



**UNIVERSIDAD DISTRITAL  
FRANCISCO JOSÉ DE CALDAS**

# **Arquitectura Propuesta para un Servicio Web Completo: Metodología de Desarrollo e Implementación**

**Ing. Gabriel Eduardo Duarte Vega**

Universidad Distrital Francisco José de Caldas  
Facultad de Ingeniería  
Departamento Especialización Ingeniería de Software  
Bogotá D.C., Colombia  
2016



# **Arquitectura Propuesta para un Servicio Web Completo: Metodología de Desarrollo e Implementación**

**Ing. Gabriel Eduardo Duarte Vega**

Tesis presentada como requisito para optar al título de:  
**Especialista en Ingeniería de Software**

Director:  
Ing. Sandro Javier Bolaños Castro, Ph.D.

Revisor:  
Ing. Jorge Mario Calvo Londono, Ph.D.

Línea de Investigación:  
Computación Aplicada

Universidad Distrital Francisco José de Caldas  
Facultad de Ingeniería  
Departamento Especialización Ingeniería de Software  
Bogotá D.C., Colombia  
2016



# Índice general

<b>Lista de figuras</b>	<b>ix</b>
<b>Lista de tablas</b>	<b>1</b>
<b>1. Introducción</b>	<b>2</b>
<b>I. CONTEXTUALIZACIÓN DE LA INVESTIGACIÓN</b>	<b>3</b>
<b>2. DESCRIPCIÓN DE LA INVESTIGACIÓN</b>	<b>4</b>
2.1. Planteamiento e Identificación del Problema . . . . .	4
2.1.1. Formulación del Problema . . . . .	4
2.1.2. Sistematización del Problema . . . . .	4
2.2. Objetivos . . . . .	5
2.2.1. Objetivo General . . . . .	5
2.2.2. Objetivos Específicos . . . . .	5
2.3. Justificación del Trabajo de Investigación . . . . .	5
2.4. Hipótesis . . . . .	5
2.5. Metodología de la Investigación . . . . .	5
2.5.1. Tipo de Estudio . . . . .	5
2.5.2. Método de Investigación . . . . .	5
2.6. Organización del trabajo . . . . .	5
2.7. Estudios de Sistemas Previos . . . . .	6
<b>3. MARCO REFERENCIAL</b>	<b>7</b>
3.1. Marco Teórico . . . . .	7
3.1.1. Patrones de Arquitectura de Software . . . . .	7
3.1.2. Protocolos y Estándares de Servicios Web . . . . .	11
3.1.3. Business Model Canvas . . . . .	12
3.1.4. Arquitectura Empresarial . . . . .	16
3.1.5. Archimate . . . . .	19
3.1.6. Archimate: Capa de Negocio . . . . .	21
3.1.7. Archimate: Capa de Aplicación . . . . .	24
3.1.8. Archimate: Capa de Infraestructura . . . . .	26
3.1.9. Archimate: Capa Motivacional . . . . .	27
3.1.10. Archimate: Puntos de Vista . . . . .	28
3.1.11. Archimate: Extensiones de Lenguaje ArchiMate . . . . .	39
3.2. Marco Conceptual . . . . .	41
3.2.1. API - Interfaz de Programación de Aplicaciones . . . . .	41
3.2.2. Arquitectura SOA . . . . .	41
3.2.3. Servicio Web . . . . .	41
3.2.4. SOA y Servicios Web . . . . .	43

3.2.5. Arquitectura de Microservicios . . . . .	43
3.2.6. RabbitMQ . . . . .	43
3.2.7. Sectores Económicos en Colombia . . . . .	44
<b>II. DESARROLLO DE LA INVESTIGACIÓN</b>	<b>46</b>
<b>4. METODOLOGÍA PROPUESTA</b>	<b>47</b>
4.1. Arquitectura Propuesta . . . . .	47
4.1.1. Investigación y Análisis Previo . . . . .	47
4.1.2. Servicio Web Completo . . . . .	48
4.1.3. Arquitectura Propuesta para un Servicio Web Completo . . . . .	50
4.2. Tecnologías Seleccionadas . . . . .	52
4.2.1. .NET Framework 4.5 . . . . .	53
4.2.2. Visual Studio 2015 Community . . . . .	55
4.2.3. Visual Studio Online . . . . .	56
4.2.4. ASP .NET - Web API 2 . . . . .	56
4.2.5. PowerShell . . . . .	57
4.2.6. Rabbit MQ . . . . .	58
4.2.7. Dot Net Nuke . . . . .	58
<b>5. EJERCICIO APLICACIÓN</b>	<b>60</b>
5.1. Arquitectura Empresarial . . . . .	60
5.1.1. Fase Preliminar . . . . .	60
5.1.2. Fase A: Visión de la Arquitectura . . . . .	65
5.1.3. Fase B: Arquitectura del Negocio . . . . .	67
5.1.4. Fase C: Arquitectura de Datos y Aplicación . . . . .	69
5.1.5. Fase D: Arquitectura de la Infraestructura . . . . .	72
5.1.6. Fase E: Oportunidades y Soluciones . . . . .	75
5.1.7. Fase F: Planear la Migración . . . . .	79
5.2. Desarrollo e Implementación . . . . .	82
5.2.1. Creando el Servicio Web . . . . .	82
5.2.2. Ruta de Acceso o Enrutamiento . . . . .	84
5.2.3. Métodos . . . . .	85
5.2.4. Seguridad Autorización de Solicitudes . . . . .	88
5.2.5. Patrón Fluent . . . . .	92
5.2.6. Parámetros de los Métodos . . . . .	95
5.2.7. CRUD de Métodos Expuestos . . . . .	98
5.2.8. Seguimiento, Trazabilidad o Auditoría . . . . .	100
5.2.9. Pruebas Unitarias . . . . .	102
<b>6. RECOLECCIÓN, ANÁLISIS Y PRESENTACIÓN DE INFORMACIÓN</b>	<b>105</b>
6.1. Recolección y Ordenamiento de la Información . . . . .	105
6.1.1. Población . . . . .	105
6.1.2. Muestra . . . . .	106
6.1.3. Presentación de los Resultados . . . . .	107
6.2. Presentación y Análisis de los Resultados . . . . .	116

<b>III. CIERRE DE LA INVESTIGACIÓN</b>	<b>118</b>
<b>7. RESULTADOS Y DISCUSIÓN</b>	<b>119</b>
<b>8. CONCLUSIONES</b>	<b>120</b>
8.0.1. Conclusiones . . . . .	120
8.0.2. Aportes originales . . . . .	121
<b>Bibliografía</b>	<b>122</b>
<b>A. Anexo: Diagrama Modelo Business Model Canvas - BMC</b>	<b>126</b>
<b>B. Anexo: BMS de una Cooperativa con Negocios de Productos de Consumo Masivo</b>	<b>127</b>
<b>C. Anexo: Sectores Económicos en Colombia</b>	<b>128</b>
<b>D. Anexo: .NET Framework 4.5</b>	<b>129</b>
<b>E. Anexo: Encuesta Aceptación, Uso, Aprendizaje y Desarrollo de Servicios Web</b>	<b>130</b>
<b>F. Anexo: Código Controladora de Aplicaciones</b>	<b>138</b>
<b>G. Anexo: Código y Comandos de PowerShell Prueba Unitaria Automatizada de la Controladora de Aplicaciones</b>	<b>140</b>



# **Lista de Figuras**

<b>3-1.</b> Diagrama Modelo Cliente-Servidor Fuente: [1]. . . . .	8
<b>3-2.</b> Diagrama Arquitectura Orientada a Servicios. . . . .	9
<b>3-3.</b> Diagrama Modelo 3 Capas Arquitectura n-Capas Fuente: [2]. . . . .	9
<b>3-4.</b> Diagrama Arquitectura Orientada a Eventos EDA Fuente: [3]. . . . .	10
<b>3-5.</b> Ejemplo arquitectura ESB para integrar aplicaciones. Fuente: [4]. . . . .	11
<b>3-6.</b> Diagrama Modelo BMC. Fuente: [5]. . . . .	12
<b>3-7.</b> Línea de Tiempo de una Arquitectura Empresarial. Fuente: [6] . . . . .	17
<b>3-8.</b> Correspondencia entre TOGAF y el Ciclo ADM de Archimate incluyendo las extensiones. Fuente: Capítulo 2 [7] . . . . .	19
<b>3-9.</b> Integración del ADM con ArchiMate. Fuente: [8] . . . . .	21
<b>3-10.</b> Metamodelo Punto de Vista Organización. . . . .	29
<b>3-11.</b> Metamodelo Punto de Vista Cooperación del Actor. . . . .	29
<b>3-12.</b> Metamodelo Punto de Vista Función del Negocio. . . . .	30
<b>3-13.</b> Metamodelo Punto de Vista Procesos de Negocio. . . . .	30
<b>3-14.</b> Metamodelo Punto de Vista Cooperación Procesos de Negocio. . . . .	31
<b>3-15.</b> Metamodelo Punto de Vista Producto. . . . .	31
<b>3-16.</b> Metamodelo Punto de Vista de Comportamiento de Aplicación. . . . .	32
<b>3-17.</b> Metamodelo Punto de Vista de Cooperación de Aplicación. . . . .	32
<b>3-18.</b> Metamodelo Punto de Vista de Estructura de Aplicación. . . . .	33
<b>3-19.</b> Metamodelo Punto de Vista de Uso de Aplicación. . . . .	33
<b>3-20.</b> Metamodelo Punto de Vista de Infraestructura. . . . .	34
<b>3-21.</b> Metamodelo Punto de Vista de Uso de Infraestructura. . . . .	34
<b>3-22.</b> Metamodelo Punto de Vista de Organización e Implementación. . . . .	35
<b>3-23.</b> Metamodelo Punto de Vista de Estructura de Información. . . . .	35
<b>3-24.</b> Metamodelo Punto de Vista de Realización del Servicio. . . . .	36
<b>3-25.</b> Metamodelo Punto de Vista de Interesados. . . . .	36
<b>3-26.</b> Metamodelo Punto de Realización de Objetivos. . . . .	36
<b>3-27.</b> Metamodelo Punto de Vista de Contribución. . . . .	37
<b>3-28.</b> Metamodelo Punto de Vista de Principios. . . . .	37
<b>3-29.</b> Metamodelo Punto de Vista de Realización de Requerimientos. . . . .	37
<b>3-30.</b> Metamodelo Punto de Vista de Motivación. . . . .	38
<b>3-31.</b> Metamodelo Punto de Vista de Proyecto. . . . .	38
<b>3-32.</b> Metamodelo Punto de Vista de Migración. . . . .	38
<b>3-33.</b> Metamodelo Punto de Vista de Migración e Implementación. . . . .	39
<b>3-34.</b> Los Servicios Web en Funcionamiento. Fuente: [9]. . . . .	42
<b>3-35.</b> Orquestación y Coreografía de Servicios Web. Fuente: [10] . . . . .	43
<b>3-36.</b> Diagrama Patrón RPC de RabbitMQ Fuente: [11]. . . . .	44
<b>3-37.</b> Sectores Económicos en Colombia. Fuente: [12] . . . . .	45
<b>4-1.</b> Diagrama de Componentes Propuesto de un Servicio Web Completo. . . . .	49
<b>4-2.</b> Arquitectura Propuesta para un Servicio Web Completo. . . . .	51
<b>4-3.</b> Arquitectura Propuesta para un Servicio Web Completo con Rabbit MQ. . . . .	52

