DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

TRABAJO DE TITULACIÓN PREVIO LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA

TEMA: "DECISION CLOUD: SISTEMA DE GESTIÓN DE SERVICIO AL CLIENTE E INTEGRACIÓN DE APLICACIONES, BASADO EN LA ARQUITECTURA ORIENTADA A MICROSERVICIOS, PARA DECISIÓN C.A."

AUTORES:
FÉRNANDEZ MONCAYO, MISAHAEL ALEXANDER
GUERRERO MUÑOZ, CARLOS JOSÉ

DIRECTOR: ING. CORAL, HENRY

SANGOLQUÍ, SEPTIEMBRE 2016

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

CERTIFICACIÓN

Certifico que el trabajo de titulación, "DECISION CLOUD: SISTEMA DE GESTIÓN DE SERVICIO AL CLIENTE E INTEGRACIÓN DE APLICACIONES, BASADO EN LA ARQUITECTURA ORIENTADA A MICROSERVICIOS, PARA DECISIÓN C.A.", elaborado por los señores MISAHAEL ALEXANDER FERNÁNDEZ MONCAYO y CARLOS JOSÉ GUERRERO MUÑOZ, ha sido revisado en su totalidad y analizado por el software anti-plagio, con un porcentaje del cero por ciento (0%), el mismo que cumple los requisitos teóricos, científicos, técnicos, metodológicos y legales vigentes establecidos por la Universidad de las Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar a los señores MISAHAEL ALEXANDER FERNÁNDEZ MONCAYO y CARLOS JOSÉ GUERRERO MUÑOZ para que lo sustente públicamente.

Sangolquí, 08 de Septiembre de 2016

ING. HENRY CORAL
DIRECTOR

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

AUTORÍA DE RESPONSABILIDAD

Nosotros, MISAHAEL ALEXANDER FERNÁNDEZ MONCAYO, con cédula de identidad N° 0704054451 y CARLOS JOSÉ GUERRERO MUÑOZ, con cédula de identidad N° 1723095228, declaramos que este trabajo denominado: "DECISION CLOUD: SISTEMA DE GESTIÓN DE SERVICIO AL CLIENTE E INTEGRACIÓN DE APLICACIONES, BASADO EN LA ARQUITECTURA ORIENTADA A MICROSERVICIOS, PARA DECISIÓN C.A.", ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme a las fuentes que se incorporan en la bibliografía.

Consecuentemente declaramos que este trabajo es de nuestra autoría, en virtud de ello nos declaramos responsables del contenido, veracidad y alcance del proyecto mencionado.

Sangolquí, 08 de Septiembre de 2016

Fernández Moncayo Misahael Alexander
C.C. 0704054451

Carlos José Guerrero Muñoz
C.C. 1723095228

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

AUTORIZACIÓN

Nosotros, MISAHAEL ALEXANDER FERNÁNDEZ MONCAYO y CARLOS JOSÉ GUERRERO MUÑOZ, autorizamos a la Universidad de las Fuerzas Armadas ESPE la publicación, en la biblioteca virtual de la Institución, el trabajo: "DECISION CLOUD: SISTEMA DE GESTIÓN DE SERVICIO AL CLIENTE E INTEGRACIÓN DE APLICACIONES, BASADO EN LA ARQUITECTURA ORIENTADA A MICROSERVICIOS, PARA DECISIÓN C.A." cuyo contenido, ideas y criterios son de nuestra responsabilidad y autoría.

Sangolquí, 08 de Septiembre de 2016

Fernández Moncayo Misahael Alexander

C.C. 0704054451

Carlos José Guerrero Muñoz

C.C. 1723095228

DEDICATORIA

Este proyecto va dedicado a mi mamá y a mi abuela, por haber estado conmigo en todo momento, ayudándome en todo lo que han podido y sacrificándose cada día para que cumpla con mis objetivos propuestos, tanto personales como profesionales. Ellas siempre han sido, son y serán mi mayor motivación para seguir adelante.

Misahael Fernández M.

Dedico este trabajo principalmente a mi mamá, por ser el pilar más importante en mi formación tanto personal como profesional. A mi padre que día tras día se sacrificó por un mejor porvenir. A mi hermana, mi cuñado y mis sobrinos que siempre me brindaron su apoyo y cariño incondicional. A mi novia por todos los momentos vividos y por ser mi motivo de superación junto con ese pequeño ser que lleva en su vientre.

Carlos Guerrero M.

AGRADECIMIENTO

A mi padre por confiar en mis capacidades para poder lograr mis objetivos y la preocupación que ha dedicado para que sea una persona ética y responsable. A mi familia que ha estado apoyándome en todo momento a lo largo de mi vida. A todas las personas que estuvieron presentes en mi vida académica, brindándome su apoyo incondicional.

A la compañía DECISIÓN c.a., por brindarnos la confianza y el apoyo necesario para poder realizar este proyecto que es el núcleo de un buen servicio hacia los clientes.

Misahael Fernández M.

A mi madre por ayudarme a ser una persona de bien y por enseñarme a valorar los buenos y malos momentos que la vida nos ofrece, y principalmente porque un buen hombre es hijo de una gran mujer. A mi familia por los consejos y por el apoyo que me brindaron en todo momento. A mis amigos que formaron una parte muy importante en mi vida estudiantil ayudándome a cumplir con este objetivo.

A la compañía DECISIÓN c.a., que nos brindó todos los recursos necesarios para llevar adelante este proyecto,

Carlos Guerrero M.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

Para el desarrollo de las empresas e incrementar su participación comercial en el mercado, es fundamental invertir estrategias que aseguren la existencia de una mejora continua en el servicio al cliente, ya que esto fideliza a los clientes y genera nuevos. Ante este escenario, este estudio presenta una propuesta, que tiene como objetivo monitorear, analizar y controlar la calidad de la atención al cliente. Se propone el diseño y desarrollo de un sistema para la gestión de servicios, basado en la arquitectura orientada a microservicios, que permite el control de un servicio requerido por el cliente. En este enfoque, se utilizó la metodología "UWE" con el propósito de asegurar la calidad del software. Los componentes del sistema están distribuidos en una instancia proporcionada por los servicios web de Amazon (AWS), los mismos se conectan mediante REST hacia una aplicación web desarrollada en HTML5 con el framework de Angular y Kendo UI, la misma que está alojada en Amazon S3 y su contenido es distribuido con CloudFront, este recurso es usado para aumentar la velocidad de carga de los archivos del aplicativo. Los datos que se generan son utilizados para medir la calidad del servicio ayudando así a realizar cambios estratégicos para su mejora; de igual manera el cliente puede ver dicha información para constatar que todo sea de su satisfacción. La implementación se realizó en la compañía Decisión c.a. donde se evaluó el funcionamiento del sistema, con respecto a la usabilidad y se encuestó a los clientes con el fin de conocer el nivel de satisfacción de los servicios recibidos.

PALABRAS CLAVES:

- GESTIÓN DEL SERVICIO
- MICROSERVICIOS
- AMAZON
- UWE
- SERVICIOS WEB
- REST
- ANGULAR

ABSTRACT

For business growth and increase its commercial market share, it is essential to invest strategies to ensure the existence of a continuous improvement in customer service, as this will create loyal customers and generates new ones. To face this scenario, this study shows a proposal, in order to monitor, analyze and control the quality of the customer support. The design and develop of a system is proposed for service management, based in microservices architecture, allowing control of a service required by the customer. In this approach, the methodology "UWE" was used in order to ensure software quality. The components of the system are distributed in an instance provided by Amazon Web Services (AWS), they are connected by REST to a web application developed in HTML5 with Angular framework and Kendo UI; this application is hosted in Amazon S3 and its content is distributed with CloudFront, this resource is used to increase speed of loading application files. The data generated is used to measure the quality of service helping to make strategic changes for improvement; likewise, the customer can see this information to make sure everything is to his satisfaction. The implementation was done in Decision c.a. company where the system functionality was evaluated with respect to usability and surveyed customers in order to meet the level of satisfaction of the services received.

KEYWORDS:

- SERVICE MANAGEMENT
- MICROSERVICES
- AMAZON
- UWE
- WEB SERVICES
- REST
- ANGULAR

CAPITULO I
INTRODUCCIÓN

1.1 ANTECEDENTES

La gestión del servicio al cliente en la actualidad ha ganado una gran importancia en el mundo, y por ende en Ecuador. Para mantenerse en el mercado como una organización exitosa, las empresas han identificado herramientas y metodologías que permiten la mejora continua en la atención a los requerimientos y problemas presentados por los clientes (Insights, 2015).

Las organizaciones pueden optar por una gran variedad de herramientas para la gestión de servicio al cliente, como, por ejemplo: iAdvice (iAdvize, 2015), LivePerson (LivePerson, 2015), etc. Por otra parte, cada organización tiene sus propias características y se adapta al mercado al cual está brindando un producto o un servicio (Hitt, México), en tal motivo muchas de las herramientas antes mencionadas no son de gran utilidad, por lo que en ocasiones es necesario desarrollar software a la medida que ofrezca beneficios como: Módulos independientes, escalabilidad y sobre todo el cumplimiento de los requerimientos de una organización (MicroServices, 2015).

DECISION c.a. es una empresa ecuatoriana especializada en la integración de soluciones de tecnología de información y sistemas de protección electrónica de alta complejidad. Esta empresa se encuentra ubicada en el Valle de los Chillos en donde lleva operando por más de 40 años en el mercado. Actualmente DECISION c.a. cuenta con un módulo de servicio al cliente y varias aplicaciones con propósitos específicos, los cuales generan varios inconvenientes (Decisión, 2014).

1.2 PROBLEMÁTICA

DECISION c.a. tiene una falencia en la gestión del servicio al cliente, además cuenta con sistemas externos e independientes entre sí, por consiguiente, es prácticamente imposible obtener información confiable, lo cual es necesario para el desarrollo de sus actividades diarias.

Para dar solución a estos inconvenientes, se propone realizar un sistema denominado DECISION CLOUD que permita la gestión de servicio al cliente, basado en la arquitectura orientada a microservicios y así lograr la integración de los sistemas externos de DECISION c.a.

La arquitectura orientada a microservicios permite una autonomía entre los módulos del sistema permitiendo flexibilidad para nuevos desarrollos y adaptar los antiguos.

1.3 JUSTIFICACIÓN

La empresa DECISION c.a. ha tomado la iniciativa de dar un valor agregado a su módulo de servicio al cliente y a empezar la integración con los aplicativos que actualmente se encuentran en funcionamiento, debido a los problemas identificados.

La comunicación del sistema a otras aplicaciones es de vital importancia para la gestión de tareas pendientes, notificaciones hacia los usuarios involucrados en el servicio que se está brindando y a su vez ampliar la funcionalidad del mismo.

Con la implementación de DECISION CLOUD se pretende lograr una integración de las aplicaciones que tiene DECISION c.a., además de proporcionar un servicio al cliente optimizado; desde que se solicita el servicio hasta su finalización.

En caso de que no implementarse DECISION CLOUD, la información se comenzará a duplicar, se escogerán datos erróneos para el análisis, la gestión de servicio al cliente será más complicada debido a la falta de depuración del proceso y por último pérdida de clientes y de datos en DECISION c.a.

1.4 OBJETIVOS

a. Objetivo General

Desarrollar e implementar un sistema de gestión de servicio al cliente e integración de aplicaciones, basado en la arquitectura orientada a microservicios, para DECISION c.a.

b. Objetivos Específicos

i. Implementar un sistema informático aplicando conceptos de la arquitectura orientada a microservicios para autonomía entre módulos.

ii. Integrar las aplicaciones externas con el nuevo sistema para un manejo correcto e integral de la información y no exista ambigüedad de datos.

iii. Desarrollar el módulo de la gestión de servicio al cliente para el crecimiento del negocio brindando.

1.5 ALCANCE

Se depurará el proceso actual de la gestión de servicio al cliente para que se adapte al nuevo sistema en DECISION c.a. Además, el sistema abarcará el proceso completo de la gestión de servicio al cliente.

El sistema integrará los sistemas externos con los que cuenta DECISION c.a. para evitar la disgregación y duplicidad de la información, logrando emitir y controlar la información para facturación relacionada con tiempo de servicio, insumos y repuestos.

CAPITULO II
MARCO TEORICO

2.1 Gestión de servicio

La gestión de servicio lo define Office of Government Commerce como: “Es un conjunto de capacidades organizativas especializadas que proporcionan valor en forma de servicios” (Office of Government Commerce, 2009).

Las organizaciones le dan suma importancia al proceso de gestión de servicio ya que esto asegura su supervivencia en el mercado y entrega mayores solvencias en las necesidades de los usuarios (Decisión, 2014).

2.1.1 Servicio

Horovitz afirma que “El servicio es el conjunto de prestaciones que el cliente espera, además del producto o del servicio básico, como consecuencia de la imagen y la reputación del mismo” (Horovitz, 2012).

Office of Government Commerce afirma que “Un servicio es un medio de entrega a los clientes facilitando los resultados que estos desean lograr sin responsabilidad sobre los costes y riesgos específicos” (Office of Government Commerce, 2009).

Servicio por tanto es un medio por el cual las organizaciones entregan un conjunto de facilidad a los clientes como un efecto de la imagen y reputación que estas han logrado, además de proporcionarle el producto u otro tipo de servicio.

Un ejemplo que sigue el patrón de un servicio es el generalizado por un sistema de almacenamiento. El almacenamiento resulta útil para mantener, organizar o asegurar activos dentro del contexto de alguna actividad, tarea o rendimiento; también crea condiciones útiles como la facilidad de acceso, una organización eficaz o la seguridad frente a las amenazas. Este sencillo patrón es intrínseco en muchos tipos de servicios de almacenamiento, cada uno especializado para dar apoyo a un tipo de resultado particular para los clientes como se indica en la Figura 1 (Office of Government Commerce, 2009).

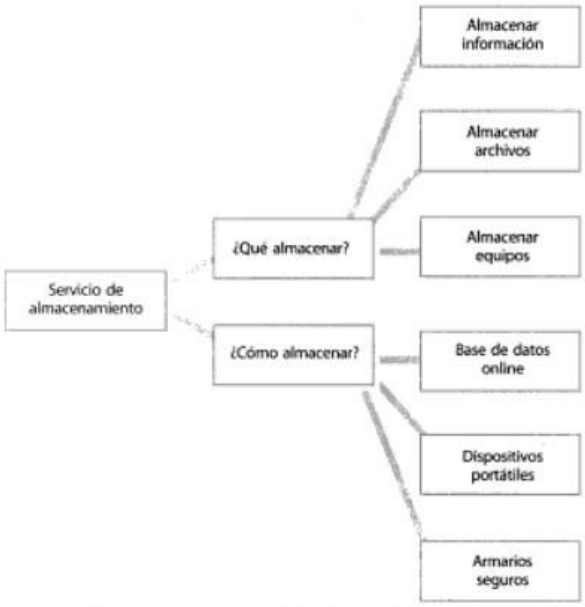

Figura 1. Patrones generalizados e instancias especializadas
Fuente: (Office of Government Commerce, pág. 16)

2.1.2 Servicio al cliente

Es el servicio que proporciona una organización al relacionarse con los clientes y la manera en que atienden sus requerimientos, esto es aprovechado por las empresas para brindar un valor agregado a sus servicios y poder ganar clientes leales y conseguir nuevos (Pearson, 2007).

2.1.3 Gestión de servicio en Ecuador

Los estudios que se han realizado en lo que respecta a la gestión de servicios en Ecuador han sido numerosos y cada vez se han implementado nuevos procesos mediante los cuales se pueden mejorar la calidad de servicio que las empresas pueden brindar a los clientes (Loza, Roa, Samaniego, 2013).

2.2 Cloud computing

Según la NIST (National Institute of Standards and Technology) cloud computing es: “Un modelo para permitir el acceso (ubicuo, fácil y bajo demanda) a través de una red a un pool compartido de recursos informáticos configurables (por ejemplo: redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser

rápidamente aprovisionados y liberados con un mínimo esfuerzo administrativo y con una mínima interacción con el proveedor de servicios" (NIST, 2011).

También llamada computación en la nube, se refiere tanto a las aplicaciones entregadas como servicio en Internet como al hardware y al software de los sistemas en los centros de datos que prestan estos servicios. Cuenta con tres modelos de entrega que son: **S**oftware **a**s **a** **S**ervice (SaaS), Infraestructure as a Service (IaaS) y Platform as a Service (PaaS). En la Figura 2 se puede visualizar los distintos módulos de entrega que ofrece el Cloud Computing.

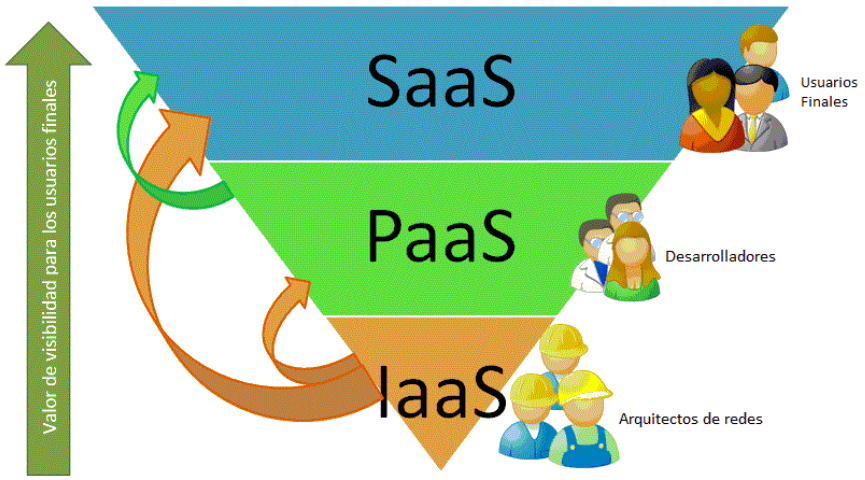

Figura 2. Modelos de entrega de Cloud Computing

Fuente: (Tecnologías aplicadas, 2011)

2.2.1 Características

La computación en la nube cuenta con las siguientes características:

- Aplicaciones bajo demanda.
- Accesibilidad
- Asignación de recursos en modo multiusuario
- Elasticidad y escalabilidad
- Supervisión del servicio
- Seguridad

Logrando ser una gran opción para software de gestión de servicio al cliente (SocieTIC, 2010).

2.2.2 Software as a Service

Para EMC el software como servicio es: “Software que se usa a través de una red sin descargarlo en un ambiente de cómputo local. Se obtiene acceso a la aplicación de software a través de Internet desde un proveedor de SaaS y se ejecuta en el ambiente de cómputo predefinido del proveedor” (EMC, 2015).

Por lo tanto, el SaaS implica proporcionar conjuntamente los medios, los servicios y la experiencia que permiten a las empresas externalizar completamente algún aspecto de su sistema informático y asimilarlo como un coste operativo en lugar de como una inversión (TraceOne, 2014).

El cliente final lo que hace es usar el software para sus propósitos sin preocuparse del mantenimiento o de las actualizaciones que este reciba a lo largo de su ciclo de vida, también puede acceder al aplicativo en cualquier momento siempre y cuando tenga conexión a internet, en la Figura 3 se puede observar el modelo SaaS definido por Security Affairs.

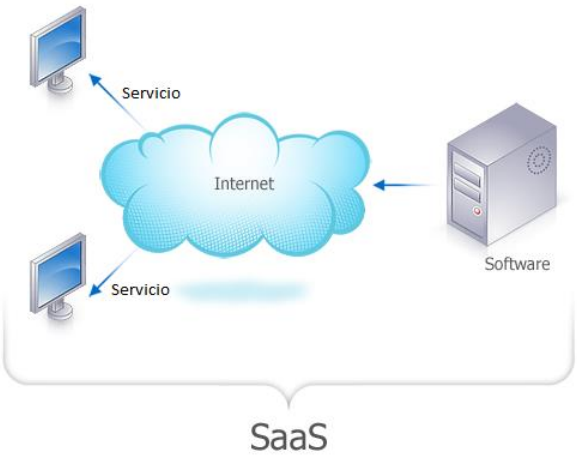

Figura 3. Modelo SaaS

Fuente: (Security Affairs, 2014)

Su funcionamiento se basa en que cada usuario conforma un grupo de usuarios en el ambiente compartido de varios grupos de usuarios del proveedor de SaaS. Puede permitirse cierta personalización del software. No obstante, no resulta habitual que el usuario tenga mucho control sobre la arquitectura de cómputo de la infraestructura, la ubicación o los niveles de servicio proporcionados como podemos visualizar en la Figura 4 (EMC, 2015).

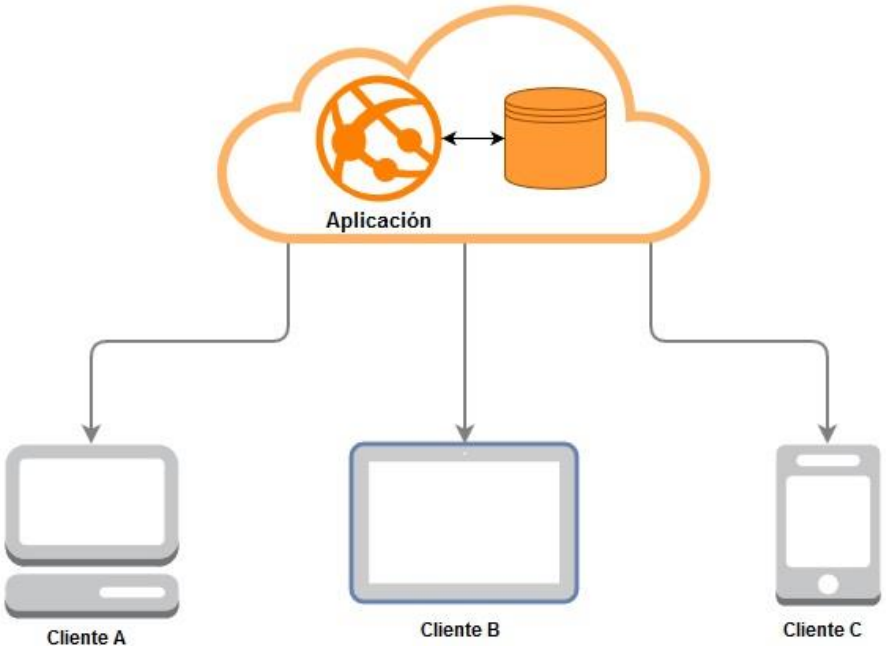

Figura 4. Funcionamiento de un SaaS

2.2.2.1 Las ventajas que ofrece este servicio son las siguientes:

- **Costes**

  La inversión inicial es exactamente cero. Además, no tiene que comprar nuevos servidores, más espacio en disco ni ordenadores. SaaS utiliza los recursos que el cliente ya posee (TRACE ONE, 2015).

  En 2006, un estudio de McKinsey calculó que el TCO (coste total de la propiedad) del software en modo SaaS es un 40% inferior al del software instalado en el ordenador del cliente (TRACE ONE, 2015).

- **Tiempo**

  Despliegue rápido en modo de actualización continua Configuración y funcionamiento más rápidos gracias a que el software está disponible a través de una simple conexión a Internet (TRACE ONE, 2015).

Accesibilidad en línea permanente desde cualquier punto, el hecho de que se acceda a él a través de Internet supone que el software en modo SaaS está disponible en todo momento y lugar (TRACE ONE, 2015).

2.3 Plataformas de integración

Una plataforma de integración en informática es aquella que puede interrelacionar datos de diferentes aplicaciones, todo esto para poder mejorar la comunicación entre sistemas heterogéneos (Carmona Barbero, 2014).

2.3.1 Historia

En los años 70 nace la idea de comunicar varios computadores u ordenadores a través de una red por medio de Sockets, también definido como el punto final de una conexión que permite realizar varias funciones o llamadas y así lograr una comunicación bidireccional entre distintas maquinas (Carmona Barbero, 2014).

Después en los años 80’s surgió un gran problema el mismo que residía en la invocación entre procesos remotos, en donde el principal funcionamiento de estos procesos era ejecutar procedimientos que pertenece a distintos entornos, dando lugar a la creación de RPC (Remote Procedure Call) que es una tecnología de gran alcance para la creación de programas cliente/servidor distribuidos (Carmona Barbero, 2014).

Los siguientes avances hacia los años 90’s con la implantación de los lenguajes orientados a objetos se centraron en la heterogeneidad del entorno ya que las redes que unen las distintas máquinas pueden usar diferentes tecnologías (Ethernet, ATM, FDDI,etc.),protocolos (TCP/IP, X.25,OSI,IPX/SPX,etc.),plataformas y lenguajes de programación; naciendo así CORBA que permite la invocación de métodos de objetos remotos sin que importe el lenguaje en el que estén escritos el llamador y el llamado, ni las plataformas (s.o. y hw.) y redes de comunicación intermedias; como también son el caso de Java RMI,DCOM/COM+/ActiveX (Carmona Barbero, 2014).

A finales de los 90’s principios de los 00’s con la adopción global de Internet, las empresas necesitaban extender sus plataformas tecnológicas para la comunicación entre socios y clientes; se tenía CORBA o DCOM, pero ninguno de los dos dominaba

el mercado, por lo que las empresas que suministraban estos canales de comunicación usaban otras tecnologías (Carmona Barbero, 2014).

Fue entonces a primeros de 2000 cuando apareció el protocolo SOAP ("Simple Object Access Protocol") con interoperatibilidad con XML ("Extensible Markup Language") y REST ("Representional State Transfer") propuesto por Roy Fielding como servicios web para solucionar el problema (Carmona Barbero, 2014).

