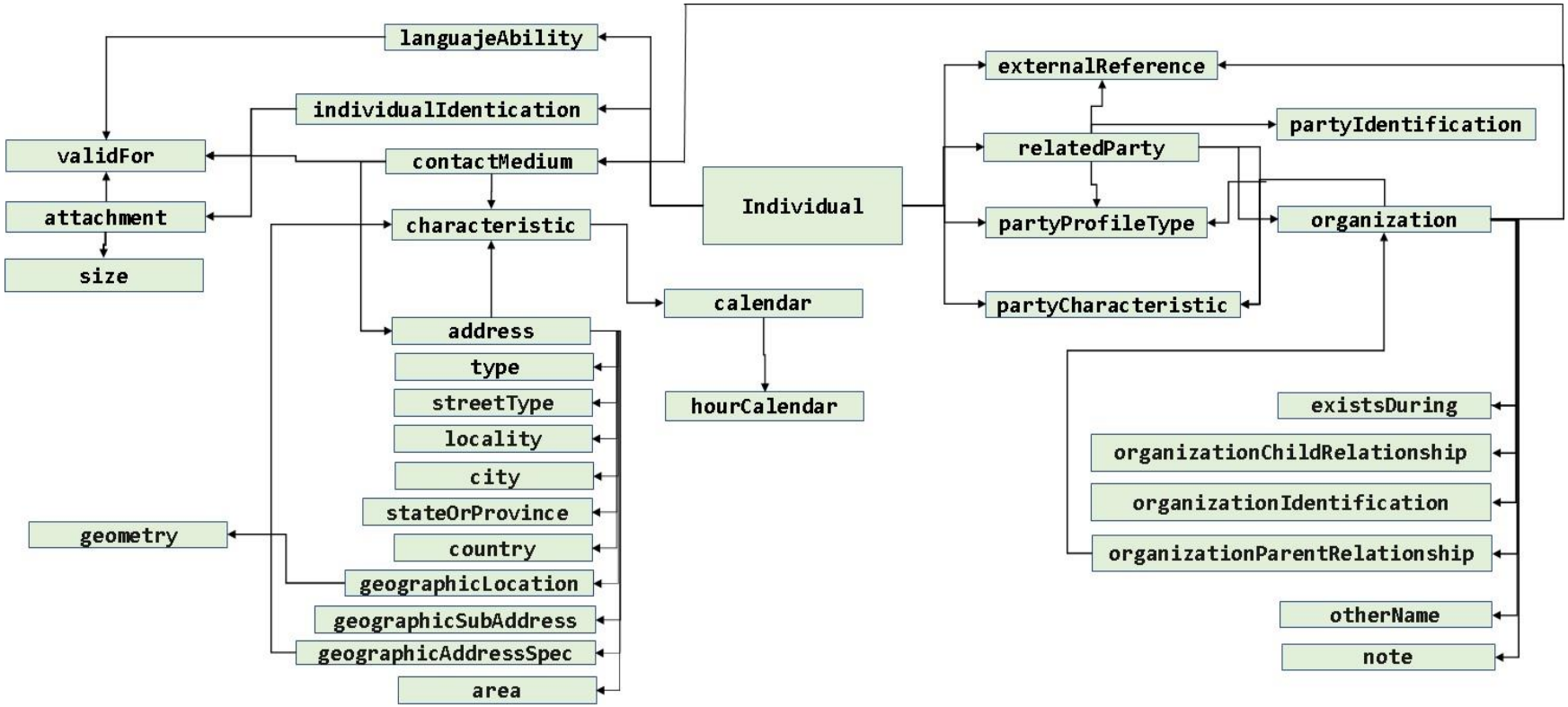

Figura 15. Modelo de datos del recurso *individual* descrito en el Anexo 1.

El recurso *individual* cuenta con atributos a nivel raíz que son id, href, birthDate, familyName, gender, givenName, maritalStatus, secondLastName, nationality, title, status, owner, createDateTime, lastUpdateDateTime, createdBy y lastUpdateBy. Los cuales en una estructura *json* tiene la forma del Listado 12.

```
{
    "id": "string",
    "href": "string",
    "birthDate": "2022-02-01T16:40:12.468Z",
    "familyName": "string",
    "gender": "string",
    "givenName": "string",
    "maritalStatus": "string",
    "secondLastName": "string",
    "nationality": "string",
    "title": "string",
    "status": "string",
    "owner": "string",
    "createDateTime": "2022-02-01T16:40:12.470Z",
    "lastUpdateDateTime": "2022-02-01T16:40:12.470Z",
    "createdBy": "string",
    "lastUpdateBy": "string"
}
```

Listado 12. Atributos a nivel raíz de Individual.

Además, cuenta con subrecursos como contactMedium, externalReference, individualIdentification, languageAbility, partyCharacteristic, relatedParty y partyProfileType. La representación de la estructura de un subrecuro puede observarse en el Listado 13.

```
{
    "id": "string",
    "href": "string",
    "birthDate": "2022-02-01T16:40:12.468Z",
    "familyName": "string",
    "gender": "string",
    "givenName": "string",
    "maritalStatus": "string",
    "secondLastName": "string",
    "nationality": "string",
    "title": "string",
    "status": "string",
    "owner": "string",
    "createDateTime": "2022-02-01T16:40:12.470Z",
    "lastUpdateDateTime": "2022-02-01T16:40:12.470Z",
    "createdBy": "string",
    "lastUpdateBy": "string",
    "languageAbility": [
      {
        "isFavouriteLanguage": true,
        "languageCode": "string",
        "languageName": "string",
        "listeningProficiency": "string",
```

```
        "readingProficiency": "string",
        "speakingProficiency": "string",
        "writingProficiency": "string",
        "validFor": {
          "endDateTime": "2022-02-01T16:40:12.469Z",
          "startDateTime": "2022-02-01T16:40:12.469Z"
        }
      }
    ]
}
```

Listado 13. Subrecurso languageAbility en estrcutura *json* de Individual.

A su vez, cada subrecurso puede contener otro subrecurso. De acuerdo con la Figura 15, el recurso *individual* tiene un subrecurso contactMedium, el cual tiene como subrecurso a characteristic, que a su vez cuentan con un subrecurso Calendar Period, el cual tiene un subrecurso HourPeriod. En el Listado 14 se observa cómo se puede representar esta estructura de subrecursos dentro de la estructura *json*, considerando los tributos de casa subrecurso.

```
{
    "id": "string",
    "href": "string",
    "birthDate": "2022-02-01T16:40:12.468Z",
    "familyName": "string",
    "gender": "string",
    "givenName": "string",
    "maritalStatus": "string",
    "secondLastName": "string",
    "nationality": "string",
    "title": "string",
    "status": "string",
    "owner": "string",
    "createDateTime": "2022-02-01T16:40:12.470Z",
    "lastUpdateDateTime": "2022-02-01T16:40:12.470Z",
    "createdBy": "string",
    "lastUpdateBy": "string",
    "contactMedium": [
      {
        "mediumType": "string",
        "preferred": true,
        "characteristic": {
          "city": "string",
          "contactType": "string",
          "country": "string",
          "emailAddress": "string",
          "faxNumber": "string",
          "phoneNumber": "string",
          "mobileNumber": "string",
          "postCode": "string",
          "webSite": "string",
          "stateOrProvince": "string",
          "street1": "string",
          "street2": "string",
          "calendar": [
```

```
            {
              "day": "string",
              "status": "string",
              "comment": "string",
              "timeZone": "string",
              "hourPeriod": [
                {
                  "endHour": "string",
                  "startHour": "string"
                }
              ]
            }
          ]
        }
    ]
}
```

Listado 14. Representación de subrecursos de un subrecurso en estructura *json* de Individual.

Cada recuadro del modelo del Figura 15 representa un nivel dentro de la estructura del *json*. El recuadro del recurso *individual* representa el nivel uno o nivel raíz en la estructura y en este nivel deben ir todos sus atributos como se muestra en el Listado 9. En el nivel dos empiezan a representarse los subrecursos, por ejemplo, languajeAbility. En el Listado 10 los atributos de este subrecurso van anidados.

Cada subrecurso de otro subrecurso representa un nivel anidado, como se muestra en el Listado 14, los atributos del subrecurso HourPeriod se encuentran anidados en un nivel 5.

El modelo de datos representa al recurso con todos sus subrecursos y todos los atributos de los mismos. Sin embargo, al momento de compartir información no es necesario que en la estructura *json* se consideren todos éstos, es mejor armar la estructura con los atributos que se cuentan o se requieren en ese momento, esto dependerá de cada desarrollo y necesidad de negocio.

Por otra parte, en el modelo de la Figura 15 se pueden observar subrecursos que hacen uso de otro subrecurso, en un mismo nivel, es decir, el recurso individual tiene como subrecurso *ExternalReference* y *relatedParty*, en el nivel dos. En un nivel tres relatedParty puede hacer uso del subrecurso ExternalReference. En el Listado 15 se muestra cómo sería la estructura *json* de acuerdo a esta particularidad del modelo.

```
{
    "id": "string",
    "href": "string",
    "birthDate": "2022-02-01T16:40:12.468Z",
    "familyName": "string",
    "gender": "string",
    "givenName": "string",
```

```
    "maritalStatus": "string",
    "secondLastName": "string",
    "nationality": "string",
    "title": "string",
    "status": "string",
    "owner": "string",
    "createDateTime": "2022-02-01T16:40:12.470Z",
    "lastUpdateDateTime": "2022-02-01T16:40:12.470Z",
    "createdBy": "string",
    "lastUpdateBy": "string",
    "relatedParty": [
      {
        "id": "string",
        "identifierType": "string",
        "href": "string",
        "name": "string",
        "role": "string",
        "partyRank": "string",
        "externalReference": [
         {
            "externalReferenceType": "string",
            "name": "string"
         }
        ]
      }
    ]
}
```

Listado 15. Representación de subrecurso a nivel de otro siendo anidado en la estructura json de Individual.

3.5.1.2 Recurso organization.

La Figura 16 muestra el modelo de datos que se utiliza para el recurso *organization* En el Anexo 2 se encuentra la estructura completa del json que representa el modelo, con todos los atributos y tipos de datos que se deben utilizar.

Organization cuenta con los atributos id, isHeadOffice, isLegalEntity, name, nameType, organizationType, tradingName, status, owner, createDateTime, lastUpdateDateTime y createdBy a nivel raíz y con los subrecursos contactMedium, externalReference, organizationChildRelationship, organizationIdentification, organizationParentRelationship, partyCharacteristic, relatedParty, partyProfileType y note. A su vez, estos subrecursos pueden tener otro subrecurso como se puede observar en la Figura 16.

La manera en la que se representan los subrecursos y atributos dentro de la estructura *json* es la misma que se explicó en el apartado 3.5.1.1 para el recurso *individual*.

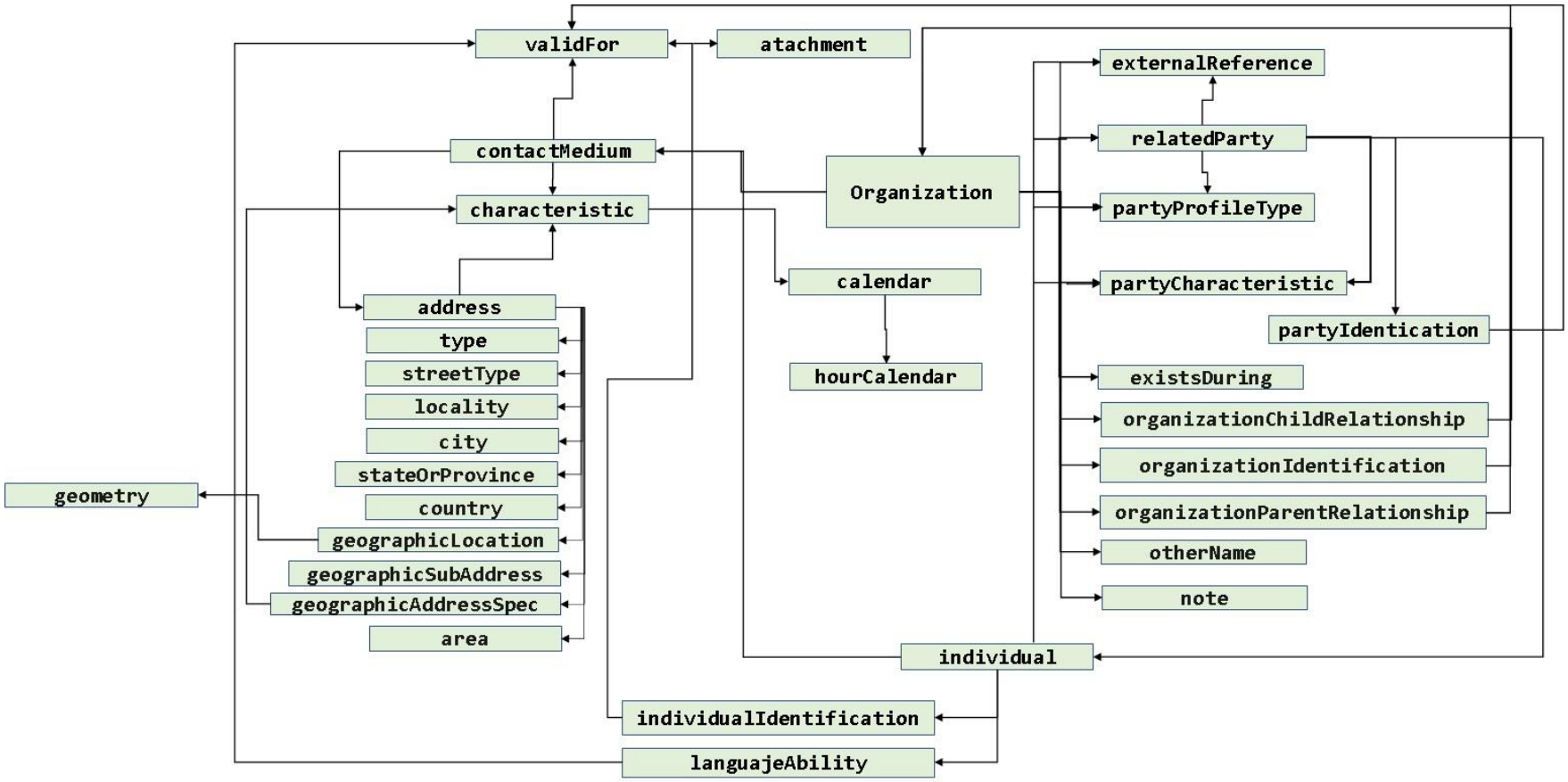

Figura 16. Modelo de datos de *organization* descrito en el Anexo 2.