<b>4-4.</b> .NET Framework 4.5. Fuente: [13]. . . . .	54
<b>4-5.</b> .NET Framework 5.0. Fuente: [14] . . . . .	55
<b>4-6.</b> Microsoft® PowerShell® ISE Fuente: [15] . . . . .	57
<b>4-7.</b> Rabbit MQ Utilizado como RPC Fuente: [11] . . . . .	58
<b>5-1.</b> BMS de una Cooperativa con Negocios de Productos de Consumo Masivo. (Para mas detalle ver anexo B) . . . . .	61
<b>5-2.</b> Organigrama Cooperativa de un Negocio de Productos de Consumo Masivo. . . . .	63
<b>5-3.</b> Punto de Vista Motivacional de un Negocios de Productos de Consumo Masivo. . . . .	65
<b>5-4.</b> Estado Actual (AS-IS) de un Negocio de Productos de Consumo Masivo. . . . .	65
<b>5-5.</b> Arquitectura del Negocio de Domicilios Futura (TO-BE). . . . .	66
<b>5-6.</b> Punto de Vista de Cooperación de Actor. . . . .	67
<b>5-7.</b> Punto de Vista de Función de Negocio. . . . .	68
<b>5-8.</b> Punto de Vista de Procesos de Negocio. . . . .	68
<b>5-9.</b> Punto de Vista de Cooperación de Procesos de Negocio. . . . .	69
<b>5-10.</b> Punto de Vista de Producto. . . . .	69
<b>5-11.</b> Punto de Vista de Comportamiento de Aplicación. . . . .	70
<b>5-12.</b> Punto de Vista de Cooperación de la Aplicación. . . . .	70
<b>5-13.</b> Punto de Vista de Estructura de la Aplicación. . . . .	71
<b>5-14.</b> Punto de Vista de Uso de la Aplicación. . . . .	71
<b>5-15.</b> Punto de Vista de Infraestructura. . . . .	72
<b>5-16.</b> Punto de Vista de Uso de Infraestructura. . . . .	72
<b>5-17.</b> Punto de Vista de Implementación en la Organización. . . . .	73
<b>5-18.</b> Punto de Vista de Estructura de Información. . . . .	74
<b>5-19.</b> Punto de Vista de Realización del Servicio. . . . .	74
<b>5-20.</b> Punto de Vista de Capas. . . . .	75
<b>5-21.</b> Punto de Vista de Implicados. . . . .	76
<b>5-22.</b> Punto de Vista de Realización de Objetivos. . . . .	76
<b>5-23.</b> Punto de Vista de Contribución de Objetivos. . . . .	77
<b>5-24.</b> Punto de Vista de Principios. . . . .	77
<b>5-25.</b> Punto de Vista de Realización de Requerimientos. . . . .	78
<b>5-26.</b> Punto de Vista del Proyecto. . . . .	79
<b>5-27.</b> Punto de Vista de la Migración. . . . .	80
<b>5-28.</b> Punto de Vista de la Implementación de la Migración. . . . .	81
<b>5-29.</b> Creando en Visual Studio un Proyecto de Módulo DNN. 2016 . . . . .	83
<b>5-30.</b> Estructura Recomendada Proyecto Módulo DNN para el Servicio Web. . . . .	83
<b>5-31.</b> Visor de Eventos de las Operaciones en el Portal de DNN . . . . .	102
<b>5-32.</b> Convenciones de los Tipos de Evento Utilizados en el Visor de Eventos de DNN . . . . .	102
<b>6-1.</b> Cantidad de Graduados en Ingeniería de Sistemas en la Universidad Nacional y Distrital en Bogotá en el 2014. Fuente: Observatorio Laboral del Ministerio de Educación. [16] . . . . .	106
<b>6-2.</b> Porcentajes de Edad de los Profesionales Encuestados. . . . .	108
<b>6-3.</b> Porcentajes por Nivel Académico de los Profesionales Encuestados. . . . .	108
<b>6-4.</b> Porcentajes por Profesión de los Profesionales Encuestados. . . . .	109
<b>6-5.</b> Porcentajes de Plataformas de Desarrollo de Software Utilizadas por los Profesionales Encuestados. . . . .	109
<b>6-6.</b> Porcentajes de Profesionales Encuestados Que Han Creado Componentes. . . . .	110
<b>6-7.</b> Porcentajes de Profesionales Encuestados Que Han Creado Aplicaciones de Consola. . . . .	110
<b>6-8.</b> Porcentajes de Profesionales Encuestados Que Han Creado Aplicaciones de Escritorio. . . . .	111

<b>6-9.</b> Porcentajes de Profesionales Encuestados Que Han Creado Portales Web. . . . .	111
<b>6-10</b> Porcentajes de Profesionales Encuestados Que Conocen La Diferencia de Conceptos entre SOA, WS y ESB. . . . .	112
<b>6-11</b> Porcentajes de Profesionales Encuestados Que Han Consumido Servicios Web. . . . .	112
<b>6-12</b> Porcentajes de Profesionales Encuestados Que Han Creado Servicios Web. . . . .	113
<b>6-13</b> Porcentajes de Tiempo Empleado en Aprender Alguna Tecnología de Servicio Web por los Profesionales Encuestados. . . . .	113
<b>6-14</b> Porcentajes de Tiempo Empleado en Crear Servicio Web por los Profesionales Encuestados. . . . .	114
<b>6-15</b> Porcentajes de Profesionales Encuestados Que Consideran Viable Económicamente Incluir un Servicio Web en Proyectos de Software. . . . .	114
<b>6-16</b> Porcentajes de Nivel de Costos de un Proyecto con Servicio Web Considerado por los Profesionales Encuestados. . . . .	115
<b>6-17</b> Porcentajes de Profesionales Encuestados Que Incluirían Servicios Web en sus Proyectos de Software. . . . .	115
<b>6-18</b> Porcentajes de Aceptación Personal de Servicios Web de los Profesionales Encuestados. . . . .	116
<b>6-19</b> Porcentajes de Aceptación de Servicios Web de las Personas Que Rodean a los Profesionales Encuestados. . . . .	116
<b>A-1.</b> Diagrama Modelo BMC. Fuente: [5]. . . . .	126
<b>B-1.</b> BMS de una Cooperativa con Negocios de Productos de Consumo Masivo. . . . .	127
<b>C-1.</b> Sectores Económicos en Colombia. Fuente: [12] . . . . .	128
<b>D-1.</b> .NET Framework 4.5. Fuente: [13] . . . . .	129
<b>E-1.</b> Formulario Encuesta: Datos del Encuestado . . . . .	130
<b>E-2.</b> Formulario Encuesta: Nivel Académico del Encuestado . . . . .	131
<b>E-3.</b> Formulario Encuesta: Conocimientos del Encuestado Primera Parte . . . . .	132
<b>E-4.</b> Formulario Encuesta: Conocimientos del Encuestado Segunda Parte . . . . .	133
<b>E-5.</b> Formulario Encuesta: Usos y creación de Servicios Web del Encuestado . . . . .	134
<b>E-6.</b> Formulario Encuesta: Tiempos de Aprendizaje y creación de Servicios Web del Encuestado . . . . .	135
<b>E-7.</b> Formulario Encuesta: Viabilidad de los Servicios Web para el Encuestado . . . . .	136
<b>E-8.</b> Formulario Encuesta: Aceptación de los Servicios Web por el Encuestado . . . . .	137



# **Lista de Tablas**

<b>3-1.</b> Conceptos Pasivos de la Capa de Negocio en Archimate.[7, Chapter 3] . . . . .	22
<b>3-2.</b> Conceptos Activos de la Capa de Negocio en Archimate.[7, Chapter 3] . . . . .	23
<b>3-3.</b> Conceptos Informacionales de la Capa de Negocio en Archimate.[7, Chapter 3] . . . . .	24
<b>3-4.</b> Elementos Estructurales de la Capa de Aplicación en Archimate.[7, Chapter 4] . . . . .	25
<b>3-5.</b> Elementos Comportamentales de la Capa de Aplicación en Archimate.[7, Chapter 4] . . . . .	26
<b>3-6.</b> Elementos de la Capa de Infraestructura en Archimate.[7, Chapter 5] . . . . .	27
<b>3-7.</b> Elementos de la Capa de Motivacional en Archimate.[7, Chapter 10] . . . . .	28

# 1. Introducción

Partiendo del Know How<sup>1</sup> adquirido por el autor de este trabajo como ingeniero de software en empresas del sector de los servicios financieros, transporte y petrolera, se observa varios inconvenientes que se resuelven mediante una arquitectura con el concepto de servicio web completo un nuevo concepto de servicio web y la tecnología en la cual se implementa para conseguir un conjunto de técnicas y métodos que disminuyan el tiempo de implementación evitando todos los inconvenientes observados, a demás de sugerir herramientas y tecnologías que disminuyen el tiempo de desarrollo.