2.3.2 Servicio web

En su charla Marcos Escover define a un servicio web como: "Un Servicio web es un componente de software que se comunica con otras aplicaciones codificando los mensajes en XML y enviando estos mensajes a través de protocolos estándares de Internet tales como el Hypertext Transfer Protocol (HTTP). Intuitivamente un Servicio web es similar a un sitio web que no cuenta con un interfaz de usuario y que da servicio a las aplicaciones en vez de a las personas. Un Servicio web, en vez de obtener solicitudes desde el navegador y retornar páginas web como respuesta, lo que hace es recibir solicitudes a través de un mensaje formateado en XML desde una aplicación, realiza una tarea y devuelve un mensaje de respuesta también formateado en XML." (Marcos Escover, 2013).

Los servicios web han ido evolucionando a través del tiempo y se han incrementado las posibilidades de comunicación entre aplicaciones codificando los mensajes en más formatos, no solo en XML sino también en formato JSON que actualmente le está ganando terreno a XML por su eficiencia y menor consumo de recursos (Nurzhan Nurseitov, 2009).

En la Figura 5 se muestra un diagrama de integración de servicios web usando Google y Salesforce.

Figura 5. Integración de servicios Web

Fuente: (Schafer Group, 2011)

2.3.2.1 RESTFul

Roy T. Fielding lo define como un conjunto coordinado de restricciones arquitectónicas que intentan minimizar la latencia y la comunicación de redes mientras que, al mismo tiempo, maximiza la independencia y escalabilidad de los componentes de implementación. REST habilita la captura y reúso de interacciones, sustituyendo dinámicamente los componentes y procesando las acciones por medio de intermediarios (Fielding & Taylor, 2000).

La Transferencia de Estado Representacional ("REpresentational State Transfer") es un estilo arquitectónico y un enfoque en las comunicaciones que suelen ser usadas por los servicios web, estos son usados por los navegadores y puede ser pensado como el lenguaje del internet (Rouse, 2014).

En REST se distinguen tres clases de elementos arquitectónicos:

- Elemento de datos,
- Elementos de conexión (conectores) y
- Elementos de procesamiento (componentes).

Elemento de datos: Un aspecto clave de REST es el estado de los elementos de datos, sus componentes se comunican mediante la transferencia de representaciones del actual estado o el deseado de los mismos (Jakl, 2005).

Una abstracción clave es el recurso "resource". Todo lo que puede ser nombrado puede ser recurso. Un recurso es un mapeo conceptual de un grupo de

entidades, solo la semántica del mismo es requerida para ser estática ya que una entidad puede cambiar a lo largo del tiempo (Jakl, 2005).

Todos los componentes REST realizan acciones en la representación de los recursos. Estas capturan el estado actual o deseado de un recurso y puede ser transferido entre componentes. Esta representación consiste en: una secuencia de bytes (el contenido), la representación de metadata (descripción del contenido) y la metadata describiendo la metadata (ej. “hash sums” o “cache-control”) (Jakl, 2005).

Conectores: Administran la comunicación de red para un componente (Jakl, 2005).

Componentes: Son identificados mediante su rol en la aplicación, como: “user agent”, el cual usa una conector cliente para iniciar las peticiones y se convierte en el último recipiente de la respuesta (Jakl, 2005).

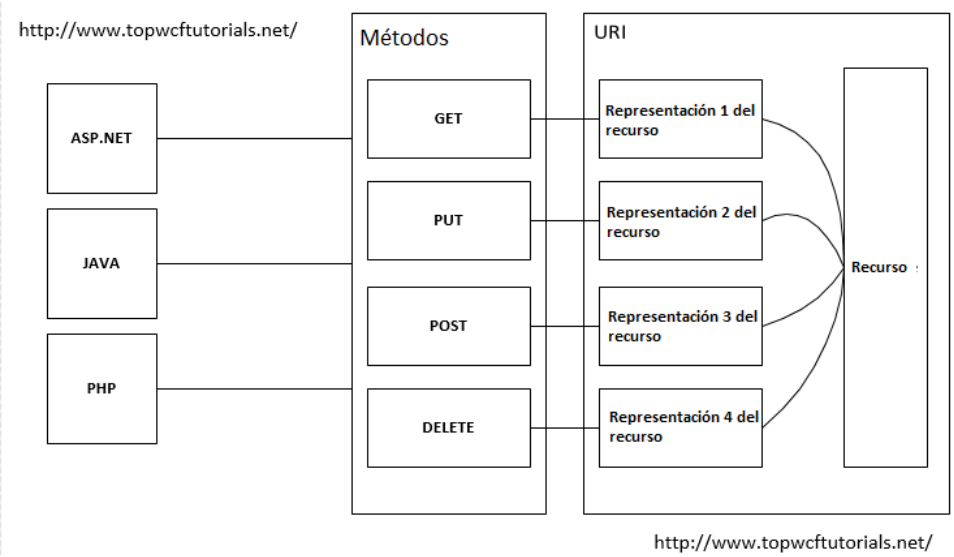

Figura 6 Diagrama básico de un servicio REST

Fuente: (Abdul, 2013)

2.3.3 Asana

Asana es una aplicación web y móvil diseñada para mejorar la comunicación y colaboración en equipo mediante la asignación de tareas y seguimiento de las mismas (Asana, 2015).

En la Figura 6 se muestra la pantalla principal de asana con las tareas asignadas y en diferentes dispositivos.

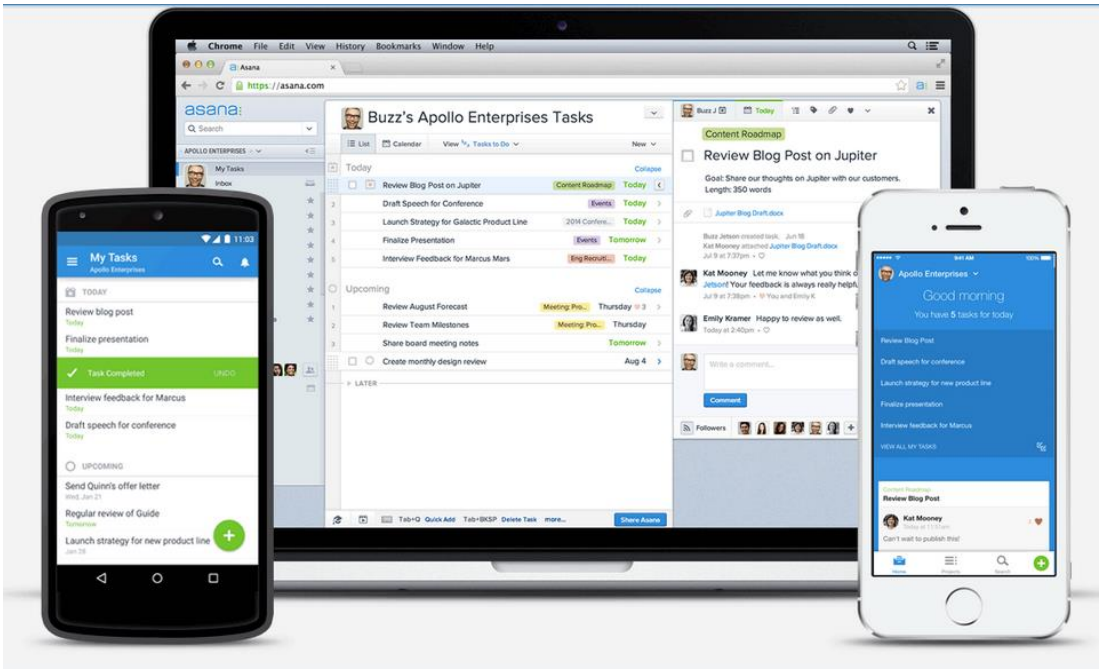

Figura 7. Pantalla principal de Asana

Fuente: (Asana, 2015)

2.3.4 Amazon Web Services (AWS)

Amazon AWS ofrece un amplio conjunto de servicios globales de computación, almacenamiento, bases de datos, análisis, aplicaciones e implementaciones que ayudan a las organizaciones a avanzar con más rapidez, reducir costos de TI y escalar aplicaciones. Estos servicios tienen la confianza de las mayores compañías y las empresas emergentes más innovadoras para respaldar una amplia variedad de cargas de trabajo, como las aplicaciones web y móviles, etc. (Amazon AWS, 2016).

2.3.4.1 Amazon Elastic Compute Cloud (EC2)

Amazon EC2 es un servicio web que ofrece una plataforma de cómputo en la nube. Este servicio como muchos que ofrece Amazon tiene dos grandes ventajas: La primera nos da la capacidad de ajustar la capacidad de nuestro servicio (Memoria

RAM, Procesador, etc) y la segunda es que se cobra el servicio por el porcentaje de uso, es decir se paga por lo que se consume (Amazon AWS, 2016).

Amazon EC2 permitirá alojar el servidor de aplicaciones el mismo que se implementará sobre el sistema operativo Amazon Linux.

2.3.4.2 Amazon Relational Database Services (RDS)

Amazon RDS es un servicio administrado de base de datos relacional que pone a su disposición seis motores de base de datos conocidos entre los que puede elegir: Amazon Aurora, MySQL, MariaDB, Oracle, Microsoft SQL Server y PostgreSQL (Amazon AWS, 2016).

Además de la misma forma que con Amazon Web Services, no se requiere ningún tipo de inversión inicial y únicamente se tendrá que pagar por los recursos que se utilice (Amazon AWS, 2016).

Amazon RDS se encargará de alojar las bases de datos para los distintos ambientes como son pruebas, producción y demostración de la aplicación.

2.3.4.3 Amazon Simple Storage Services (S3)

Amazon S3 almacena datos como objetos dentro de recursos conocidos como "buckets". Además de almacenar todos los objetos que desee dentro de un bucket, podrá realizar operaciones de escritura, lectura y eliminación de los objetos almacenados en el contenedor (Amazon AWS, 2016).

Este servicio es el encargado de almacenar el cliente de la aplicación, es decir es el que se conectará con el servidor de aplicaciones alojado en Amazon EC2, que a su vez se conectará con la base de datos almacenada en Amazon RDS.

2.3.4.4 Amazon Simple Email Service (SES)

Amazon SES es un servicio de envío de correo electrónico, el cual nos permitirá enviar las distintas notificaciones generadas por la aplicación a los distintos usuarios generadores de la información (Amazon AWS, 2016).

2.3.4.5 Amazon CloudFront

Amazon CloudFront es un servicio de red de entrega de contenidos, para poder acceder de una manera más rápida al sitio web de la aplicación (Amazon AWS, 2016).

2.3.4.6 Amazon Route53

Amazon Route 53 es un servicio web DNS (Sistema de nombres de dominio) escalable y de alta disponibilidad en la nube. Está diseñado para redirigir a los usuarios finales a las aplicaciones en Internet convirtiendo nombres legibles para las personas como www.example.com en direcciones IP numéricas como 192.0.2.1 que utilizan los equipos para conectarse entre ellos (Amazon AWS, 2016).

2.4 Plataforma JEE

Oracle lo define como: "Conjunto de especificaciones y prácticas coordinadas que juntas permiten soluciones para el desarrollo, despliegue y gestión de aplicaciones multicapa centradas en servidor". En la Figura 7 se puede visualizar el esquema de la plataforma JEE (Oracle, 2014).

Figura 8. Esquema de JEE
Fuente: (Oracle, 2014)

2.4.1 Historia

Fue desarrollada por Sun Microsystems, a continuación, se detalla la historia de la plataforma JEE.

- **1996 -** Java Development Kit (JDK) 1.02: colección ordenada de bibliotecas de clases y paquetes.

- **1999 -** JDK1.2, Java 2 Platform: adicional al JDK, paquetes opcionales para mensajes, generación dinámica de páginas Web o programas de email en Java. Dividida en 3 ediciones:
    - Java 2 Platform, Standard Edition (J2SE): contiene el JDK actual y las APIs estándar. Desarrollo de aplicaciones de Desktop y applets.
    - Java 2 Platform, Enterprise Edition (J2EE): basada en J2SE, extiende el lado del servidor.
    - Java 2 Platform, Micro Edition (J2ME): especial para móviles, pagers, palmtops (embedded systems).
- **2006 -** Aparece la especificación Java EE 5.0, desarrollada bajo el JSR244. Cambia la denominación de la tecnología: de J2EE a JEE
- **2009 -** Aparece Java EE 6.0
- **2013 –** Finalmente aparece Java EE 7.0

2.4.2 Arquitectura

La arquitectura JEE soporta el desarrollo de aplicaciones n-capas para el lado del servidor, utilizando diferentes componentes como son: EJBs, Containers, APIs, etc. En la Figura 8 se puede observar la arquitectura de aplicación de J2EE.

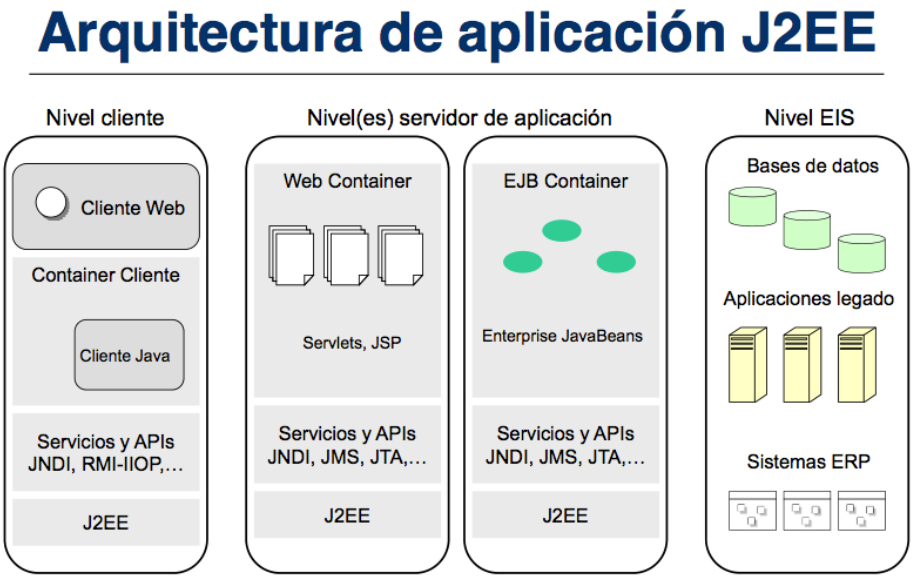

Figura 9. Arquitectura J2EE

Fuente: (UNIVERSIDAD CARLOS III DE MADRID, 2009)

2.4.3 Modelo Vista Controlador (MVC)

El patrón de arquitectura MVC (Modelo Vista Controlador) es un patrón que define la organización independiente del Modelo (Objetos de Negocio), la Vista (interfaz con el usuario u otro sistema) y el Controlador (controlador del workflow de la aplicación) (Ejemplos TIW, 2012).

De esta forma, dividimos el sistema en tres capas donde, como explicaremos más adelante, tenemos la encapsulación de los datos, la interfaz o vista por otro y por último la lógica interna o controlador (Ejemplos TIW, 2012).

El patrón de arquitectura "modelo vista controlador", es una filosofía de diseño de aplicaciones, compuesta por:

- **Modelo**
  - Contiene el núcleo de la funcionalidad (dominio) de la aplicación.
  - Encapsula el estado de la aplicación.

- **Vista**
  - Es la presentación del Modelo.
  - Puede acceder al Modelo, pero nunca cambiar su estado.
  - Puede ser notificada cuando hay un cambio de estado en el Modelo.
- **Controlador**
  - Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente (Ejemplos TIW, 2012).
  - Para entender cómo funciona nuestro patrón Modelo vista controlador, se debe entender la división a través del conjunto de estos tres elementos y como estos componentes se comunican unos con los otros y con otras vistas y controladores externos a el modelo principal. Para ello, es importante saber que el controlador interpreta las entradas del usuario, enviado el mensaje de acción al modelo y a la vista para que se proceda con los cambios que se consideren adecuados (Ejemplos TIW, 2012).

A continuación, en la Figura 9 se muestra la interacción e integración del modelo, la vista y el controlador.

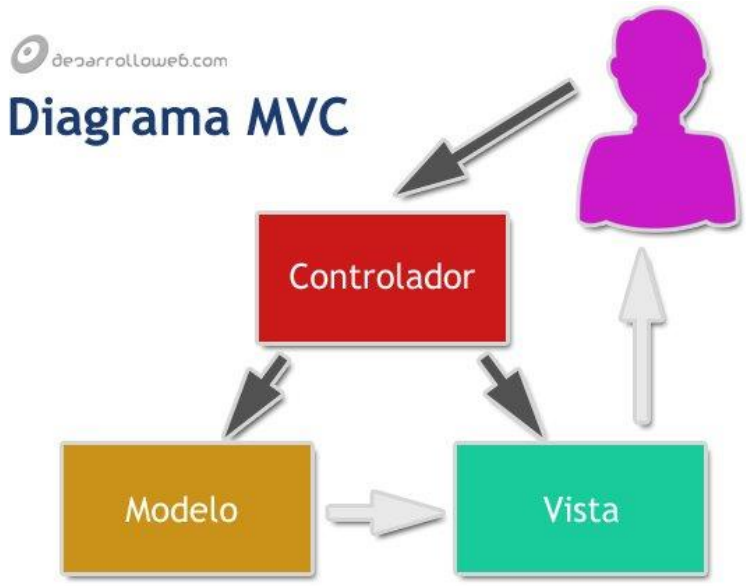

Figura 10. Interacción e integración MVC
Fuente: (Alvarez, 2014)

2.5 AngularJS

Es un framework de JavaScript para la creación de aplicaciones web, extendiendo la funcionalidad de HTML para lograr un desarrollo más ágil. Una de sus ventajas es la utilización de una programación declarativa para un mejor rendimiento en la carga de páginas web en el navegador (Google, 2015).

2.5.1 Compilador HTML de Angular

Permite al desarrollador enseñar al navegador nueva sintaxis HTML, este ayuda a unir comportamiento a cualquier elemento o atributos e inclusive crear nuevos con los comportamientos modificados. Angular llama a estas extensiones de comportamientos *directivas* (Google, 2015).

HTML tiene muchos constructores para formatear el documento en una manera declarativa. Por ejemplo, si algún elemento necesita centrado, no es necesario proveer instrucciones al navegador de cómo el tamaño de la ventana necesita ser dividido en la mitad para encontrar el centro y este necesita estar alineado con el centro del texto.

Simplemente se añade un atributo “align="center"” a cualquier elemento para lograr el comportamiento deseado (Google, 2015).

Angular viene pre-empaquetado con directivas comunes las cuales son útiles para construir cualquier aplicación. Toda esta compilación toma lugar en el navegador; no del lado del servidor (Google, 2015).

El compilador es un servicio de Angular el cual recorre el DOM para buscar atributos. El proceso de compilación se lleva a cabo en dos fases:

1. **Compilar:** Recorre el DOM y recolecta todas las directivas. El resultado es una función de enlace.
2. **Enlace:** Combina las directivas con el “scope” y produce una vista en vivo. Cualquier cambio en el modelo del “scope” son reflejados en la vista y cualquier interacción de un usuario en la vista es reflejada en el modelo del “scope”.

Es importante notar que Angular opera en los nodos DOM en lugar de cadenas de texto. Usualmente, no se aprecia esta restricción porque, cuando la página carga, el navegador convierte el HTML en DOM automáticamente (Google, 2015).

La compilación del HTML ocurre en tres fases:

1. “$compile” recorre el DOM y empareja las directivas. Si el compilador encuentra que un elemento coincide con una directiva, entonces esta es añadida a la lista de directivas. Un elemento puede coincidir con múltiples directivas.
2. Una vez que se han identificado que todas las directivas coinciden con el elemento del DOM, el compilador ordena estas directivas por prioridad.
   Cada función de compilación no tiene la oportunidad de modificar el DOM. Cada función de compilación retorna una función de enlace (“link function”). Estas funciones son compuestas en una función de enlace “combinada”, la cual invoca una “link function” para cada directiva.

3. $compile enlaza la plantilla con el “scope” llamando a la función de enlace combinada del paso anterior. Esta llamará la función de enlace de cada directiva, registrando escuchas (“listeners”) en el elemento.

2.5.2 Expresiones

Son fragmentos de código que son colocados principalmente en los enlaces de interpolación, como por ejemplo “{{ textBinding }}” , pero también se puede utilizar directamente una directiva como “ng-click="functionExpression()"” (Google, 2015).

A continuación, se muestran algunos ejemplos válidos de expresiones:

- 1+2
- a+b
- user.name
- items[index]

Aparte del operador ternario a ? b : c no se pueden escribir sentencias de control de flujo, la razón es que la lógica debe estar en el controlador, no en las vistas. Si es necesario una condicional real o un bucle, se debe delegar a un método de JavaScript (Google, 2015).

2.5.3 Módulos

Se puede definir como un contenedor de las diferentes partes de la aplicación: controladores, servicios, filtros, etc. Los módulos especifican cómo la aplicación debe iniciar, existen algunas ventajas con este enfoque:

- El proceso declarativo es fácil de entender.
- Se puede empaquetar el código en módulos reusables.
- Pueden ser cargados en cualquier orden o paralelamente.

Un módulo es una colección de bloques de configuración y ejecución los cuales son aplicados durante el proceso de arranque de la aplicación. En su forma más sencilla los módulos constan de dos tipos de bloques:

- **Bloque de configuración:** Son ejecutados en la fase del registro de los proveedores y configuración. Solo proveedores y constantes pueden ser inyectadas en este bloque. Esto se realiza para prevenir que algunos servicios accidentalmente sean inicializados sin estar completamente configurados.
- **Bloque de ejecución:** Son ejecutados después que el inyector es creado y usados para el inicio de la aplicación. Solo instancias y constantes pueden ser inyectadas en este bloque. Esto se realiza para evitar una mayor configuración del sistema durante la ejecución de la aplicación.

```
angular.module('myModule', []).
config(function(injectables) { // provider-injector
  // Este es un ejemplo de un bloque de configuración.
  // Se puede tener cuántos sean necesarios.
  // Solo se pueden inyectas proveedores (providers)
  // dentro de estos bloques.
}).
run(function(injectables) { // instance-injector
  // Este es un ejemplo de un bloque de ejecución.
  // Se puede tener cuántos sean necesarios.
  // Solo se pueden inyecter instancias (no proveedores)
  // dentro de estos bloques
});
```

Figura 11 Ejemplo para declarar el bloque de configuración y ejecución

Fuente: (Google, 2015)

2.5.3.1 Dependencias

Los módulos pueden listar otros, como sus dependencias. La dependencia de un módulo implica que el módulo requerido debe ser cargado antes de que el módulo solicitante lo haga. Cada uno de estos puede ser cargado una sola vez, incluso si múltiples módulos lo quieren (Google, 2015).

2.5.3.2 Creación y recuperación

Para crear un módulo se debe usar la sentencia angular.module('myModule', []) , para recupéralo angular.module('myModule'). Se

debe tener mucho cuidado en estas sintaxis ya que se puede sobrescribir todo el módulo en lugar de recuperarlo (Google, 2015).

2.5.4 Directivas

Son marcadores del DOM (Document Object Model) como un atributo, nombre del elemento, comentario o clase CSS, que le dice al compilador de HTML de Angular para atar un comportamiento especificado a ese elemento del DOM, o incluso para transformarlos (Google, 2015).

Angular cuenta con una serie de directivas incorporadas, como “ngBind”, “ngModel” y “ngClass”. Así como se pueden crear controladores y servicios, se pueden crear directivas para que Angular pueda usarlas. Cuando Angular arranca la aplicación, el compilador HTML atraviesa el DOM haciendo un match contra los elementos del DOM (Google, 2015).

2.5.5 Enlace de datos

Es la sincronización automática entre los datos del modelo y los componentes de la vista. La manera en que Angular implementa el enlace de datos permite que el modelo sea tratado como un solo recurso en la aplicación. La vista es una proyección del modelo en todo tiempo, es decir, si el modelo cambia, estos son reflejados en dicha vista y viceversa (Google, 2015).

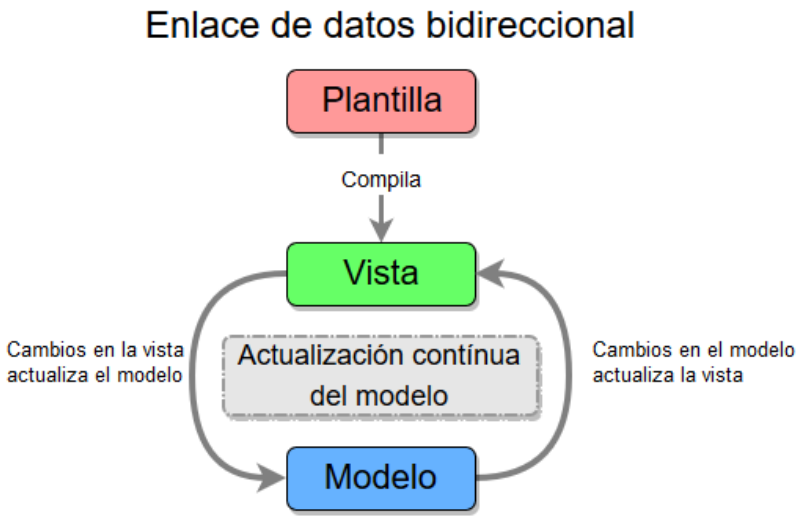

Figura 12 Enlace de datos entre una plantilla, la vista y el modelo

Fuente: (Google, 2015)

2.5.6 Controladores

Un controlador es definido como una función constructora de JavaScript que es usado como aumento de “Angular Scope”. Cuando un controlador es atado al DOM vía la directiva “ng-controller”, Angular va a inicializar un nuevo objeto “Controller”, usando la función constructora del controlador especificado. Un nuevo “child scope” se creará y permitirá como parámetro inyectable de la función constructora del controlador como “$scope” (Google, 2015).

Típicamente, cuando se crea una aplicación se necesita configurar el estado inicial para el “$scope” de Angular, esto se realiza atando las propiedades al objeto “$scope”. Las propiedades contienen el “view model” (el modelo que va a ser presentado por la vista) (Google, 2015).