Tanto en la Figura 15 como en la Figura 16, del modelo de datos de *individual* y *orgatization*, respectivamente, se puede observar que el subrecurso ContatcMedium tiene como subrecurso GeographicAddressRef, este a su vez tiene los subrecursos geographicLocation, geographicSubAddress, area, geographicAddressSpec y characteristic. Estos subrecursos permiten recuperar información sobre direcciones, por ejemplo, la dirección de un cliente. En el diseño original de las especificaciones de *PartyManagement* no existen estos subrecursos como tal. Sin embargo, existe un *API* llamada *Geographic Address Management* que es la que permite recuperar información sobre direcciones, esto significa, que se puede hacer uso de diferentes especificaciones de las *API* de *TM Forum*, usarlas en conjunto y crear una sola especificación de acuerdo a las necesidades de cada desarrollo.

En este caso en específico, el uso que se puede dar al combinar ambas *API* es validar si el cliente tiene una dirección asignada, en caso de no ser así, crear la dirección con la información del *json* en el sistema y asociarla a dicho cliente. La información debe existir en la estructura del *json* del recurso *individual* u *organization*. También se podría hacer solo uso de la *API Geographic Address Management* para realizar la manipulación de las direcciones como consultar, crear, y actualizar.

3.5.2 Especificación Geographic Address Management.

La *API Geographic Address Management* permite gestionar información de direcciones o lugares geográficos. Pueden ser direcciones de un Cliente, de un Contacto o de una Sucursal, de los cuales la organización debe almacenar su información para su uso posterior. En este caso se utiliza para guardar direcciones de los clientes o cuentas de la organización.

El recurso principal de esta *API* es *GeograficAdreesValidattion* con atributos como id, externalId, status, state, lastUpdateDateTime, createdBy, lastUpdateBy, @type y @schemaLocation. Sus subrecursos son GeographicAddress, charactristic, GeographicAddressSpec. A su vez GeographicAddress con sus subrecursos Area, geographicLocation, geographicSubAddress. Los niveles en los recursos y sub recurso se maneja de igual forma que en la *API PartyManagement* explicados en el recurso Individual.

La Figura 17 muestra el modelo de datos de *Geographic Address Management*. Como puede Observarse en esta *API* también existen subrecursos en un mismo nivel que pueden hacer uso de otro subrecurso, como es el caso del subrecurso validAddress que está en el mismo nivel que el subrecurso GeographicAddressSpec y en caso de ser necesario GeographicAddressSpec puede utilizarse en un nivel inferior.

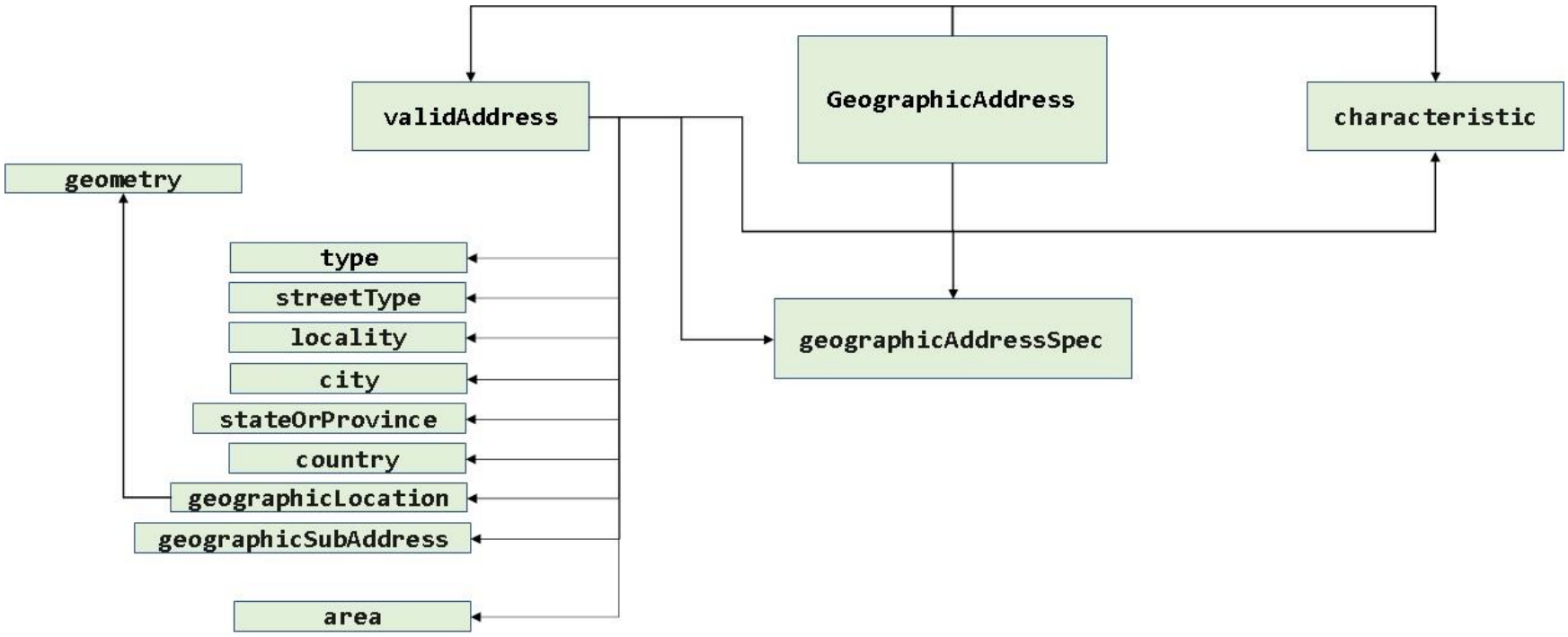

Figura 17. Modelo de datos de *GeograficAdrees*.

Como se mencionó anteriormente las especificaciones de la *API GeograficAdreesValidattion* se usan dentro de las especificaciones de *PartyManagement* para complementar la información de dirección del cliente. En el Listado 16 se muestra la estructura del *json* para poder enviar o recibir dicha información. La información debe de ir dentro del subrecurso contacMediumm de *individual* u *organization*.

```
{
  "contactMedium": [
    {
      "mediumType": "string",
      "preferred": true,
      "address": {
        "id": "string",
        "externalId": "string",
        "href": "string",
        "type": {
          "id": "string",
          "name": "string",
          "type": "string",
          "description": "string"
        },
        "streetNr": "string",
        "streetName": "string",
        "streetType": {
          "id": "string",
          "name": "string",
          "type": "string",
          "description": "string"
        },
        "postcode": "string",
        "locality": {
          "id": "string",
          "name": "string",
          "type": "string",
          "description": "string"
        },
        "city": {
          "id": "string",
          "name": "string",
          "type": "string",
          "description": "string"
        },
        "stateOrProvince": {
          "id": "string",
          "name": "string",
          "type": "string",
          "description": "string"
        },
        "country": {
          "id": "string",
          "name": "string",
          "type": "string",
          "description": "string"
        },
```

```
"@type": "string",
"@schemaLocation": "string",
"geographicLocation": {
  "id": "string",
  "href": "string",
  "name": "string",
  "geometryType": "string",
  "accuracy": "string",
  "spatialRef": "string",
  "@type": "string",
  "@schemaLocation": "string",
  "geometry": [
    {
      "x": "string",
      "y": "string",
      "z": "string"
    }
  ]
},
"geographicSubAddress": [
  {
    "id": "string",
    "type": "string",
    "name": "string",
    "description": "string",
    "subUnitType": "string",
    "subUnitNumber": "string",
    "levelType": "string",
    "levelNumber": "string",
    "buildingName": "string",
    "privateStreetNumber": "string",
    "privateStreetName": "string",
    "@type": "string",
    "@schemaLocation": "string"
  }
],
"geographicAddressSpec": [
  {
    "id": "string",
    "href": "string",
    "type": "string",
    "@type": "string",
    "@schemaLocation": "string",
    "characteristic": [
      {
        "id": "string",
        "name": "string",
        "valueType": "string",
        "value": "string"
      }
    ]
  }
],
"area": [
  {
    "id": "string",
    "name": "string",
```

```
            "type": "string",
            "description": "string"
          }
        ],
        "characteristic": [
          {
            "id": "string",
            "name": "string",
            "value": "string"
          }
        ],
        "owner": "string"
      }
    }
  ]
}
```

Listado 16. Representación de subrecurso *GeograficAdrees* dentro del subrecurso ContactMedium en la API PartyManagement.

El hecho de hacer uso de dos especificaciones de un *API* en una misma estructura ayuda a reducir código al momento del desarrollo, en este caso, se evita el uso de otra clase *wrappe*r para el llamado de la estructura de *GeograficAdreesValidattion*. Se pueden manejar las estructuras por separado y esto no tendrá ningún efecto negativo dentro del desarrollo, sólo impactaría en la cantidad de componentes y código dentro del mismo. En el Listado 17 se muestra la estructura del *json* para la *API Geographic Address Management*.

```
{
  "id": "string",
  "externalId": "string",
  "status": "string",
  "state": "string",
  "createdDateTime": "2022-02-01T23:28:27.181Z",
  "lastUpdateDateTime": "2022-02-01T23:28:27.182Z",
  "createdBy": "string",
  "lastUpdateBy": "string",
  "@type": "string",
  "@schemaLocation": "string",
  "validAddress": {
    "id": "string",
    "externalId": "string",
    "href": "string",
    "type": {
      "id": "string",
      "name": "string",
      "type": "string",
      "description": "string"
    },
    "streetNr": "string",
    "streetName": "string",
    "streetType": {
      "id": "string",
      "name": "string",
      "type": "string",
```

```
    "description": "string"
  },
  "postcode": "string",
  "locality": {
    "id": "string",
    "name": "string",
    "type": "string",
    "description": "string"
  },
  "city": {
    "id": "string",
    "name": "string",
    "type": "string",
    "description": "string"
  },
  "stateOrProvince": {
    "id": "string",
    "name": "string",
    "type": "string",
    "description": "string"
  },
  "country": {
    "id": "string",
    "name": "string",
    "type": "string",
    "description": "string"
  },
  "@type": "string",
  "@schemaLocation": "string",
  "geographicLocation": {
    "id": "string",
    "href": "string",
    "name": "string",
    "geometryType": "string",
    "accuracy": "string",
    "spatialRef": "string",
    "@type": "string",
    "@schemaLocation": "string",
    "geometry": [
      {
        "x": "string",
        "y": "string",
        "z": "string"
      }
    ]
  },
  "geographicSubAddress": [
    {
      "id": "string",
      "type": "string",
      "name": "string",
      "description": "string",
      "subUnitType": "string",
      "subUnitNumber": "string",
      "levelType": "string",
      "levelNumber": "string",
      "buildingName": "string",
```

```
      "privateStreetNumber": "string",
      "privateStreetName": "string",
      "@type": "string",
      "@schemaLocation": "string"
    }
  ],
  "area": [
    {
      "id": "string",
      "name": "string",
      "type": "string",
      "description": "string"
    }
  ],
  "geographicAddressSpec": [
    {
      "id": "string",
      "href": "string",
      "type": "string",
      "@type": "string",
      "@schemaLocation": "string",
      "characteristic": [
        {
          "id": "string",
          "name": "string",
          "value": "string",
          "valueType": "string"
        }
      ]
    }
  ],
  "characteristic": [
    {
      "id": "string",
      "name": "string",
      "value": "string",
      "valueType": "string"
    }
  ],
  "owner": "string"
},
"geographicAddressSpec": [
  {
    "id": "string",
    "href": "string",
    "type": "string",
    "@type": "string",
    "@schemaLocation": "string",
    "characteristic": [
      {
        "id": "string",
        "name": "string",
        "value": "string",
        "valueType": "string"
      }
    ]
  }
```

```
  ],
  "characteristic": [
    {
      "id": "string",
      "name": "string",
      "value": "string",
      "valueType": "string"
    }
  ]
}
```

Listado 17. Estructura *json* del *API Geographic Address Management*.

3.5.3 Definición detalle de la respuesta de la petición.

Toda *API* debe contar con la definición del detalle de la respuesta que es empleado para el manejo del código de estatus al interactuar con el recurso, con la finalidad de poder brindar un mayor detalle de la ejecución de las operaciones del mismo. Cuando surge algún error al momento de realizar una petición se debe contar con un mensaje descriptivo de tal error, para esto es que sirve el detalle de la respuesta. En la Figura 18 se puede observar el modelo de datos para dicho detalle.

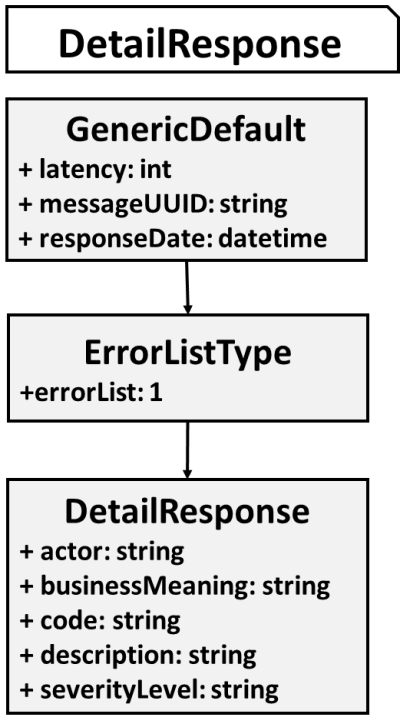

Figura 18. Modelo de detalles de la respuesta.