La investigación de esta problemática en la cual se ve envuelta el diseño y desarrollo de los servicios web, se realizó por el interés de conocer qué componentes mínimos definen la arquitectura que incluya servicios web sin importar el contexto donde se aplique.

El desarrollo del trabajo inicia con el análisis de las arquitecturas existentes definiendo un nuevo concepto, el de servicio web completo del cual se presenta una arquitectura con sus componentes y relaciones.

Luego comentan las tecnologías seleccionadas para implementar los componentes de la arquitectura y finalmente se aplica mediante una serie de pasos, utilizando técnicas, métodos y patrones el desarrollo e implementación del servicio web.

La característica principal de este proyecto es la utilización de técnicas, métodos y tecnologías en forma ordenada, lógica y coherente. Más que llegar a ser un manual o un tutorial permite orientar al lector a tener una visión clara de la definición de una buena arquitectura, selección de tecnologías e implementación adecuadas en muy poco tiempo en cualquier área donde se quiera intercomunicar sistemas.

Profundizar la indagación desde la perspectiva tecnológica y de innovación, fue un interés académico. Asimismo, aportar nuevo conocimiento exploratorio inductivo.

En el ámbito profesional, como ingeniero, el interés versa en conocer las tecnologías como variables independientes de las condiciones en las que se cree el servicio.

En la bibliografía consultada no se tratan temas que propongan una arquitectura que permita dividir el trabajo en componentes y que al mismo tiempo contemple un conjunto de características deseadas con el uso de una tecnología específica que disminuya drásticamente el tiempo de desarrollo e implementación.

La arquitectura y tecnologías propuestas son la clave para evitar reprocesos y sobrecostos.

---

<sup>1</sup>Know-how —del inglés «saber cómo»— o conocimiento fundamental, es un neologismo anglosajón que hace referencia a una forma de transferencia de tecnología. Se ha empezado a utilizar habitualmente en los últimos tiempos en el ámbito del comercio internacional para denominar a los conocimientos preexistentes, no siempre académicos, que incluyen técnicas, información secreta, teorías e incluso datos privados —como clientes o proveedores—.[17]

**Parte I.**

# **CONTEXTUALIZACIÓN DE LA INVESTIGACIÓN**

## **2. DESCRIPCIÓN DE LA INVESTIGACIÓN**

### **2.1. Planteamiento e Identificación del Problema**

Existen diversas tecnologías para crear servicios web cada una con su documentación y ejemplos, incluso hay generadores de código automáticos, pero estos no tratan temas sobre integrar varias tecnologías para cumplir ciertos requerimientos funcionales y no funcionales y permitir que su construcción sea fácil y rápida.

Aunado a lo anterior se encuentra una cantidad abrumadora de información por cada tecnología existente, cuyo tiempo o curva de aprendizaje e implementación es muy largo y a esto se le suma la baja aceptación y desconocimiento de los servicios web por parte de desarrolladores, empresas de software y su uso en las organizaciones, academia e investigación.

El autor observa que no utilizar servicios web trae consecuencias: sistemas que no permitan intercambiar información de forma autónoma y automática, no interconectar los sistemas internos entre sí en las organizaciones o con sistemas externos, desarrolladores o empresas de software, no contar con un sistema de comunicación seguro, rápido, fácil de estudiar y modificar disminuyendo costos y tiempo en proyectos de investigación, por ejemplo en temas como la computación en la nube, el Internet de las cosas, por nombrar algunos.

El tratamiento ha seguir para evitar el problema encontrado es proponer una arquitectura, definir un servicio web completo y seleccionar una tecnología que disminuya el tiempo de desarrollo e implementación para llevarlo a cabo.

#### **2.1.1. Formulación del Problema**

¿Cómo diseñar una arquitectura para implementar servicios web completos de forma rápida y sencilla que cumplan un conjunto de características y cualidades deseadas utilizando una tecnología específica?

#### **2.1.2. Sistematización del Problema**

- ¿Cuáles deben ser los componentes que debe tener una arquitectura que incluya un servicio web para hacerla escalable, segura y eficiente?
- ¿Qué técnicas y métodos de la tecnología seleccionada debe tenerse en cuenta para crear en muy poco tiempo un servicio web?
- ¿Cómo utilizar la arquitectura con la tecnología seleccionada en la implementación de una solución en un tema específico?

## **2.2. Objetivos**

### **2.2.1. Objetivo General**

Proponer una arquitectura que mediante tecnologías específicas permitan crear servicios web con características y cualidades específicas.

### **2.2.2. Objetivos Específicos**

- Proponer una arquitectura para crear servicios web con las características y cualidades necesarias mediante la combinación de componentes de varias tecnologías.
- Seleccionar las tecnologías más recientes para el desarrollo e implementación de un servicio web completo siguiendo la arquitectura propuesta.
- Aplicar la arquitectura propuesta mediante la tecnología seleccionada como parte de la solución en la transformación digital de un negocio.

## **2.3. Justificación del Trabajo de Investigación**

La motivación para desarrollar esta investigación es aportar una arquitectura que integre tecnologías que sirva de referencia para otras investigaciones o para la creación de servicios web en la academia, organización e industria, de tal forma que tenga una ventaja significativa con métodos tradicionales y empíricos, permitiendo observar la validez del método a través de su aplicación y comparación, y finalmente buscando incentivar el uso de servicios web para el intercambio de información entre sistemas de forma eficaz, eficiente y segura.

## **2.4. Hipótesis**

Si se ordenan y acotan las técnicas empleadas con la tecnología seleccionada, entonces, disminuirá la complejidad y el tiempo de desarrollo e implementación de la arquitectura propuesta con su conjunto de características y cualidades deseadas.

## **2.5. Metodología de la Investigación**

### **2.5.1. Tipo de Estudio**

Esta investigación será de tipo exploratoria. Este es tal vez un primer acercamiento al conocimiento del problema que se plantea identificando sus elementos y características para resolverlo y servirá como base para la realización de nuevas investigaciones por otros autores.

### **2.5.2. Método de Investigación**

El método de investigación es inductivo - deductivo.

## **2.6. Organización del trabajo**

Como punto de partida se inicio acudiendo a la experiencia personal y la documentación existente sobre los servicios web, luego se realizó una búsqueda bibliográfica para encontrar temas relacionados con estos, los

cuales se analizarán y explicarán en el marco teórico y conceptual.

Luego se propone una arquitectura y sus componentes, se define el concepto de servicio web completo, se selecciona la tecnología a utilizar para desarrollarlo.

Con los resultados de la encuesta realizada se pudo comparar los tiempos de desarrollo e implementación al aplicar la arquitectura propuesta para un servicio web como parte de la solución en una transformación digital de un negocio.

## 2.7. Estudios de Sistemas Previos

En la actualidad, existe un incremento importante en el desarrollo de sistemas basados en una arquitectura orientada a servicios. Dichos sistemas aprovechan la gran oferta de servicios web en inglés Web Services (WS) existentes en la red para implementar funcionalidades.

Este cambio de paradigma es tan grande que incluso se han definido lenguajes formales de alto nivel que permiten describir un proceso de negocio mediante servicios web e incluso a partir de modelos para generar el código automáticamente.

También se han propuesto técnicas para implementar servicios web con lenguajes de alto nivel como BPEL (Business Process Execution Language) que mediante su orquestación define el flujo completo de un proceso, por ejemplo el de un negocio específico.[18]

## 3. MARCO REFERENCIAL

### 3.1. Marco Teórico

#### 3.1.1. Patrones de Arquitectura de Software

En el diseño de la arquitectura de software los **elementos y relaciones** están definidos a un alto nivel mediante los **requisitos funciones y no funcionales**. Los requisitos definen la arquitectura, por ejemplo, si se quiere implementar una sitio web en el que se pida autenticación para acceder, su arquitectura estará marcada por los requisitos de seguridad, confidencialidad e interfaz, los cuales definen finalmente las tecnologías y sus interacciones. [19, 20]

Teniendo en cuenta los requisitos y restricciones se proporcionan los patrones de diseño arquitectónicos, ya que estos ayudan a resolver problemas de calidad, rendimiento y disponibilidad.

#### **Dominio de los patrones de arquitectura**

Estos patrones se encuentran en 4 dominios:[21]

- **Control de acceso:** personas que pueden acceder al sistema.
- **Concurrencia:** manejar múltiples tareas en paralelo.
- **Distribución:** sistemas distribuidos y la forma en que se comunican entre sí.
- **Persistencia:** almacenar datos en bases de datos para leer o modificarlos.

#### **Relación entre patrones de arquitectura y los requerimientos no funcionales**

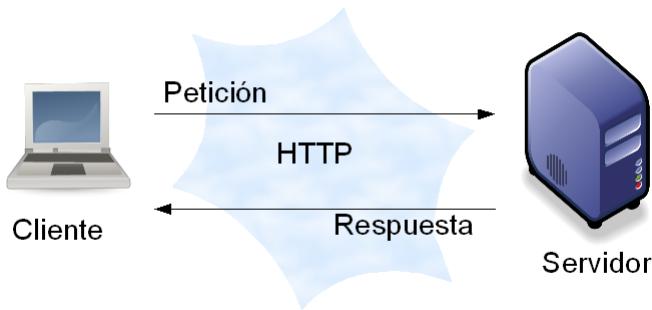
Los patrones tienen relación con los requisitos no funcionales más característicos como:

- **Escalabilidad y mantenimiento:** define características de rendimiento, concurrencia y capacidad del sistema.
- **Seguridad:** autentica y autoriza el acceso a la información o datos sensibles, evita o resuelve amenazas que ponen en riesgo la confidencialidad e integridad de la información.
- **Pruebas de aceptación y verificación:** permite conocer el estado de verificación y los resultados del sistema.
- **Documentación:** determina la documentación técnica y de comprensión del sistema.
- **Recursos:** define los límites en costos, físicos, tecnológicos y humanos.
- **Interoperabilidad:** Garantizar que el software pueda ser implementado en cualquier sistemas operativo.
- **Salvaguarda y replicación de datos:** Garantizar tener la información replicada en caso de un fallo, a demás de poder replicar el sistema en otros escenarios.