Para poder reaccionar a eventos o ejecutar operaciones en la vista, se debe proveer de un comportamiento al “scope”. Un comportamiento puede ser un método y estos métodos estarán disponibles en las plantillas y/o vistas (Google, 2015).

En el siguiente ejemplo se usa el controlador para añadir un método, el cuál divide un número para dos:

```
var myApp = angular.module('myApp',[]);

myApp.controller('HalfController', ['$scope', function($scope) {
  $scope.half = function(value) { return value / 2; };
}]);
```

Figura 13 Se añade al scope el método para dividir un número a la mitad

Fuente: (Google, 2015)

```
<div ng-controller="HalfController">
  The half of <input ng-model="num"> is {{ half(num) }}
</div>
```

Figura 14 El método “half” puede ser invocado con una expresión de Angular

Fuente: (Google, 2015)

Un controlador solo debe contener la lógica del negocio que se necesita para una simple vista. El camino más común para mantener el controlador sólo con lo necesario es encapsular el trabajo que no debe realizar en los servicios y usar estos servicios vía inyección de dependencia (Google, 2015).

2.5.7 Scopes

Es un objeto que hace referencia al modelo de la aplicación. Es un contexto de ejecución para expresiones; estos arreglados en una estructura jerárquica con la estructura DOM de la aplicación. Los "scopes" pueden observar expresiones y propagar eventos (Google, 2015).

El "scope" es un pegamento entre el controlador de la aplicación y la vista. Durante la fase de enlace de la plantilla las directivas preparan las expresiones "$watch" en el "scope". El "$watch" permite a la directiva ser notificada de los cambios de las propiedades, lo que a su vez admite que la directiva renderice el valor actualizado para el DOM (Google, 2015).

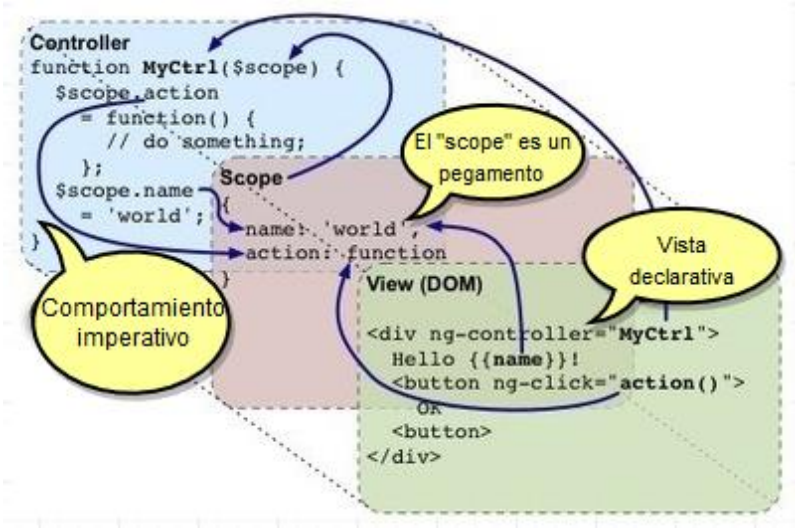

Figura 15 Interacción del "scope" con el controlador y la vista

Fuente: (Google, 2015)

Ambos controladores y directivas tienen referencia al *scope*, pero entre sí. Este arreglo asila el controlador de la directiva así como del DOM. Esto es un punto importante ya que hace la vista de los controladores agnóstica (Google, 2015).

```
angular.module('ejemploScope', [])
.controller('MiControlador', ['$scope', function($scope) {
  $scope.nombreUsuario = 'Mundo';

  $scope.decirHola = function() {
    $scope.saludo = '¡Hola ' + $scope. nombreUsuario + '!';
  };
}]);

<div ng-controller="MiControlador">
  Tu nombre:
    <input type="text" ng-model="nombreUsuario">
    <button ng-click='decirHola()'>saludar</button>
  <hr>
  {{saludo}}
</div>
```

Figura 16 Ejemplo del funcionamiento del “scope”.

Fuente: (Google, 2015)

En la figura anterior se puede apreciar que “MiControlador” asigna “Mundo” a la propiedad “nombreUsuario” del “scope”. El “scope”entonces notifica al “input” de la asignación, la cual renderiza el input con el “nombreUsuario” pre-llenado. Esto demuestra como un controlador escribe datos dentro del “scope” (Google, 2015).

De manera similar el controlador puede asignar un comportamiento al “scope” como es visto por el método “decirHola”, el cual es invocado cuando el usuario realiza un clic en el botón “saludar”. Este método puede leer la propiedad “nombreUsuario” y crear la propiedad “saludo”. Esto demuestra que las propiedades del “scope” se actualizan automáticamente cuando son destinadas a las entradas del HTML (Google, 2015).

Lógicamente la renderización de “{{saludo}}” involucra:

1. Obtener el “scope” asociado con el nodo DOM donde “{{saludo}}” es definido en la plantilla.
2. Evaluar la expresión “saludo” contra el “scope” obtenido en el paso anterior y asigna el resultado al texto del elemento DOM.

Cada aplicación tiene exactamente un “root scope”, pero puede tener varios “scopes” hijos, porque algunas directivas crean nuevos hijos. Cuando un nuevo

"scope" es creado, se añade como hijo. Esto crea una estructura de árbol la cual es paralela al DOM donde son unidas (Google, 2015).

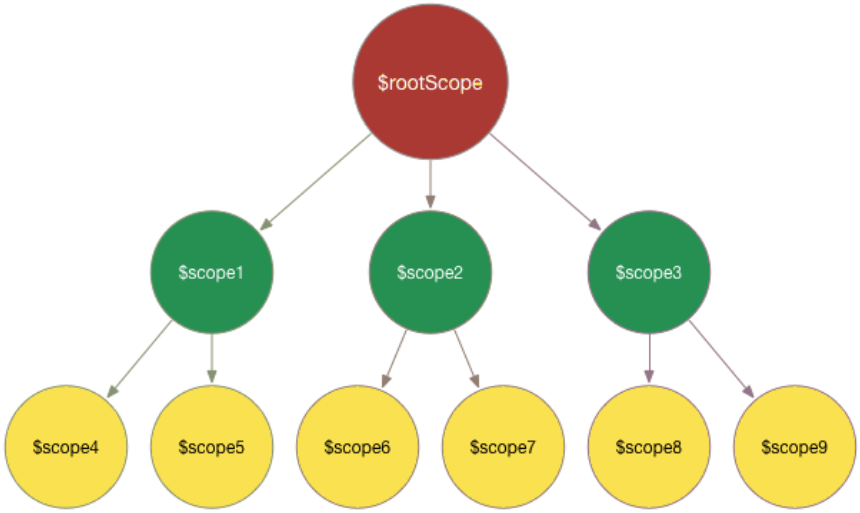

Figura 17 Árbol de "scopes"
Fuente: (Craven, 2015)

Cuando Angular evalúa por ejemplo "{{nombre}}", primero busca el "scope" asociado con el elemento dado por la propiedad "nombre". Si la propiedad no es encontrada, busca el "scope" padre y continúa hasta que alcanza el "root scope" (Google, 2015).

2.5.7.1 Características

Acorde a Google las características de los "scopes" son las siguientes:

- Proveen de APIs para observar los cambios en los modelos, como "$watch".
- Proveen de APIs para propagar cualquier cambio en el modelo a través del sistema hacia la vista, como "$apply".
- Pueden ser anidados para limitar el acceso a los componentes de la aplicación mientras proveen acceso a las propiedades compartidas del modelo. Un "scope" anidado pueden ser "child scopes" o "isolate scopes". Un "child scope" tiene propiedades heredadas del "scope" padre, mientras que un "isolate scope" no.
- Proveen contextos contra cuál de las expresiones son evaluadas. Por ejemplo la expresión "{{nombreUsuario}}" no tiene significado, a menos que sea

evaluado en un “scope” específico con que se define la propiedad “nombreUsuario”.

2.5.7.2 Ciclo de vida

El flujo normal de un navegador en la recepción de un evento es que se ejecuta un correspondiente “callback” de JavaScript. Una vez dicho “callback” se complete, el navegador re-renderiza el DOM y lo retorna para la espera de más eventos (Google, 2015).

Cuando un navegador llama dentro de JavaScript el código, este se ejecuta fuera del contexto de ejecución de Angular, lo que quiere decir que Angular no es consciente de las modificaciones del modelo. Para procesar adecuadamente estas modificaciones la ejecución tiene que entrar en el contexto de Angular usando el método “$apply”; solo de esta manera Angular será consciente de los cambios (Google, 2015).

Después de evaluar la expresión, el método “$apply” realiza un “$digest”. En la fase “$digest” el “scope” examina las expresiones del “$watch” y las compara con el valor previo. Este chequeo es realizado asincrónicamente. Por lo tanto esa asignación como $scope.username="angular" no será una causa inmediata de que “$watch” sea notificado, en su lugar la esta notificación es retrasada hasta la fase del “$digest”. Este retraso es deseable, ya que se fusionan las múltiples actualizaciones de los modelos dentro de una notificación “$watch” así como garantiza que durante dicha notificación ningún otro “$watch” este corriendo. Si un “$watch” cambia el valor del modelo, será forzado a realizar un ciclo “$digest” adicional (Google, 2015).

De acuerdo a Google el ciclo de vida de los “scopes” son:

1. **Creación:** El “root scope” es creado durante el arranque de la aplicación vía “$injector”. Durante el enlace de la plantilla, algunas directivas crean nuevos “child scopes”.
2. **Registro de “watcher”:** Durante el enlace de la plantilla, las directivas registran “watches” en el “scope”. Estos “watches” son usados para para propagar los valores del modelo hacia el DOM.

3. **Cambio del modelo:** Para que los cambios sean adecuadamente observados, se deben realizar solo con el "scope.$apply()". Las APIs de Angular lo realizan implícitamente, por lo tanto no es necesario una llamada extra de cuando se está realizado el trabajo en los controladores o en algunos servicios.
4. **Observación del cambio:** Al final de "$apply", Angular realiza el ciclo "$digest" en el "root scope", por lo tanto se propaga a través de todos los "child scopes". Durante el ciclo "$digest", todas las expresiones "$watch" o funciones son revisadas para detectar cambios en los modelos, si se detecta algún cambio el "$watch listener" es llamado.
5. **Destrucción del "scope":** Cuando un "child scope" ya no es necesitado, es responsabilidad de su creador destruirlo vía la API "scope.$destroy()". Esto detendrá la propagación de las llamadas "$digest" dentro del "child scope" y permite que la memoria usada por el mismo sea reclamada por el recolector de basura.

2.5.8 Filtros

Los filtros dan formato a los valores de una expresión para ser mostrada al usuario. Pueden ser usadas en plantillas, controladores o servicios. Angular permite el uso de una colección de filtros incorporados como: "currency, json, uppercase,etc.", pero se pueden definir nuevos (Google, 2015).

2.5.9 Servicios

Son objetos sustituibles que se conectan usando inyección de dependencia, estos se pueden usar para organizar y compartir código a través de la aplicación. Estos servicios son inicializados solo cuando son requeridos o dependientes de un componente de la aplicación y cada uno de estos componentes obtiene una única instancia generada por el "service Factory" (Google, 2015).

```
var miModulo = angular.module('miModulo', []);
miModulo.factory('servicioId', function() {
  var nuevaInstanciaServicio;
  // cuerpo de la función “factory” que construye nuevaInstanciaSer
  return nuevaInstanciaServicio;
});
```

Figura 18 Registro de un servicio

Fuente: (Google, 2015)

2.6 Arquitectura orientada a microservicios

2.6.1 Microservicios

Sam Newman lo define como: “Los microservicios son pequeños y autónomos” (Newman, 2015).

Son aplicaciones pequeñas que funcionan de manera independiente que realizan sus funciones sin depender de otra aplicación, lo único que hacen es acceder a los servicios de otra para procesos específicos en la Figura 19 se puede visualizar la arquitectura básica de microservicios (Alvarez, 2014).

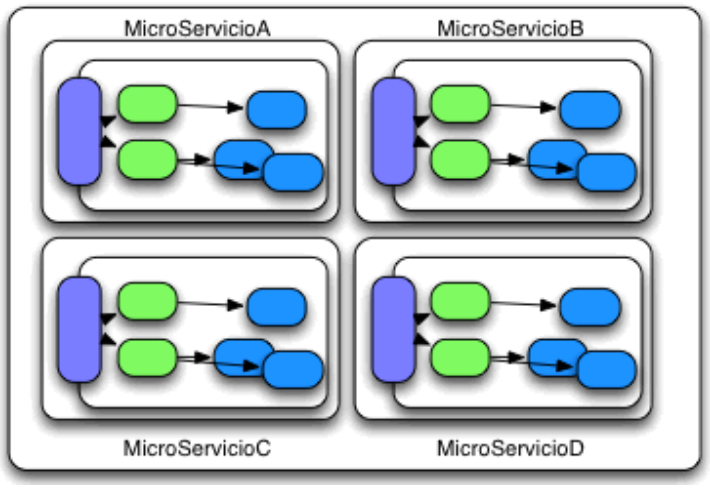

Figura 19 Arquitectura básica de microservicios

Fuente: (Álvarez, 2015)

Al ser cada microservicio independiente existen varias ventajas. Primero es más fácil su desarrollo ya que son relativamente pequeños y fáciles de abordar. Segundo si un microservicio falla no afecta a los demás ya que son totalmente independientes. Por otro lado, facilita el despliegue ya que no hay que desplegar aplicaciones gigantescas sino "microservicios" de tal forma que si solo se debe que actualizar uno de ellos pues solo redesplegamos este. Por otro lado, y una de las cosas más importantes podemos acceder a estos microservicios desde todo tipo de dispositivo ya que los tenemos publicados vía REST o algún tipo de RPC, en la Figura 20 se puede ver la forma como se accede a los distintos microservicios (Álvarez, 2015).

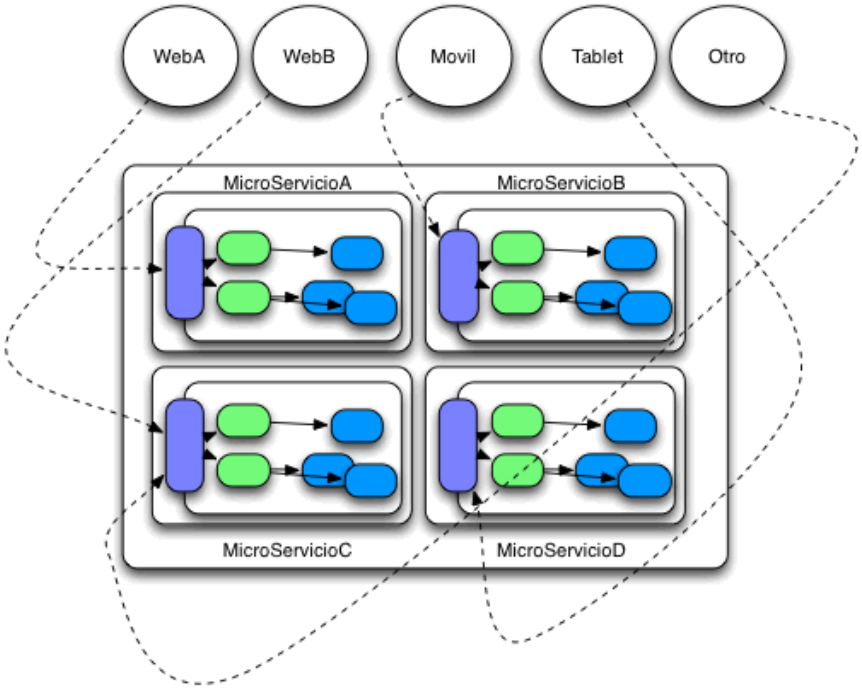

Figura 20 Acceso a microservicios

Fuente: (Álvarez, 2015)

2.6.2 Heterogeneidad de tecnologías

Con un sistema con múltiples módulos, se puede elegir diferentes tecnologías para cada uno. Esto permite elegir una herramienta adecuada para cada tipo de trabajo, en lugar de seleccionar una más estandarizada que a la final puede terminar por complicar el desarrollo para ciertas tareas. (Newman, 2015)

Si una parte del sistema necesita mejorar el rendimiento se puede optar por el uso de una tecnología que permita alcanzar el nivel requerido de dicho rendimiento. También se puede elegir cómo se guardan los datos en las diferentes partes del sistema. En la figura 21 se puede apreciar el uso heterogéneo de tecnologías. (Newman, 2015)

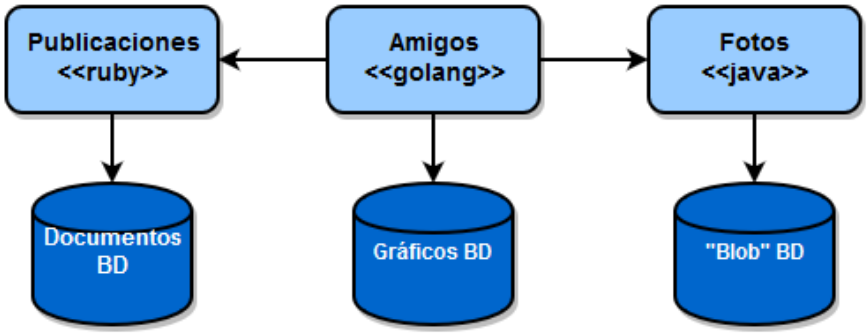

Figura 21 Los microservicios ayudan a integrar diferentes tecnologías

Fuente: (Newman, 2015)

2.6.3 Escalabilidad

Con un sistema monolítico se debe escalar todo junto, pero con pequeños servicios se puede realizar el escalamiento sólo en los que se necesiten, permitiendo ejecutar el sistema en un más pequeño y menos poderoso hardware, como se muestra en la figura 22 (Newman, 2015).

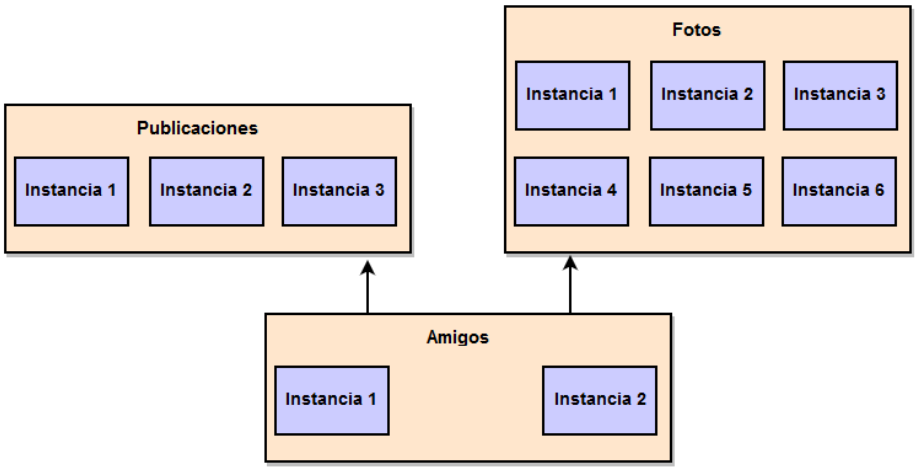

Figura 22 Se puede seleccionar qué microservicio escalar

Fuente: (Newman, 2015)

2.6.4 Modelar un servicio

Los esfuerzos para modelar un buen servicio se basan en dos conceptos los cuales son: acoplamiento débil y alta cohesión (Newman, 2015).

2.6.4.1 Acoplamiento débil

Cualquier cambio que se realice a un servicio con un acoplamiento débil no requerirá realizar un cambio en otro. Todo el objetivo de un microservicio es la posibilidad de ser cambiado y desplegado, sin necesidad de cambiar otra parte del sistema (Newman, 2015).

2.6.4.2 Alta cohesión

Sebe deben establecer límites junto con el dominio de problema para ayudar a asegurar que los comportamientos relacionados se encuentren en un solo lugar y la comunicación con otros límites sea lo más débil posible (Newman, 2015).

2.6.5 Contextos acotados

Es el patrón principal en el Domain-Driven Design, trabaja con largos modelos que se dividen en contextos (bounded contexts) y comienzan a ser explícitos en sus interrelaciones. Se trata de separar elementos comunes en los denominados modelos y determinar la relación que tiene entre ellos, en la siguiente figura se muestra un ejemplo de contextos acotados (Fowler, 2014).

Para un mejor entendimiento, un contexto es como la membrana de las células, las mismas que separan el exterior del interior y determinan que puede pasar (Newman, 2015).

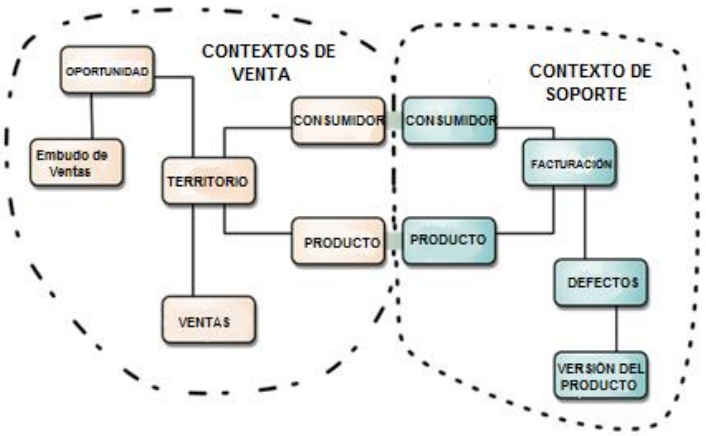

Figura 23 Contextos acotados (bounded conexts)

Fuente: (Fowler, 2014)

2.6.6 Integración

Existen muchas tecnologías que permiten la integración de los microservicios, ya sea mediante llamadas remotas de procedimientos (RPC), por sus siglas en inglés Remote Procedure Calls, REpresentational State Transfer (REST), etc.

REST y HTTP permite el uso de diferentes formatos textuales, esto brinda al cliente mayor flexibilidad para consumir los recursos. Uno de ellos es JSON y debido a su simplicidad el consumo es mucho más fácil (Newman, 2015).

2.6.7 Composición de la API

Una vez que se define el tipo de integración de los microservicios, se construye la interfaz de usuario que interactúa directamente con la API. Una interfaz basada en la web puede usar peticiones GET mediante JavaScript para obtener datos, o peticiones POST para cambiarlos, como se muestra en la figura 24. Inclusive para aplicaciones móviles nativas, iniciar las comunicaciones HTTP son muy sencillas (Newman, 2015).

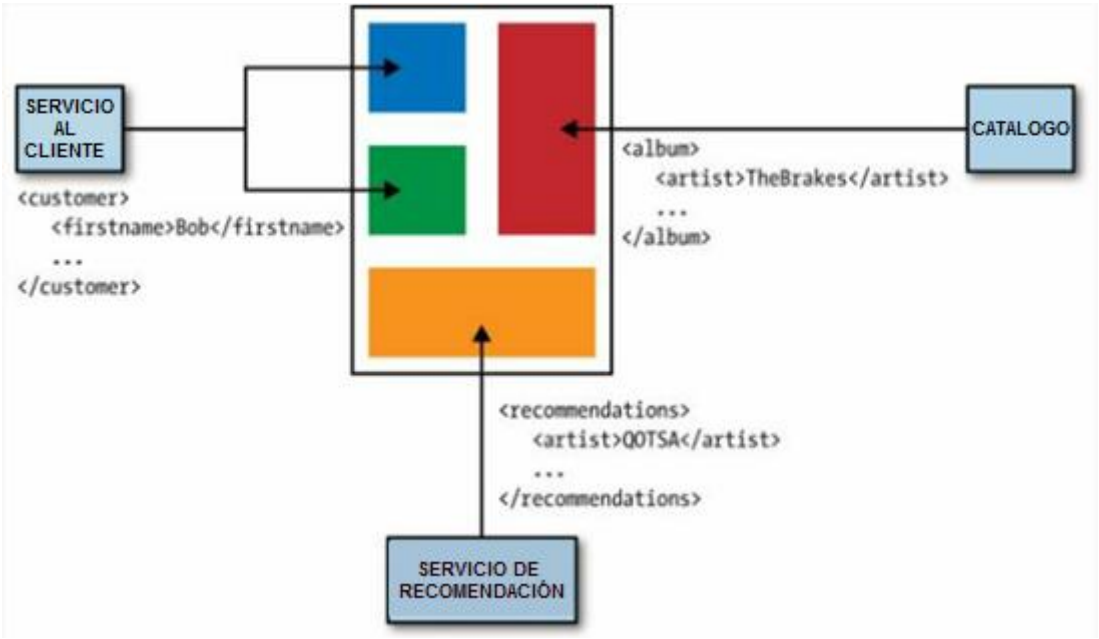

Figura 24 Usando múltiples APIs para presentar una interfaz de usuario

Fuente: (Newman, 2015)

Sin embargo, realizar múltiples llamadas puede ser intensa para los dispositivos móviles y afectar el plan de planos (Newman, 2015).

Para solucionar este problema se puede hacer uso de un Gateway, que permite unir múltiples llamadas a los servicios, exponer una sola al cliente y agregar más dependiendo de los nuevos requerimientos, como se puede apreciar en la siguiente figura (Newman, 2015).