El Listado 18 muestra la estructura *json* en la que se debe enviar o recibir un mensaje de respuesta de error, considerando el modelo de la Figura 18, tomando en cuenta los recursos, subrecurso, atributos y tipos de datos del modelo.

```
{
        "messageUUID": "string",
        "responseDate": "2022-02-01T23:54:07.092Z",
        "latency": 0,
        "errorList": {
          "error": [
            {
              "code": "string",
              "severityLevel": "string",
              "description": "string",
              "actor": "string",
              "businessMeaning": "string"
            }

          ]
        }
}
```

Listado 18. Estructura *json* de mensaje de error.

3.6. Exponer servicios de Salesforce hacia un sistema externo.

Exponer un servicio significa que dentro de Salesforce se encuentran todos los componentes, en este caso, clases necesarias para que cualquier sistema externo realice alguna petición hacia Salesforce y éste envíe una respuesta.

Para ejemplificar como un sistema externo consume servicios disponibles en Salesforce se puede hacer uso de la herramienta *Postman* o *workbench*, las cuales nos permiten realizar las peticiones como si se estuvieran haciendo desde cualquier otro sistema y ver los resultados de las misma.

Se realiza la exposición del servicio PartyManagement Organización, el cual permite consultar, crear o actualizar desde un sistema externo el objeto Account(clientes) dentro de Salesforce, usando los métodos *GET*, *POST* y *PATCH*; mediante *Postman*.

Primero se debe de obtener el *token* para que se autorice la petición en Salesforce, para ello es necesario contar con el usuario, contraseña, *cliente_id* y *client_secret*. Por ejemplo, usando los datos de la Tabla 13, se pueden configurar como parámetros en *Postman* para obtener el *token* de acceso hacia *Salesforce*, utilizando el método *POST*.

Tabla 13. Datos necesarios para obtener *token*.

Parámetro	Valor
grant_type	Password
client_id	3MVG9AzPSkglhtpsEf1EFWhN7zc2vPdy5s4PvP46WR8T.Y8kKXWlxzqrBWSXCugyP7liHsG6nPJNB1CyabP5V
client_secret	AFAEE48E2B60C90C880C916D72B9811690929628A3F76E127F9960E0B7450334
Password	Chile@2022
Username	conectividadtranscapacl@amx.com

Una vez que se configuran los parámetros en *Postman* se obtienen los valores que se observan en el Listado 19. El valor que se debe de tomar es el que se encuentra dentro del parámetro ***access_token*.**


```
{
    "access_token": "00D3F0000001PtB!AR8AQIfxE0ZwNvcis2gTDlxN9R58zUWCfsczrVoXRyTF
y3A6rktf5ujzQF3oN8gXi0qaaiGNyT57lDeBbZqq7HsjBwxHcJZ5",
    "instance_url": "https://transforma--transdev.my.salesforce.com",
    "id": "https://test.salesforce.com/id/00D3F0000001PtBUAU/0053F000005fTGIQA2",
    "token_type": "Bearer",
    "issued_at": "1640213422199",
    "signature": "TqCzLEzfnaKdqwGqgonHe70XsuLa3zSyIQsGnE/u6t0="
}
```


Listado 19. Valor *token* para conectarse a *Salesforce*.

Una vez obtenido el *token* se pueden realizar cualquiera de las peticiones que soporte el *API*. Por ejemplo, si se desea crear un cliente en *Salesforce* desde cualquier sistema externo se necesitan la *url* del recurso al que se va a acceder, el formato *json* con los datos necesarios del cliente, el token y el método *HTTP*, en este caso *POST*.

La *ur*l utilizada en este ejemplo es **https://transforma--transdev.my.salesforce.com/services/apexrest/party/organization**.

El token es el obtenido en el Listado 19. La estructura *json* del *body* que se envía en el *request* de la petición se muestra en el Listado 20.


```
{
  "id": "0000AAAA",
  "tradingName": "Cliente Prueba",
  "name": "Rexa 2.1",
  "status": " ",
  "lastUpdateBy": "JPLOPEZ",
  "createdBy": "CRIQUELME",
  "lastUpdateDateTime": "2020-07-22T14:46:44.095Z",
  "createDateTime": "2020-09-03T14:46:44.095Z",
  "organizationType": "Grandes Empresas",
  "owner": "CAROLINA.VIVANCO",
  "relatedParty": [
    {
      "role": "CL_Clientes",
      "partyRank": "Estratégico",
      "partyProfileType": [
        {
          "name": "Estrátegico (B)",
          "category": "SUBCATEGORIA"
        },
        {
          "name": "convergente",
          "category": "tipoCliente"
        }
      ],
      "name": "",
      "isLegalEntity": false,
      "identifierType": "customerId",
      "id": "0000AAwIua",
```

```
    "externalReference": [
      {
        "name": "",
        "externalReferenceType": "SGACustomerId"
      }
    ]
  }
],
"partyProfileType": [
  {
    "name": "Otras actividades",
    "id": "",
    "description": "",
    "category": "sector"
  },
  {
    "name": "Hoteles, campamentos y otros tipos de hospedaje",
    "id": "",
    "description": "",
    "category": "subSector"
  },
  {
    "name": "Corporativo",
    "id": "",
    "description": "",
    "category": "segmento"
  },
  {
    "name": "Multinacionales",
    "id": "",
    "description": "",
    "category": "SegmentoVenta"
  },
  {

    "name": "GE1 EMPRESAS RM",
    "id": "",
    "description": "",
    "category": "CanalVenta"
  },
  {
    "name": "Universidades",
    "id": "",
    "description": "",
    "category": "ActividadEconomica"
  },
  {
    "name": "Fundacion",
    "id": "",
    "description": "",
    "category": "GiroComercial"
  }
],
"partyCharacteristic": [
  {
    "value": "0",
    "name": "Numero de Empleados"
  },
  {
    "value": "Corporaciones",
    "name": "TipoContribuyente"
  }
],
```

```
"organizationIdentification": [
  {
    "validFor": {
      "startDateTime": "2020-07-22T14:46:44.095Z",
      "endDateTime": "2020-07-22T14:46:44.095Z"
    },
    "issuingDate": "2020-08-27T14:46:44.095Z",
    "issuingAuthority": "",
    "identificationType": "RUT",
    "identificationId": "6936073",
    "checkDigit": "7"
  }
],
"note": [
  {
    "text": "Cliente Pruebas Integracion",
    "date": "2020-08-22T14:46:44.095Z",
    "id": "",
    "author": "Ejecutivo"
  }
],
"contactMedium": [
  {
    "mediumType": "mainAddress",
    "characteristic": {
      "phoneNumber": ""
    },
    "address": {
      "id": "0123456789wIua",
      "externalId": "0123456789wIua",
      "streetNr": "1001wIua",
      "streetName": "ChapultepeC",
      "postcode": "78100",
      "owner": "string",
      "country": {
        "type": "country",
        "name": "CL",
        "id": "1"
      },
      "stateOrProvince": {
        "type": "state",
        "name": "Baca",
        "id": "9"
      },
      "city": {
        "type": "city",
        "name": "municipio norte",
        "id": "0987654321"
      },
      "locality": {
        "type": "locality",
        "name": "Concepcionada",
        "id": "3"
      },
      "type": {
        "type": "addressType",
        "name": "addressType",
        "id": "16",
        "description": ""
      },
      "streetType": {
        "type": "string",
        "name": "avenida",
```

```
    "id": "itipo_via"
  },
  "geographicLocation": {
    "spatialRef": "",
    "geometry": [
      {
        "y": "34.000000",
        "x": "21.000000"
      }
    ]
  },
  "geographicSubAddress": [
    {
      "id": "0987654321",
      "type": "block",
      "name": "ninosheroes"
    },
    {
      "id": "0987654321",
      "type": "building",
      "subUnitType": "tipoedificacion",
      "subUnitNumber": "Númerodependencia",
      "levelType": "Floor",
      "levelNumber": "2",
      "buildingName": "interior",
      "privateStreetNumber": "8"
    }
  ],
  "geographicAddressSpec": [
    {
      "type": "Factibilidad",
      "characteristic": [
        {
          "valueType": "",
          "value": "demo1",
          "name": "product",
          "id": ""
        },
        {
          "valueType": "",
          "value": "sfdc",
          "name": "technology",
          "id": ""
        },
        {
          "valueType": "",
          "value": "demo",
          "name": "priority",
          "id": ""
        },
        {
          "valueType": "",
          "value": "213",
          "name": "Atributo Cobertura",
          "id": ""
        }
      ]
    }
  ],
  "characteristic": [
    {
      "value": "Edificio Corporativo",
      "name": "ADDRESSASSOCIATEDREFERENCE",
```

```
            "id": ""
          },
          {
            "value": "S/C",
            "name": "economicSocialGroup",
            "id": "6"
          },
          {
            "value": "HP",
            "name": "RealEstateProject",
            "id": "16"
          },
          {
            "value": "0",
            "name": "autogeneration",
            "id": ""
          }
        ],
        "area": [
          {
            "type": "street",
            "name": "CALLE57",
            "id": "id_calle"
          }
        ]
      }
    },
    {
      "mediumType": "emailAddress",
      "characteristic": {
        "emailAddress": "contacto@miniso.cl.test"
      }
    },
    {
      "mediumType": "telephoneNumber",
      "characteristic": {
        "phoneNumber": "6716253482"
      }
    },
    {
      "mediumType": "mobileNumber",
      "characteristic": {
        "mobileNumber": "1829384756"
      }
    },
    {
      "mediumType": "webSite",
      "characteristic": {
        "mobileNumber": "www.rexa.cl"
      }
    }
  ]
}
```

Listado 20. Estructura *json* utilizada para crear un cliente en *Salesforce*.

Cuando el cliente se crea correctamente en *Salesforce*, el servicio devuelve un código de respuesta 201 y un *json* de respuesta o response de la petición en el que se encuentra el Id del cliente con el que fue creado en *Salesforce*. En el Listado 21 se muestra la estructura del mensaje de respuesta a la petición de creación.

```
{
    "tradingName": "Cliente Prueba",
    "lastUpdateDateTime": "2021-12-22T22:33:26.000Z",
    "id": "0013F00000mQNsMQAW",
    "createDateTime": "2021-12-22T22:33:26.000Z"
}
```

Listado 21. Estructura *json* del response al crear un cliente en *Salesforce*.

Dentro del *json* de respuesta se encuentra el id con el que el cliente fue creado en *Salesforce*, en este caso y de acuerdo al Listado 21 0013F00000mQNsMQAW, siendo este el Código del Cliente. Con este código se puede realizar la búsqueda del cliente para verificar que efectivamente se creó.

En algunas ocasiones el servicio puede devolver algún mensaje de error, esto en base a las reglas de negocio que se tienen dentro de *Salesforce* o por algún error técnico de conexión, por ejemplo, que no se encuentre activo el servidor.

Un error común que suele presentarse al crear la información es la duplicidad de la misma. Otro caso común es que se viole alguna regla de negocio, como, por ejemplo, que en campo con valores predeterminados se quiera almacenar un valor no considerado dentro de la lista. Pueden existir más escenarios de errores lo importante es que exista un mensaje en donde se describa porque se presenta el error. En el Listado 22 se presenta la estructura *json* con la cual se deben de reportar los errores que se presenten en cada petición, en este caso el error es por duplicidad de datos. Se obtiene un código de respuesta 500 con el mensaje **Insert failed. First exception on row 0; first error: DUPLICATES_DETECTED, Favor de verificar la dirección, ya existe dirección con esta información.: []"**

```
{
    "responseDate":"2021-12-22 16:52:50",
    "messageUUID":"9954e699-2482-43f3-b737-08661e71d2a1",
    "latency": 289,
    "errorList": {
        "error": [
            {
                "severityLevel": "3: Fatal (Error)",
                "description": "Class.INT_CuentaCL_Service.createCustomer: line 2
18, column 1\nClass.INT_PartyManagement_Rest.createCustomerCL: line 408, column 1
\nClass.INT_PartyManagement_Rest.partyPost: line 97, column 1",
                "code": "500",
                "businessMeaning": "Insert failed. First exception on row 0; firs
t error: DUPLICATES_DETECTED, Favor de verificar la dirección, ya existe direcció
n con esta información.: []"
            }
        ]
    }
}
```

Listado 22. Estructura *json* de mensaje de error por duplicidad de datos.

Una vez que se tiene creado el Cliente se puede consultar su información usando el método *GET*. En este caso, la consulta de un cliente se puede realizar de

dos maneras, por ID Externo de Cliente, que es el Id con el que el cliente se creó en el sistema externo, o por Identificador Tributario del Cliente. Para realizar la consulta se necesita la *url* con los parámetros necesarios y el *token*.