### Patrones de arquitectura

A continuación se presentarán los **patrones de arquitectura de software** más comunes, indicando sus principales características y su relación con los requisitos no funcionales en los que se hace énfasis.[19, 20, 21]

**Modelo cliente-servidor** Este modelo se caracteriza por tener dos partes: un **cliente** y un **servidor**. El modelo básicamente consiste en hacer peticiones desde el cliente hacia el servidor y recibir una respuesta de este. Las peticiones y respuestas son información en forma de texto HTML, imágenes u otros elementos. [20] Ver la figura 3-1.

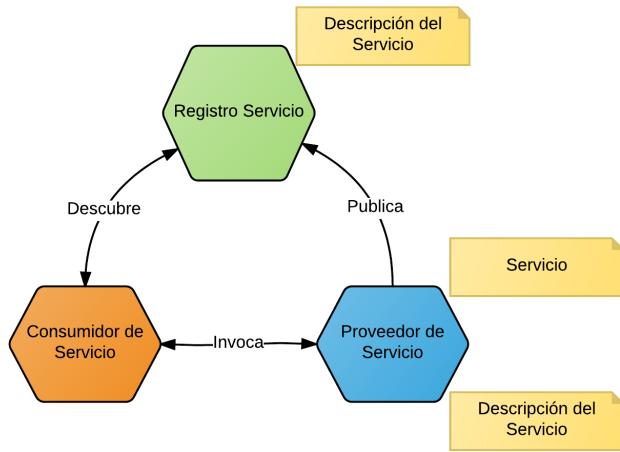


**Figura 3-1.:** Diagrama Modelo Cliente-Servidor Fuente: [1].

Estas peticiones viajan normalmente por TCP que es una de las capas del modelo OSI de transporte. El servidor puede soportar múltiples conexiones simultáneas de diferentes clientes.[3, 22] Por lo tanto, el requisito funcional más importante es la concurrencia y escalabilidad.

**Arquitectura orientada a servicios SOA** La arquitectura orientada a servicios (en inglés: Service Oriented Architecture) es un patrón que establece restricciones a tener en cuenta por el desarrollador del software: respuestas rápidas, adaptativas y posibles cambios en los requisitos. Este patrón **orquesta** el funcionamiento del software a partir de un servidor central (Registro Servicio) quien es el que reparte los servicios entre diferentes servidores y estos a su vez lo llevan hacia sus clientes respectivos.

En la figura 3-2 se pueden observar el paradigma descubrir, publicar e invocar de la arquitectura SOA. El consumidor de servicios localiza dinámicamente (descubre) un servicio consultando el registro del servicio el cual tiene la descripción de los servicios. Una vez localiza el servicio, si existe, el registro proporciona al consumidor la interfaz de contrato y la dirección del servicio proveedor. El consumidor invoca al proveedor de servicio para llevar a cabo los procesos requeridos.



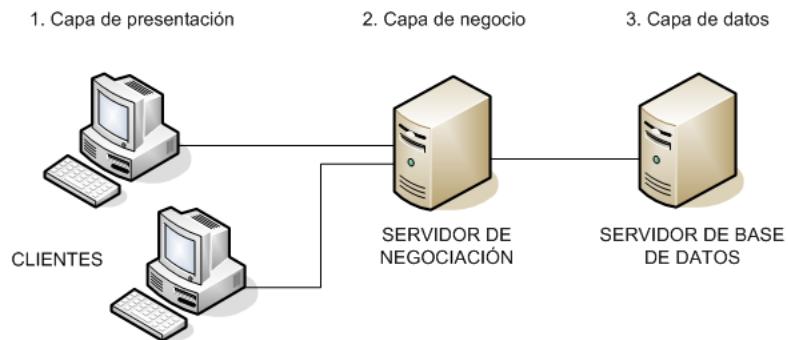
**Figura 3-2.:** Diagrama Arquitectura Orientada a Servicios.

Las conexiones se realizan mediante petición - respuesta, igual que el modelo cliente-servidor, siendo este la base de un SOA. [3] El requisito no funcional que más se ajusta es la escalabilidad.

**Modelo cliente-servidor multinivel** En este modelo se varía el modelo cliente-servidor mediante la distribución de capas para procesar las peticiones.[20] Ver la figura 3-3.

El modelo más usado es el de tres capas que son:

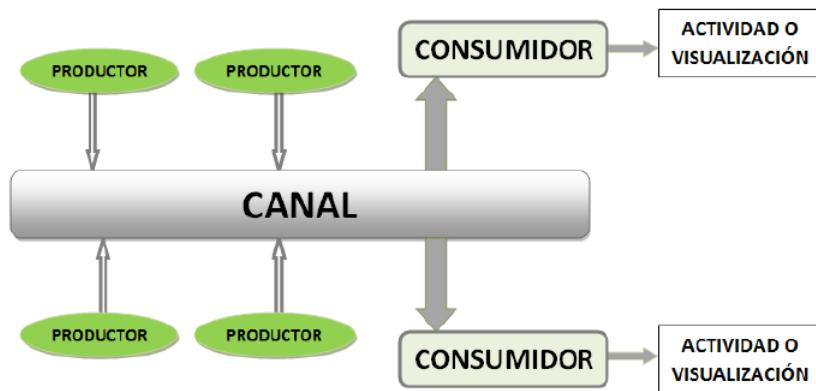
1. **Capa de presentación:** que básicamente es la interfaz gráfica del usuario (capturando movimiento, eventos y acciones).
2. **Capa de negocio:** almacena la lógica de la aplicación (procesos del negocio, librerías, core).
3. **Capa de datos:** donde se encuentra la información almacenada en bases de datos.



**Figura 3-3.:** Diagrama Modelo 3 Capas Arquitectura n-Capas Fuente: [2].

La orquestación en este modelo la realiza la capa de negocios que es la que interactúa con la capa de presentación y la capa de datos.[3, 23] El requisito no funcional más importante es la escalabilidad y mantenimiento.

**Arquitectura orientada a eventos EDA** La arquitectura orientada a objetos (en inglés: Event Driven Architecture EDA) monitoriza, visualiza, produce y **reacciona a eventos**.<sup>1</sup> Este patrón tiene 4 elementos: **productores, canal, consumidores y actividad**. Un productor genera un evento el cual se transmite por un canal hasta llegar al consumidor el cual lleva a cabo alguna actividad. Ver la figura 3-4.



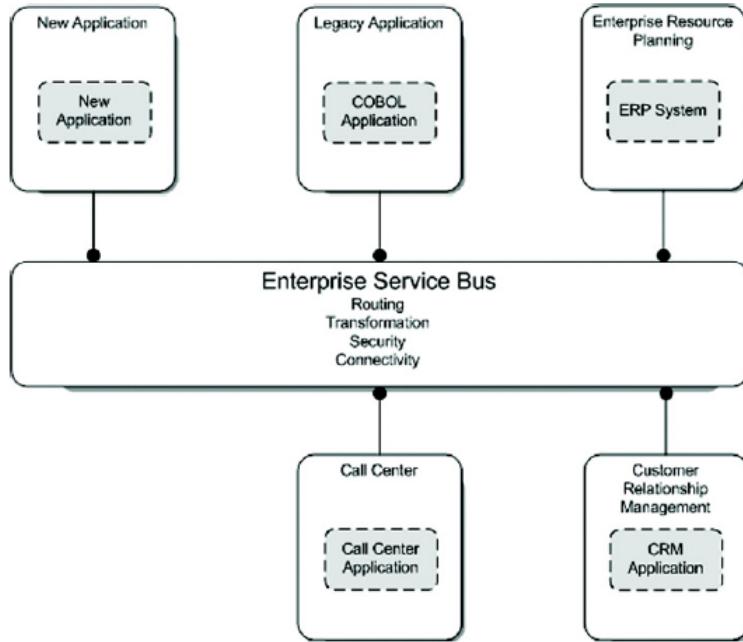
**Figura 3-4.**: Diagrama Arquitectura Orientada a Eventos EDA Fuente: [3].

El requisito no funcional que se espera es la escalabilidad y la concurrencia. Esta arquitectura está muy normalizada a eventos de tipo asíncrono y no predecibles.[3]

**Bus de Servicios Empresariales** Un bus de servicios empresarial (en inglés Enterpriser Service Bus - ESB) son **elementos de integración** (multiprotocolo y multipropósito) en SOA que combinan servicios web, mensajería, enrutamiento y enriquecimiento de datos, políticas de seguridad entre otro.[4]

Integra los enfoques dirigidos por eventos (EDA) y orientado a servicios (SOA). Su principal característica es permitir la interacción entre aplicaciones heterogéneas. En la figura 3-5 se observa una arquitectura que utiliza un ESB para integrar distintas aplicaciones con diferentes orígenes y tecnologías que comparten en común un ESB para intercambiar información entre ellas.

<sup>1</sup>Los **eventos** son **cambios** de estado en un parámetro, sistema o elemento. [3]



**Figura 3-5.:** Ejemplo arquitectura ESB para integrar aplicaciones. Fuente: [4].

### 3.1.2. Protocolos y Estándares de Servicios Web

La comunicación desde y hacia un servicio web implica un determinado lenguaje, con una estructura definida. Por esta razón encontramos protocolos y estándares que nos definen la forma en que los servicios web se comunican, la estructura de la información que se debe enviar y recibir, entre muchas otras cuestiones técnicas.