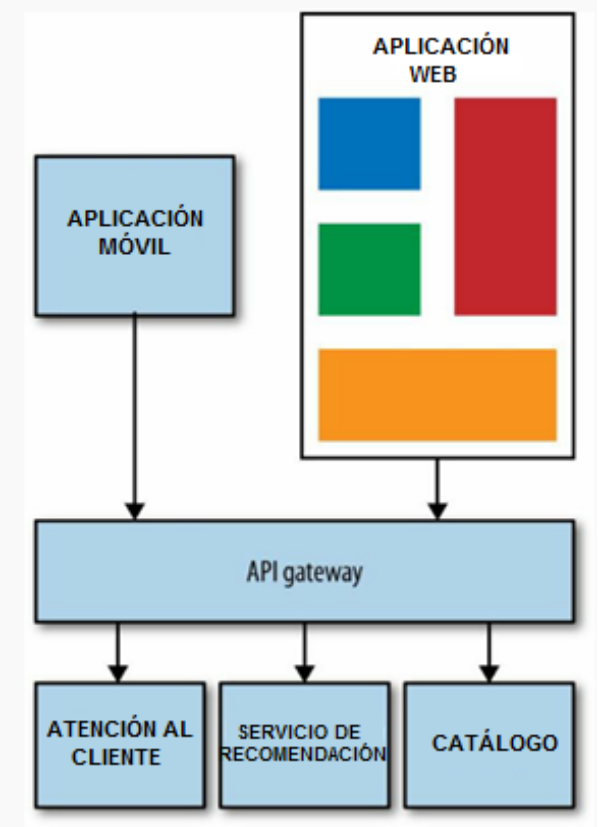

Figura 25. Usando un Gateway para manejar las llamadas a las interfaces
Fuente: (Newman, 2015)

2.6.8 Seguridad

Autenticación y autorización son el centro de los conceptos cuando se trata de personas y cosas que interactúan con el sistema. En el contexto de seguridad, autenticación es el proceso por el cual una parte dice ser quien es. Para un humano, lo más común es autenticarse ingresando un nombre de usuario y una contraseña; se asume que solo ésa parte tiene acceso a dicha información, y por lo tanto la persona que ingresa la misma debe ser ese usuario (Newman, 2015).

Autorización es un mecanismo con el cual se mapea las acciones que tiene el usuario que se ha autenticado, es decir si dicho usuario puede visualizar o manipular cierta parte de la información del sistema (Newman, 2015).

2.6.8.1 HTTPS

El protocolo seguro de transferencia de hipertexto está destinado a que los datos sean enviados por un canal seguro. De esta manera el cliente gana una fuerte garantía

de que el servidor está comunicándose con él. Además, brinda una protección adicional contra las personas que espían el tráfico entre el cliente y el servidor. (Callegati, Cerroni, & Marco, 2009)

2.6.8.2 Autenticación basada en tokens

Permite al usuario que ingrese el nombre de usuario y contraseña para poder obtener un token que le permite acceder a un recurso específico. En otras palabras en lugar de autenticarse con nombre de usuario y contraseña para cada recurso protegido, el usuario se autentica de esta manera una sola vez, con un límite de sesión activa, obtiene un token de igual manera con un límite de duración y es usado para la autenticación durante la sesión. El esquema básico se muestra en la figura 26.

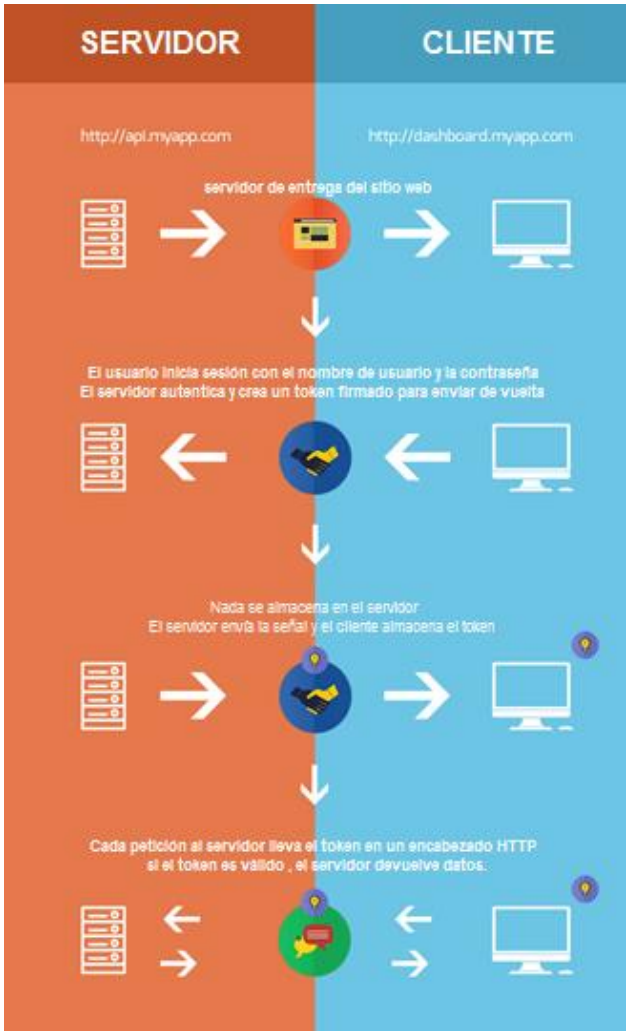

Figura 26. Diagrama básico de autenticación por token

2.6.9 Principio de los microservicios

Estos principios ayudan a crear pequeños autónomos servicios que trabajar bien juntos, estos se muestran en la figura 27. (Newman, 2015)

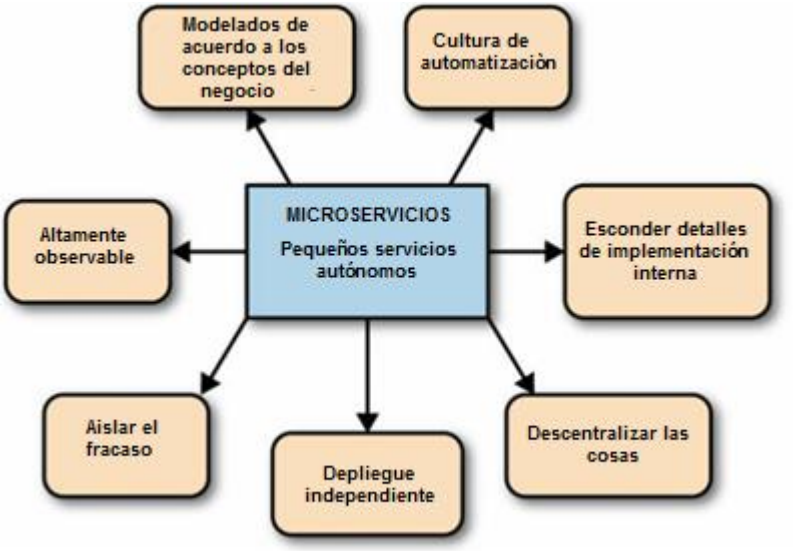

Figura 27 Principios de los microservicios

Fuente: (Newman, 2015)

2.6.9.1 Modelados de acuerdo a los conceptos del negocio

Las interfaces estructuradas alrededor del contexto de los límites del negocio son más estables que aquellas que son estructuradas en conceptos técnicos. Modelando el dominio en el cual el sistema opera, no solo se forman más interfaces estables, sino que se asegura la capacidad de reflejar los cambios en los procesos de negocio más fácilmente. (Newman, 2015)

2.6.9.2 Adoptar una cultura de automatización

En vista que los microservicios añaden mucha complejidad debido al número de partes móviles con las que se debe lidiar. Adquirir una cultura de automatización es esencial para poder manejar este problema, como las pruebas automatizadas para asegurar que todos los servicios digan trabajando. Tener una línea de comandos que despliegue de la misma manera todo, puede ayudar y esto puede ser la clave para adoptar una entrega continua para dar un feedback de la calidad de cada microservicio. (Newman, 2015)

2.6.9.3 Esconder los detalles de implementación interna

Para maximizar la habilidad de que un servicio evolucione independientemente de los demás, es vital que se oculte los detalles de implementación. Modelar con contextos acotados puede ayudar, esto facilita el enfoque en los modelos que deben ser compartidos y los que deben ocultarse. Los servicios deben además ocultar sus bases de datos para evitar caer dentro del orden más común de acoplamiento.

2.6.9.4 Descentralizar todas las cosas

Para maximizar la autonomía que los microservicios hacen posible, se necesita constantemente buscar la oportunidad de delegar la toma de decisiones y el control de los equipos que poseen los propios servicios. Esto proceso comienza con la adopción del propio-servicio (self-service) cuando sea posible, permitiendo a las personas desplegar el software bajo demanda, haciendo el desarrollo y las pruebas lo más fácil posible y evitar dividir el equipo para realizar estas actividades. (Newman, 2015)

Asegurarse que los servicios pertenezcan a los equipos es un paso importante, crear equipos responsables de los cambios que se hicieron, incluso cuando ellos decidan cuando liberar estos cambios. Haciendo uso del código abierto interno asegura a las personas a realizar cambios en servicios que pertenecen a otros equipos, sin embargo, se debe tener en cuenta que requiere trabajo implementar. (Newman, 2015)

2.6.9.5 Independientemente desplegable

Se debe asegurar que los microservicios pueden y sean desplegados ellos mismos. Aun cuando cambios de última hora sean requeridos, se debe buscar que puedan coexistir endpoints versionados para permitir a los clientes cambiarlos con el tiempo. Esto optimiza la velocidad en que se liberan nuevas funcionalidades, así como se incrementa la autonomía de los equipos dueños de estos microservicios. (Newman, 2015)

Adoptando el modelo un servicio por host, se reduce los efectos secundarios que pueden causar que el despliegue de un servicio afecte a otro que no se encuentre relacionado. (Newman, 2015)

2.6.9.6 Aislar el fracaso

Una arquitectura de microservicios puede ser más flexible que un sistema monolítico, pero solo si se entiende y planifica los fallos. Si no se toma en cuenta el hecho de que una llamada puede fallar, el sistema puede sufrir catastróficos fallos en cascada y esto hará notar que el sistema que es más frágil que antes.

2.6.9.7 Altamente observable

No se puede confiar en la observación del comportamiento de una única instancia del servicio de una máquina para visualizar si el sistema funciona correctamente. En su lugar, se necesita una vista unificada de lo que está pasando. Usar el monitoreo semántico para observar si el sistema se comporta correctamente, esto se logra inyectando transacciones sintéticas para simular el comportamiento de un usuario real. Agregar logs y estadísticas permite conocer mejor el problema que se haya presentado. (Newman, 2015)

2.7 Metodología UWE

Las metodologías de desarrollo de software nos permiten gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito, y además logrando tener un producto software de calidad acorde a las necesidades del mercado (usr.code, 2016).

La metodología UWE realiza procesos detallados y exhaustivos en el diseño de la aplicación web, ya que cuenta con procesos iterativos e incrementales, y además incluye flujos de trabajo y puntos de control en las distintas fases que posee (Quiroga, 2016).

2.7.1 Definición

Es un método de ingeniería del software para el desarrollo de aplicaciones Web basado en UML (Lenguaje Unificado de Modelado). Cualquier tipo de diagrama UML puede ser usado, porque UWE es una extensión de UML (Perdita Stevens, 2002).

2.7.2 Características

Las principales características de la metodología UWE son las siguientes:

- Uso de una notación estándar (UML)
- Definición de métodos
- Especificación de restricciones

La figura 28 muestra las dimensiones del modelamiento de la metodología de desarrollo UWE. Direccionada en 3 ejes X, Y, Z, donde se puede observar en el eje X las fases de análisis, diseño e implementación; en el eje Y las vistas como son contenido, estructura de navegación y presentación; en el eje Z muestra los aspectos estructura y comportamiento (ESPE, 2014).

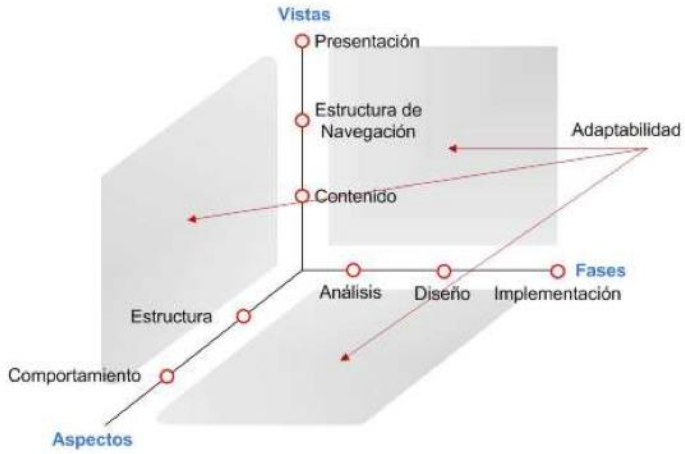

Figura 28. Diagrama en X,Y,Z de UWE
Fuente: (ESPE, 2015)

2.7.3 Actividades de Modelado

Las actividades que propone la metodología UWE son cuatro, las mismas que nos permiten recolectar toda la información relacionada a la especificación de requerimientos del aplicativo web basándose en estándares reconocidos como como UML y en el lenguaje de especificación de restricciones asociado OCL (Object Contraint Language) (Hernández, 2016).

2.7.3.1 Análisis de requisitos

En esta fase se realiza la especificación de requerimientos funcionales y no funcionales que el aplicativo web debe cumplir, para esto el modelado de requisitos consta de dos partes:

- Casos de uso de la aplicación y sus relaciones.
- Actividades describiendo los casos de uso en detalle

2.7.3.2 Diseño conceptual

Se plasma los requerimientos definidos en la fase de análisis de requisitos en un diagrama de clases donde se representará los conceptos detallados.

2.7.3.3 Diseño navegacional

El diseño navegacional se lo realiza mediante la creación de dos modelos los cuales se detallarán a continuación:

- Modelo de espacio de navegación: Especifica que objetos pueden ser visitados a través de la aplicación.
- Modelo de estructura de navegación: Conjunto de estructuras de accesos para la navegación.

2.7.3.4 Diseño de presentación

Es una representación física de las vistas de la aplicación web, estas representaciones se las realizan en base a dos modelos:

- Estructura de Vista: Vista general de la presentación del aplicativo web.
- Interfaz de Usuario: Vista en detalle de los distintos elementos de la interfaz.

CAPITULO III
ANALISIS Y DISEÑO DEL PROYECTO

Para el análisis, diseño, desarrollo e implementación del sistema Decision Cloud Gestión de Servicios (dcGS) se utilizó la metodología UWE, la misma que permite generar una retroalimentación continua en las distintas etapas del proyecto.

Para la recolección de información se trabajó directamente con la gerencia general de la compañía DECISIÓN c.a., la misma que nos brindó todas las reglas de negocio necesarias para elaborar un documento de especificación de requerimientos de software el cual nos permitió elaborar los diagramas del sistema.

3.1 Análisis

A continuación, se define las pautas generales y especificación de requerimientos funcionales y no funcionales del sistema dcGS.

3.1.1 Ámbito del sistema

- Se depurará el proceso actual de la gestión de servicio al cliente aplicando estándares que puedan ser de utilidad en el dcGS.
- El sistema abarcará el proceso completo de la gestión de servicio al cliente.
- El sistema integrará los sistemas externos con los que cuenta DECISION c.a. para evitar la disgregación y duplicidad de la información.
- Permitirá emitir y controlar la información para facturación relacionada con tiempo de servicio, insumos y repuestos.

3.1.2 Definiciones, Acrónimos y Abreviaturas

3.1.2.1 Del negocio

- **dcGS**: Decision Cloud Gestión de Servicios.
- **Empresa Contratante**: Empresa que contrata el servicio dcGS. Una empresa contratante puede ser de dos tipos:
    - **Empresa Proveedora de Servicio:** Empresa que provee servicios, es decir su razón de ser es brindar servicios a sus distintos clientes.
    - **Empresa Receptora de Servicio:** Empresa que requiere servicios, es decir que solicita servicios a sus proveedores para un buen desempeño de sus funciones.

- **Cliente**: Son las Empresas Clientes (Receptoras de Servicio) que requieren servicios de la Empresa Proveedora de Servicio.
- **Proveedor**: Son las Empresas Proveedoras que brindan servicio a la Empresa Receptora de Servicio.
- **Puntos de Servicio**: Son las distintas ubicaciones donde se brindará el servicio y estará ligada a una Empresa Receptora de Servicio.
- **Contrato servicio:** El contrato servicio es aquel que está ligado con la contratación del dcGS por parte de la Empresa Contratante.
- **Contrato convenio:** El contrato convenio es aquel que está ligado con el contrato de servicio entre la Empresa Contratante y sus Clientes en el caso de ser Proveedora de Servicio o con sus Proveedores en el caso de ser Receptora de Servicio.

3.1.2.2 Del sistema

- **Administrar**: Acción de agregar, modificar, eliminar y consultar la información de un determinado objeto o persona.
- **Pruebas**: Proceso mediante el cual se realizan actividades para verificar la óptima función de un módulo o de la totalidad del sistema.
- **CRUD**: Es el acrónimo de Crear, Obtener, Actualizar y Borrar uno o varios registros de una tabla de la base de datos.

3.1.2.3 De tecnología

- **Sistema de gestión de Base de Datos**: Es un tipo de software específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.
- **Base de Datos:** Es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso.
- **Aplicación**: Es un programa informático diseñado para facilitar al usuario la realización de un determinado tipo de trabajo.
- **PostgreSQL**: Sistema de gestión de base de datos relacional.
- **Java**: Lenguaje de Programación Orientado Objetos.

3.1.3 Características de los usuarios

El sistema Decision Cloud Gestión de Servicio constará de varios tipos de usuarios que interactuarán y administrarán con el sistema, los usuarios son:

- Super Administrador.
- Administrador del Sistema.
- Supervisor del Servicio al Cliente.
- Personas.

3.1.4 Requerimientos funcionales

3.1.4.1 Especificaciones generales del sistema

- **RF.dcGS.General.001**

  El sistema debe ser completamente modular manejando los input, output y process de una manera independiente de tal manera que los distintos módulos puedan ser implantados independientemente.

- **RF.dcGS.General.002**

  El sistema debe permitir la integración entre módulos y el intercambio de información entre ellos.

- **RF.dcGS.General.003**

  El sistema debe realizar procedimientos de backup para posibles interrupciones de servicios de internet.

- **RF.dcGS.General.004**

  El sistema debe permitir adicionar sin restricciones, información proveniente de un archivo externo en las módulos que necesiten esta función.

- **RF.dcGS.General.005**

  El sistema debe permitir identificar claramente los módulos y funciones que permite (ingreso/edición de datos, consulta, gráfico, etc).

- **RF.dcGS.General.006**

  El sistema debe permitir que los campos puedan ser opcional y/o requerido dependiendo del cliente y/o ambiente de trabajo, además debe permitir que el campo pueda ser visualizado u ocultado, los mismos que se deben definir en la

implantación y además pueden ser fácilmente gestionado por el administrador del sistema.

- **RF.dcGS.General.007**

  La información de todos y cada uno de los campos puede provenir de ingreso manual, búsqueda de tabla del propio módulo, otro sistema o módulo, hoja de cálculo/archivo externo. Los mismo que serán definidos al momento de la implantación y además pueden ser fácilmente gestionado por el administrador del sistema.

- **RF.dcGS.General.008**

  En todos los módulos del sistema se debe manejar procesos de validación de datos sea en procesos en línea, antes de grabarlos o en procesos batch para la emisión de reportes.

- **RF.dcGS.General.009**

  Todas las operaciones del sistema deben tener doble validación (operación/autorización), independientemente que algunas funciones en la práctica se asignen al mismo usuario.

- **RF.dcGS.General.010**

  La interfaz de usuario del sistema debe de ser única para todos los módulos (barra de herramientas, menús, símbolos, etc).

- **RF.dcGS.General.011**

  Todas las consultas y reportes deben permitir la posibilidad de ingresar dos rangos de fechas para ejecutar reportes/consultas comparativas.

- **RF.dcGS.General.012**

  Todas las consultas y reportes deben permitir la presentación a nivel de detalle o de totales en el caso de ser requerido.

- **RF.dcGS.General.013**

  Todas las consultas deben permitir la exportación a hojas de cálculo, en los distintos formatos que se definirán posteriormente por cada módulo.

- **RF.dcGS.General.014**

  El sistema debe tener contadores que permitan facturar el servicio por diferentes conceptos ya sea por Reporte de Servicio, Mbytes, Usuarios, etc.

- **RF.dcGS.General.015**

Debido a que el sistema se venderá como servicio, el mismo debe advertir con anticipación la fecha límite de uso antes de la cancelación del servicio, además el sistema debe permitir la habilitación/deshabilitación manual o automática del acceso al sistema.

- **RF.dcGS.General.016**

  Debe orientarse a los dos posibles mercados. De un Proveedor con varios clientes o de un cliente con varios proveedores.

3.1.4.2 Módulo de Seguridad

El módulo de seguridad permitirá a los usuarios acceder al sistema y realizar las tareas dependiendo de los permisos que se les haya asignado. Para controlar estos permisos el módulo de seguridad estará implementado con la función RBAC (Control de Acceso Basado en Roles) con lo cual se cumplirá los siguientes requisitos:

- **RF.dcGs.Seguridad.001**

  Los usuarios del sistema deben estar asociados a uno o más perfiles de acceso para el uso del sistema.

- **RF.dcGs.Seguridad.002**

  Los permisos y privilegios asociados a un usuario, estarán registrados en los perfiles de acceso que se le asigne a cada uno.

- **RF.dcGs.Seguridad.003**

  La aplicación estará definida en un inicio por los siguientes perfiles de usuario:

  - Super Administrador.
  - Administrador del Sistema.
  - Supervisor Servicio al Cliente.
  - Personas

- **RF.dcGs.Seguridad.004**

  El usuario debe tener registrado una hora de inicio y de fin para el uso del sistema, en caso de que no tenga registrada una hora de inicio y fin de uso del sistema, dicho usuario podrá acceder al sistema a cualquier hora.

- **RF.dcGs.Seguridad.005**

  El sistema debe permitir doble autenticación en el caso de que se requiera (VIA EMAIL).

- **RF.dcGs.Seguridad.006**

  Si existen 5 intentos de acceso al sistema fallidos la cuenta del usuario se bloqueará y deberá comunicarse con el administrador del sistema para habilitar nuevamente su cuenta.

- **RF.dcGs.Seguridad.007**

  Por control de seguridad se deberá permitir a un usuario iniciar sesión una sola vez, es decir no puede iniciar sesión en varios dispositivos al mismo tiempo.

- **RF.dcGs.Seguridad.008**

  Por control de seguridad se deberá registrar la fecha y hora de la última conexión del usuario al sistema.

- **RF.dcGs.Seguridad.009**

  El sistema debe permitir al Administrador del Sistema de una manera fácil los usuario, roles y permisos.

3.1.4.3 Módulo de Servicio al cliente

El módulo de servicio al cliente permitirá a los usuarios generar los siguientes puntos:

- Solicitud del servicio

  Creación de las solicitudes correspondiente con los datos necesarios para brindar el servicio, esto lo realizará el empleado del cliente.

- Parte de servicio

  Asignación de recursos necesarios para el cumplimiento de los objetivos establecidos por la solicitud de servicio (Técnicos, materiales, repuestos, etc.).

- Recepción y evaluación del servicio

  Se refiere a la aceptación del cliente del servicio recibido y a la evaluación del cliente y del técnico proveedor del servicio.

- Validación y autorización.

  Se refiere al proceso en donde se verifica toda la información y se autoriza el reporte como tal.

- Reporte de servicio

  Visualización de los distintos reportes del módulo.

- Administrar plantillas

Permite administrar la parte gráfica del módulo de servicio, es decir administrar los campos del módulo como se indica en el requerimiento RF.dcGS.EG.006.

Para poder realizar cada una de estas tareas se necesita iniciar sesión con el perfil de usuario correspondiente.

Los input y output (Entradas y salida de información) fueron establecidos en documentos establecidos por DECISION c.a. y firmados por los responsables del proyecto.

3.1.4.4 Otros requerimientos

- Arquitectura
    - El sistema debe ser 100% una aplicación web y toda la parametrización y administración debe realizarse desde un navegador con acceso a internet.
    - El sistema deberá funcionar de una manera independiente del navegador que se utilice.
    - El sistema tendrá interfaces gráficas de administración y todas sus interfaces serán en idioma español.
- Backups
    - El sistema deberá proveer mecanismos para generar backup´s periódicamente de la información que se mantiene en el sistema. Los backup´s deben ser responsabilidad del administrador del sistema.
- Integración
    - El sistema deberá integrarse con aplicaciones externas tales como Asana y las distintas API's de Google.

3.2 Diseño

3.2.1 Casos de uso

"Un caso de uso expresa todas las formas de usar un sistema para alcanzar una meta particular para un usuario. En conjunto, los casos de uso le proporcionan todos los caminos útiles de usar el sistema e ilustran el valor que este provee" (Jacobson, 2013).

Los casos de uso son técnicas para la especificación de requisitos funcionales, es la descripción de una secuencia de interacciones entre el sistema y uno o más actores. Los actores son personas u otros sistemas que interactúan con el sistema cuyos requisitos se están describiendo (Jacobson, 2013), por lo tanto, este documento tiene la finalidad de describir los modelos de casos de uso para el sistema dcGS con sus respectivos actores.

3.2.1.1 Modelo de actores

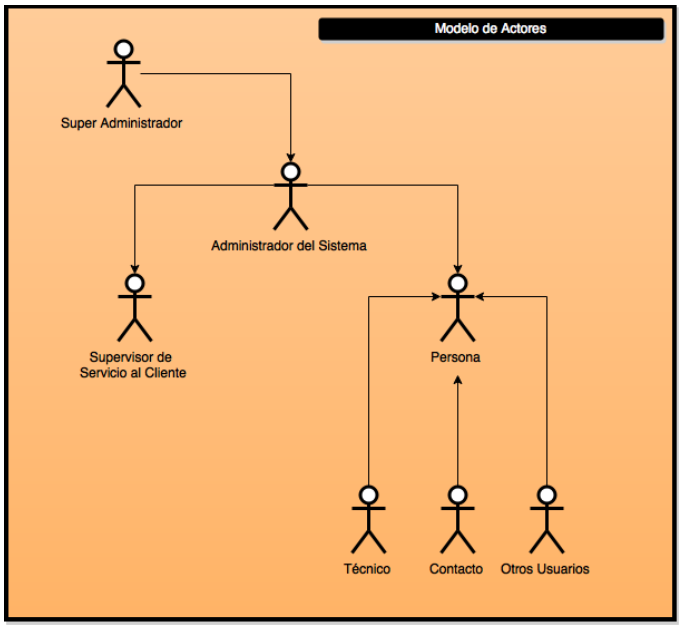

Figura 29. Modelo de actores

Tabla 1.