La *url* utilizada para consultar por Id Externo del Cliente:

https://transforma--transdev.my.salesforce.com/services/apexrest/party/organization?relatedParty.id=0000AAwIua&relatedParty.identifierType=customerId&relatedParty.role=custormer

La *url* para consultar por Identificador Tributario del Cliente:

https://transforma--transdev.my.salesforce.com/services/apexrest/party/organization?organizationIdentification.identificationId=6936073&organizationIdentification.checkDigit=7&organizationIdentification.identificationType=RUT

La consulta se puede realizar utilizando alguna de las dos *url* arriba mencionadas. Los parámetros utilizados para la consulta por Id Externo son relatedParty.id, relatedParty.identifierType y relatedParty.role.

relatedParty.id=0000AAwIua
relatedParty.identifierType=customerId
relatedParty.role=custormer

Los parámetros utilizados para la consulta por Identificador Tributario son organizationIdentification.identificationId, organizationIdentification.checkDigit y organizationIdentification.identificationType.

organizationIdentification.identificationId=6936073
organizationIdentification.checkDigit=7
organizationIdentification.identificationType=RUT

La consulta se realiza con el método *GET* configurando los parámetros en *Postman* y se obtiene un código de respuesta 200, lo que significa, que la consulta fue exitosa y se devuelve como respuesta el *json* representado en el Listado 23.


```
[
    {
        "tradingName": "Cliente Prueba",
        "relatedParty": [
            {
                "role": "custormer",
                "partyRank": "Estratégico",
                "partyProfileType": [
                    {
                        "name": "Convergente",
                        "category": "tipoCliente"
                    }
                ],
                "identifierType": "customerId",
                "id": "0000AAwIua",
```

```
            "externalReference": [
                {
                    "name": "0013F00000mQNsMQAW",
                    "externalReferenceType": "SalesForceCustomerId"
                }
            ]
        }
    ],
    "partyProfileType": [
        {
            "name": "Corporativo",
            "category": "segmento"
        },
        {
            "name": "Multinacionales",
            "category": "segmentoVenta"
        },
        {
            "name": "GE1 Empresas RM",
            "category": "canalVenta"
        },
        {
            "name": "Otras actividades",
            "category": "sector"
        },
        {
            "name": "Hoteles, campamentos y otros tipos de hospedaje",
            "category": "subSector"
        },
        {
            "name": "Universidades",
            "category": "actividadEconomica"
        },
        {
            "name": "Fundacion",
            "category": "GiroComercial"
        }
    ],
    "partyCharacteristic": [
        {
            "value": "0",
            "name": "Numero de Empleados"
        }
    ],
    "organizationType": "Grandes Empresas",
    "organizationIdentification": [
        {
            "identificationType": "RUT",
            "identificationId": "6936073"
        }
    ],
    "name": "Rexa 2.1",
    "lastUpdateDateTime": "2021-12-22T22:33:26.000Z",
    "id": "0013F00000mQNsMQAW",
    "createDateTime": "2021-12-22T22:33:26.000Z",
    "contactMedium": [
        {
            "preferred": true,
            "mediumType": "emailAddress",
            "characteristic": {
                "emailAddress": "contacto@miniso.cl.test"
            }
        },
```

```
{
    "preferred": true,
    "mediumType": "telephoneNumber",
    "characteristic": {
        "phoneNumber": "6716253482"
    }
},
{
    "preferred": true,
    "mediumType": "mobileNumber",
    "characteristic": {
        "mobileNumber": "1829384756"
    }
},
{
    "mediumType": "mainAddress",
    "address": {
        "streetType": {
            "type": "Avenida"
        },
        "streetNr": "1001wIua",
        "streetName": "ChapultepeC",
        "stateOrProvince": {},
        "postcode": "78100",
        "locality": {},
        "id": "0123456789wIua",
        "geographicSubAddress": [
            {
                "type": "building",
                "subUnitType": "Númerodependencia",
                "levelType": "Floor",
                "levelNumber": "2",
                "buildingName": "interior"
            },
            {
                "type": "block"
            }
        ],
        "geographicLocation": {
            "geometry": [
                {
                    "y": "34.000000",
                    "x": "21.000000"
                }
            ]
        },
        "geographicAddressSpec": [
            {
                "characteristic": [
                    {
                        "value": "demo1",
                        "name": "product"
                    },
                    {
                        "value": "sfdc",
                        "name": "technology"
                    },
                    {
                        "value": "demo1",
                        "name": "priority"
                    },
                    {
                        "value": "213",
```

```
                                    "name": "Atributo Cobertura"
                                }
                            ]
                        }
                    ],
                    "externalId": "0123456789wIua",
                    "city": {},
                    "characteristic": []
                }
            }
        ]
    }
]
```

Listado 23. Estructura *json* con información del Cliente consultado.

En caso de no encontrar información al momento de consultar se obtendrá un código de respuesta 404, con el mensaje "*Resource not found*": El Listado 24 representa el mensaje de error.

```
{
    "responseDate": "2021-12-23 12:53:43",
    "messageUUID": "cea1da85-60e2-4bab-a565-659df6db73a0",
    "latency": 80,
    "errorList": {
        "error": [
            {
                "severityLevel": "3: Fatal (Error)",
                "description": "Class.INT_CuentaCL_Helper.getAccounts: line 33, c
olumn 1\nClass.INT_CuentaCL_Service.getAccountsWrapper: line 98, column 1\nClass.
INT_PartyManagement_Rest.getCustomersCL: line 373, column 1\nClass.INT_PartyManag
ement_Rest.partyGet: line 46, column 1",
                "code": "404",
                "businessMeaning": "Resource not found"
            }
        ]
    }
}
```

Listado 24. Estructura *json* con mensaje de error cuando hay información en la consulta.

Si falta algún parámetro en la *url* se devuelve un código de respuesta 400, con el mensaje que se observa en el Listado 25, indicando que faltan parámetros obligatorios en la *url*.

```
{
    "responseDate": "2021-12-23 12:54:48",
    "messageUUID": "844bbc5a-445c-4ebe-b8b8-e6164e060442",
    "latency": 57,
    "errorList": {
        "error": [
            {
                "severityLevel": "3: Fatal (Error)",
                "description": "Not found the mandatory params",
                "code": "400",
                "businessMeaning": "Not found the mandatory params"
            }
        ]
    }
}
```

Listado 25. Estructura *json* con mensaje de error por falta de parámetros en la consulta.

En caso de faltar alguna diagonal en la *url* se obtiene un código de respuesta 404, con el mensaje de error presentado en el Listado 26.

```
[
    {
        "errorCode": "NOT_FOUND",
        "message": "Could not find a match for URL"
    }
]
```

Listado 26. Estructura *json* con mensaje de error por *url* errónea en la consulta.

Así mismo, se puede realizar la actualización de algún o algunos campos del cliente creado. La estructura *json* para modificar un cliente es la misma estructura con la que se creó, representada en el Listado 21. A manera de ejemplo se va a modificar la estructura del *json* para actualizar los campos Teléfono, Teléfono Móvil y Correo electrónico. Los valores que se tenían cuando se creó son los mostrados en el Listado 27.

```
    {
      "mediumType": "emailAddress",
      "characteristic": {
        "emailAddress": "contacto@miniso.cl.test"
      }
    },
    {
      "mediumType": "telephoneNumber",
      "characteristic": {
        "phoneNumber": "6716253482"
      }
    },
    {
      "mediumType": "mobileNumber",
      "characteristic": {
        "mobileNumber": "1829384756"
      }
    }
```

Listado 27. Estructura *json* con valores Teléfono, Teléfono Móvil y Correo electrónico al momento de crear el Cliente.

La actualización se debe realizar por Código de Cliente o por ID Externo del cliente, para ello al final de la *url* se debe agregar como parámetro alguno de estos dos códigos.:

Url para consultar un cliente utilizando como parámetro Id del mismo:
https://transforma--transdev.my.salesforce.com/services/apexrest/party/organization/0013F00000mQNsMQAW

Url para consultar un cliente utilizando como parámetro Id Externo del mismo:
https://transforma--transdev.my.salesforce.com/services/apexrest/party/organization/0000AAwIua

Para actualizar los valores se cambia la información en la estructura *json* de acuerdo al Listado 28:

```
{
  "mediumType": "emailAddress",
  "characteristic": {
    "emailAddress": "prueba.chile@miniso.cl"
  }
},
{
  "mediumType": "telephoneNumber",
  "characteristic": {
    "phoneNumber": "6716786990"
  }
},
{
  "mediumType": "mobileNumber",
  "characteristic": {
    "mobileNumber": "6789456723"
  }
}
```

Listado 28. Estructura *json* con valores Teléfono, Teléfono Móvil y Correo electrónico para actualizar el Cliente.

La actualización se realiza desde *Postman*, utilizando alguna de las dos *url* mencionadas. El método *PATCH* y el *json* del Listado 21, modificando los valores de los campos a actualizar.

Si la actualización del recurso se realiza correctamente se devuelve un código de respuesta 200 y la estructura *json* mostrada en el Listado 29, la cual contiene el Nombre del Cliente, el Id y a la Fecha de creación y actualización.

```
{
    "tradingName": "Cliente Prueba",
    "lastUpdateDateTime": "2021-12-23T19:14:00.000Z",
    "id": "0013F00000mQNsMQAW",
    "createDateTime": "2021-12-22T22:33:26.000Z"
}
```

Listado 29. Estructura *json* devuelta al actualizar un Cliente.

En caso de que el recurso a actualizar no se encuentre dentro de Salesforce se devuelve un código de respuesta 500 con el mensaje que se muestra en el Listado 30.


```
{
    "responseDate": "2021-12-23 13:18:58",
    "messageUUID": "8a3d5053-2390-4d1d-a8a8-fedbd4838175",
    "latency": 55,
    "errorList": {
        "error": [
            {
                "severityLevel": "3: Fatal (Error)",
                "description": "Class.INT_CuentaCL_Service.updateCustomer: line 4
63, column 1\nClass.INT_PartyManagement_Rest.updateCustomerCL: line 445, column 1
\nClass.INT_PartyManagement_Rest.partyPatch: line 158, column 1",
                "code": "500",
                "businessMeaning": "List has no rows for assignment to SObject"
            }
        ]
    }
}
```


Listado 30. Estructura *json* de error al no encontrar información que actualizar.

Para verificar que los datos se actualizaron correctamente se busca el cliente con el id 0013F00000mQNsMQAW dentro de *Salesforce*, el cual se obtiene del *json* de respuesta del Listado 29.

Todas las peticiones de consulta, creación o actualización de los recursos en *Salesforce* son transparentes para los usuarios, es decir, generalmente no saben en qué momento ocurren ni cómo funcionan técnicamente, pero pueden beneficiarse de los resultados obtenidos.

3.7. Consumir servicios externos desde Salesforce.

Consumir un servicio significa que desde *Salesforce* se realizan peticiones hacia el sistema externo y éste nos devuelve una respuesta. Para mostrar cómo se consume un servicio se usa el *API PartyManagement* con el recurso *individual*, que permite consultar, crear o actualizar contactos.

Dentro de *Salesforce* se utilizan contactos para almacenar información acerca de personas. Los contactos se asocian habitualmente con una cuenta o cliente, aunque, también se pueden asociar con otros registros como oportunidades.

El flujo definido para consumir el servicio de contactos es:

1.- Se crea un contacto en *Salesforce* y se guarda.

2.- Al momento de guardar el contacto se dispara la petición de consulta hacia el sistema externo. El campo Id Externo de contacto al momento de crear se encuentra vacío.

3.- Si dentro del response o respuesta del sistema externo indica que ya existe el contacto, se actualiza el *Salesforce* el ID Externo del contacto, en caso de no existir en el sistema externo, se dispara la petición de crear contacto.

4.- Se recibe la respuesta del sistema externo, en caso de ser exitosa, se actualiza el ID Externo del Contacto, en caso contrario, se actualiza el campo Respuesta Servicio.

5.- Una vez que el contacto está creado, se pueden actualizar campos y guardar la información, en ese momento se dispara la petición de actualización hacia el sistema externo. Sólo en caso de que el response no sea exitoso se actualiza el campo Respuesta Servicio para almacenar el error.

Cabe aclarar que, al momento de crear un contacto, éste debe estar asociado a un cliente, el cual debe existir tanto en *Salesforce* como en el sistema externo. Si se relaciona a un cliente no existente en el sistema externo se obtendrá un error.

Una vez creado el contacto se dispara la petición hacia el sistema externo, en donde, la primera acción a realizar es la consulta del contacto por medio del Id externo del mismo. En el método *Get* se utiliza únicamente la *url* para realizar el *request*, utilizando los parámetros relatedParty.id, relatedParty.role, relatedParty.identifierType e id.

https://APIservicedesa.clarochile.cl:7006/PartyManagement/tmf-API/party/v1/individual?relatedParty.id=03368699&relatedParty.role=customer&relatedParty.identifierType=customerId&id=null

Los valores de los parámetros para realizar la consulta del contacto en el sistema externo o legado son:

relatedParty.id=03368699
relatedParty.role=customer
relatedParty.identifierType=customerId
id=null

Al realizar la consulta se obtiene como código de respuesta 404, que significa que no se encuentra el contacto en el sistema externo. El *json* del response del sistema externo se muestra en el Listado 31.

```
{
  "messageUUID": "e2835afe-56c7-bf9e-d1e9-6c6faae50a9c",
  "responseDate": "2022-01-04T16:35:43.024Z",
  "latency": 0,
  "errorList": {
    "error": [
      {
```

```
        "code": "100",
        "severityLevel": "3",
        "description": "Not Found",
       "businessMeaning": "Error: SF_PROC_ConsultaContacto, No se encontro en
      VTATABCLI cliente de codigo : null"
      }
    ]
  }
}
```


Listado 31. Estructura *json* de error al no encontrar información de un Contacto en el sistema externo.