Los siguientes son los protocolos y estándares básicos que un servicio web puede utilizar:[24]

- **XML (eXtensible Markup Language):** es un formato o lenguaje de marcado para describir contenido de forma separada de su formato. Dentro de este estándar hay estándares como el DTD o XSD para la definición del lenguaje y el XSL-FO y XSLT para la transformación y presentación.
- **JSON (JavaScript Object Notation):** es un formato o lenguaje ligero de intercambio de datos más simple de leer y escribir por humanos.
- **SOAP (Simple Object Access Protocol):** este protocolo complejo define una gramática XML con el formato de los documentos que se deben intercambiar entre quien realiza la solicitud (cliente) y quien la responde (servicio) por lo general se utiliza en los servicios web.
- **REST (Representational State Transfer):** es un estándar de transferencia de representación de estado el cual no tiene estado (en inglés stateless), lo que quiere decir que, entre dos llamadas cualesquiera, el servicio pierde todos sus datos.
- **WSDL (web Services Description Language):** este estándar define la gramática XML pero de la interfaz del servicio web o sea los métodos y parámetros que se exponen tanto a la entrada como salida necesarios para invocar los servicios.

- **UDDI (Universal Description Discovery):** este estándar pertenece a OASIS<sup>2</sup>, permite configurar un servicio de registro o directorio donde los proveedores de servicios publican sus servicios, como un servicio de páginas amarillas de servicios web.

### 3.1.3. Business Model Canvas

El modelo de negocio Canvas (Business Model Canvas - BMC) es una **herramienta de gestión estratégica y empresarial**. Esta herramienta permite describir, diseñar, desafiar, inventar, y transformar el modelo de negocio de una organización.<sup>[5]</sup>

El modelo BMC propone un diagrama como herramienta para la gestión estratégica, el cual se usa para interpretar y conocer el negocio. Ver la figura 3-6. Para más detalle ver diagrama en el anexo A.

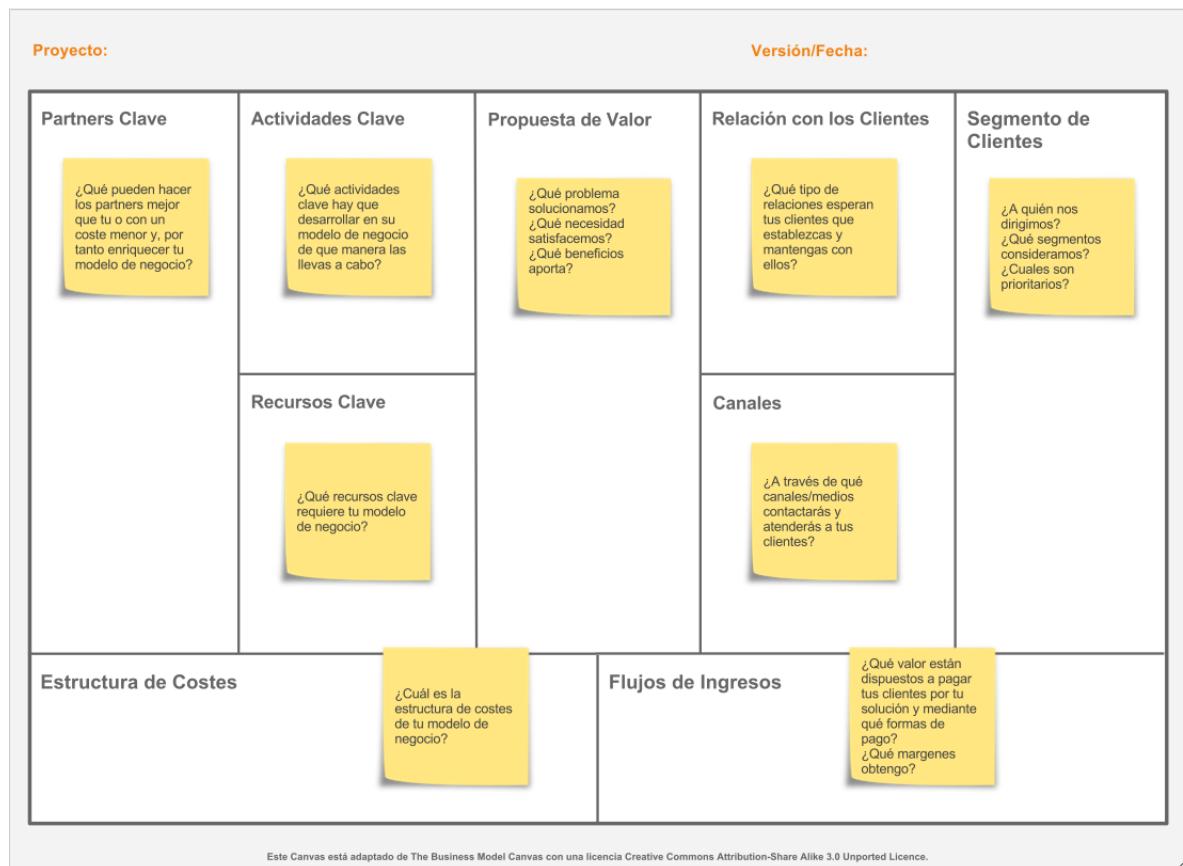


Figura 3-6.: Diagrama Modelo BMC. Fuente: [5].

#### Segmento de Clientes

Este segmento identifica los clientes del negocio, analiza el mercado de masas, el nicho específico o segmento de enfoque del negocio. Para determinar esos clientes se plantean dos preguntas:

<sup>2</sup><http://www.uddi.org>

- ¿Para quién se está creando valor?
- ¿Cuáles son los clientes más importantes?

### **Propuestas de Valor**

Este segmento entrega las propuestas que buscan dar un valor al cliente. Las características más importantes de una propuesta de valor son:

- Novedad
- Actuación
- Personalización
- Diseño
- Marca
- Precio
- Reducción de costo
- Reducción de riesgos
- Accesibilidad
- Conveniencia o Usabilidad

Para determinar las propuestas de valor se plantean las siguientes preguntas:

- ¿Qué valor se ofrece a los clientes?
- ¿Cuál es el problema que se le ayuda a resolver?
- ¿Qué paquetes de productos y servicios se le ofrece a cada segmento de clientes?
- ¿Qué necesidades del cliente se está satisfaciendo?

### **Relaciones con el Cliente**

Este segmento indica las relaciones con el cliente a fin de poder satisfacer sus necesidades. Por ejemplo: asistencia personalizada, personal técnico, autoservicio, servicios automatizados. Para determinar la relación con el cliente se plantean las siguientes preguntas:

- ¿Qué tipo de relación establecer y mantener con los clientes?
- ¿Cómo se integran las relaciones en el modelo de negocio?
- ¿Cuáles son los costos?

### Canales

Este segmento muestra los canales a través de los cuales el segmento de clientes puede acceder, informarse o comunicarse. Para descubrir los canales se plantean las siguientes preguntas:

- ¿Cómo se está llegando a ellos ahora?
- ¿Cómo se integran los canales?
- ¿Cuáles funcionan mejor?
- ¿Cuáles son los más rentables?
- ¿Cómo estamos integrándolos con las rutinas de los clientes?

A continuación se presentan las cuatro fases que se analizan de un canal:

- **Conocimiento:** ¿Cómo aumentar la conciencia sobre los productos y servicios de la empresa?
- **Evaluación:** ¿Cómo ayudar a los clientes a evaluar la propuesta de valor?
- **Compra:** ¿Cómo permitir que los clientes compren productos y servicios específicos?
- **Entrega:** ¿Cómo ofrecer una propuesta de valor a los clientes?

### Actividades Claves

Este segmento identifica las actividades clave que se requieren para la propuesta de valor. Se identifican los canales de distribución, las relaciones con el cliente, los ingresos corrientes. Estas actividades se pueden clasificar en categorías:

- Producción
- Resolución de problemas
- Funcionamiento e infraestructura

### Recursos Claves

Este segmento identifica los recursos claves que requiere la propuesta de valor como por ejemplo: los canales de distribución, las relaciones del cliente o ingresos necesarios. Estos recursos se pueden clasificar en tipos:

- Físico
- Intelectual (patentes de marcas, derechos de autor, datos)
- Humano
- Financiero

### Asociaciones Clave

Este segmento enumera las asociaciones claves para el negocio como lo son: los socios, proveedores, recursos, actividades. Estas asociaciones claves buscan como motivación la optimización y economía, reducción de riesgos e incertidumbre, la adquisición de recursos y actividades particulares.

**Flujos de Ingresos**

Este segmento representa el flujo de ingresos, respondiendo a preguntas como:

- ¿Qué valor están dispuestos a pagar los clientes?
- ¿Por qué pagarían los clientes?
- ¿Cómo están pagando actualmente?
- ¿Cómo prefieren pagar?

Clasificación de los flujos de ingresos:

**Tipos:**

- Venta de activos
- Tarifa de uso
- Tasas de suscripción
- Préstamos, Renting o leasing
- Concesión de licencias
- Honorarios de corretaje
- Publicidad

**Precios fijos:**

- Precio de lista
- Característica de producto
- Segmento de clientes
- Volumen

**Fijación de precios dinámicos (Negociación):**

- Negociación
- Mercado en tiempo real

**Estructura de Costos**

Este segmento muestra los costos más importantes inherentes a nuestro modelo de negocio, como por ejemplo el costo de los recursos y actividades clave. Se deben tener en cuenta características de la estructura de costos como lo son: costos fijos (salarios, alquileres, servicios públicos, etc), costos variables, economías de escala y de alcance.

### 3.1.4. Arquitectura Empresarial

El concepto Arquitectura Empresarial se origina en el Marco de Trabajo definido por Jhon Zachman en 1984 y publicado por IBM en 1987 [25]. El enfoque es garantizar que todos los elementos o recursos de una empresa estén bien organizados y relacionados como un sistema integral.

La arquitectura empresarial es la organización de los componentes, interrelaciones internas, externas y gobernabilidad de un sistema. Esto quiere decir que se tiene en cuenta el modelo de negocio y de operaciones, la estructura organizacional, procesos de negocio, datos, aplicaciones y tecnología. [26]

Se puede decir que una arquitectura es el diseño y descripción de una organización o entidad como un sistema en términos de sus componentes y relaciones y que la arquitectura empresarial es construir modelos (modelamiento) de la realidad. Estos modelos se reflejan en los entregables.