Descripción de los actores del sistema

Actor	Descripción
Super Administrador	Persona encargada de administrar todo el sistema, independientemente de la Empresa contratante del servicio dcGS..
Administrador del Sistema	Persona encargada de administrar todo lo referente a la Empresa contratante del servicio dcGS
Supervisor de Servicio al Cliente	Persona encargada de gestionar los reportes de servicios.
Persona	Personas con distintos roles, permisos y accesos dependiendo de los requerimientos de la Empresa contratante del servicio dcGs.

3.2.1.2 Nivel 1

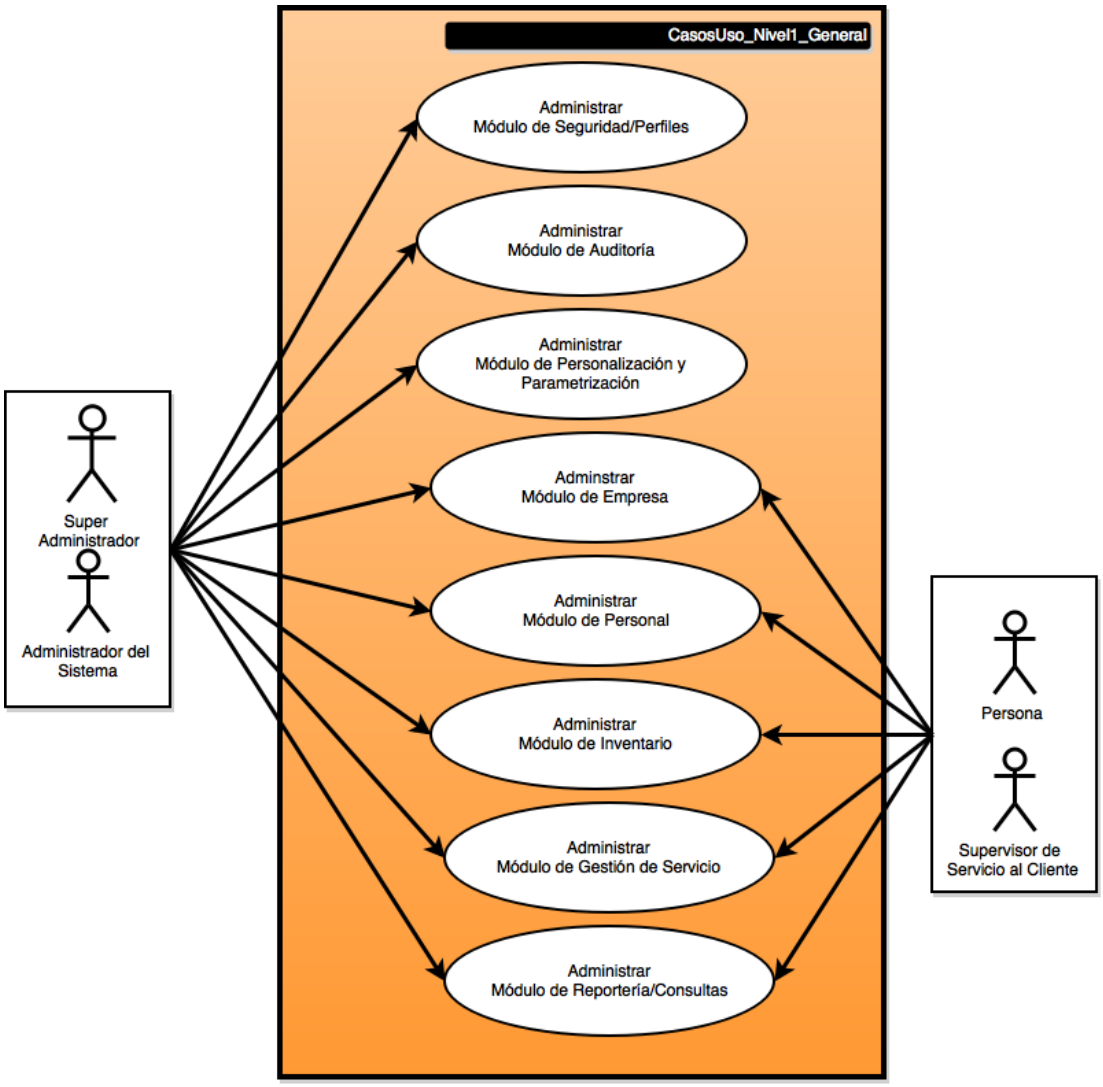

Figura 30 Caso de uso general del Sistema dcGS

Tabla 2

Caso de uso general 01

Nombre del Requerimiento:	**CU-NIVEL1-GENERAL-01 Módulo de Seguridad/Perfiles**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	El Módulo de Seguridad/Perfiles permitirá administrar los distintos roles, usuarios, accesos y acciones que tendrá el sistema.
Actor:	Super Administrador, Administrador del Sistema.
Precondición:	Ninguna
Secuencia normal:	1. Administración de Roles. 2. Administración de Usuarios.

Continúa

	3. Administración de Accesos. 4. Administración de Acciones.
Postcondición:	
Excepciones:	
Comentarios	Toda la administración debe ser orientada a la escalabilidad del sistema.

Tabla 3

Caso de uso general 02

Nombre del Requerimiento:	**CU-NIVEL1-GENERAL-02** **Módulo de Auditoría**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	El Módulo de Auditoría permitirá visualizar el log de registros que se realizarón en el sistema.
Actor:	Super Administrador, Administrador del Sistema.
Precondición:	Ninguna
Secuencia normal:	1. Visualización del log de auditoría.
Postcondición:	
Excepciones:	
Comentarios	

Tabla 4

Caso de uso general 03

Nombre del Requerimiento:	**CU-NIVEL1-GENERAL-03** **Módulo de Personalización/Parametrización**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	El Módulo de Personalización/Parametrización permitirá gestionar la siguiente información referente a la Empresa contratante del servicio dcGS: • Información empresarial (Nombre, siglas, imagen corporativa, etc). • Personalización de campos (etiquetas, obligatoriedad, visible, etc). • Administración del contrato de servicio. Además permitirá administrar los parámetros generales para el uso del sistema.
Actor:	Super Administrador, Administrador del Sistema.
Precondición:	Ninguna
Secuencia normal:	1. Administración Empresa Contratante. 2. Administración de Contrato de Servicio. 3. Personalización de Campos. 4. Parametrizaciones Generales.
Postcondición:	
Excepciones:	
Comentarios	

Tabla 5

Caso de uso general 04

Nombre del Requerimiento:	**CU-NIVEL1-GENERAL-04 Módulo de Empresa.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	El Módulo de Empresa permitirá gestionar la información referente a los clientes o proveedores de la Empresa contratante del dcGS con sus respectivos convenios de contratos.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona..
Precondición:	Ninguna
Secuencia normal:	1. Administración de Empresa Cliente/Proveedora de Servicio. 2. Administración de Puntos de Servicio. 3. Administración de Convenios de Contratos.
Postcondición:	
Excepciones:	
Comentarios	

Tabla 6

Caso de uso general 05

Nombre del Requerimiento:	**CU-NIVEL1-GENERAL-05 Módulo de Personal.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	El Módulo de Persona permitirá gestionar la información referente a las personas(salario, horarios, etc.) que tendrán una relación directa o indirecta con el sistema de servicio dcGS.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Ninguna
Secuencia normal:	1. Administración de Personal.
Postcondición:	
Excepciones:	
Comentarios	

Tabla 7

Caso de uso general 07

Nombre del Requerimiento:	**CU-NIVEL1-GENERAL-06 Módulo de Inventario.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	El Módulo de Inventario permitirá gestionar la información referente a materiales e insumos, repuestos, activos, equipos y sistemas de DECISIÓN c.a.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Ninguna

Continúa

Secuencia normal:	1. Administración de Materiales e Insumos. 2. Administración de Repuestos. 3. Administración de Activos. 4. Administración de Equipos. 5. Administración de Sistemas.
Postcondición:	
Excepciones:	
Comentarios	

Tabla 8

Caso de uso general 07

Nombre del Requerimiento:	**CU-NIVEL1-GENERAL-07** **Módulo de Gestión de Servicio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	El Módulo de Reporte de Servicio es la razón de ser del sistema dcGS, el cual permitirá gestionar el servicio brindado o requerido por la Empresa contratante.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Ninguna
Secuencia normal:	1. Administración del Reporte de Servicio. 2. Firma y evaluación del Reporte de Servicio. a. Supervisor del CSC y contacto que firma la recepción del servicio. 3. Validación y autorización del Reporte de Servicio.
Postcondición:	
Excepciones:	
Comentarios	

Tabla 9

Caso de uso general 08

Nombre del Requerimiento:	**CU-NIVEL1-GENERAL-08** **Módulo de Consultas.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	El Módulo de Consultas permitirá visualizar la información previamente guardada en la base de datos.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona..
Precondición:	Ninguna
Secuencia normal:	1. Selección de Consulta. 2. Selección de Matrices. 3. Selección de Reportes
Postcondición:	
Excepciones:	
Comentarios	

3.2.1.3 Nivel 2

3.2.1.3.1 Módulo de Empresa

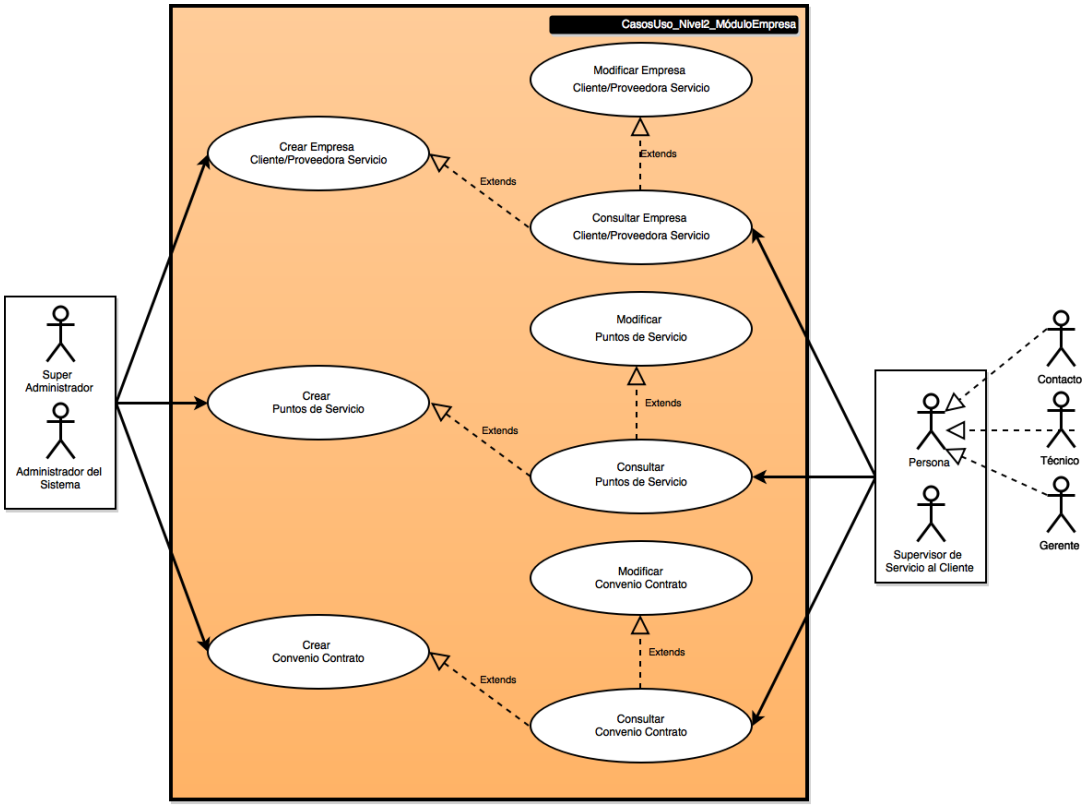

Figura 31 Casos de uso del módulo de empresa

Tabla 10

Caso de uso empresa 01

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOEMPRESA-01 Crear Empresa Cliente/Proveedoras Servicio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Este módulo permitirá gestionar la información referente a la Empresa cliente que requiere un servicio o a la empresa proveedora que brinda un servicio.
Actor:	Super Administrador, Administrador del Sistema.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Empresa.**
Secuencia normal:	1. El sistema desplegará un formulario correspondiente para el nuevo registro de la Empresa Cliente/Proveedora de Servicio con los campos establecidos en el documento dcGS2015 INPUT EMPRESAS 25062015. 2. El sistema solicitará al actor que guarde o cancele los datos de la nueva Empresa Cliente/Proveedora de Servicio. 3. El sistema verificará que todos los campos obligatorios se encuentren lleno.

Continúa

Postcondición:	1. El sistema guardará la nueva Empresa Cliente/Proveedora de Servicio en la base de datos.
Excepciones:	1. Si ya existe una Empresa Cliente/Proveedora Servicio similar en la base de datos el sistema mostrará un mensaje de advertencia y el caso de uso queda sin efecto. 2. Si el actor escoge la opción guardar datos, estos se almacenan en la base de datos, de lo contrario queda sin efecto el caso de uso. 3. Si el actor no ingreso todos los campos obligatorios, el sistema mostrará un mensaje de advertencia en cada uno de los campos donde se genere el error.
Comentarios	

Tabla 11

Caso de uso empresa 02

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOEMPRESA-02** **Modificar Empresa Cliente/Proveedora Servicio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá modificar la información referente a la Empresa Cliente/Proveedora de Servicio.
Actor:	Super Administrador, Administrador del Sistema.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Empresa.** Debe ejecutarse el caso de uso **Crear Empresa Cliente/Proveedora Servicio.**.
Secuencia normal:	1. El sistema permitirá seleccionar la Empresa Cliente/Proveedora de Servicio a modificar. 2. El sistema desplegará un formulario con la información correspondiente a la Empresa Cliente/Proveedora de Servicio con los campos establecidos en el documento dcGS2015 INPUT EMPRESAS 25062015. 3. El sistema solicitará al actor que actualice o cancele los datos de la Empresa Cliente/Proveedora de Servicio modificada. 4. El sistema verificará que todos los campos obligatorios se encuentren registrados.
Postcondición:	1. El sistema actualizará la Empresa Cliente/Proveedora de Servicio en la base de datos.
Excepciones:	1. Si el actor escoge la opción actualizar datos, estos se almacenan en la base de datos, de lo contrario queda sin efecto el caso de uso. 2. Si el actor no ingreso todos los campos obligatorios, el sistema mostrará un mensaje de advertencia en cada uno de los campos donde se genere el error.
Comentarios	

Tabla 12

Caso de uso empresa 03

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOEMPRESA-03 Consultar Empresa Cliente/Proveedora Servicio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá consultar las Empresas Cliente/Proveedora de Servicio.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Empresa.** Debe ejecutarse el caso de uso **Crear Empresa Cliente/Proveedora Servicio.**
Secuencia normal:	1. El sistema permitirá buscar una Empresa Cliente/Proveedora de Servicio. 2. El sistema desplegará un formulario con la información correspondiente a la Empresa Cliente/Proveedora de Servicio con los campos establecidos en el documento dcGS2015 INPUT EMPRESAS 25062015.
Postcondición:	1. El usuario podrá obtener la información de la Empresa Cliente/Proveedora de Servicio.
Excepciones:	1. En el caso de no existir la Empresa Cliente/Proveedora de Servicio, el sistema debe mostrar un mensaje de información de registro inexistente.
Comentarios	

Tabla 13

Caso de uso empresa 04

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOEMPRESA-04 Crear Punto de Servicio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Este módulo permitirá gestionar la información referente a los puntos de servicio de una empresa que requiere servicios.
Actor:	Super Administrador, Administrador del Sistema.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Empresa.**
Secuencia normal:	1. El sistema desplegará un formulario correspondiente para el nuevo registro del Punto de Servicio con los campos establecidos en el documento dcGS2015 INPUT EMPRESAS 25062015. 2. El sistema solicitará al actor que guarde o cancele los datos del nuevo Punto de Servicio. 3. El sistema verificará que todos los campos obligatorios se encuentren lleno.
Postcondición:	1. El sistema guardará el nuevo Punto de Servicio en la base de datos.

Continúa

Excepciones:	1. Si ya existe un Punto de Servicio similar en la base de datos el sistema mostrará un mensaje de advertencia y el caso de uso queda sin efecto. 2. Si el actor escoge la opción guardar datos, estos se almacenan en la base de datos, de lo contrario queda sin efecto el caso de uso. 3. Si el actor no ingreso todos los campos obligatorios, el sistema mostrará un mensaje de advertencia en cada uno de los campos donde se genere el error.
Comentarios	

Tabla 14

Caso de uso empresa 05

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOEMPRESA-05** **Modificar Punto de Servicio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá modificar la información referente al Punto de Servicio.
Actor:	Super Administrador, Administrador del Sistema.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Empresa.** Debe ejecutarse el caso de uso **Crear Punto de Servicio.**
Secuencia normal:	1. El sistema permitirá seleccionar el Punto de Servicio a modificar. 2. El sistema desplegará un formulario con la información correspondiente al Punto de Servicio con los campos establecidos en el documento dcGS2015 INPUT EMPRESAS 25062015. 3. El sistema solicitará al actor que actualice o cancele los datos del Punto de Servicio modificado. 4. El sistema verificará que todos los campos obligatorios se encuentren lleno.
Postcondición:	1. El sistema actualizará el Punto de Servicio en la base de datos.
Excepciones:	1. Si el actor escoge la opción actualizar datos, estos se almacenan en la base de datos, de lo contrario queda sin efecto el caso de uso. 2. Si el actor no ingreso todos los campos obligatorios, el sistema mostrará un mensaje de advertencia en cada uno de los campos donde se genere el error.
Comentarios	

Tabla 15

Caso de uso empresa 06

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOEMPRESA-06 Consultar Punto de Servicio**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá consultar los Puntos de Servicio.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Empresa.** Debe ejecutarse el caso de uso **Crear Punto de Servicio.**
Secuencia normal:	1. El sistema permitirá buscar un Punto de Servicio. 2. El sistema desplegará un formulario con la información correspondiente al Punto de Servicio con los campos establecidos en el documento dcGS2015 INPUT EMPRESAS 25062015.
Postcondición:	1. El usuario podrá obtener la información del Punto de Servicio.
Excepciones:	1. En el caso de no existir el Punto de Servicio, el sistema debe mostrar un mensaje de información de registro inexistente.
Comentarios	

Tabla 16

Caso de uso empresa 07

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOEMPRESA-07 Crear Contrato Convenio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Este módulo permitirá gestionar el contrato que tiene la empresa contratante con sus clientes o proveedores de servicio.
Actor:	Super Administrador, Administrador del Sistema.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Empresa.**
Secuencia normal:	1. El sistema desplegará un formulario correspondiente para el nuevo registro del Contrato/Convenio de Servicio con los campos establecidos en el documento dcGS2015 INPUT EMPRESAS 25062015. 2. El sistema solicitará al actor que guarde o cancele los datos del nuevo Contrato/Convenio de Servicio. 3. El sistema verificará que todos los campos obligatorios se encuentren registrados.
Postcondición:	1. El sistema guardará el nuevo Contrato/Convenio en la base de datos.
Excepciones:	1. Si el actor escoge la opción guardar datos, estos se almacenan en la base de datos, de lo contrario queda sin efecto el caso de uso.

Continúa

	2. Si el actor no ingresa todos los campos obligatorios, el sistema mostrará un mensaje de advertencia en cada uno de los campos donde se genere el error.
Comentarios	Se debe mantener un contrato vigente con cada empresa contratante del servicio dcGS.

Tabla 17

Caso de uso empresa 08

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOEMPRESA-08 Modificar Contrato Convenio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá modificar el Contrato/Convenio que tiene la Empresa Contratante con las empresas Clientes/Proveedoras de servicios.
Actor:	Super Administrador, Administrador del SIstema.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Empresas.** Debe ejecutarse el caso de uso **Crear Contrato/Convenio.**
Secuencia normal:	1. El sistema permitirá seleccionar el Contrato/Convenio a modificar. 2. El sistema desplegará un formulario con la información correspondiente para la modificación del Contrato/Convenio con los campos establecidos en el documento dcGS2015 INPUT EMPRESAS 25062015. 3. El sistema solicitará al actor que actualice o cancele los datos del Contrato/Convenio. 4. El sistema verificará que todos los campos obligatorios se encuentren lleno.
Postcondición:	1. El sistema actualizará el nuevo Contrato/Convenio en la base de datos.
Excepciones:	1. Si el actor escoge la opción actualizar datos, estos se almacenan en la base de datos, de lo contrario queda sin efecto el caso de uso. 2. Si el actor no ingreso todos los campos obligatorios, el sistema mostrará un mensaje de advertencia en cada uno de los campos donde se genere el error.
Comentarios	

Tabla 18

Caso de uso empresa 09

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOEMPRESA-09 Consultar Contrato Convenio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá consultar el Contrato/Convenio.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Empresa.** Debe ejecutarse el caso de uso **Crear Contrato Convenio.**
Secuencia normal:	1. El sistema permitirá buscar un Contrato/Convenio. 2. El sistema desplegará un formulario con la información correspondiente al Contrato/Convenio.
Postcondición:	1. El usuario podrá obtener la información del Contrato/Convenio a consultar.
Excepciones:	1. En el caso de no existir el Contrato/Convenio el sistema debe mostrar un mensaje de información de registro inexistente.
Comentarios	

3.2.1.3.2 Módulo de Persona

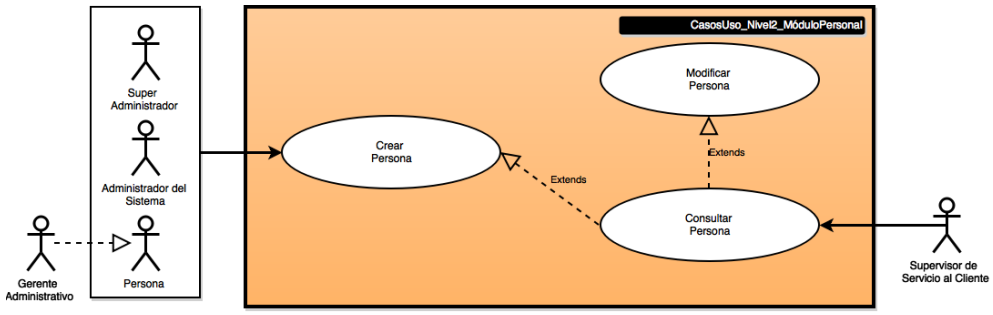

Figura 32 Casos de uso del módulo de persona

Tabla 19

Caso de uso persona 01

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOPERSONA-01 Crear Persona.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá gestionar la información referente a recursos humanos que tendrán una relación directa o indirecta con el sistema de servicio dcGS
Actor:	Super Administrador.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Persona.**

Continúa

Secuencia normal:	1. El sistema desplegará un formulario correspondiente para el nuevo registro de la Persona con los campos establecidos en el documento dcGS 2015 INPUT PERSONAS 20150707. 2. El sistema solicitará al actor que guarde o cancele los datos de la nueva Persona. 3. El sistema verificará que todos los campos obligatorios se encuentren lleno.
Postcondición:	1. El sistema guardará la nueva Persona en la base de datos.
Excepciones:	1. Si ya existe una Persona similar en la base de datos el sistema mostrará un mensaje de advertencia y el caso de uso queda sin efecto. 2. Si el actor escoge la opción guardar datos, estos se almacenan en la base de datos, de lo contrario queda sin efecto el caso de uso. 3. Si el actor no ingreso todos los campos obligatorios, el sistema mostrará un mensaje de advertencia en cada uno de los campos donde se genere el error.
Comentarios	

Tabla 20

Caso de uso persona 02

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOPERSONA-02** **Modificar Persona.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá modificar la información referente a la Persona.
Actor:	Super Administrador.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Persona.** Debe ejecutarse el caso de uso **Crear Persona.**
Secuencia normal:	1. El sistema permitirá seleccionar la Persona a modificar. 2. El sistema desplegará un formulario con la información correspondiente a la Persona con los campos establecidos en el documento dcGS 2015 INPUT PERSONAS 20150707. 3. El sistema solicitará al actor que actualice o cancele los datos de la Persona modificada. 4. El sistema verificará que todos los campos obligatorios se encuentren lleno.
Postcondición:	1. El sistema actualizará a la Persona en la base de datos.
Excepciones:	1. Si el actor escoge la opción actualizar datos, estos se almacenan en la base de datos, de lo contrario queda sin efecto el caso de uso. 2. Si el actor no ingreso todos los campos obligatorios, el sistema mostrará un mensaje de advertencia en cada uno de los campos donde se genere el error.
Comentarios	

Tabla 21

Caso de uso persona 03

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOPERSONA-03** **Consultar Persona.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá consultar a las Personas registradas en el dcGS.
Actor:	Super Administrador.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Persona.** Debe ejecutarse el caso de uso **Crear Persona.**
Secuencia normal:	1. El sistema permitirá buscar a una Persona. 2. El sistema desplegará un formulario con la información correspondiente a la Persona con los campos establecidos en el documento dcGS 2015 INPUT PERSONAS 20150707..
Postcondición:	1. El usuario podrá obtener la información de la Persona.
Excepciones:	1. En el caso de no existir la Persona, el sistema debe mostrar un mensaje de información de registro inexistente.
Comentarios	