Como no se encuentra el contacto se ejecuta inmediatamente el método *POST* para su creación en el sistema externo. Para poder crearse el contacto es necesario la *url* y el *body* del *request*. La *Url*, https://APIservicedesa.clarochile.cl:7006/PartyManagement/tmf-API/party/v1/individual

El Listado 32 muestra la estructura *json* del *request* hacia el sistema externo para la creación del contacto.


```
{
  "title": "Catedrático",
  "secondLastName": "Fuentes Perez",
  "relatedParty": [
    {
      "role": "customer",
      "identifierType": "customerId",
      "id": "03368699",
      "externalReference": [
        {
          "name": "0011g00000xYXEnAAO",
          "externalReferenceType": "SalesForceCustomerId"
        }
      ]
    }
  ],
  "partyProfileType": [
    {
      "name": "CL - Contactos",
      "id": "0125A000001RdBnQAK",
      "category": "TipoRegistro"
    }
  ],
  "partyCharacteristic": [
    {
      "value": "USD",
      "name": "Divisa"
    },
    {
      "value": "Comercial",
      "name": "Cargo"
    },
    {
      "value": "false",
      "name": "hasBiometrics"
    }
  ],
  "owner": "00032020",
  "nationality": "Chilena",
```

```
"lastUpdateBy": "00032020",
"individualIdentification": [
  {
    "identificationType": "RUT",
    "identificationId": "17899516",
    "checkDigit": "2"
  }
],
"givenName": "Jose",
"gender": "Masculino",
"familyName": "Fuentes Perez",
"externalReference": [
  {
    "name": "0031g00000r4tOPAAY",
    "externalReferenceType": "SalesForceIndividualId"
  }
],
"createdBy": "00032020",
"contactMedium": [
  {
    "preferred": true,
    "mediumType": "Contacto",
    "address": {
      "streetType": {},
      "stateOrProvince": {},
      "locality": {
        "type": "locality"
      },
      "geographicSubAddress": [
        {
          "type": "building",
          "levelType": "Floor"
        },
        {
          "type": "block"
        }
      ],
      "geographicLocation": {
        "geometry": [
          {}
        ]
      },
      "geographicAddressSpec": [
        {
          "type": "Factibilidad",
          "characteristic": [
            {
              "value": "null",
              "name": "Product"
            },
            {
              "value": "BIk",
              "name": "technology"
            },
            {
              "value": "null",
              "name": "priority"
            }
          ]
        }
      ],
      "city": {
        "type": "city"
```

```
        },
        "characteristic": []
      }
    },
    {
      "preferred": true,
      "mediumType": "emailAddress",
      "characteristic": {
        "emailAddress": "j.fuentes@claro.com"
      }
    },
    {
      "preferred": true,
      "mediumType": "telephoneNumber",
      "characteristic": {
        "phoneNumber": "5634567091"
      }
    },
    {
      "preferred": true,
      "mediumType": "mobileNumber",
      "characteristic": {
        "mobileNumber": "5634567092"
      }
    },
    {
      "preferred": true,
      "mediumType": "AttentionTime",
      "characteristic": {
        "calendar": [
          {
            "hourPeriod": [
              {
                "startHour": "10:00",
                "endHour": "12:00"
              }
            ],
            "day": "Miércoles",
            "comment": "Vespertino"
          }
        ]
      }
    }
  ],
  "birthDate": "1980-01-08T00:00:00.000Z"
}
```


Listado 32. Estructura *json* enviada al sistema externo para la creación de un Contacto.

Si se crea correctamente el contacto en el sistema externo se obtiene el response de la petición mostrado en el Listado 33. El código de respuesta a la petición es 201.

```
{
  "id": "03025563",
  "relatedParty": [
    {
      "id": "03368699",
      "identifierType": "customerId",
      "role": "customer"
    }
  ]
}
```

Listado 33. Estructura *json* obtenida del sistema externo al momento de crear un Contacto.

Por lo tanto, en *Salesforce* se actualiza el campo Id Externo del Contacto con el valor 03368699 que se encuentra en la respuesta del sistema externo del Listado 33. Este id o código permite tener la información consistente entre *Salesforce* y el sistema externo.

En caso de que exista algún error al momento de la creación, en el response del servicio se indicará cual es el problema. El campo Id Externo del Contacto no se actualiza, se actualiza el campo Respuesta del Servicio para mostrar que existe un error en el sistema Externo.

El Listado 34 muestra un ejemplo de error, al realizar la petición el servicio se obtiene un código de respuesta 409, que significa conflicto, en este caso, se envía un campo con longitud mayor a lo que esperaba el sistema externo, debido a ello no puede almacenar la información y devuelve un error como respuesta.

```
{
  "messageUUID": "2fb7d206-743e-fb26-694c-bf1979183b86",
  "responseDate": "2022-01-04T17:06:31.85Z",
  "latency": 2,
  "errorList": {
    "error": [
      {
        "code": "409",
        "severityLevel": "3",
        "description": "Internal server Error",
        "businessMeaning": "Problema interno con servicio
cl.clarochile.openAPI.partymanagement.dao.IndividualDAO.executeSFPROCMainContacto
: ERROR En SF_PROC_MainContacto ORA-12899: value too large for column
\"MARKETING\".\"SF_INT_CONTACTO\".\"CODCLI\" (actual: 10, maximum: 8)"
      }
    ]
  }
}
```

Listado 34. Estructura *json* de error obtenida al no crearse el Contacto en el sistema externo.

Si se realiza una consulta a un contacto que ya existe tanto en *Salesforce* como en el sistema externo, se obtiene como código de respuesta 200. Como el contacto ya tiene un Id Externo, esta petición es completamente imperceptible para el usuario, aparentemente no se realiza ninguna acción, sin embargo, se actualiza

el campo Id Externo, esto con el objetivo de que en determinado momento si en el sistema externo por alguna razón se actualizó se homologuen los valores con *Salesforce*. Para realiza la consulta se usa la url:
https://APIservicedesa.clarochile.cl:7006/PartyManagement/tmf-API/party/v1/individual?relatedParty.id=03367868&relatedParty.role=customer&relatedParty.identifierType=customerId&id=03025561

La estructura *json* del response del sistema externo al momento de realizar la consulta es como la del Listado 35.


```
[
  {
    "id": "03025561",
    "birthDate": "2000-10-15T03:00:00Z",
    "familyName": "CONTACT CHILE A",
    "gender": "Masculino",
    "givenName": "PRUEBA A",
    "maritalStatus": "Otro",
    "secondLastName": "CONTACT CHILE A",
    "nationality": "chilena",
    "title": "Catedrático",
    "contactMedium": [
      {
        "mediumType": "Contacto",
        "address": {
          "type": {},
          "streetNr": "203",
          "streetName": "Arco 2 BCD 034",
          "streetType": {
            "name": "Calle"
          },
          "postcode": "78100",
          "locality": {
            "type": "locality"
          },
          "city": {
            "type": "city"
          },
          "stateOrProvince": {
            "type": "state"
          },
          "country": {
            "type": "country"
          },
          "geographicLocation": {
            "geometry": [
              {}
            ]
          },
          "geographicSubAddress": [
            {
              "type": "building",
              "subUnitNumber": "88"
            }
          ],
          "geographicAddressSpec": [
            {
              "type": "Factibilidad",
              "characteristic": [
                {}
              ]
```

```
          }
        ],
        "area": [
          {
            "type": "street"
          }
        ],
        "characteristic": [
          {
            "name": "AddressAssociatedReference"
          },
          {
            "name": "economicSocialGroup"
          },
          {
            "name": "RealEstateProject"
          },
          {
            "name": "autogeneration"
          }
        ]
      }
    },
    {
      "mediumType": "emailAddress",
      "characteristic": {
        "emailAddress": "prueba.chile@gmail.com"
      }
    },
    {
      "mediumType": "telephoneNumber",
      "characteristic": {
        "phoneNumber": "5623456789"
      }
    },
    {
      "mediumType": "mobileNumber",
      "characteristic": {
        "mobileNumber": "5623456780"
      }
    },
    {
      "mediumType": "webSite",
      "characteristic": {}
    },
    {
      "mediumType": "AttentionTime",
      "characteristic": {
        "calendar": [
          {
            "day": "Miércoles",
            "comment": "Vespertino",
            "hourPeriod": [
              {
                "endHour": "21:00",
                "startHour": "17:15"
              }
            ]
          }
        ]
      }
    }
  ],
```

```
    "externalReference": [
      {
        "externalReferenceType": "SalesForceIndividualId",
        "name": "0033F00000ZXoWsQAL"
      }
    ],
    "individualIdentification": [
      {
        "identificationId": "12870631",
        "identificationType": "RUT",
        "checkDigit": "3"
      }
    ],
    "languageAbility": [
      {
        "languageName": "Español"
      }
    ],
    "partyCharacteristic": [
      {
        "name": "Divisa",
        "value": "CLP"
      },
      {
        "name": "Cargo",
        "value": "Comercial"
      },
      {
        "name": "DependeDe"
      }
    ],
    "relatedParty": [
      {
        "id": "03367868",
        "identifierType": "customerId",
        "role": "customer",
        "externalReference": [
          {
            "externalReferenceType": "SalesForceCustomerId",
            "name": "0013F00000O3YQ0QAN"
          }
        ]
      }
    ],
    "status": "Activo",
    "partyProfileType": [
      {
        "id": "54",
        "description": "Aviso de Cobranza",
        "name": "TipoRegistro",
        "category": "TipoRegistro"
      }
    ],
    "owner": "00032020",
    "lastUpdateDateTime": "2022-01-14T17:11:26Z",
    "createdBy": "00032020",
    "lastUpdateBy": "00032020"
  }
]
```


Listado 35. Estructura *json* del *response* del sistema externo al consultar un Contacto.

Para actualizar un contacto, la estructura del *body* de la petición es exactamente igual a la que se usa para la creación, sólo es necesario considerar los nuevos valores. Es importante recordar que para poder realizar la actualización de un contacto en el sistema externo en *Salesforce* debe existir informado el campo Id Externo debido a que esta es la llave de búsqueda en el sistema legado. La petición de la actualización se realizará al momento de dar clic en el botón guardar.

Una vez que se guarda la información en *Salesforce* se ejecuta la petición de actualización hacia el sistema externo usando el método *PATCH*, para ello se usa la siguiente *url*, la cual contiene como parámetro el Id Externo del contacto con el que se identifica en dicho sistema:

https://APIservicedesa.clarochile.cl:7006/PartyManagement/tmf-API/party/v1/individual/03025563

La estructura *json* del *request* de la petición se muestra en el Listado 36.

```
{
  "title": "Catedrático",
  "secondLastName": "Fuentes Perez",
  "relatedParty": [
    {
      "role": "customer",
      "identifierType": "customerId",
      "id": "03368699",
      "externalReference": [
        {
          "name": "0011g00000xYXEnAAO",
          "externalReferenceType": "SalesForceCustomerId"
        }
      ]
    }
  ],
  "partyProfileType": [
    {
      "name": "CL - Contactos",
      "id": "0125A000001RdBnQAK",
      "category": "TipoRegistro"
    }
  ],
  "partyCharacteristic": [
    {
      "value": "USD",
      "name": "Divisa"
    },
    {
      "value": "Comercial",
      "name": "Cargo"
    },
    {
      "value": "false",
      "name": "hasBiometrics"
    }
  ],
  "owner": "00032020",
  "nationality": "Chilena",
  "lastUpdateBy": "00032020",
  "individualIdentification": [
    {
```

```
    "identificationType": "RUT",
    "identificationId": "17899516",
    "checkDigit": "2"
  }
],
"id": "03025563",
"givenName": "Jose",
"gender": "Masculino",
"familyName": "Fuentes Perez",
"externalReference": [
  {
    "name": "0031g00000r4tOPAAY",
    "externalReferenceType": "SalesForceIndividualId"
  }
],
"createdBy": "00032020",
"contactMedium": [
  {
    "preferred": true,
    "mediumType": "Contacto",
    "address": {
      "streetType": {},
      "stateOrProvince": {},
      "locality": {
        "type": "locality"
      },
      "geographicSubAddress": [
        {
          "type": "building",
          "levelType": "Floor"
        },
        {
          "type": "block"
        }
      ],
      "geographicLocation": {
        "geometry": [
          {}
        ]
      },
      "geographicAddressSpec": [
        {
          "type": "Factibilidad",
          "characteristic": [
            {
              "value": "null",
              "name": "Product"
            },
            {
              "value": "BIk",
              "name": "technology"
            },
            {
              "value": "null",
              "name": "priority"
            }
          ]
        }
      ],
      "city": {
        "type": "city"
      },
      "characteristic": []
```

```
      }
    },
    {
      "preferred": true,
      "mediumType": "emailAddress",
      "characteristic": {
        "emailAddress": "j.fuentes@claro.com"
      }
    },
    {
      "preferred": true,
      "mediumType": "telephoneNumber",
      "characteristic": {
        "phoneNumber": "5634567091"
      }
    },
    {
      "preferred": true,
      "mediumType": "mobileNumber",
      "characteristic": {
        "mobileNumber": "56345673478"
      }
    },
    {
      "preferred": true,
      "mediumType": "AttentionTime",
      "characteristic": {
        "calendar": [
          {
            "hourPeriod": [
              {
                "startHour": "10:00",
                "endHour": "12:00"
              }
            ],
            "day": "Martes",
            "comment": "Vespertino"
          }
        ]
      }
    }
  ],
  "birthDate": "1980-01-08T00:00:00.000Z"
}
```


Listado 36 Estructura *json* del *request* enviada al sistema externo para actualizar información de un contacto.

Si la actualización se realiza de manera exitosa en el sistema externo se obtiene como código de respuesta 200 y la estructura j*son* mostrada en el Listado 37, la cual contiene el Id Externo del Contacto y el Id Externo del Cliente al que se encuentra ligado el contacto.

```
{
  "id": "03025563",
  "relatedParty": [
    {
      "id": "03368699",
      "identifierType": "customerId",
      "role": "customer"
    }
  ]
}
```

Listado 37. Estructura *json* de response del sistema externo al momento de actualizar un Contacto.

En dado caso que se desee actualizar un contacto que no existe en el sistema externo se obtendrá un código de 404, además, se actualiza el campo Respuesta del Servicio para guardar el error que se obtiene de respuesta.

Usando el Id Externo 00025565 que no existe en el sistema legado se envía la petición con la *url*:

https://APIservicedesa.clarochile.cl:7006/PartyManagement/tmf-API/party/v1/individual/00025565

Se obtiene como resta del sistema externo la estructura *json* del Listado 38 indicando que no existe información.

```
{
  "messageUUID": "b1048e7c-216b-27f8-5dca-866513863e4c",
  "responseDate": "2022-01-04T19:15:02.821Z",
  "latency": 0,
  "errorList": {
    "error": [
      {
        "code": "100",
        "severityLevel": "3",
        "description": "Not Found",
       "businessMeaning": "Error: SF_PROC_ModificaContacto En VTATABCNTCLI
      NO existe Contacto de Codigo : 00025565"
      }
    ]
  }
}
```

Listado 38. Estructura *json* de error enviada por el sistema externo al momento de actualizar un contacto que no se encuentra.