La arquitectura empresarial permite adquirir un discurso para adentrarse al alma de la organización y convencerlos que TI no es una muleta o soporte. A demás permite adquirir una forma de comunicación de alto nivel para que las personas comprendan los términos según su nivel. La palabra alto nivel es muy recurrente en arquitectura y permite hacer abstracciones muy sintetizadas.

Algunas definiciones para contextualizar: [26, 7, 27]

- **Empresa:** Una empresa está definida como un conjunto de organizaciones que tienen metas y resultados comunes.
- **Arquitectura:** La arquitectura se puede entender como una estructura de componentes, sus interrelaciones, principios y guías que gobiernan su diseño y evolución a través del tiempo. Entonces, la arquitectura empresarial es la descripción de la operación de una empresa, sus sistemas y tecnologías de información.
- **Formas de ver una Arquitectura:** Existen 3 formas de ver una arquitectura: disciplina, producto y proceso.
- **Disciplina:** Gobierna los cambios de la arquitectura.
- **Producto:** Diseño que muestra la coherencia entre los productos, procesos, organización, suministro de información y la infraestructura, basado en una visión y ciertos puntos de partida, principios y preferencias.
- **Proceso:** Indica la gente y los recursos y define la forma de trabajo.
- **Modelo:** Es una abstracción del mundo real.
- **Metamodelo:** Es una abstracción que sobresalta las características de un modelo en sí mismo.

#### Arquitectura de Conceptos Básicos

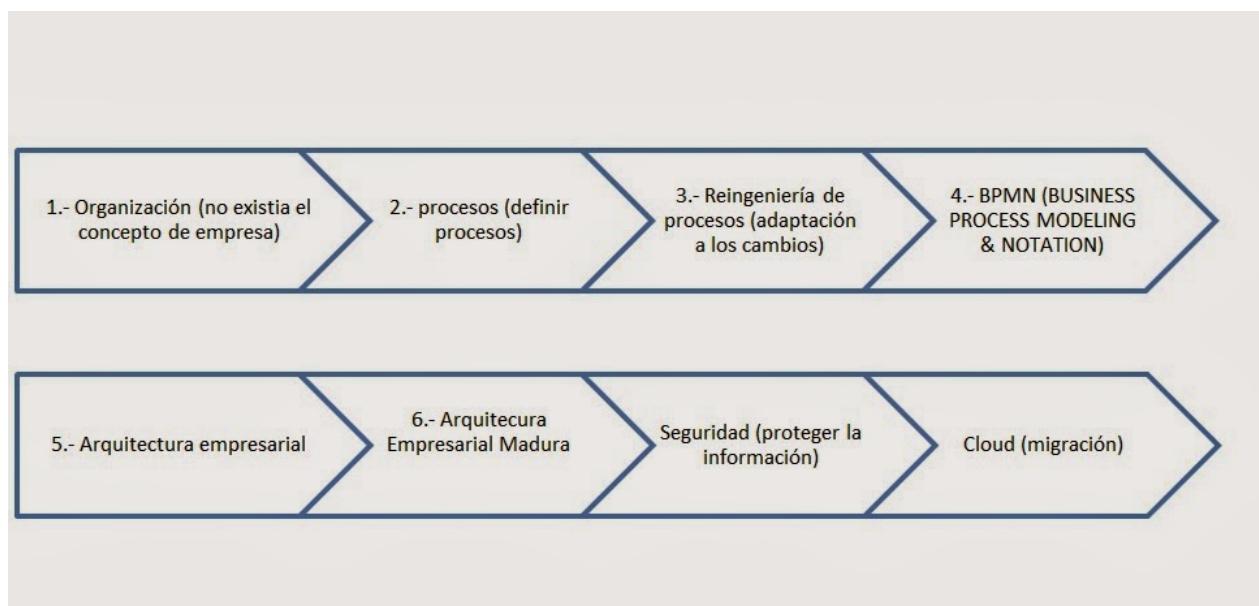
Cuando se conversa sobre arquitectura empresarial es válido plantear las siguientes preguntas: [27, 28, 29, 30, 26]

- ¿Cuál es el objetivo de la Arquitectura Empresarial?
- ¿Cuál es la definición de Arquitectura Empresarial?
- ¿Cómo fue la evolución de la Arquitectura Empresarial?

- ¿Cuáles son las maneras de visualizar una Arquitectura?

A continuación las respuestas a las preguntas: [6]

- **Objetivo de la Arquitectura Empresarial:** Integrar los componentes que conforman a nuestra empresa, creando un ambiente capaz de responder a las necesidades del cambio, utilizando TIC (Tecnologías de la Información Computacional) como un factor clave del éxito.
- **Definición de Arquitectura Establecida por el Massachusetts Institute of Technology (MIT):** Es la organización lógica de procesos de negocio e infraestructura de tecnologías de la información que refleja la integración y estandarización de los requerimientos de nuestro modelo operativo.
- **Línea del tiempo de la arquitectura empresarial:** La línea de tiempo de una arquitectura muestra el proceso que se lleva a cabo desde el concepto de empresa hasta la migración. Ver la figura 3-7.



**Figura 3-7.: Línea de Tiempo de una Arquitectura Empresarial.** Fuente: [6]

### Formas de ver una Arquitectura

Existen 3 formas de ver una arquitectura: disciplina, producto y proceso.

- **Disciplina:** Gobierna los cambios de la arquitectura.
- **Producto:** Diseño que muestra la coherencia entre los productos, procesos, organización, suministro de información y la infraestructura, basado en una visión y ciertos puntos de partida, principios y preferencias.
- **Proceso:** Indica la gente y los recursos y define la forma de trabajo.

### Arquitectura Empresarial en el contexto de TOGAF

El grupo abierto (The Open Group)[31] es un consorcio a nivel mundial que mediante IT posibilita el cumplir y alcanzar los objetivos de los negocios. Este grupo ofrece un conjunto de servicios para mejorar la eficiencia,

gestionar los requerimientos actuales y emergentes a demás de establecer políticas y compartirlas mediante las mejores prácticas.

TOGAF (en inglés The Open Group Architeceture Framework)[32] es un consorcio formado por profesionales en tecnologías de información con el objetivo de entregar estándares en el mundo de la arquitectura de tecnologías de información.

ISO/IEC 42010:2007[33] define “arquitectura” como: ”La organización fundamental de un sistema, compuesta por sus componentes, las relaciones entre ellos y su entorno, así como los principios que gobiernan su diseño y evolución.”

TOGAF adopta y amplía esta definición. En TOGAF, “arquitectura” tiene dos significados según el contexto:

1. Una descripción formal de un sistema, o un plano detallado del sistema al nivel de sus componentes para orientar su implementación.
2. La estructura de componentes, sus interrelaciones, y los principios y guías que gobiernan su diseño y evolución a través del tiempo.

### 3.1.5. Archimate

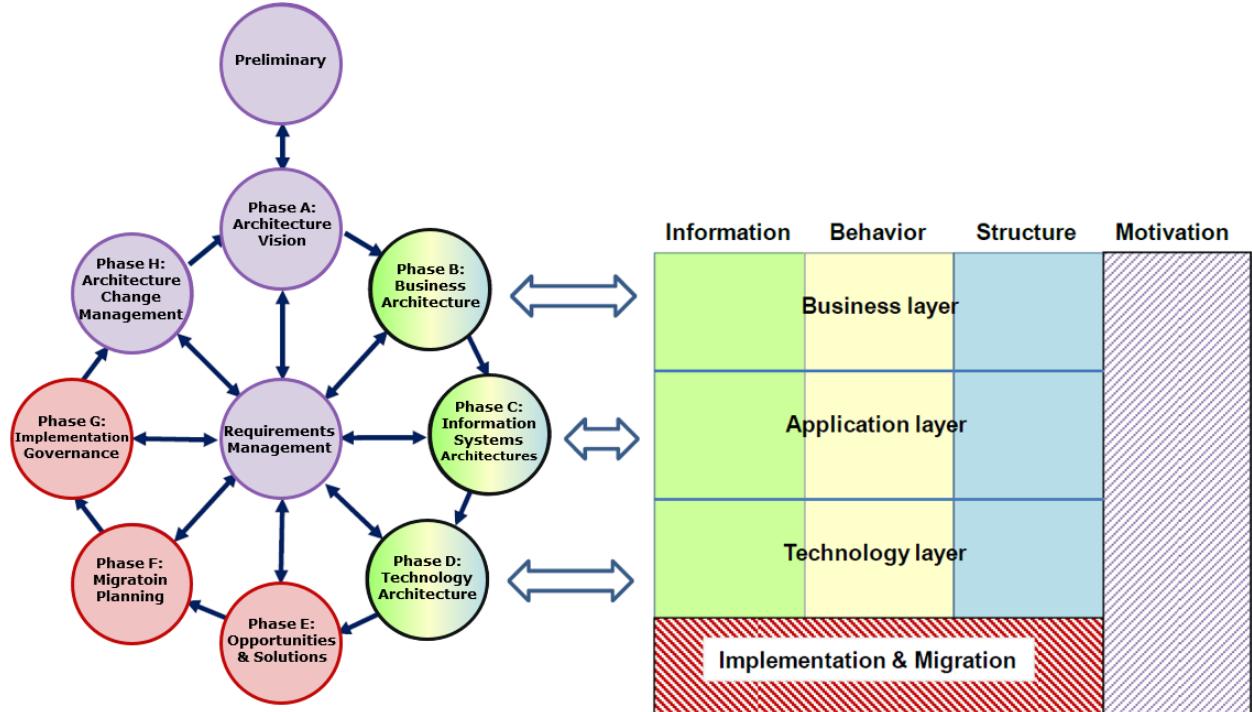
Archimate es un lenguaje arquitectónico que proporciona los elementos de información necesarios para mostrar la funcionalidad dentro de un proyecto de arquitectura empresarial. Archimate tiene la ventaja como lenguaje que lo puede usar un experto del dominio y un experto técnico.