3.2.1.3.3 Módulo de Gestión de Servicio

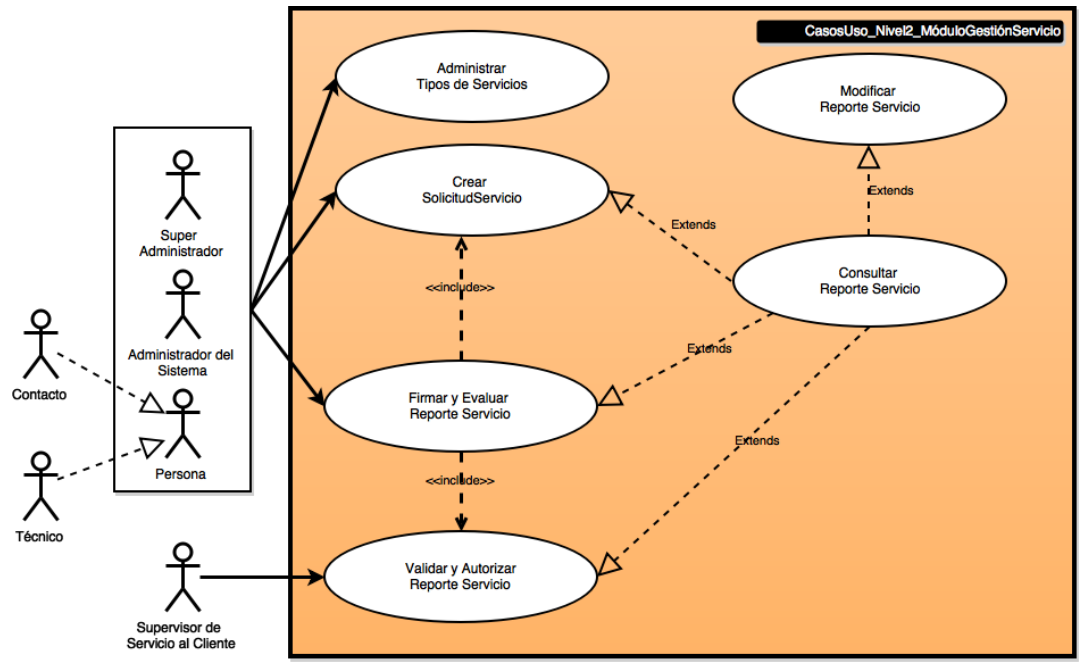

Figura 33 Casos de uso del módulo de servicio

Tabla 22

Caso de uso servicio 01

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOGESTIÓNSERVICIO-01 Administrar tipos de Servicios.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá Gestionar los tipos de servicio que tendrá el sistema.
Actor:	Super Administrador.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Gestión de Servicio.**
Secuencia normal:	1. El sistema permitirá crear, modificar y consultar los distintos tipos de servicios que tendrá el sistema. 2. El sistema desplegará un formulario con la información correspondiente a los tipos de servicio.
Postcondición:	1. El usuario podrá crear, modificar o visualizar la información de los tipos de servicio.
Excepciones:	
Comentarios	

Tabla 23

Caso de uso servicio 02

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOGESTIÓNSERVICIO-02 Crear Solicitud de Servicio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá crear la información referente a una solicitud de servicio.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Gestión de Servicio.**
Secuencia normal:	1. El sistema desplegará un formulario correspondiente para la nueva Solicitud de Servicio con los campos establecidos en el documento dcGS2015 INPUT SOLICITUD DE SERVICIO 20150707. 2. El sistema solicitará al actor que guarde o cancele los datos de la nueva Solicitud de Servicio. 3. El sistema verificará que todos los campos obligatorios se encuentren registrados.
Postcondición:	1. El sistema guardará la nueva Solicitud de Servicio en la base de datos.
Excepciones:	1. Si el actor escoge la opción guardar datos, estos se almacenan en la base de datos, de lo contrario queda sin efecto el caso de uso. 2. Si el actor no ingreso todos los campos obligatorios, el sistema mostrará un mensaje de advertencia en cada uno de los campos donde se genere el error.
Comentarios	

Tabla 24

Caso de uso servicio 03

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOGESTIÓNSERVICIO-03 Modificar Reporte de Servicio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá modificar la información referente al Reporte de Servicio.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Gestión de Servicio.** Debe ejecutarse el caso de uso **Crear Solicitud de Servicio.**
Secuencia normal:	1. El sistema permitirá seleccionar una solicitud de servicio a modificar. 2. El sistema desplegará un formulario con la información correspondiente al Reporte de Servicio con los campos establecidos en el documento dcGS2015 INPUT REPORTE DE SERVICIO 29062015. 3. El sistema solicitará al actor que actualice o cancele los datos del Reporte de Servicio. 4. El sistema verificará que todos los campos obligatorios se encuentren lleno.
Postcondición:	1. El sistema actualizará el Reporte de Servicio en la base de datos.
Excepciones:	1. Si el actor escoge la opción actualizar datos, estos se almacenan en la base de datos, de lo contrario queda sin efecto el caso de uso. 2. Si el actor no ingreso todos los campos obligatorios, el sistema mostrará un mensaje de advertencia en cada uno de los campos donde se genere el error.
Comentarios	

Tabla 25

Caso de uso servicio 04

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOGESTIÓNSERVICIO-04 Consultar Reporte de Servicio.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá consultar los Reportes de Servicio.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Gestión de Servicio.** Debe ejecutarse el caso de uso **Crear Solicitud de Servicio.**

Continúa

Secuencia normal:	1. El sistema permitirá buscar a una Solicitud de Servicio. 2. El sistema desplegará un formulario con la información correspondiente al Reporte de Servicio con los campos establecidos en el documento dcGS2015 INPUT REPORTE DE SERVICIO 29062015.
Postcondición:	1. El usuario podrá obtener la información del Reporte de Servicio.
Excepciones:	
Comentarios	

Tabla 26

Caso de uso servicio 05

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOGESTIÓNSERVICIO-05 Validar y Autorizar Reporte de Servicio**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá validar y autorizar un Reportes de Servicio.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Gestión de Servicio.**
Secuencia normal:	1. El sistema permitirá al actor validar el servicio brindado. 2. Una vez evaluado el sistema permitirá al actor autorizar el reporte de servicio.
Postcondición:	
Excepciones:	
Comentarios	

Tabla 27

Caso de uso servicio 06

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOGESTIÓNSERVICIO-06 Validar y Autorizar Reporte de Servicio**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá validar todos los detalles del Reporte de Servicio y autorizar para su respectivo cobro.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Gestión de Servicio.**
Secuencia normal:	1. El sistema permitirá validar los detalles del reporte de servicio.

Continúa

	2. El sistema debe llevar un registro de los cambios realizados. 3. Finalmente debe permitir autorizar el reporte de servicio.
Postcondición:	
Excepciones:	
Comentarios	

3.2.1.3.4 Módulo de Consultas

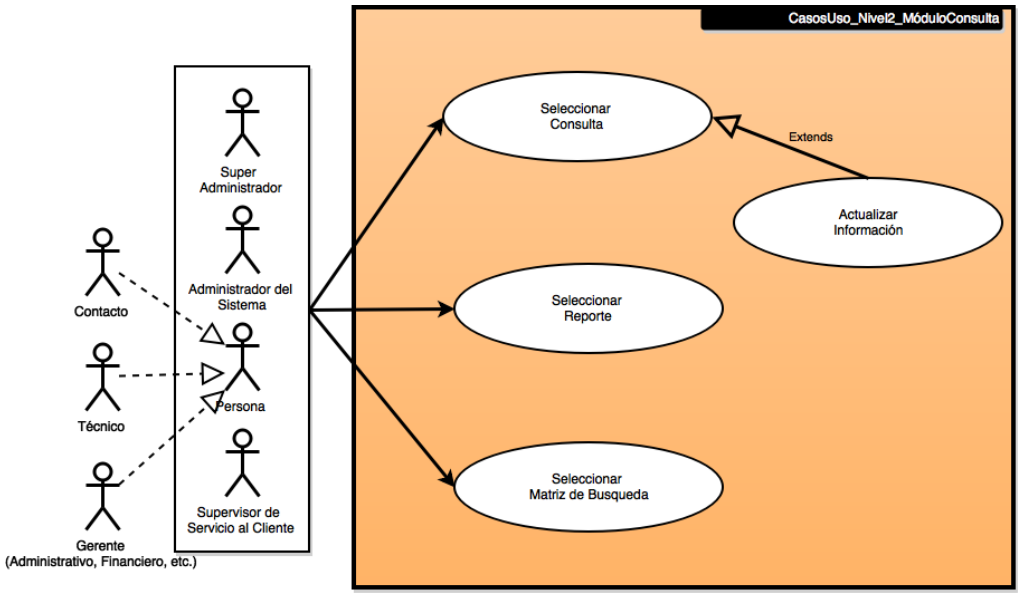

Figura 34 Casos de uso del módulo de consultas

Tabla 28

Caso de uso consulta 01

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOCONSULTA-01 Seleccionar Consulta.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá realizar las consultas al sistema.
Actor:	Super Administrador, , Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Consultas.**
Secuencia normal:	1. El sistema permitirá seleccionar la consulta deseada. 2. El sistema mostrará los datos de dicha consulta.
Postcondición:	
Excepciones:	
Comentarios	

Tabla 29

Caso de uso consulta 02

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOCONSULTA-02 Actualizar Información.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá actualizar ciertos datos de las consultas al sistema.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Consultas.**
Secuencia normal:	1. El sistema permitirá seleccionar la consulta deseada. 2. El sistema permitirá actualizar ciertos datos de dicha consulta.
Postcondición:	
Excepciones:	
Comentarios	

Tabla 30

Caso de uso consulta 03

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOCONSULTA-03 Seleccionar Reporte.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá consultar los reportes del sistema.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Consultas.**
Secuencia normal:	1. El sistema permitirá seleccionar el reporte deseado. 2. El sistema mostrará los datos de dicho reporte.
Postcondición:	
Excepciones:	
Comentarios	

Tabla 31

Caso de uso consulta 04

Nombre del Requerimiento:	**CU-NIVEL2-MÓDULOCONSULTA-04 Seleccionar Matriz.**
Requerimientos del Sistema Asociado:	Reunión interna DCA.
Descripción:	Permitirá consultar mediante la matrices de búsquedas del sistema.
Actor:	Super Administrador, Administrador del Sistema, Supervisor de Servicio al Cliente, Persona.
Precondición:	Debe ejecutarse el caso de uso **Administrar Módulo de Consultas.**
Secuencia normal:	1. El sistema permitirá seleccionar la matriz de búsqueda deseada. 2. El sistema permitirá ingresar los datos relacionados a dicha matriz. 3. El sistema mostrará los datos de la matriz de búsqueda.
Postcondición:	
Excepciones:	
Comentarios	

3.2.2 Diseño conceptual

A continuación, se mostrará los diagramas de clases realizados en la etapa de diseño.

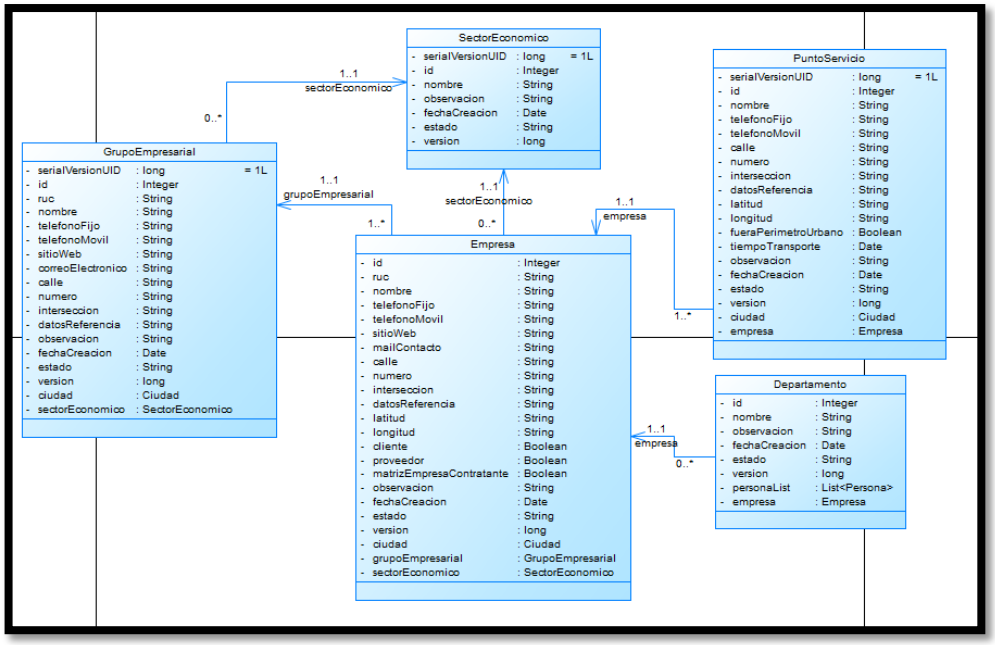

Figura 35 Diagrama de clases del módulo de empresa.

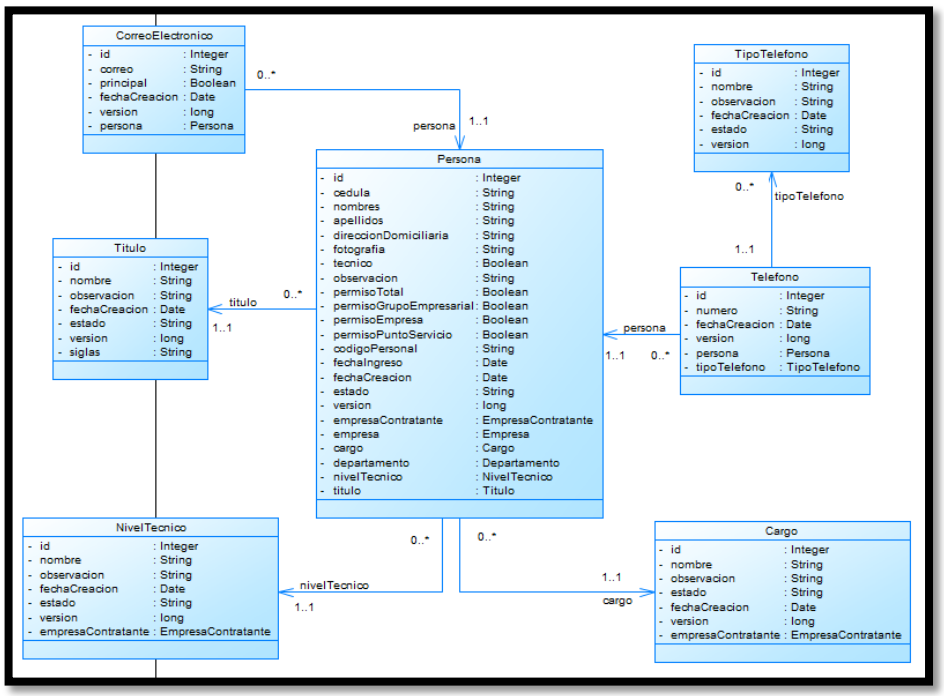

Figura 36 Diagrama de clases del módulo de persona.

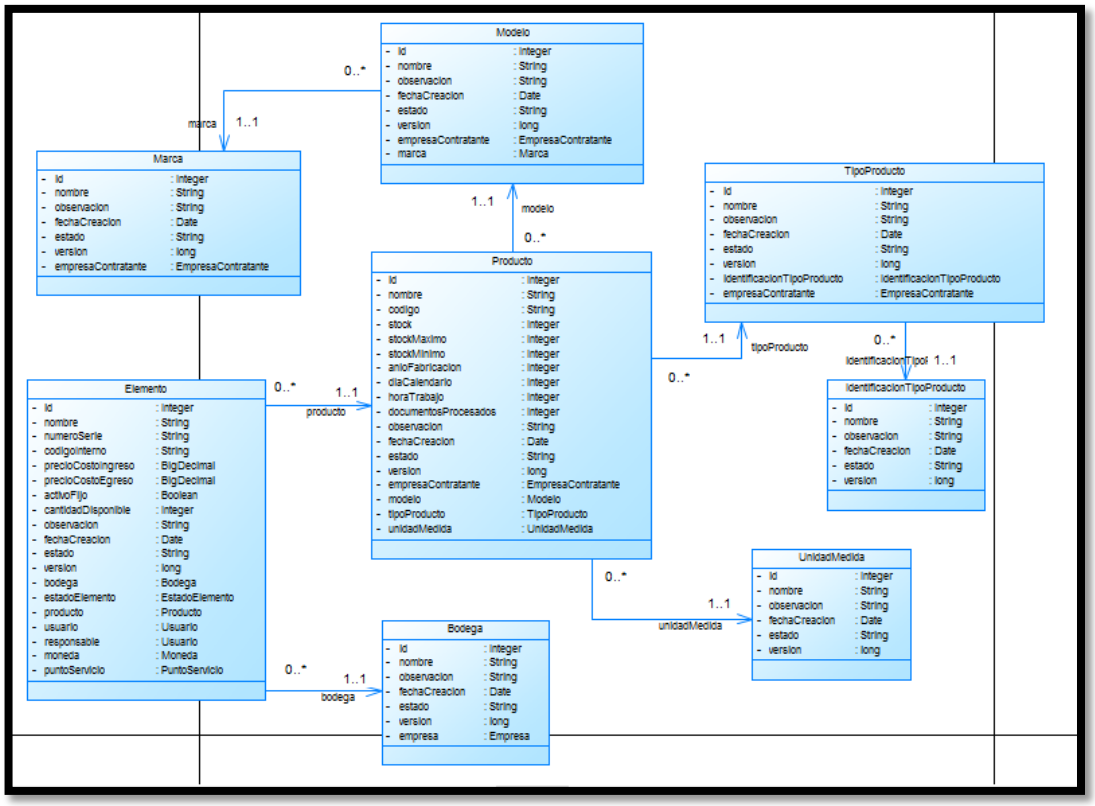

Figura 37 Diagrama de clases del módulo de inventario

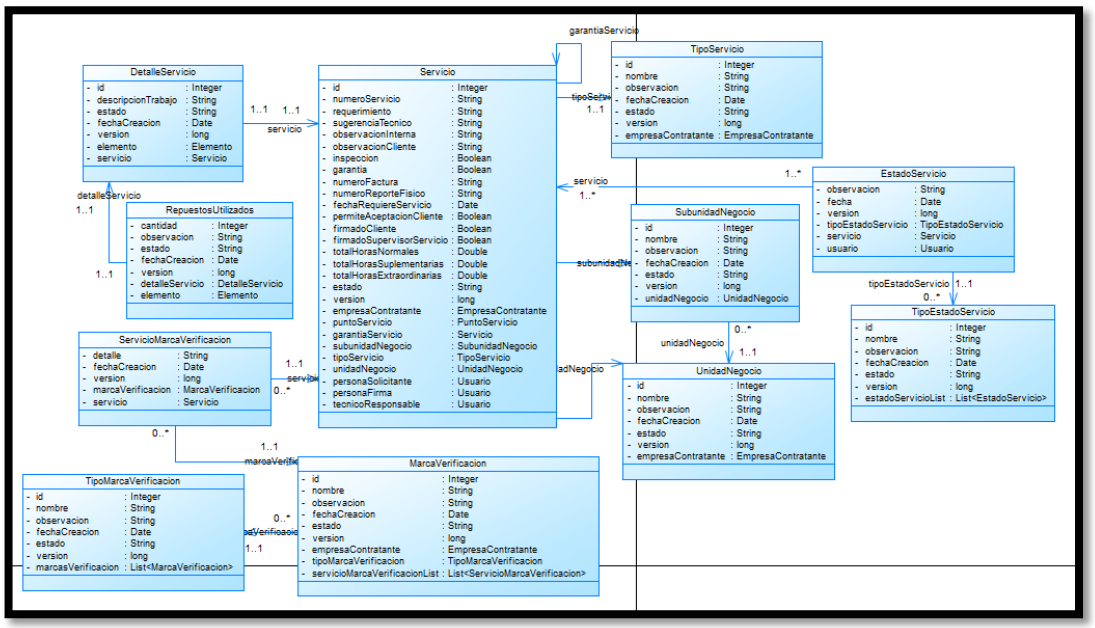

Figura 38 Diagrama de clases del módulo de servicio

3.2.3 Diseño Navegacional

El diseño navegacional permite saber cómo están enlazadas las páginas de la aplicación web, para ello se realizó cuatro diagramas de una forma genérica por módulos.

Los siguientes diagramas hacen referencia a los módulos de empresa (Figura 27), persona (Figura 28) e inventario (Figura 29), solo se representa a la entidad principal de cada módulo debido a que los funcionamientos de las otras páginas de estos módulos son similares.

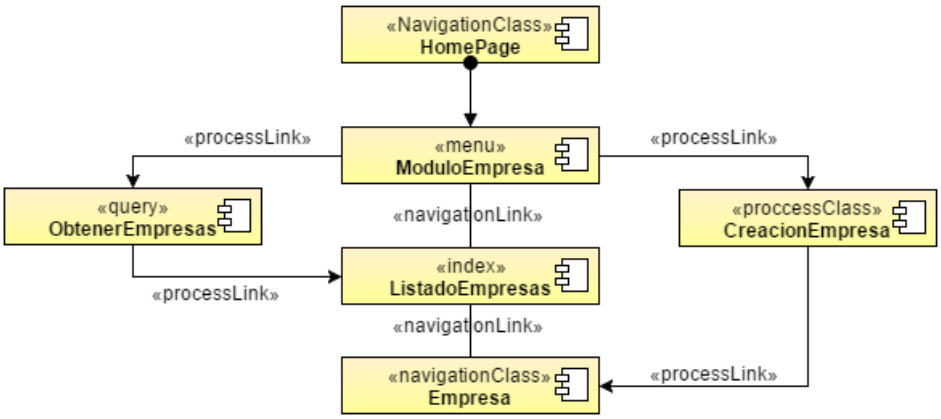

Figura 39 Diagrama de navegación del módulo de empresa

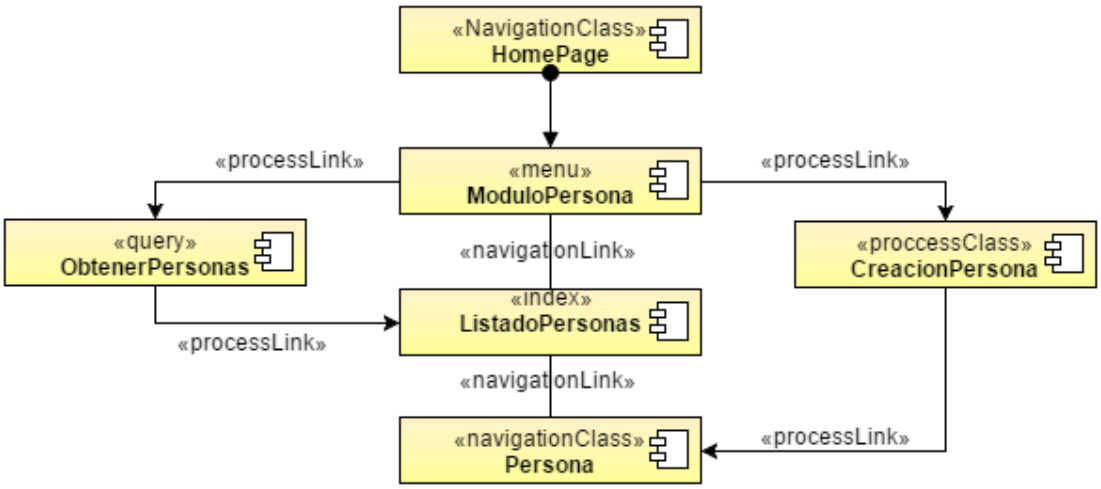

Figura 40 Diagrama de navegación del módulo de persona

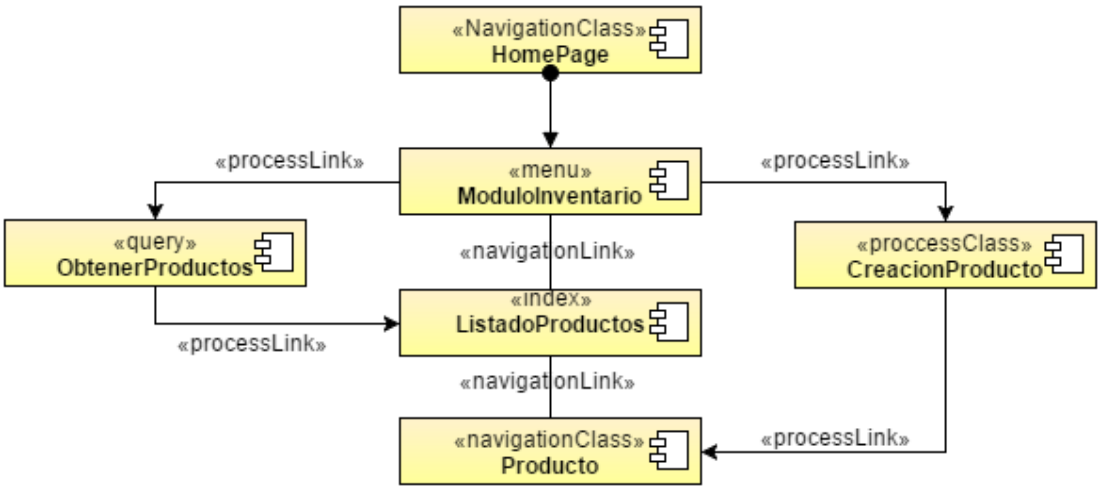

Figura 41 Diagrama de navegación del módulo de inventario

Finalmente, en la Figura 30 se muestra el diagrama de navegación del módulo de servicio.