Resumiendo, al consumir un servicio desde *Salesforce* hacia un sistema externo o sistema legado, se pueden utilizar los métodos *GET, POST y PATCH*. En cada petición que se hace por servicio ésta puede ser exitosa o no, es decir, si una petición es exitosa se obtiene la información esperada, en caso contrario, se obtiene un mensaje de error. El método *GET* y *PATCH* se pueden considerar exitosos cuando se obtiene un código de respuesta 200, para los demás códigos se obtiene un mensaje de error. Para el método *POST* se obtiene un código de respuesta 201 cuando es exitoso. De acuerdo a esto, se podría decir que los códigos de respuesta

exitosos son 200 y 201, todo los demás son códigos de respuesta de error. Los códigos de respuesta fueron definidos en la sección 2.1.4 de este trabajo.

Los mensajes de error varían de acuerdo al código de respuesta y estos son definidos por el sistema externo. Se deben considerar los códigos de respuesta definidos en el contrato del *API* utilizada

Conclusiones.

En el presente trabajo se ha documentado cómo se pueden integrar varios sistemas de software con diferentes arquitecturas, de tal manera que, conviviendo en conjunto puedan comunicarse y compartir información entre ellos. Para ello se utilizó como software central o principal *Salesforce* el cual puede comunicarse con diferentes sistemas externos a través de servicios web y *Open APIs* para poder compartir información entre ellos.

Después del análisis se concluye que se cumple el objetivo de esta tesina porque efectivamente mediante el uso de *Open API*s desarrolladas por *TM Forum* se establece una comunicación mediante el envío o recepción de peticiones *HTTP* en formato *json*. Previamente se tienen que configurar los metadatos de *Salesforce* con todos los datos de autentificación y las direcciones *IP* a las que se tiene acceso, sino se realiza dicha configuración no se logra la comunicación entre aplicaciones.

Dados los avances de tecnología que existen en estos momentos, es viable utilizar un modelo de comunicación como el descrito en este trabajo, es decir, hacer uso de *Salesforce* basada en la nube para poder centralizar toda la información de la organización en tiempo real a través de diferentes canales de comunicación lo que permitirá tenerla disponible en cualquier momento. Lo podrán utilizar empresas que cuenten con sistemas construidos en diferentes plataformas y lenguajes de programación que deseen integrar su información. Por otra parte, si la empresa tiene diferentes ubicaciones geográficas esto podrá ser de gran ayuda para integrar todos sus sistemas.

Las *APIs* juegan un papel importante en la transformación digital de cualquier organización, ya que, a través de ellas se integran o comunican los sistemas sin importar lenguajes de programación o sistemas operativos, trabajando de tal manera que puedan interactuar con aplicaciones web o móviles, agilizando procesos en cualquier área y ayudando a mejorar la toma de decisiones.

Glosario.

ARIN: *American Registry of Internet Numbers*. Organización en Estados Unidos que gestiona las direcciones IP del país, y sus territorios asignados. Debido a que las direcciones en Internet deben de ser únicas, y los espacios de direcciones en internet son limitados, es necesaria una organización que controle y asigne los bloques numéricos (Wikipedia, 2021).

Apex: Lenguaje de programación utilizado en Salesforce (*Trailhead Salesforce*, 2021).

Apex REST API: Es lo mismo que *API REST*, pero se utiliza para acceder a los métodos y clases de Apex de modo que las aplicaciones externas puedan acceder al código a través de la arquitectura *REST* (*Trailhead Salesforce*, 2021).

API: *Application Programming Interface* o Interfaz de Programación de Aplicaciones permite establecer una comunicación con una base de datos, un sistema operativo o un protocolo de comunicaciones (Definición DE, 2021).

API REST: Es la interfaz de programación de aplicaciones de transferencia de estado representacional, más comúnmente conocida como servicio web *REST API*. Significa que cuando se llama a una *API REST*, el servidor transfiere información de los recursos solicitados al cliente (*Trailhead Salesforce*, 2021).

APNIC: *Asia Pacific Network Information Centre,* es el registro regional de direcciones de Internet para la región Asia-Pacífico, proporciona servicios de registro y asignación de recursos numéricos que respaldan el funcionamiento global de Internet (Wikipedia, 2021).

B2B: *Business to Business*, que significa de negocio a negocio, es un modelo de transmisión de información en la red relacionado con las transacciones comerciales que las empresas realizan (Sánchez, 2015).

B2B2C: *Business to Business to Costumer*, que significa Empresa a Empresa a Cliente, es un modelo en el que la empresa accede al cliente final mediante otro negocio, en el que además los clientes finales reconocen la marca del prestador del servicio (Wikipedia, 2021).

CRUD: hace referencia a *Create*, *Read*, *Update* y *Delete*. Resume las funciones requeridas por un usuario para crear y gestionar datos (*Salesforce Developers*, 2021).

Cloud Computing: computación en la nube que ofrece servicios a través de la conectividad y gran escala de Internet (*Salesforce*, 2021).

CRM: *Customer Relationship Management*, Gestión de relaciones con el Cliente (*Salesforce*, 2021).

Endpoint: Es un tipo de nodo de red de comunicación. Es un dispositivo informático remoto que se comunica a través de una red a la que está conectado, por ejemplo, un servidor, un teléfono, un pc o una tablet (Pons, 2021).

HTTP: *Hypertext Transfer Protocol*, traducida a nuestro idioma como Protocolo de Transferencia de Hipertexto. Se trata de un protocolo de comunicación que posibilita la circulación de información a través de la *World Wide Web* (www) mediante un esquema cliente-servidor (Wikipedia, 2021).

JSON: *JavaScript Object Notation* o Notación de Objeto de JavaScript es un formato ligero de intercambio de datos, que resulta sencillo de leer y escribir para los programadores y simple de interpretar y generar para las máquinas (Wikipedia, 2021).

Metadatos: son datos que describen otros datos. En general, un grupo de metadatos se refiere a un grupo de datos que describen el contenido informativo de un objeto al que se denomina recurso (*Salesforce*, 2021).

OAS: *Open API Specification* define una descripción de interfaz estándar, independiente del lenguaje de programación para las *API REST* (*Swagger*, 2021).

OAuth: *Open Authorization* es un protocolo para pasar la autorización de un servicio a otro sin compartir las credenciales de usuario reales, como un nombre de usuario y contraseña. Con esta herramienta, un usuario puede iniciar sesión en una plataforma y luego estar autorizado para realizar acciones y ver datos en otra plataforma (López, 2020).

On Demand: es un sistema en el que los usuarios pagan por un servicio solo cuando hacen uso de este, lo que permite a los clientes manejar de manera más eficiente sus gastos y sin la obligación de tener un costo fijo que deban pagar (Ventura, 2021).

On Premiss: en español significa en instalaciones propias, se refiere a utilizar servidores y software propio de la empresa (Ventura, 2021).

Open API: es un estándar para la descripción de las interfaces de programación, puede utilizarse para describir, desarrollar, probar y documentar las APIs compatibles con REST (*Swagger*, 2021).

PaaS: *Platform as Service* o plataforma como servicio es un conjunto de servicios basados en la nube que permite a los desarrolladores y usuarios empresariales crear aplicaciones a una velocidad que las soluciones en las instalaciones no pueden alcanzar. Al tratarse de un servicio basado en la nube, no hay necesidad de preocuparse por la configuración y el mantenimiento de servidores, parches, actualizaciones y autenticaciones, entre muchas otras tareas: los usuarios pueden centrarse en crear la mejor experiencia de usuario posible (*Salesforce*, 2021).

Path: Es conocido como la dirección *URL* que aparece en la que se encuentra un determinado recurso (Wikipedia, 2021).

REST: *Representational State Transfer* o transferencia de estado representacional es una interfaz para conectar varios sistemas basados en el protocolo HTTP, sirve para obtener y generar datos, devolviendo esos datos en formatos muy específicos, como *XML* y *JSON* (Wikipedia, 2021).

RIPE: *Réseaux IP Européens*, es un foro colaborativo para los grupos interesados en redes IP. El objetivo de *RIPE* es asegurar la coordinación administrativa y técnica necesaria para que funcione Internet dentro de la región de Europa, Oriente Medio y partes de Asia Central (Wikipedia, 2021).

SaaS: *Software as Service* o software como servicio ofrece a los usuarios la posibilidad de conectarse a aplicaciones alojadas en la nube a través de Internet (*Salesforce*, 2021).

SFDC: Salesforce.com (*Salesforce*, 2021).

Sistema legado: también conocido como sistema *legacy* o sistema heredado, y no es más que un sistema, tecnología o aplicación de software antiguo o desactualizado que sigue en uso dentro de una organización porque sigue desempeñando las funciones para las que fue diseñado. Por lo general, los sistemas legacy ya no cuentan con soporte y mantenimiento y están limitados a nivel de crecimiento. Sin embargo, no pueden reemplazarse fácilmente (Stackscale, 2021).

SOSL: *Salesforce Object Search Language* o lenguaje de búsqueda de objetos de Salesforce, el cual permite realizar búsquedas de texto en los registros. A diferencia de SOQL, SOSL permite consultar múltiples tipos de objetos al mismo tiempo. Además, en el caso SOSL, es posible usar una palabra coincidente para establecer la coincidencia con campos, mientras que SOQL requiere la frase exacta (*Salesforce*, 2021).

SOQL: *Salesforce Object Query Language* o lenguaje de consulta de objetos de Salesforce. Puede usarse para leer la información almacenada en la base de datos de la organización (*Salesforce*, 2021).

Swagger: es una serie de reglas, especificaciones y herramientas que permiten documentar *APIs*. De esta manera, se realiza la documentación que sea realmente útil para las personas que la necesitan. *Swagger* ayuda a crear documentación que todo el mundo entienda (Chakray, 2021)

URI: *Uniform Resource Identificato*r o Identificador Uniforme de Recursos, es la ruta que permite acceder a un recurso y debe ser un identificador único para cada objeto o *API* (Wikipedia, 2021).

URL: U*niform Resource Locator* o Localizador Uniforme de Recursos, es la dirección específica que se asigna a cada uno de los recursos disponibles en la red con la finalidad de que estos puedan ser localizados o identificados (Wikipedia, 2021).

Whitelist: Una lista blanca es una lista o registro de entidades que, por una razón u otra, pueden obtener algún privilegio particular, servicio, movilidad, acceso o reconocimiento. Por el contrario, la lista negra es la compilación que identifica a quienes serán denegados, no reconocidos u obstaculizados (Wikipedia, 2021).

Workflow: Flujo de trabajo que define la automatización de tareas de una empresa (*Salesforce*, 2021).

YAML: *Yet Another Markup Language* que significa no es otro lenguaje de marcado, es un formato estándar de serialización de datos legible por humanos que intenta reducir la abstracción y hacer la programación más sencilla (Wikipedia, 2021).

Referencias.

BBVA (7 de junio de 2016). Qué es un API y qué puede hacer por mi negocio. https://www.bbvaAPImarket.com/es/mundo-API/que-es-una-API-y-que-puede-hacer-por-mi-negocio/

Chakray (s.f). REST fácil: diseño, evolución y conexión API.
Recuperado el 25 de ocutubre de 2021 de https://www.chakray.com/es/ebooks/

Chakray (s.f). API Management: ¿Cómo acelerar el crecimiento de mi negocio con APIs Open Source?
Recuperado el 25 de ocutubre de 2021 de https://www.chakray.com/es/ebooks/

Da Silva, Douglas. (23 de marzo de 2020). Conoce las características de un CRM: las ventajas para tu empresa. *Blog de Zendesk*.
https://www.zendesk.com.mx/blog/caracteristicas-de-un-crm/

Definición.DE. (s.f). https://definicion.de/

Fernández Corre, Mayelín. & Anías Calderón, Caridad. Aplicaciones del Modelo SID.Telem@tica (La Habana), Cuba, 2013 Vol. 12 Núm. 3 Sep-Dic, Pág. 61-70.
https://biblat.unam.mx/es/revista/telemtica-la-habana

FORBES (11 de diciembre de 2020). Transformación digital orientada al cliente con visión 360°.
https://www.forbes.com.mx/ad-transformacion-digital-con-vision-360-salesforce/

FORRESTER (04 de febrero de 2018). Five Forces Shape The Modern CRM Landscape. *Featured Blogs*. https://www.forrester.com/blogs/five-forces-shape-the-modern-crm-landscape/

Glender, Carlos (mayo 2018). El componente tecnológico en la estrategia CRM.
https://www.gartner.com/

Hayafuji, Cecilia (23 de agosto de 2019). APIs: ¿por qué son indispensables en la transformación digital? HAL.
https://automation.hal.company/blog/apis-por-que-son-indispensables-en-la-transformacion-digital

IDC (octubre 2021). Worldwide Semiannual Software Tracker.
https://www.idc.com/

IDC (18 de enero de 2022). La consolidación digital de empresas mexicanas dependerá de la automatización.
https://idconline.mx/corporativo/2022/01/18/consolidacion-digital-de-empresas-mexicanas-dependera-de-automatizacion

Informática (s.f). Guía de 30 minutos para la integración de aplicaciones y Multi-Cloud/Hybrid API. Recuperado el 28 de octubre de 2021 de https://www.informatica.com/es/

Jin, Brenda. & Sahni, Saurabh. (2018). Designing Web APIs. United States of America. O'Reilly Media.

López Magaña, Luis Miguel (17 de enero de 2020). Qué es OAuth 2. https://openwebinars.net/blog/que-es-oauth2/

Masse, Mark. (2012). REST API Design Rulebook. United States of America. O'Reilly Media.