Archimate define que la arquitectura empresarial es la identificación de unos problemas y el desarrollo que hace un grupo de personas para resolverlos utilizando tecnologías de información.

Archimate representa los conceptos de las capas mediante grafos, algunos tomados del lenguaje unificado de modelado del inglés **Unified Modeling Language (UML)** y los presenta en distintos colores. Los colores son para generar conceptos similares en las capas y su trazabilidad.

**Correspondencia entre Archimate y TOGAF** Archimate está relacionado con TOGAF, pero no es TOGAF aunque se mezclan muy bien ambos. En la figura 3-8 es fácil de identificar la correspondencia entre el ciclo ADM (izquierda) y TOGAF (derecha), debido al uso de colores:

- La fase A, H junto con Requerimientos y Preliminar agrupadas de color lila, representan lo motivacional y nos responde el **porqué** vamos a hacer la arquitectura.
- Las fases B, C y D agrupadas cada una de tres colores que representan la información (Verde), el comportamiento (Amarillo) y la estructura (Azul), nos responde el **qué** vamos a hacer.
- Las fases E, F y G agrupadas de color rojo representan la migración y nos responde el **cómo** vamos a hacer la arquitectura.



**Figura 3-8.: Correspondencia entre TOGAF y el Ciclo ADM de Archimate incluyendo las extensiones.**  
Fuente: Capítulo 2 [7]

### El Ciclo ADM

El Método de Desarrollo de la Arquitectura en inglés **Architecture Development Method (ADM)**, permite iniciar la arquitectura desde una línea base o estado actual (AS-IS) hacia una arquitectura objetivo o futura (TO-BE).

**Fases del Ciclo ADM** El método ADM consta de ocho fases identificadas por las letras A, B, C, D, E, F, G y H, y dos secciones: Preliminar y Administración de Requerimientos. Ver figura 3-8

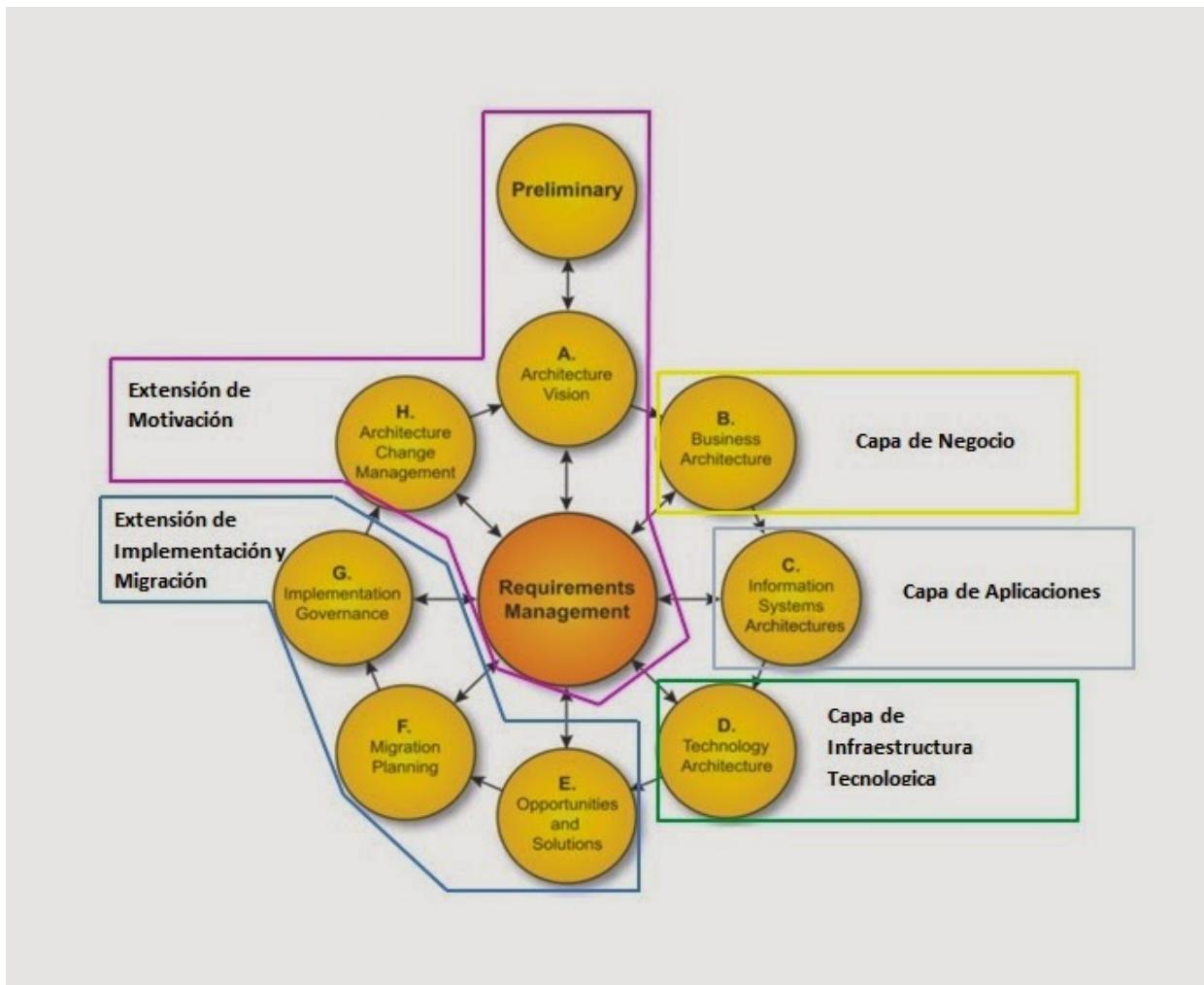
- La fase A define la visión de la arquitectura, alcance y punto al que se quiere llegar. En esta fase se presentan el diagrama del estado actual (AS-IS) de la arquitectura del negocio, aplicación y/o infraestructura y el estado al que quiero llegar o estado futuro (TO-BE).
- Las fases B, C, D son los niveles de abstracción del negocio, aplicación o infraestructura en más detalle. Cada fase puede estar representada por puntos de vista, como se verá mas adelante.
- La fase E lista las oportunidades y soluciones<sup>3</sup> para lograr el objetivo o estado futuro (TO-BE) propuesto en la arquitectura empresarial.
- En la fase F se identifican y seleccionan los activos para llevar a cabo la arquitectura de un estado AS-IS a un estado TO-BE. Como en las anteriores se identifica que se pacta, que se modifica y que se elimina, los activos pueden ser los diagramas o entregables candidatos, clasificados en paquetes de trabajo para ver como impactan la organización u otras arquitecturas.
- Las fases G y H son la gobernabilidad y gestión de control de cambios de la arquitectura empresarial antes, durante y después de cada iteración entre el estado AS-IS a TO-BE.

**Archimate y su Integración con el ADM** ArchiMate cuenta con 43 elementos gráficos, los cuales se distribuyen en 3 capas (Negocio, Aplicaciones e Infraestructura Tecnológica) y 2 extensiones (Extensión de Motivación y Extensión de Implementación y Migración). Además cuenta con 12 relaciones, todos los elementos están en un 99 % alineado al Framework TOGAF.

A continuación se verán como se agrupan las fases del ciclo ADM en capas y extensiones. Ver figura 3-9

---

<sup>3</sup>Oportunidades y soluciones puede interpretarse como procesos, tareas, soluciones de software.



**Figura 3-9.: Integración del ADM con ArchiMate.** Fuente: [8]

### 3.1.6. Archimate: Capa de Negocio

Es una de las capas más importantes debido a que el lenguaje que se utiliza, permite hablar en términos de las entidades del negocio, por lo que es importante distribuir adecuadamente la semántica.

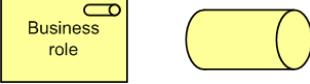
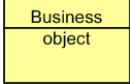
Esta capa gira en torno a tres dimensiones de comportamiento: procesos, servicios y producto (centro del negocio). La indagación que uno hace al modelar esta capa es convertirla en software.

Las capas agregan conceptos que soportan las etapas del ADM de TOGAF en las fases B, C y D que se encuentran relacionadas con el negocio, aplicación o datos e infraestructura.

#### Conceptos Pasivos:

Es la dimensión estructural con entidades del negocio que se etiquetan con sustantivos, cuyo concepto más importante es el actor. Ver tabla: 3-1

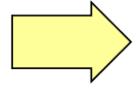
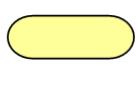
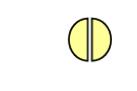
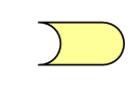
**Tabla 3-1.:** Conceptos Pasivos de la Capa de Negocio en Archimate. [7, Chapter 3]

Concepto	Descripción	Representación
<b>Actor</b>	Representa una entidad organizacional que es capaz de mejorar su comportamiento. Una persona o una unidad administrativa. Ejemplo: Un actor puede ser el desarrollador.	
<b>Rol</b>	Tiene que ver con un comportamiento específico, asignación funcional específica. Aunque se puede asociar directamente a un actor pero por ahora no es necesario. Ejemplo: Un rol puede ser el de líder de pruebas.	
<b>Colaboración</b>	Es una sociedad de roles, un agregado de uno o más roles de negocio. Ejemplo: los comités.	
<b>Interfaz</b>	Punto de acceso relacionado con la forma en que se conecta un actor con un negocio. Ejemplo: En el contexto de una universidad la forma en que se comunica un estudiante con el administrativo de la facultad es la interfaz y pueden ser mediante teléfono, correo electrónico e incluso un mismo compañero de estudio que sea representante de los estudiantes.	
<b>Localización</b>	Tiene que ver más con ubicación espacial. Ejemplo: La ubicación de la empresa, una arquitectura back office y front office.	
<b>Objeto</b>	Representa el objeto del negocio como concepto pasivo. Ejemplo: en el contexto universitario para la carrera de especialización de ingeniería de software el objeto de negocio es ser “Especialista”.	