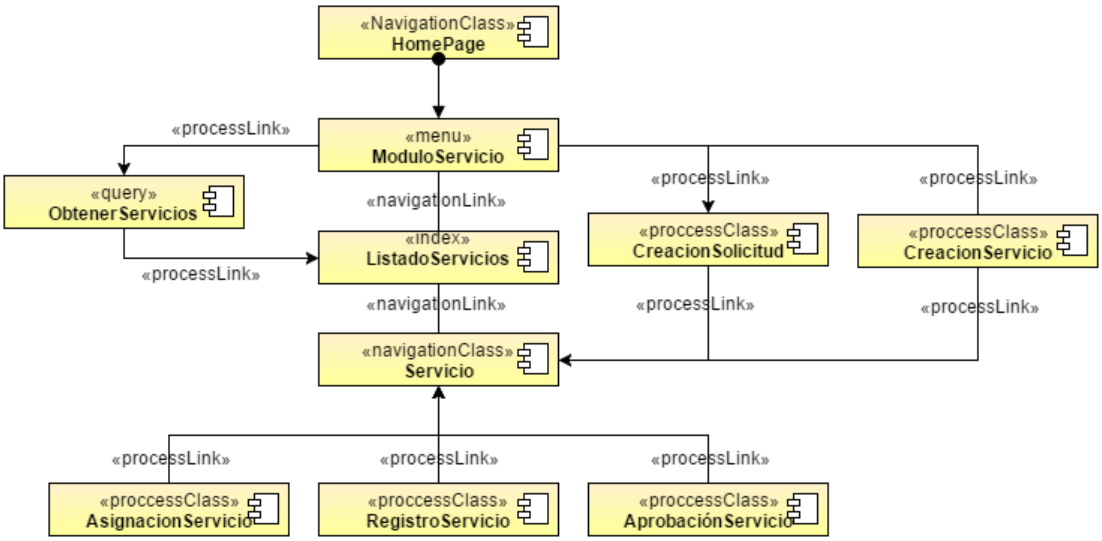

Figura 42 Diagrama de navegación del módulo de servicio

3.2.4 Diseño de presentación

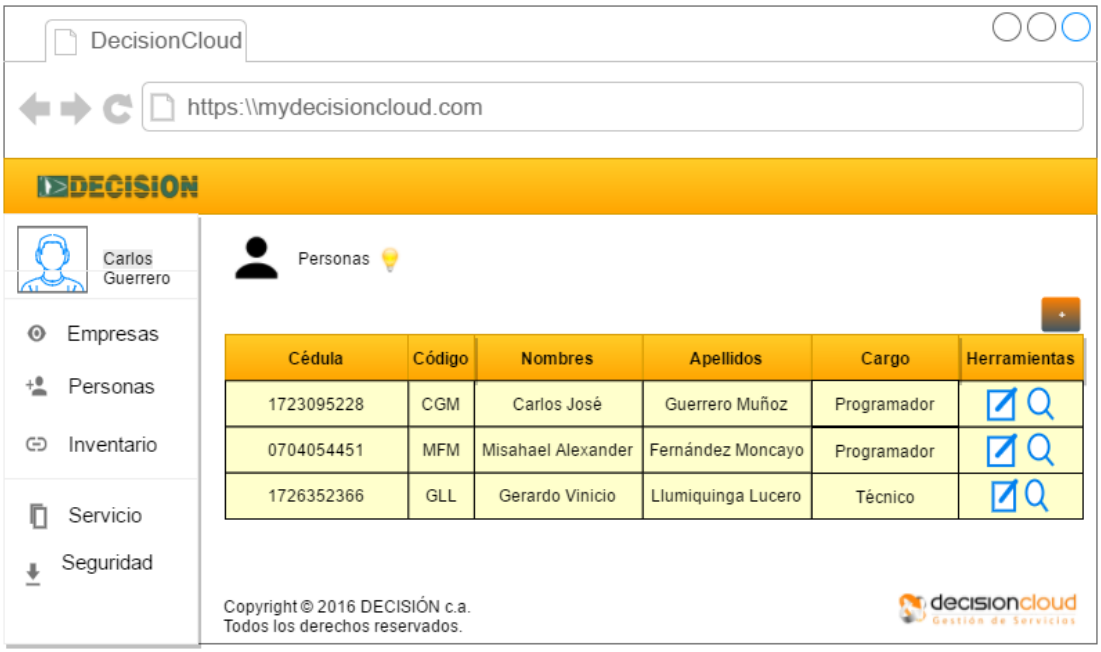

Figura 43 Vista general de la aplicación

3.2.5 Contextos acotados (Bounded contexts)

En esta sección se describe cómo se separan los contextos y se determinan los modelos que deben ser compartidos en los mismos. Para una separación correcta, se deben analizar los requerimientos funcionales del sistema.

Para el diseño del sistema se han identificado tres contextos principales, los cuales son: servicio, inventario y persona. A continuación, se procede a describir cada uno para conocer los modelos y aquellos que están destinados a ser compartidos con los demás contextos.

- **Contexto de servicio:** Se encuentra todo lo relacionado a la gestión del servicio como: el tipo de servicio, el cliente o proveedor solicitante, el técnico responsable, las horas trabajadas, el trabajo realizado, los elementos atendidos y/o utilizados, las observaciones, etc.
- **Contexto de inventarios:** Se encuentra todo lo relacionado al movimiento de elementos, a la creación de nuevos productos, a los pedidos o importaciones de productos, la reposición de elementos, los tipos de producto, los tipos de movimiento, etc.

- **Contexto de personas:** Se encuentra todo lo relacionado a los recursos humanos como: el salario, el horario, el historial de la persona, los permisos, las vacaciones, etc.

Para identificar los modelos que se van a compartir a los demás contextos se comienza analizando el contexto que potencialmente puede necesitar información de otro, en este caso es el de servicio, ver figura 44.

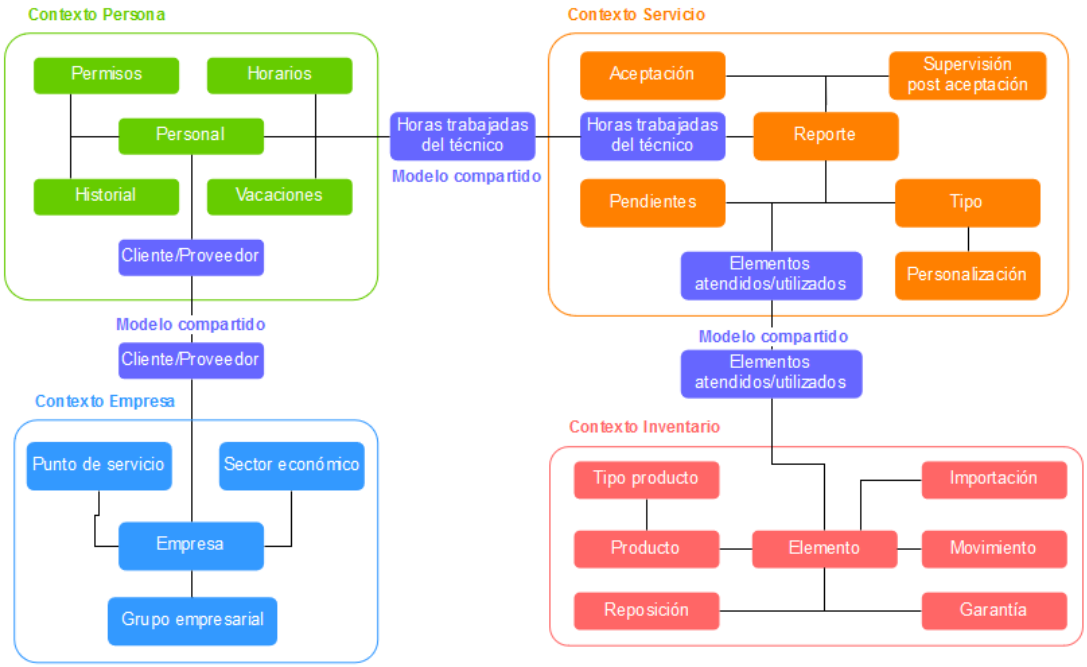

Figura 44 Contextos acotados

Las horas trabajadas contienen la información del técnico y el tiempo que ha empleado en realizar el trabajo, entonces para el contexto de personas esta información ayuda a determinar si el horario en el que trabajó el técnico se encuentra o no en el horario normal de trabajo, caso contrario se procede a contabilizar horas extras y suplementaras para su posterior pago.

Los elementos atendidos y/o utilizados se componen de dos partes: los atendidos, que contienen la información del elemento, que en este caso se denomina equipo/sistema, al cual el técnico ha dado servicio y los utilizados que contienen la información del elemento utilizado para la atención de dicho equipo/sistema; por lo tanto para el contexto de inventarios esta información es utilizada para saber cuántos

elementos se han utilizado en el servicio y si necesitan reposición, para contar con un stock de elementos al día.

CAPITULO IV
DESARROLLO E IMPLEMENTACIÓN

4.1 Desarrollo

4.1.1 Arquitectura orientada a microservicios

El siguiente diagrama muestra la arquitectura general del sistema dcGS que se implementó en DECISIÓN c.a.

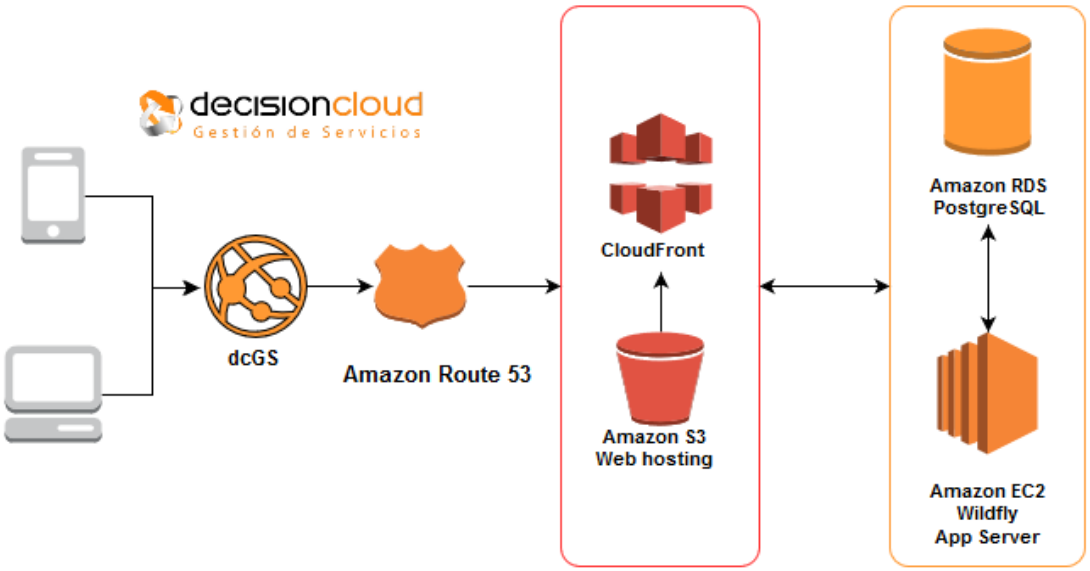

Figura 45 Arquitectura física general del sistema

Utilizando AWS (Amazon Web Services) se configura el entorno de desarrollo de la aplicación.

En el siguiente diagrama se muestra el módulo *gestión del servicio* y su interacción con la aplicación cliente.

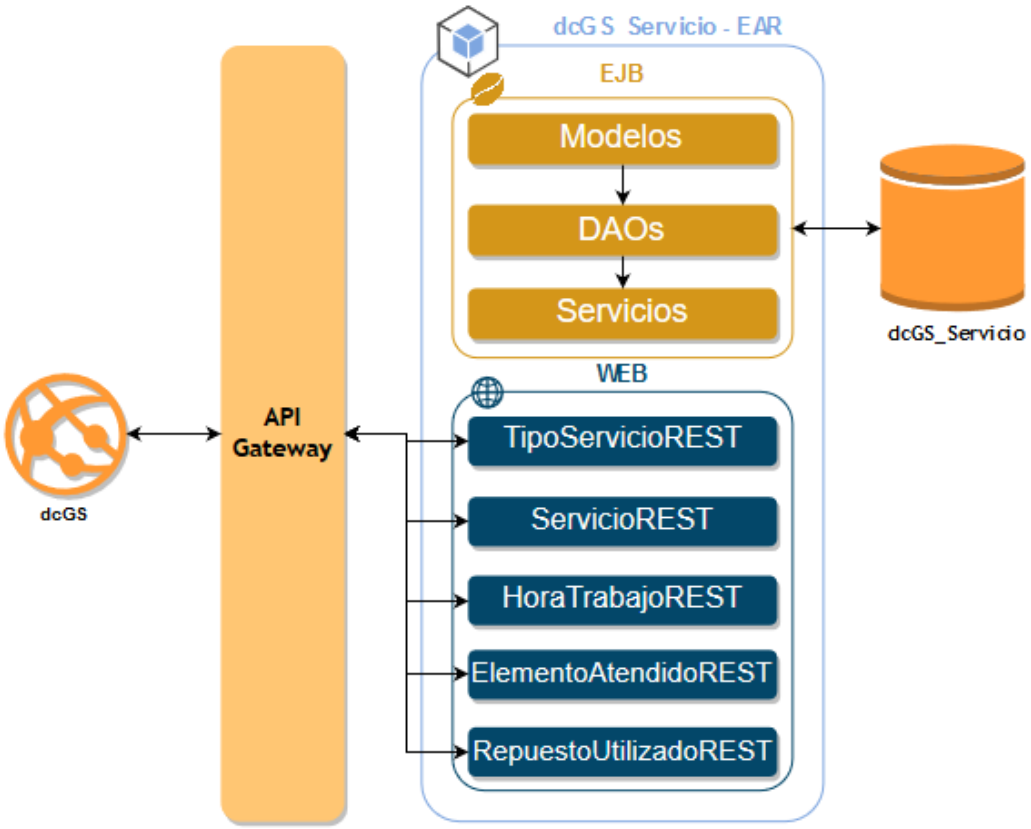

Figura 46 Diagrama detallado de un microservicio – Gestión del servicio

En el siguiente diagrama se muestra el diagrama global de la arquitectura a microservicios y cómo es su relación con cada componente.

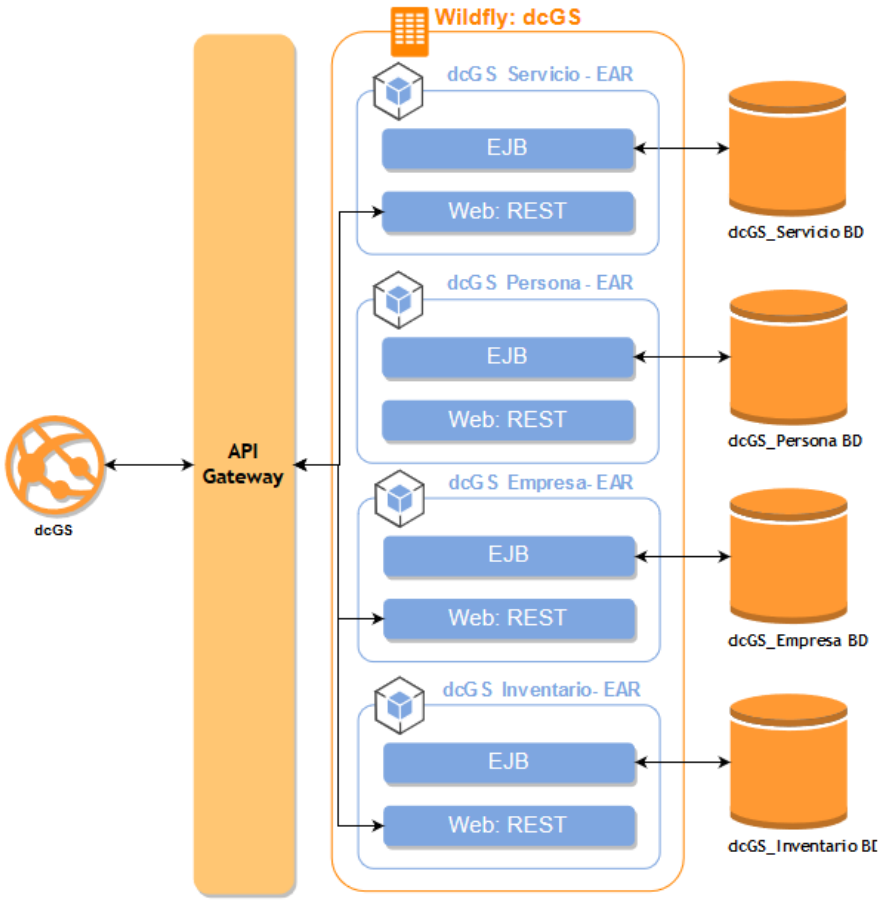

Figura 47 Diagrama global de la aplicación orientada a microservicios

Cada microservicio cuenta con su propia base de datos y se encuentra desplegado como una aplicación independiente en el servidor de aplicaciones Wildfly, en donde exponen sus servicios REST a la API Gateway que a su vez se encarga de la comunicación con la aplicación Angular y poder realizar las operaciones destinadas.

La única comunicación que existe entre los microservicios es a través del aplicativo Angular, que es el encargado de enviar y recibir los datos correspondientes para que sean procesados en cada uno de estos microservicios.

A continuación, se muestra el diagrama de despliegue de un microservicio.

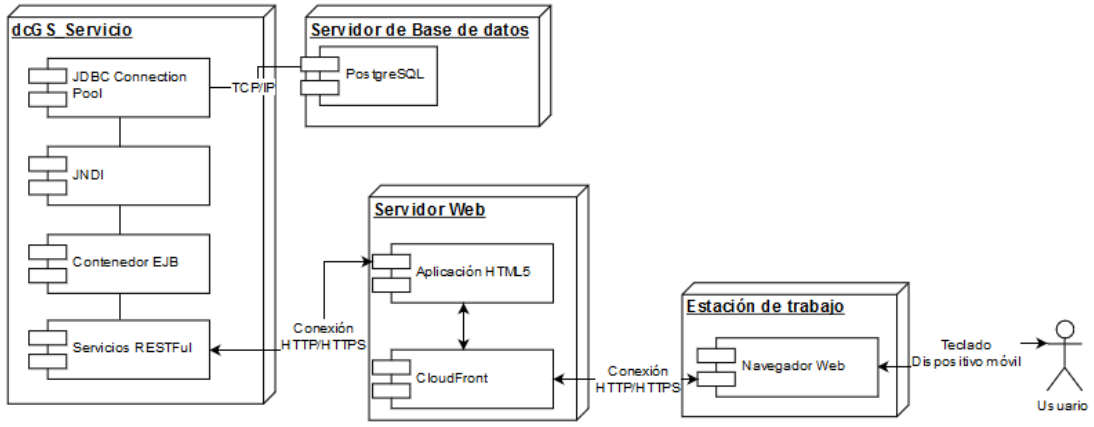

Figura 48 Diagrama de despliegue de la gestión del servicio

A continuación, se muestra el diagrama de despliegue global del sistema.

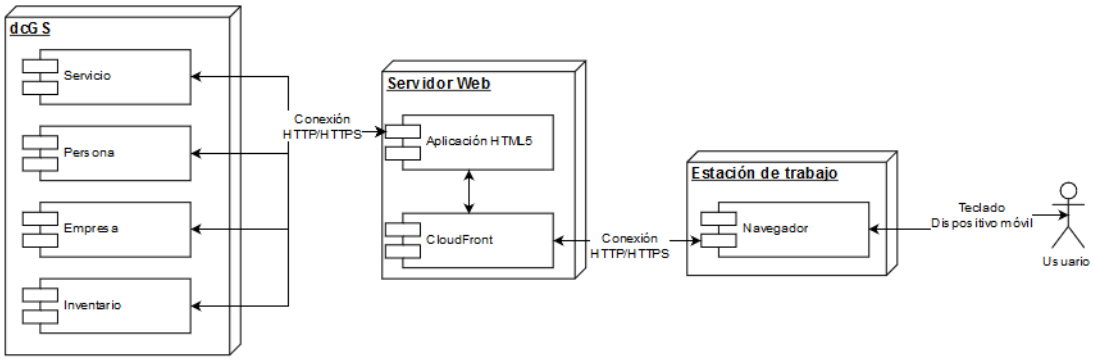

Figura 49 Diagrama de despliegue global

4.1.1.1 Servicios RESTFul

A continuación, se describe los servicios RESTful que se manejan en el servidor de la aplicación.

Tabla 32

Servicio RESTFul: Módulo empresa

MÓDULO DE EMPRESA

HTTP	URL	PARAMETRO	RESULTADO
GET	/empresa	empresa	Obtener todos
GET	/empresa/id	empresa.id	Obtener por id
POST	/empresa	empresa	Crear
PUT	/empresa	empresa	Editar

Continúa

GET	/grupoEmpresarial	grupoEmpresarial	Obtener todos
GET	/grupoEmpresarial/id	grupoEmpresarial.id	Obtener por id
POST	/grupoEmpresarial	grupoEmpresarial	Crear
PUT	/grupoEmpresarial	grupoEmpresarial	Editar
GET	/sectorEconomico	sectorEconomico	Obtener todos
GET	/sectorEconomico/id	sectorEconomico.id	Obtener por id
POST	/sectorEconomico	sectorEconomico	Crear
PUT	/sectorEconomico	sectorEconomico	Editar
GET	/puntoServicio	puntoServicio	Obtener todos
GET	/puntoServicio/id	puntoServicio.id	Obtener por id
POST	/puntoServicio	puntoServicio	Crear
PUT	/puntoServicio	puntoServicio	Editar
GET	/departamento	departamento	Obtener todos
GET	/departamento/id	departamento.id	Obtener por id
POST	/departamento	departamento	Crear
PUT	/departamento	departamento	Editar

Tabla 33

Servicio RESTFul: Módulo persona

MÓDULO DE PERSONA

HTTP	URL	PARAMETRO	RESULTADO
GET	/persona	persona	Obtener todos
GET	/persona/id	persona.id	Obtener por id
POST	/persona	persona	Crear
PUT	/persona	persona	Editar
GET	/cargo	cargo	Obtener todos
GET	/cargo/id	cargo.id	Obtener por id
POST	/cargo	cargo	Crear
PUT	/cargo	cargo	Editar
GET	/titulo	titulo	Obtener todos
GET	/titulo/id	titulo.id	Obtener por id

Continúa

POST	/titulo	titulo	Crear
PUT	/titulo	titulo	Editar
GET	/nivelTecnico	nivelTecnico	Obtener todos
GET	/nivelTecnico/id	nivelTecnico.id	Obtener por id
POST	/nivelTecnico	nivelTecnico	Crear
PUT	/nivelTecnico	nivelTecnico	Editar

Tabla 34

Servicio RESTFul: Módulo inventario

MÓDULO DE INVENTARIO

HTTP	URL	PARAMETRO	RESULTADO
GET	/marca	marca	Obtener todos
GET	/marca/id	marca.id	Obtener por id
POST	/marca	marca	Crear
PUT	/marca	marca	Editar
GET	/modelo	modelo	Obtener todos
GET	/modelo/id	modelo.id	Obtener por id
POST	/modelo	modelo	Crear
PUT	/modelo	modelo	Editar
GET	/tipoProducto	tipoProducto	Obtener todos
GET	/tipoProducto/id	tipoProducto.id	Obtener por id
POST	/tipoProducto	tipoProducto	Crear
PUT	/tipoProducto	tipoProducto	Editar
GET	/unidadMedida	unidadMedida	Obtener todos
GET	/unidadMedida/id	unidadMedida.id	Obtener por id
POST	/unidadMedida	unidadMedida	Crear
PUT	/unidadMedida	unidadMedida	Editar
GET	/bodega	bodega	Obtener todos
GET	/bodega/id	bodega.id	Obtener por id
POST	/bodega	bodega	Crear

Continúa

PUT	/bodega	bodega	Editar
GET	/elemento	elemento	Obtener todos
GET	/elemento/id	elemento.id	Obtener por id
POST	/elemento	elemento	Crear
PUT	/elemento	elemento	Editar
GET	/producto	producto	Obtener todos
GET	/producto/id	producto.id	Obtener por id
POST	/producto	producto	Crear
PUT	/producto	producto	Editar

Tabla 35

Servicio RESTFul: Módulo servicio

MÓDULO DE SERVICIO

HTTP	URL	PARAMETRO	RESULTADO
GET	/tipoServicio	tipoServicio	Obtener todos
GET	/tipoServicio/id	tipoServicio.id	Obtener por id
POST	/tipoServicio	tipoServicio	Crear
PUT	/tipoServicio	tipoServicio	Editar
GET	/tipoEstadoServicio	tipoEstadoServicio	Obtener todos
GET	/tipoEstadoServicio/id	tipoEstadoServicio.id	Obtener por id
POST	/tipoEstadoServicio	tipoEstadoServicio	Crear
PUT	/tipoEstadoServicio	tipoEstadoServicio	Editar
GET	/estadoServicio	estadoServicio	Obtener todos
GET	/estadoServicio/id	estadoServicio.id	Obtener por id
POST	/estadoServicio	estadoServicio	Crear
PUT	/estadoServicio	estadoServicio	Editar
GET	/subunidadNegocio	subunidadNegocio	Obtener todos
GET	/subunidadNegocio/id	subunidadNegocio.id	Obtener por id
POST	/subunidadNegocio	subunidadNegocio	Crear
PUT	/subunidadNegocio	subunidadNegocio	Editar
GET	/unidadNegocio	unidadNegocio	Obtener todos

Continúa

GET	/unidadNegocio/id	unidadNegocio.id	Obtener por id
POST	/unidadNegocio	unidadNegocio	Crear
PUT	/unidadNegocio	unidadNegocio	Editar
GET	/tipoMarcaVerificacion	tipoMarcaVerificacion	Obtener todos
GET	/tipoMarcaVerificacion/id	tipoMarcaVerificacion.id	Obtener por id
POST	/tipoMarcaVerificacion	tipoMarcaVerificacion	Crear
PUT	/tipoMarcaVerificacion	tipoMarcaVerificacion	Editar
GET	/servicioVerificacion	servicioVerificacion	Obtener todos
GET	/servicioVerificacion/id	servicioVerificacion.id	Obtener por id
POST	/servicioVerificacion	servicioVerificacion	Crear
PUT	/servicioVerificacion	servicioVerificacion	Editar
GET	/repuestosUtilizados	repuestosUtilizados	Obtener todos
GET	/repuestosUtilizados/id	repuestosUtilizados.id	Obtener por id
POST	/repuestosUtilizados	repuestosUtilizados	Crear
PUT	/repuestosUtilizados	repuestosUtilizados	Editar
GET	/detalleServicio	detalleServicio	Obtener todos
GET	/detalleServicio/id	detalleServicio.id	Obtener por id
POST	/detalleServicio	detalleServicio	Crear
PUT	/detalleServicio	detalleServicio	Editar
GET	/servicio	servicio	Obtener todos
GET	/servicio/id	servicio.id	Obtener por id
POST	/servicio	servicio	Crear
PUT	/servicio	servicio	Editar

4.1.2 Integración de plataformas

4.1.2.1 Amazon

Amazon AWS nos proporciona varias APIs para distintos lenguajes de programación. Para este desarrollo se utilizó la API de java para realizar la integración a los diferentes servicios que ofrece Amazon.