Microsoft (s.f). Cómo sacar el máximo partido al software de CRM. Recuperado el 13 de octubre de 2021 de https://dynamics.microsoft.com/es-mx/crm/crm-software/

Navalón, Javier (12 de octubre de 2019). Descubre qué es Salesforce, mucho más que un CRM. *Showerthinking*. https://www.showerthinking.es/blog/que-es-salesforce/

NEXTUP (s.f). Guía para comprar un CRM. Recuperado el 3 de octubre de 2021 de https://nextup.com.mx/

NTS (5 de enero de 2022). Mejores CRM de ventas 2022: ¿Cuál es el ideal para tu empresa? *Salesforce Blog*. https://www.nts-solutions.com/blog/mejores-crm.html

Oracle (s.f). ¿Qué es la CRM? Recuperado el 13 de octubre de 2021 de https://www.oracle.com/mx/cx/what-is-crm

Orozco, Igor. et al. LA NUEVA ERA DE LOS NEGOCIOS: COMPUTACIÓN EN LA NUBE. Télématique, vol. 15, núm. 2, julio-diciembre, 2016, pp. 172-191

Panchal, Rahul (31 de mayo de 2021). API Development: Complete guide to building robust APIs. *Rlogical*. https://www.rlogical.com/blog/API-development-complete-guide-to-building-robust-APIs/

Peláez, Bruno. & Bonanno, Samantha (10 de enero de 2020). 10 características del CRM que debes tener en cuenta para elegir el correcto. Capterra. https://www.capterra.es/blog/1194/10-caracteristicas-del-crm-que-debes-tener-en-cuenta-para-elegir-el-correcto

Pons, Luis (15 de junio de 2021). ¿Qué son los Endpoints y para qué sirven? https://www.icm.es/

Richardson, Leonard. & Amundsen, Mike. (2013). RESTful Web APIs. United States of America. O'Reilly Media.

Red Hat (s.f). ¿Qué es una API?
Recuperado el 22 de octubre de 2021 de
https://www.redhat.com/es/topics/API/what-are-application-programming-interfaces

Salesforce (s.f). CRM 101: What is CRM? Recuperado el 13 de octubre de 2021 de https://www.salesforce.com/crm/what-is-crm/

Salesforce (15 de septiembre de 2021). Integration Patterns and Practices, v53.0
https://www.salesforce.com/

Salesforce (mayo 2016). Salesforce for Communications, v4.0.
https://www.salesforce.com/

Salesforce (s.f). ¿Qué es Cloud Computing?
Recuperado el 10 de noviembre de 2021 de https://www.salesforce.com/mx/cloud-computing/

Salesforce Developers (s.f). https://developer.salesforce.com.

Sanz, Oscar (15 de abril de 2020). Buenas prácticas en el diseño de APIs.
Buenas prácticas en el diseño de APIs (enmilocalfunciona.io)

SAP (s.f). ¿Qué es CRM hoy en día? Recuperado el 13 de octubre de 2021 de
https://www.sap.com/latinamerica/insights/what-is-crm.html

SOFDOIT. (s.f). Tipos de CRM y tabla comparativa: CRM operativo, colaborativo y analítico ¿Cuál elegir? Recuperado el 31 de octubre de 2021 de
https://www.softwaredoit.es/software-crm-guias/tabla-comparativa-tipos-crm.html

Stackscale (31 de agosto de 2021). ¿Qué es un sistema legacy?
https://www.stackscale.com/es/blog/sistemas-legacy/

Sturgeon, Philip. (2015). Build APIs You Won't Hate. United States of America. LeanPub

Swagger (s.f). OpenAPI Specification.
Recuperado el 26 de noviembre de 2021 de https://swagger.io/specification/

TM Forum (s.f). About us. Recuperado el 3 enero de 2022 de
https://www.tmforum.org/about-tm-forum/

TM Forum (s.f). Developer Quick Start Guide.
https://projects.tmforum.org/wiki/display/API/Developer+Quick+Start+Guide

TM Forum (marzo 2015). Frameworx Open Digital API Business Guide.
https://tmforum.org/

TM Forum (s.f). Open API Table. Recuperado el 10 de noviembre de 2021 de https://projects.tmforum.org/wiki/display/API/Open+API+Table

Trailhead Salesforce (s.f). https://trailhead.salesforce.com/es-MX

Ventura, Isaac (s.f). Tipos de CRM según el tipo de empresa. *El Blog de AMOCRM*. Recuperado el 9 de noviembre de 2021 de https://www.amocrm.com/es/blog/tipos-de-crm/

Yang, Royston (24 de diciembre de 2019). 3 Software-as-a-Service Companies That Could Double. https://www.fool.com/investing/2019/12/24/3-software-as-a-service-companies-could-double.aspx

Anexos

Anexo 1

Estructura *json* completa que representa el modelo de datos del recurso *Individual*, contiene todos sus subrecursos, atributos y tipo de datos a utilizar.


```
{
  "id": "string",
  "href": "string",
  "birthDate": "2022-02-01T16:40:12.468Z",
  "familyName": "string",
  "gender": "string",
  "givenName": "string",
  "maritalStatus": "string",
  "secondLastName": "string",
  "nationality": "string",
  "title": "string",
  "fullName": "string",
  "age": "string",
  "contactMedium": [
    {
      "mediumType": "string",
      "preferred": true,
      "characteristic": {
        "city": "string",
        "contactType": "string",
        "country": "string",
        "emailAddress": "string",
        "faxNumber": "string",
        "phoneNumber": "string",
        "mobileNumber": "string",
        "postCode": "string",
        "webSite": "string",
        "stateOrProvince": "string",
        "street1": "string",
        "street2": "string",
        "calendar": [
          {
            "day": "string",
            "status": "string",
            "comment": "string",
            "timeZone": "string",
            "hourPeriod": [
              {
                "endHour": "string",
                "startHour": "string"
              }
            ]
          }
        ]
      },
      "address": {
        "id": "string",
        "externalId": "string",
        "href": "string",
        "type": {
          "id": "string",
          "name": "string",
          "type": "string",
          "description": "string"
        },
        "streetNr": "string",
        "streetName": "string",
        "streetType": {
          "id": "string",
```

```
  "name": "string",
  "type": "string",
  "description": "string"
},
"postcode": "string",
"locality": {
  "id": "string",
  "name": "string",
  "type": "string",
  "description": "string"
},
"city": {
  "id": "string",
  "name": "string",
  "type": "string",
  "description": "string"
},
"stateOrProvince": {
  "id": "string",
  "name": "string",
  "type": "string",
  "description": "string"
},
"country": {
  "id": "string",
  "name": "string",
  "type": "string",
  "description": "string"
},
"@type": "string",
"@schemaLocation": "string",
"geographicLocation": {
  "id": "string",
  "href": "string",
  "name": "string",
  "geometryType": "string",
  "accuracy": "string",
  "spatialRef": "string",
  "@type": "string",
  "@schemaLocation": "string",
  "geometry": [
    {
      "x": "string",
      "y": "string",
      "z": "string"
    }
  ]
},
"geographicSubAddress": [
  {
    "id": "string",
    "type": "string",
    "name": "string",
    "description": "string",
    "subUnitType": "string",
    "subUnitNumber": "string",
    "levelType": "string",
    "levelNumber": "string",
    "buildingName": "string",
    "privateStreetNumber": "string",
    "privateStreetName": "string",
    "@type": "string",
    "@schemaLocation": "string"
  }
],
"geographicAddressSpec": [
  {
    "id": "string",
    "href": "string",
    "type": "string",
    "@type": "string",
```

```
            "@schemaLocation": "string",
            "characteristic": [
              {
                "id": "string",
                "name": "string",
                "valueType": "string",
                "value": "string"
              }
            ]
          }
        ],
        "area": [
          {
            "id": "string",
            "name": "string",
            "type": "string",
            "description": "string"
          }
        ],
        "characteristic": [
          {
            "id": "string",
            "name": "string",
            "value": "string"
          }
        ],
        "owner": "string"
      },
      "validFor": {
        "endDateTime": "2022-02-01T16:40:12.468Z",
        "startDateTime": "2022-02-01T16:40:12.468Z"
      }
    }
  ],
  "externalReference": [
    {
      "externalReferenceType": "string",
      "name": "string"
    }
  ],
  "individualIdentification": [
    {
      "identificationId": "string",
      "identificationType": "string",
      "issuingAuthority": "string",
      "checkDigit": "string",
      "issuingDate": "2022-02-01T16:40:12.468Z",
      "validFor": {
        "endDateTime": "2022-02-01T16:40:12.468Z",
        "startDateTime": "2022-02-01T16:40:12.468Z"
      },
      "attachment": {
        "id": "string",
        "href": "string",
        "attachmentType": "string",
        "content": "string",
        "description": "string",
        "mimeType": "string",
        "name": "string",
        "url": "string",
        "size": {
          "amount": 1,
          "units": "string"
        },
        "validFor": {
          "endDateTime": "2022-02-01T16:40:12.469Z",
          "startDateTime": "2022-02-01T16:40:12.469Z"
        }
      }
    }
  ],
```

```
"languageAbility": [
  {
    "isFavouriteLanguage": true,
    "languageCode": "string",
    "languageName": "string",
    "listeningProficiency": "string",
    "readingProficiency": "string",
    "speakingProficiency": "string",
    "writingProficiency": "string",
    "validFor": {
      "endDateTime": "2022-02-01T16:40:12.469Z",
      "startDateTime": "2022-02-01T16:40:12.469Z"
    }
  }
],
"partyCharacteristic": [
  {
    "id": "string",
    "name": "string",
    "valueType": "string",
    "value": "string"
  }
],
"relatedParty": [
  {
    "id": "string",
    "identifierType": "string",
    "href": "string",
    "name": "string",
    "role": "string",
    "partyRank": "string",
    "partyProfileType": [
      {
        "id": "string",
        "description": "string",
        "name": "string",
        "category": "string"
      }
    ],
    "externalReference": [
      {
        "externalReferenceType": "string",
        "name": "string"
      }
    ],
    "partyIdentification": [
      {
        "identificationId": "string",
        "identificationType": "string",
        "issuingAuthority": "string",
        "issuingDate": "2022-02-01T16:40:12.469Z",
        "checkDigit": "string",
        "validFor": {
          "endDateTime": "2022-02-01T16:40:12.469Z",
          "startDateTime": "2022-02-01T16:40:12.469Z"
        }
      }
    ],
    "isLegalEntity": true,
    "individual": "string",
    "organization": {
      "id": "string",
      "externalId": "string",
      "href": "string",
      "isHeadOffice": true,
      "isLegalEntity": true,
      "name": "string",
      "nameType": "string",
      "organizationType": "string",
      "tradingName": "string",
      "existsDuring": {
```

```
  "endDateTime": "2022-02-01T16:40:12.469Z",
  "startDateTime": "2022-02-01T16:40:12.469Z"
},
"contactMedium": [
  {
    "mediumType": "string",
    "preferred": true,
    "characteristic": {
      "city": "string",
      "contactType": "string",
      "country": "string",
      "emailAddress": "string",
      "faxNumber": "string",
      "phoneNumber": "string",
      "mobileNumber": "string",
      "postCode": "string",
      "webSite": "string",
      "stateOrProvince": "string",
      "street1": "string",
      "street2": "string",
      "calendar": [
        {
          "day": "string",
          "status": "string",
          "comment": "string",
          "timeZone": "string",
          "hourPeriod": [
            {
              "endHour": "string",
              "startHour": "string"
            }
          ]
        }
      ]
    },
    "address": {
      "id": "string",
      "externalId": "string",
      "href": "string",
      "type": {
        "id": "string",
        "name": "string",
        "type": "string",
        "description": "string"
      },
      "streetNr": "string",
      "streetName": "string",
      "streetType": {
        "id": "string",
        "name": "string",
        "type": "string",
        "description": "string"
      },
      "postcode": "string",
      "locality": {
        "id": "string",
        "name": "string",
        "type": "string",
        "description": "string"
      },
      "city": {
        "id": "string",
        "name": "string",
        "type": "string",
        "description": "string"
      },
      "stateOrProvince": {
        "id": "string",
        "name": "string",
        "type": "string",
        "description": "string"
```

```
},
"country": {
  "id": "string",
  "name": "string",
  "type": "string",
  "description": "string"
},
"@type": "string",
"@schemaLocation": "string",
"geographicLocation": {
  "id": "string",
  "href": "string",
  "name": "string",
  "geometryType": "string",
  "accuracy": "string",
  "spatialRef": "string",
  "@type": "string",
  "@schemaLocation": "string",
  "geometry": [
    {
      "x": "string",
      "y": "string",
      "z": "string"
    }
  ]
},
"geographicSubAddress": [
  {
    "id": "string",
    "type": "string",
    "name": "string",
    "description": "string",
    "subUnitType": "string",
    "subUnitNumber": "string",
    "levelType": "string",
    "levelNumber": "string",
    "buildingName": "string",
    "privateStreetNumber": "string",
    "privateStreetName": "string",
    "@type": "string",
    "@schemaLocation": "string"
  }
],
"geographicAddressSpec": [
  {
    "id": "string",
    "href": "string",
    "type": "string",
    "@type": "string",
    "@schemaLocation": "string",
    "characteristic": [
      {
        "id": "string",
        "name": "string",
        "valueType": "string",
        "value": "string"
      }
    ]
  }
],
"area": [
  {
    "id": "string",
    "name": "string",
    "type": "string",
    "description": "string"
  }
],
"characteristic": [
  {
    "id": "string",
```

```
          "name": "string",
          "value": "string"
        }
      ],
      "owner": "string"
    },
    "validFor": {
      "endDateTime": "2022-02-01T16:40:12.470Z",
      "startDateTime": "2022-02-01T16:40:12.470Z"
    }
  }
],
"externalReference": [
  {
    "externalReferenceType": "string",
    "name": "string"
  }
],
"organizationChildRelationship": [
  {
    "relationshipType": "string",
    "organization": {
      "id": "string",
      "href": "string",
      "name": "string"
    }
  }
],
"organizationIdentification": [
  {
    "identificationId": "string",
    "identificationType": "string",
    "issuingAuthority": "string",
    "issuingDate": "2022-02-01T16:40:12.470Z",
    "checkDigit": "string",
    "validFor": {
      "endDateTime": "2022-02-01T16:40:12.470Z",
      "startDateTime": "2022-02-01T16:40:12.470Z"
    },
    "@baseType": "string",
    "@schemaLocation": "string",
    "@type": "string"
  }
],
"organizationParentRelationship": {
  "relationshipType": "string",
  "organization": {
    "id": "string",
    "href": "string",
    "name": "string"
  }
},
"otherName": [
  {
    "name": "string",
    "nameType": "string",
    "tradingName": "string",
    "@baseType": "string",
    "@schemaLocation": "string",
    "@type": "string"
  }
],
"partyCharacteristic": [
  {
    "id": "string",
    "name": "string",
    "valueType": "string",
    "value": "string"
  }
],
"relatedParty": [
```

```
          "string"
        ],
        "partyProfileType": [
          {
            "id": "string",
            "description": "string",
            "name": "string",
            "category": "string"
          }
        ],
        "note": [
          {
            "id": "string",
            "author": "string",
            "text": "string",
            "date": "2022-02-01T16:40:12.470Z"
          }
        ],
        "status": "string",
        "owner": "string",
        "createDateTime": "2022-02-01T16:40:12.470Z",
        "lastUpdateDateTime": "2022-02-01T16:40:12.470Z",
        "createdBy": "string",
        "lastUpdateBy": "string",
        "state": "string",
        "@baseType": "string",
        "@schemaLocation": "string",
        "@type": "string"
      },
      "partyCharacteristic": [
        {
          "id": "string",
          "name": "string",
          "valueType": "string",
          "value": "string"
        }
      ],
      "@baseType": "string",
      "@schemaLocation": "string",
      "@type": "string",
      "@referredType": "string"
    }
  ],
  "status": "string",
  "partyProfileType": [
    {
      "id": "string",
      "description": "string",
      "name": "string",
      "category": "string"
    }
  ],
  "owner": "string",
  "createDateTime": "2022-02-01T16:40:12.470Z",
  "lastUpdateDateTime": "2022-02-01T16:40:12.470Z",
  "createdBy": "string",
  "lastUpdateBy": "string"
}
```


Anexo 2

Estructura json completa que representa el modelo de datos del recurso *organization*, contiene todos sus subrecusrso, atributos y tipo de datos a utilizar.


```
{
  "id": "string",
  "externalId": "string",
  "href": "string",
```

```
"isHeadOffice": true,
"isLegalEntity": true,
"name": "string",
"nameType": "string",
"organizationType": "string",
"tradingName": "string",
"existsDuring": {
  "endDateTime": "2022-02-01T23:54:07.047Z",
  "startDateTime": "2022-02-01T23:54:07.047Z"
},
"contactMedium": [
  {
    "mediumType": "string",
    "preferred": true,
    "characteristic": {
      "city": "string",
      "contactType": "string",
      "country": "string",
      "emailAddress": "string",
      "faxNumber": "string",
      "phoneNumber": "string",
      "mobileNumber": "string",
      "postCode": "string",
      "webSite": "string",
      "stateOrProvince": "string",
      "street1": "string",
      "street2": "string",
      "calendar": [
        {
          "day": "string",
          "status": "string",
          "comment": "string",
          "timeZone": "string",
          "hourPeriod": [
            {
              "endHour": "string",
              "startHour": "string"
            }
          ]
        }
      ]
    },
    "address": {
      "id": "string",
      "externalId": "string",
      "href": "string",
      "type": {
        "id": "string",
        "name": "string",
        "type": "string",
        "description": "string"
      },
      "streetNr": "string",
      "streetName": "string",
      "streetType": {
        "id": "string",
        "name": "string",
        "type": "string",
        "description": "string"
      },
      "postcode": "string",
      "locality": {
        "id": "string",
        "name": "string",
        "type": "string",
        "description": "string"
      },
      "city": {
        "id": "string",
        "name": "string",
        "type": "string",
```

```
  "description": "string"
},
"stateOrProvince": {
  "id": "string",
  "name": "string",
  "type": "string",
  "description": "string"
},
"country": {
  "id": "string",
  "name": "string",
  "type": "string",
  "description": "string"
},
"@type": "string",
"@schemaLocation": "string",
"geographicLocation": {
  "id": "string",
  "href": "string",
  "name": "string",
  "geometryType": "string",
  "accuracy": "string",
  "spatialRef": "string",
  "@type": "string",
  "@schemaLocation": "string",
  "geometry": [
    {
      "x": "string",
      "y": "string",
      "z": "string"
    }
  ]
},
"geographicSubAddress": [
  {
    "id": "string",
    "type": "string",
    "name": "string",
    "description": "string",
    "subUnitType": "string",
    "subUnitNumber": "string",
    "levelType": "string",
    "levelNumber": "string",
    "buildingName": "string",
    "privateStreetNumber": "string",
    "privateStreetName": "string",
    "@type": "string",
    "@schemaLocation": "string"
  }
],
"geographicAddressSpec": [
  {
    "id": "string",
    "href": "string",
    "type": "string",
    "@type": "string",
    "@schemaLocation": "string",
    "characteristic": [
      {
        "id": "string",
        "name": "string",
        "valueType": "string",
        "value": "string"
      }
    ]
  }
],
"area": [
  {
    "id": "string",
    "name": "string",
```

```
            "type": "string",
            "description": "string"
          }
        ],
        "characteristic": [
          {
            "id": "string",
            "name": "string",
            "value": "string"
          }
        ],
        "owner": "string"
      },
      "validFor": {
        "endDateTime": "2022-02-01T23:54:07.047Z",
        "startDateTime": "2022-02-01T23:54:07.047Z"
      }
    }
  ],
  "externalReference": [
    {
      "externalReferenceType": "string",
      "name": "string"
    }
  ],
  "organizationChildRelationship": [
    {
      "relationshipType": "string",
      "organization": {
        "id": "string",
        "href": "string",
        "name": "string"
      }
    }
  ],
  "organizationIdentification": [
    {
      "identificationId": "string",
      "identificationType": "string",
      "issuingAuthority": "string",
      "issuingDate": "2022-02-01T23:54:07.047Z",
      "checkDigit": "string",
      "validFor": {
        "endDateTime": "2022-02-01T23:54:07.047Z",
        "startDateTime": "2022-02-01T23:54:07.047Z"
      },
      "@baseType": "string",
      "@schemaLocation": "string",
      "@type": "string"
    }
  ],
  "organizationParentRelationship": {
    "relationshipType": "string",
    "organization": {
      "id": "string",
      "href": "string",
      "name": "string"
    }
  },
  "otherName": [
    {
      "name": "string",
      "nameType": "string",
      "tradingName": "string",
      "@baseType": "string",
      "@schemaLocation": "string",
      "@type": "string"
    }
  ],
  "partyCharacteristic": [
    {
```

```
      "id": "string",
      "name": "string",
      "valueType": "string",
      "value": "string"
    }
  ],
  "relatedParty": [
    {
      "id": "string",
      "identifierType": "string",
      "href": "string",
      "name": "string",
      "role": "string",
      "partyRank": "string",
      "partyProfileType": [
        {
          "id": "string",
          "description": "string",
          "name": "string",
          "category": "string"
        }
      ],
      "externalReference": [
        {
          "externalReferenceType": "string",
          "name": "string"
        }
      ],
      "partyIdentification": [
        {
          "identificationId": "string",
          "identificationType": "string",
          "issuingAuthority": "string",
          "issuingDate": "2022-02-01T23:54:07.048Z",
          "checkDigit": "string",
          "validFor": {
            "endDateTime": "2022-02-01T23:54:07.048Z",
            "startDateTime": "2022-02-01T23:54:07.048Z"
          }
        }
      ],
      "isLegalEntity": true,
      "individual": {
        "id": "string",
        "href": "string",
        "birthDate": "2022-02-01T23:54:07.048Z",
        "familyName": "string",
        "gender": "string",
        "givenName": "string",
        "maritalStatus": "string",
        "secondLastName": "string",
        "nationality": "string",
        "title": "string",
        "fullName": "string",
        "age": "string",
        "contactMedium": [
          {
            "mediumType": "string",
            "preferred": true,
            "characteristic": {
              "city": "string",
              "contactType": "string",
              "country": "string",
              "emailAddress": "string",
              "faxNumber": "string",
              "phoneNumber": "string",
              "mobileNumber": "string",
              "postCode": "string",
              "webSite": "string",
              "stateOrProvince": "string",
              "street1": "string",
```

```
  "street2": "string",
  "calendar": [
    {
      "day": "string",
      "status": "string",
      "comment": "string",
      "timeZone": "string",
      "hourPeriod": [
        {
          "endHour": "string",
          "startHour": "string"
        }
      ]
    }
  ]
},
"address": {
  "id": "string",
  "externalId": "string",
  "href": "string",
  "type": {
    "id": "string",
    "name": "string",
    "type": "string",
    "description": "string"
  },
  "streetNr": "string",
  "streetName": "string",
  "streetType": {
    "id": "string",
    "name": "string",
    "type": "string",
    "description": "string"
  },
  "postcode": "string",
  "locality": {
    "id": "string",
    "name": "string",
    "type": "string",
    "description": "string"
  },
  "city": {
    "id": "string",
    "name": "string",
    "type": "string",
    "description": "string"
  },
  "stateOrProvince": {
    "id": "string",
    "name": "string",
    "type": "string",
    "description": "string"
  },
  "country": {
    "id": "string",
    "name": "string",
    "type": "string",
    "description": "string"
  },
  "@type": "string",
  "@schemaLocation": "string",
  "geographicLocation": {
    "id": "string",
    "href": "string",
    "name": "string",
    "geometryType": "string",
    "accuracy": "string",
    "spatialRef": "string",
    "@type": "string",
    "@schemaLocation": "string",
    "geometry": [
```

```
            {
              "x": "string",
              "y": "string",
              "z": "string"
            }
          ]
        },
        "geographicSubAddress": [
          {
            "id": "string",
            "type": "string",
            "name": "string",
            "description": "string",
            "subUnitType": "string",
            "subUnitNumber": "string",
            "levelType": "string",
            "levelNumber": "string",
            "buildingName": "string",
            "privateStreetNumber": "string",
            "privateStreetName": "string",
            "@type": "string",
            "@schemaLocation": "string"
          }
        ],
        "geographicAddressSpec": [
          {
            "id": "string",
            "href": "string",
            "type": "string",
            "@type": "string",
            "@schemaLocation": "string",
            "characteristic": [
              {
                "id": "string",
                "name": "string",
                "valueType": "string",
                "value": "string"
              }
            ]
          }
        ],
        "area": [
          {
            "id": "string",
            "name": "string",
            "type": "string",
            "description": "string"
          }
        ],
        "characteristic": [
          {
            "id": "string",
            "name": "string",
            "value": "string"
          }
        ],
        "owner": "string"
      },
      "validFor": {
        "endDateTime": "2022-02-01T23:54:07.048Z",
        "startDateTime": "2022-02-01T23:54:07.048Z"
      }
    }
  ],
  "externalReference": [
    {
      "externalReferenceType": "string",
      "name": "string"
    }
  ],
  "individualIdentification": [
```

```
      {
        "identificationId": "string",
        "identificationType": "string",
        "issuingAuthority": "string",
        "checkDigit": "string",
        "issuingDate": "2022-02-01T23:54:07.048Z",
        "validFor": {
          "endDateTime": "2022-02-01T23:54:07.048Z",
          "startDateTime": "2022-02-01T23:54:07.048Z"
        },
        "attachment": {
          "id": "string",
          "href": "string",
          "attachmentType": "string",
          "content": "string",
          "description": "string",
          "mimeType": "string",
          "name": "string",
          "url": "string",
          "size": {
            "amount": 1,
            "units": "string"
          },
          "validFor": {
            "endDateTime": "2022-02-01T23:54:07.048Z",
            "startDateTime": "2022-02-01T23:54:07.048Z"
          }
        }
      }
    ],
    "languageAbility": [
      {
        "isFavouriteLanguage": true,
        "languageCode": "string",
        "languageName": "string",
        "listeningProficiency": "string",
        "readingProficiency": "string",
        "speakingProficiency": "string",
        "writingProficiency": "string",
        "validFor": {
          "endDateTime": "2022-02-01T23:54:07.048Z",
          "startDateTime": "2022-02-01T23:54:07.048Z"
        }
      }
    ],
    "partyCharacteristic": [
      {
        "id": "string",
        "name": "string",
        "valueType": "string",
        "value": "string"
      }
    ],
    "relatedParty": [
      "string"
    ],
    "status": "string",
    "partyProfileType": [
      {
        "id": "string",
        "description": "string",
        "name": "string",
        "category": "string"
      }
    ],
    "owner": "string",
    "createDateTime": "2022-02-01T23:54:07.048Z",
    "lastUpdateDateTime": "2022-02-01T23:54:07.048Z",
    "createdBy": "string",
    "lastUpdateBy": "string"
  },
```

```
      "organization": "string",
      "partyCharacteristic": [
        {
          "id": "string",
          "name": "string",
          "valueType": "string",
          "value": "string"
        }
      ],
      "@baseType": "string",
      "@schemaLocation": "string",
      "@type": "string",
      "@referredType": "string"
    }
  ],
  "partyProfileType": [
    {
      "id": "string",
      "description": "string",
      "name": "string",
      "category": "string"
    }
  ],
  "note": [
    {
      "id": "string",
      "author": "string",
      "text": "string",
      "date": "2022-02-01T23:54:07.048Z"
    }
  ],
  "status": "string",
  "owner": "string",
  "createDateTime": "2022-02-01T23:54:07.048Z",
  "lastUpdateDateTime": "2022-02-01T23:54:07.048Z",
  "createdBy": "string",
  "lastUpdateBy": "string",
  "state": "string",
  "@baseType": "string",
  "@schemaLocation": "string",
  "@type": "string"
}
```