A continuación, se detalla cómo se realizó la integración con algunos de estos servicios.

4.1.2.1.1 Amazon SES

Amazon SES permite enviar correos electrónicos a los distintos usuarios del sistema dcGS, los correos se enviaron desde dcgs@mydecisioncloud.com realizando la siguiente configuración:

Verificación de un nuevo dominio

Se debe verificar un dominio y realizar las configuraciones para poder enviar correos electrónicos para lo cual se debe realizar el proceso Verify a New Domain y colocar el dominio que ya está configurado previamente en Route53.

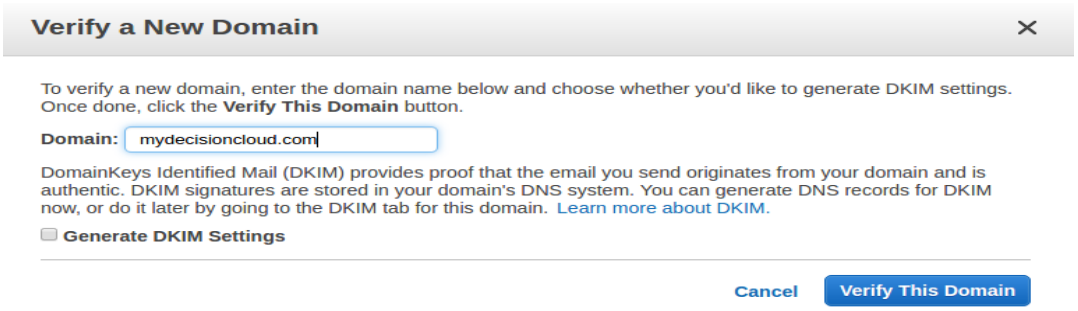

Figura 50 Verificación de un nuevo dominio

Una vez verificado se crea un nuevo registro automáticamente denominado MX en Route53 el cual permite enviar correos desde ese dominio.

Verificación del correo electrónico

Luego se debe verificar un nuevo correo electrónico desde donde se enviarán las notificaciones del sistema, para lo cual se accede a Verify a New Email Address.

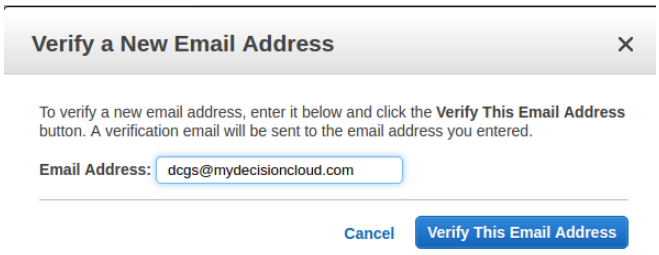

Figura 51 Verificación de un nuevo correo electrónico

Integración con JAVA

Para la integración de JAVA con Amazon SES es necesario añadir la librería javamail.jar y también se debe generar credenciales STMP para generar la conexión con el servicio de Amazon.

4.1.2.1.2 Amazon S3

Amazon S3 se integra con el aplicativo dcGS para el manejo de imágenes y archivos adjuntos.

Las imágenes que se almacenan en el servicio S3, son las fotografías de los usuarios mientras que los archivos adjuntos son los documentos que se atan a un servicio.

Integración con JAVA

Para la integración de JAVA con Amazon S3 es necesario añadir la librería AmazonS3Client que viene incluida en el SDK de Amazon, además es necesario crear credenciales AWS que contiene una llave pública y una privada.

```
aws_access_key_id  =  your_access_key_id
aws_secret_access_key  =  your_secret_access_key
```

Figura 52 Credenciales AWS

4.1.2.2 Asana

Para poder realizar la integración con Asana se debe registrar la aplicación que actúa como servidor, en este caso el microservicio *Gestión del Servicio*, en la página web proporcionada por el aplicativo.

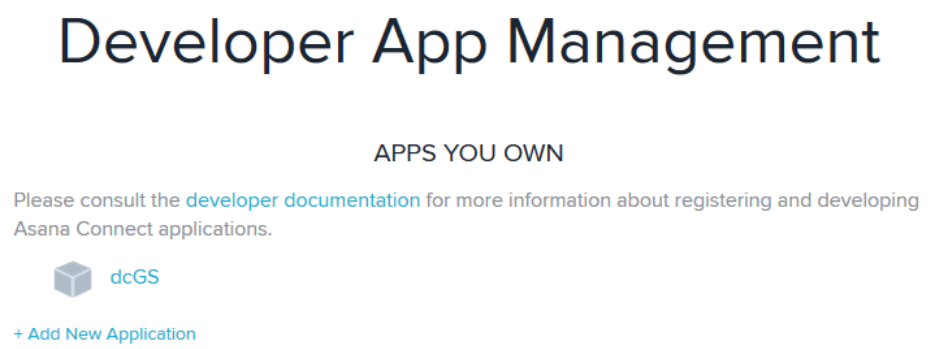

Figura 53 Aplicación registrada en la página de Asana

Una vez registrada la aplicación Asana genera un ID y una contraseña, estas se colocaron en un método en el microservicio para poder realizar la autenticación, cabe recalcar que estas credenciales deben ser manejadas con mucho cuidado y sólo usuarios que se encuentren autorizados para este fin.

Posteriormente se debe generar un token por el usuario que se va a autenticar en el Asana para la creación de tareas y su respectiva asignación.

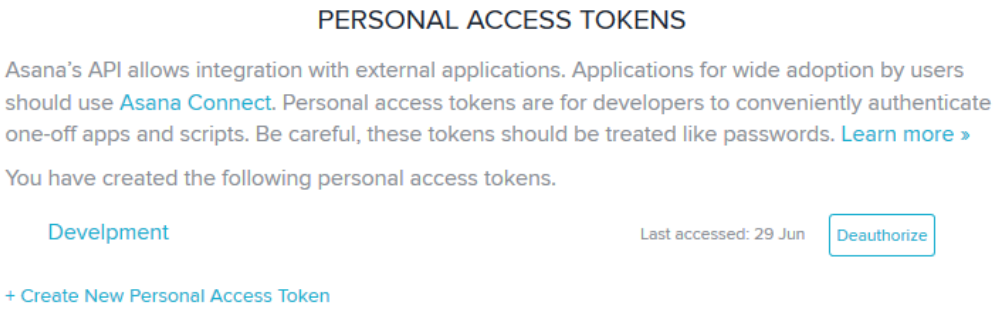

Figura 54 Token personal generado por el usuario

Este token se ingresó como parte de la autenticación en el microservicio, junto con las credenciales generadas previamente. Una vez hecho esto se realizó la comunicación exitosamente; este token debe ser manipulado por una persona autorizada, ya que el mismo equivale a la clave del usuario y podrá acceder a todos sus datos.

La comunicación se realizó mediante la interface RESTful, donde se envían ciertos datos requeridos por el Asana para poder manipular el aplicativo desde la

programación del microservicio, como la generación automática de tareas para que sean gestionadas por el usuario correspondiente.

4.2 Implementación

4.2.1 Configuración del entorno de desarrollo

Para la configuración del entorno de desarrollo se utilizó los servicios ofrecidos por Amazon AWS.

Amazon AWS tiene una amplia y completa guía para desarrolladores de todos sus servicios, es por tal razón que no se realizará un detalle minucioso de los pasos a seguir, sino más bien se hará énfasis en algunos procesos particulares que requieren una aclaración.

4.2.1.1 Configuración del servidor de aplicaciones (Amazon EC2)

Amazon EC2 cumple un rol muy importante en el desarrollo del aplicativo dcGS, porque es en donde se implementará el servidor de aplicaciones con los distintos entornos como: pruebas, producción y demostración.

Creación de la instancia

De todas las instancias que ofrece Amazon EC2, se eligió la versión denominada t2.micro con el sistema operativo Amazon Linux, debido a que es una capa gratuita que nos ofrece 750 horas de uso mensuales por un año.

La instancia t2.micro es una instancia muy pequeña pero muy útil para entornos de desarrollo con base de datos pequeñas, además en sus casos de uso indica que es para aplicaciones web diseñadas como microservicios.

Las características principales de este tipo de instancia son las siguientes:

- Procesadores Intel Xeon de alta frecuencia con Turbo hasta 3,3 GHz.
- Memoria RAM: 1Gb
- CPU Virtual: 1

La instancia por defecto viene configurada con IP dinámica, es decir cambia la IP cada vez que se reinicie la instancia, por lo tanto, se adquirió una IP elástica ofrecida

por Amazon que básicamente es una IP estática pública y la que permitirá acceder a los Servicios RESTful de la aplicación.

Servidor de aplicaciones

Como servidor de aplicaciones se utilizó Wildfly 9 implementado los perfiles Java EE7 full, fue instalado como servicio en la instancia creada en Amazon Linux y donde se desplegó los distintos microservicios de los diferentes entornos de desarrollo.

4.2.1.2 Configuración del servidor web (Amazon S3)

Amazon S3 es el encargado de alojar el cliente del aplicativo dcGS, en este servicio es necesario crear 3 contenedores (Buckets) los mismo que permitirán alojar los 3 ambientes de pruebas, producción y demostración.

Una vez creado los distintos buckets es necesario habilitarlos como web hosting y editar los permisos junto con la configuración CORS, los cuales permitirán acceder desde cualquier origen al cliente de la aplicación

4.2.1.3 Configuración de la base de datos (Amazon RDS)

Amazon RDS brinda el servicio de alojamiento de bases de datos, por lo tanto, este servicio será el encargado de alojar las bases de datos para los distintos entornos de desarrollo.

Amazon RDS ofrece bases de datos que usan MySQL, Oracle, PostgreSQL, Microsoft SQL, entre otros.

Creación de la instancia postgreSQL

Para el aplicativo dcGS se usó la base de datos postgreSQL 9.5 y se utilizó la instancia db.t2.micro, este tipo de instancia cuenta con las siguientes características:

- Rendimiento de red: Bajo
- Memoria RAM: 1GB
- CPU Virtual: 1

Conexión a la instancia postgreSQL

Una vez creada la instancia, es necesario ingresar a las propiedades y copiar el url del Endpoint que es la dirección que permitirá conectarse a la base de datos. Para la conexión se utilizó pgAdmin III que es el administrador de base de datos de postgreSQL

4.2.1.4 Distribución de contenido (CloudFront)

CloudFront permite distribuir contenido a través de la red para poder acceder de una manera rápida al cliente del aplicativo dcGS.

CloudFront está estrechamente ligado a los buckets del s3 ya que desde los mismo se distribuirá el contenido.

4.2.1.5 Configuración del dominio (Amazon Route53)

Para la puesta en producción del sistema, la compañía DECISIÓN c.a. adquirió un dominio llamado mydecisioncloud.com. Route53 permite administrar todos los registros relacionados con este dominio.

4.2.1.6 Configuración de envio de correos electrónicos (Amazon SES)

Amazon SES, permitirá enviar correos electrónicos desde el aplicativo dcGS, para lo cual se configuró un correo dcgs@mydecisioncloud.com desde donde se enviarán todas las notificaciones del sistema.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- La arquitectura orientada a microservicios permitió separar un software complejo y grande en pequeños sistemas escalables y robustos que se comunican entre sí para brindar alta disponibilidad y simplificando el mantenimiento y desarrollo de nuevos requerimientos.
- El monitoreo y las pruebas para asegurar el correcto funcionamiento de los microservicios se torna complejo debido a la cantidad concebida de los mismos, ya que todos estos módulos tienen un comportamiento dinámico.
- La comunicación entre los microservicios puede resultar compleja debido a la información que debe ser distribuida a cada uno de estos y a las llamadas de los servicios REST para procesar los datos enviados.
- Se logró la integración del aplicativo Decision Cloud con las aplicaciones externas de DECISIÓN c.a, obteniendo una mayor efectividad en el control de las novedades del servicio generadas por el cliente, para gestionar dichas novedades se realizó la comunicación con el API que provee ASANA, el mismo que permite generar tareas con su respectivo responsable y seguidores (Ver figura 55), obteniendo así un control total de los pendientes generados elevando el grado de satisfacción con respecto al servicio, como se ilustra en la Figura 56.

Figura 55. Tareas generadas automáticamente por el sistema

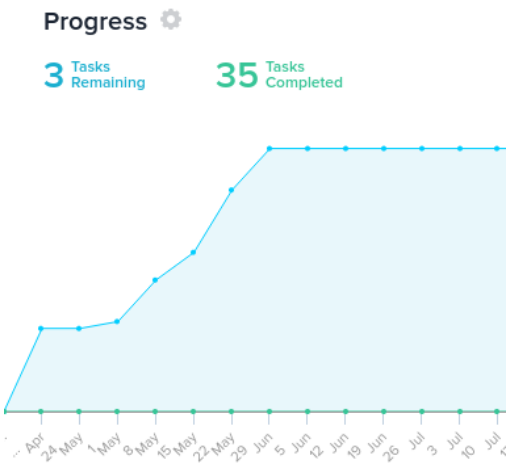

Figura 56. Tareas generadas automáticamente por el sistema

- En la evaluación mensual de clientes de la compañía DECISIÓN c.a. Aplicada a 24 de los 27 clientes activos finalizada el 30 junio de 2016, debido a la puesta en producción del aplicativo dcGS el nivel del servicio mejoró con respecto al mes anterior donde el 54% de los clientes está muy satisfecho con el servicio recibido, el 42% satisfecho y un 4% insatisfecho con el servicio (Ver figura 57).

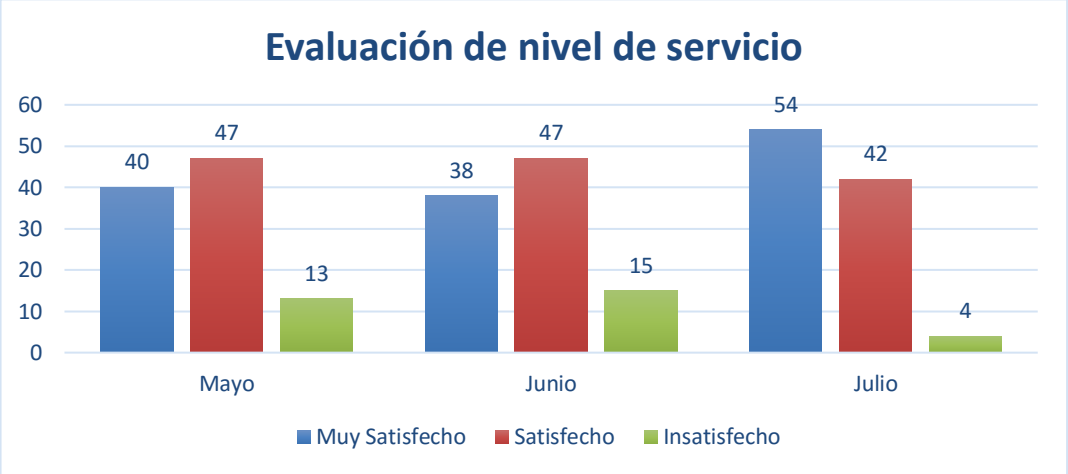

Figura 57 Resultado de evaluación realizada a los clientes de DECISIÓN c.a.

5.2 RECOMENDACIONES

- Es recomendable el uso de esta arquitectura para sistemas grandes, ya que ayuda a separarlos en partes más pequeñas de fácil mantenimiento, por tal razón se deben analizar los requerimientos funcionales del software.
- Existen servicios en Amazon que permiten el monitoreo de los microservicios, para poder verificar su correcto funcionamiento, por otra parte para las pruebas se encuentran herramientas como Arquillian para poder codificar pruebas reales.
- Para la comunicación entre los microservicios se recomienda el desarrollo de una aplicación, con ayuda de la arquitectura REST, para que reciba toda la información, sea procesada y enviada a cada uno de estos microservicios.
- La seguridad y confiabilidad de que la información no sea visualizada por usuarios o entidades no autorizadas es muy importante, por esta razón es

indispensable contar con un proceso de autenticación por tokens, para asegurar que el usuario es quien dice ser.

- Es de mucha utilidad tener un sistema de gestión de servicios al cliente debido a que se optimiza el tiempo de búsqueda de información de los servicios brindados en un intervalo de tiempo y permite obtener de una manera rápida datos necesarios para generar informes, consultas, reportes de servicio, facturación de horas, facturación de repuestos, etc.

REFERENCIAS BIBLIOGRAFICAS

- Peel, M. (1991). *El servicio al cliente: guía para mejorar la atención y la asistencia.* España: Deusto.
- PrimeFaces. (30 de 04 de 2015). *PrimeFaces*. Obtenido de http://www.primefaces.org/
- Telerik. (30 de 04 de 2015). *Telerik - Kendo UI*. Obtenido de http://www.telerik.com/kendo-ui
- Caliskan, M. (1996). *PrimeFaces Cookbook.* Birmingham: PACKPUB.
- Decision c.a. (11 de 05 de 2015). *Decision*. Obtenido de http://www.decision.com.ec/
- Hitt, B. P. (México). *Administración.* 2006: Michael, A. Hitt.
- iAdvize. (11 de 05 de 2015). *iAdvize*. Obtenido de http://www.iadvize.com
- Insights. (11 de 05 de 2015). *Insights*. Obtenido de http://insights.la/2015/03/13/la-importancia-del-servicio-al-cliente/
- LivePerson. (11 de 05 de 2015). *LivePerson*. Obtenido de http://www.liveperson.com/
- LMU – Ludwig-Maximilians-Universität München. (11 de 05 de 2015). *UWE – UML-based Web Engineering*. Obtenido de http://uwe.pst.ifi.lmu.de/teachingTutorialSpanish.html
- Marchioni, F. (s.f.). *Practical Java EE7 development on WildFly*.
- MicroServices. (11 de 05 de 2015). *MicroServices*. Obtenido de http://microservices.io/patterns/microservices.html

- Peel, M. (1991). *El servicio al cliente: guía para mejorar la atención y la asistencia.* España: Deusto.
- PrimeFaces. (30 de 04 de 2015). *PrimeFaces*. Obtenido de http://www.primefaces.org/

- Spona, H. (2008). *Programación de Base de Datos con mysql y php.* Barcelona: MARCOMBO.

- Telerik. (30 de 04 de 2015). *Telerik - Kendo UI*. Obtenido de http://www.telerik.com/kendo-ui

- Writing@CSU. (11 de 05 de 2015). *The Witring Studio*. Obtenido de http://writing.colostate.edu/guides/guide.cfm?guideid=60
- Asana. (13 de 05 de 2015). *Asanna*. Obtenido de https://asana.com/
- Google. (13 de 05 de 2015). *Google Developers*. Obtenido de https://developers.google.com/products/
- JBoss. (13 de 05 de 2015). *WildFly*. Obtenido de http://wildfly.org/
- TechTarget. (05 de 13 de 2015). *Search*. Obtenido de http://searchsoa.techtarget.com/definition/J2E
- Best, R. J. (2007). *Marketing estratégico*. Madrid: Pearson Prentice Hall.
- München, L. –L.-M.-U. (27 de 08 de 2014). *UWE – UML-based Web Engineering*. Recuperado el 15 de 01 de 2015, de http://uwe.pst.ifi.lmu.de/aboutUwe.html
- Tocto, E. (2011). Process Optimization and Quantification using BPM. 22.
- ESPE. (2011). *UNIVERSIDAD DE LAS FUERZAS ARMADAS- ESPE*. Obtenido de http://www.espe.edu.ec/
- Euclides. (2011). *Grupo Euclides*. Obtenido de http://www.grupoeuclides.com/es/soluciones/automatizacion-de-procesos-documentales
- Muñiz, L. (2004). ERP: GUIA PRACTICA PARA LA SELECCION E IMPLANTACION. España: Planeta DeAgostini Profesional y Formación,SL.
- Rossi, G., Pastor, O., Schwabe, D., & Olsina, L. (2008). Web Engineering: Modelling and Implementing Web Applications. Poznen: Springer Science & Business Media.
- PIÑEIRO GÓMEZ, J. M. (2014). *UF2176 - Definición y manipulación de datos.* España: Ediciones Paraninfo, S.A.
- affiliates, O. a. (05 de 2013). *GlassFish Server Open Source Edition.* Recuperado el 26 de 02 de 2015, de GlassFish Server: https://glassfish.java.net/documentation.html

- *NetBeans IDE 8.0.2 Information*. (s.f.). Recuperado el 26 de 02 de 2015, de https://netbeans.org: https://netbeans.org/community/releases/80/
- Forsythe, L., Herrera, A., Muñoz, A., Samaniego, M., & Villarreal, A. (2010). "Plataformas J2SE, J2EE, J2ME".
- Kawaguchi, K. (19 de 06 de 2007). *Introducing Metro*. Recuperado el 26 de 02 de 2015, de java.net: https://weblogs.java.net/blog/kohsuke/archive/2007/06/introducing_met.html
- affiliates, O. C. (2014). *JSR 344: JavaServerTM Faces 2.2*. Recuperado el 26 de 02 de 2015, de Java Community Access: https://www.jcp.org/en/jsr/detail?id=344
- Oracle. (s.f.). *¿Qué es Java?* Recuperado el 26 de 02 de 2015, de Oracle: https://www.java.com/es/about/
- Monson-Haefel, R. (2004). *Enterprise JavaBeans.* O'Reilly.
- Eubanks, B. D. (2005). *Wicked Cool Java: code bits, open-source libraries, and project ideas.* No Starch Press.
- Nurzhan Nurseitov, M. P. (12 de 4 de 2009). Comparison of JSON and XML Data Inter change Formats: A Case Study. Bozeman, Montana, Estados Unidos.
- Google. (20 de Marzo de 2015). *Google Empresa*. Recuperado el 21 de Junio de 2015, de Google: https://www.google.com/about/company/
- Asana. (1 de Junio de 2015). *About.* Recuperado el 21 de Junio de 2015, de Asana: https://asana.com/company
- Schafer Group. (16 de Noviembre de 2011). *Web Service Integration Services*. Recuperado el 21 de Junio de 2015, de Schafer Group: http://www.theschafergroup.com/Blog/BlogArticles/tabid/85/articleType/ArticleView/articleId/8/Web-Service-Integration-Services.aspx

- Carmona Barbero, P. (4 de Febrero de 2014). *ProjETSII: servicio para la gestión de trabajos académicos.* Recuperado el 21 de Junio de 2015, de Universidad de Sevilla: https://projetsii.informatica.us.es/attachments/download/2760/Plataformas_de_integraci%C3%B3n.Servicios_Web_REST_y_SOAP.pdf

- Oracle. (1 de Diciembre de 2014). *Java Enterprise Edition.* Recuperado el 21 de Junio de 2015, de Oracle: http://www.oracle.com/technetwork/java/javaee/overview/index.html

- UNIVERSIDAD CARLOS III DE MADRID. (3 de Mayo de 2009). Arquitecturas Empresariales y la plataforma J2EE. *Junio*(21). Madrid, Madrid, España.

- Ejemplos TIW. (12 de Mayo de 2012). *Patrón de arquitectura Modelo Vista Controlador (MVC)*. Recuperado el 21 de Junio de 2015, de Ejemplos TIW: http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html

- Spring. (10 de Abril de 2015). *Introduction to Spring Web MVC*. Recuperado el 21 de Junio de 2015, de Spring: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html

- Oracle. (1 de Mayo de 2015). *Java Servlet Technology*. Recuperado el 21 de Junio de 2015, de Oracle: http://www.oracle.com/technetwork/java/index-jsp-135475.html

- Alvarez, M. (2 de Enero de 2014). *¿Qué es MVC?* Recuperado el 21 de Junio de 2015, de Desarrollo Web: http://www.desarrolloweb.com/articulos/que-es-mvc.html

- Google. (12 de Marzo de 2015). *Introduction*. Recuperado el 21 de Junio de 2015, de Angular JS: https://angularjs.org/

- Álvarez, C. (23 de Enero de 2015). *Microservicios*. Recuperado el 21 de Junio de 2015, de Arquitectura Java: http://www.arquitecturajava.com/que-es-un-microservicio/

- München, L. –L.-M.-U. (4 de Septiembre de 2013). *UWE – UML-based Web Engineering*. Recuperado el 21 de Junio de 2015, de About UWE: http://uwe.pst.ifi.lmu.de/aboutUwe.html