### Conceptos Activos:

Es la dimensión que representan el comportamiento o acciones de las estructuras del negocio y se etiquetan con verbos. El concepto más importante es el paradigma de procesos. Ver tabla: **3-2**

**Tabla 3-2.: Conceptos Activos de la Capa de Negocio en Archimate.[7, Chapter 3]**

Concepto	Descripción	Representación
<b>Proceso</b>	El proceso es un elemento de comportamiento interno o privado por lo que se debe proteger ya que ahí se refleja el Know How de la organización. Los procesos debemos etiquetarlos como verbos sustantivados.	 
<b>Servicio</b>	Es un comportamiento interno o externo y es lo que al cliente le interesa.	 
<b>Función</b>	Es lo que hace un actor o rol. Es un comportamiento ejecutado por un actor que de alguna manera le apuntan a generar, manejar recursos o administrar los objetivos organizacionales.	 
<b>Interacción</b>	La interacción describe básicamente el comportamiento de las colaboraciones de un negocio. Se puede ver como una función de la colaboración solo que se define con más precisión y se establece de esta manera. Las responsabilidades no ingresan al manual de funciones. Ejemplo: Un comité administrativo.	 
<b>Evento</b>	Podemos asociarlo con procesos de salida y entrada o un flujo de trabajo (en inglés: work flow). Los eventos son vistos también como funciones de entrada y salida debido a que se supone que con esto se puede iniciar y finalizar un proceso.	 

#### Conceptos Informacionales:

Es la dimensión que representan la información que gira alrededor del producto cuyo concepto más importante es el producto. Ver tabla: **3-3**

**Tabla 3-3.: Conceptos Informacionales de la Capa de Negocio en Archimate.[7, Chapter 3]**

Concepto	Descripción	Representación
<b>Producto</b>	Un producto es una colección de servicios. En términos generales, los servicios se pueden informatizar. El producto tiene que ir acompañado de contratos y acuerdos.	
<b>Contrato</b>	Un contrato es una especificación formal de acuerdos.	
<b>Servicio</b>	Los servicio satisfacen las necesidades de los clientes. Los servicios en conjunto forman el producto.	
<b>Valor</b>	Es una medida cualitativa o cuantitativa del producto. Ejemplo: En un contexto de especialización académica, si el producto es el trabajo de grado entonces el valor será como soluciona y se aplica la solución a una problemática social. Otros valores serían: facilidad de uso, expectativas generadas a futuro, nuevo conocimiento generado.	
<b>Significado</b>	Representación de conocimiento o experiencia con respecto al negocio. Ejemplo: habilidad innata para hacer pruebas.	
<b>Representación</b>	Forma perceptible de materializar el significado. Ejemplo: En un contexto universitario la representación a un título adquirido es el diploma.	

### 3.1.7. Archimate: Capa de Aplicación

Es una de las capas más interesantes debido a que el lenguaje que se utiliza nos permite hablar de componentes de software. Recordemos que la arquitectura de software hereda y basa su modelo de las arquitecturas, utilizando el concepto de componente. Basta con saber que se le debe pasar al componente para tener una estructura que garantice el ciclo de vida.

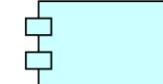
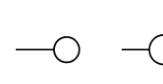
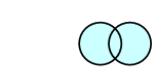
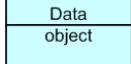
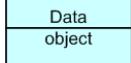
Esta capa maneja un lenguaje de descripción de arquitectura en inglés **Architecture Description Language - ADL**, utiliza los siguientes elementos: componentes, interfaces, conectores y restricciones.

Esta capa tiene dos dimensiones: la dimensión estructural y la dimensión de comportamiento. Sus conceptos se representan de color azul celeste.

#### Dimensión Estructural

A continuación los elementos estructurales de la capa de aplicación:

**Tabla 3-4.: Elementos Estructurales de la Capa de Aplicación en Archimate.[7, Chapter 4]**

Concepto	Descripción	Representación
<b>Componente</b>	Unidad modular de caja negra que realiza un conjunto de interfaces y además tiene como propiedad la re-ubicabilidad binaria que significa que el componente es portable entre plataformas.	 
<b>Interfaz</b>	Módulos abstractos o pegamento de los componentes. El estándar Corba creado por el OMG en inglés Object Management Group desarrollaron el lenguaje IDL, una propuesta que trató de generar un espacio de conversación genérica, una especificación para hacer que los conectores sean flexibles y un protocolo con el que finalmente se hacen metadatos como las interfaces.	 
<b>Colaboración</b>	Brindan el soporte para el ADL. Es un agregado de dos o más componentes. Así como en la colaboración de negocio agregabamos roles, acá agregamos componentes. Ejemplo: modelamiento visual y el componente de modelamiento de datos pueden llegar a tener una colaboración en una arquitectura MVC. Asociar varios componentes es una colaboración: un xml es una colaboración porque luego cada componente lo utiliza y visualiza en con una estructura ya sea en árboles o en grafos.	 
<b>Colaboración</b>	Abstracción de muy alto nivel la cual no es un modelo de datos. Se supone que debemos generar abstracciones que nos hagan síntesis de muchas cosas. Objeto pasivo que almacena gran cantidad de información.	
<b>Colaboración</b>	Abstracción de muy alto nivel la cual no es un modelo de datos. Se supone que debemos generar abstracciones que nos hagan síntesis de muchas cosas. Objeto pasivo que almacena gran cantidad de información.	

### Dimensión Comportamental

A continuación los elementos comportamentales de la capa de aplicación:

**Tabla 3-5.: Elementos Comportamentales de la Capa de Aplicación en Archimate.[7, Chapter 4]**

Concepto	Descripción	Representación
<b>Función</b>	Si en la función de negocio el responsable es el actor, la función de aplicación son las responsabilidades del componente.	 
<b>Interacción</b>	Son los elementos que describen el comportamiento.	 
<b>Servicio</b>	Es lo realizado por un componente, es lo que vamos a exponer, comportamiento y lo que resuelve el sistema.	 

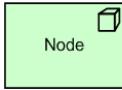
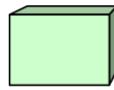
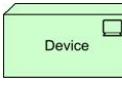
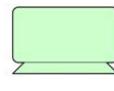
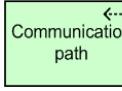
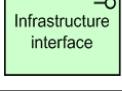
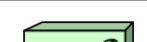
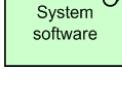
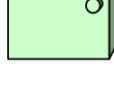
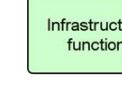
### 3.1.8. Archimate: Capa de Infraestructura

Esta capa representa sus conceptos en color verde. Acá el concepto de infraestructura de servicio es clave.

#### Elementos:

A continuación los elementos de la capa de infraestructura:

**Tabla 3-6.: Elementos de la Capa de Infraestructura en Archimate.[7, Chapter 5]**

Concepto	Descripción	Representación
<b>Nodo</b>	Es un recurso computacional sobre el cual los artefactos se pueden almacenar o desplegar para su ejecución, más general, dispositivo más específico. Por ejemplo servidores de aplicaciones, servidores de bases de datos, la nube.	 
<b>Dispositivo</b>	Cualquier equipo o recurso de hardware que puede consumir o almacenar a través de un nodo información. Por ejemplo dispositivos móviles inteligentes, computadoras. Ejemplo: si se quisiera modelar un cluster, debería hacerlo como una agrupación de nodos, donde cada nodo podría ser un dispositivo móvil.	 
<b>Red</b>	Es el medio de comunicación entre dos dispositivos. Algo más general es que las redes sirven para modelar topologías. La red sirve para modelar un problema de red. Ejemplo: internet, wifi, red lan.	 
<b>Camino de comunicación</b>	Es el enlace entre dos o más nodos mediante el cual pueden intercambiar información. El path sirve para modelar una ruta. Ejemplo: protocolo http, https, ftp.	 
<b>Interface</b>	Punto de acceso donde un nodo ofrece servicio y puede acceder o ser accedido por otros nodos o componentes de aplicaciones. Ejemplo: servicio web.	  
<b>Sistemas</b>	Un sistema de software específico de componentes que están soportados por la infraestructura. Ejemplo: Sistemas operativos, manejadores de bases de datos, lenguajes.	 
<b>Artefacto</b>	Elemento de información que es producida por la parte física por el proceso de desarrollo. Ejemplo: librería.	 
<b>Función</b>	Tiene unas funciones, las responsabilidades de los nodos.	 

### 3.1.9. Archimate: Capa Motivacional

En esta capa encontramos conceptos que amplían y ayudan a la noción que tenemos de la organización. Aclarar y definir estos conceptos nos ayudan a entender la organización de forma más sencilla.

Tiene un enfoque de objetivos. La parte más importante son las personas directamente interesadas e involucradas con el resultado de la arquitectura. El color es fucsia porque hace alusión a la parte motivacional.

### Elementos:

A continuación los elementos de la capa de motivacional que generan conceptos hacia la organización buscando que el sistema se comience a entender cómo la organización misma.

**Tabla 3-7.: Elementos de la Capa de Motivacional en Archimate.[7, Chapter 10]**

Concepto	Descripción	Representación
<b>Interesados</b>	Son las personas directamente interesadas, las personas que incluyen y que están involucradas con el resultado de la arquitectura.	
<b>Manejador</b>	Es la visión del interesado. Es algo que crea y motiva el cambio organizacional. Parecido a las funciones de negocio pero de más alto nivel que se convierte en una misión del interesado.	
<b>Valoración</b>	En la capa de negocio teníamos el valor del negocio. Se puede asociar al discurso de planeación estratégica (DOFA) donde cada uno de estos es una valoración.	
<b>Objetivo</b>	Es el pivote clave de todos los puntos de vista. El objetivo que el interesado tiene para alcanzar un logro. Es la meta que organizacionalmente recae sobre la función del actor del negocio.	
<b>Requerimiento</b>	Es un mandato o necesidad que debe ser realizado por un sistema. Es lo que se espera del sistema al final de la arquitectura.	
<b>Restricción</b>	Es el cumplimiento obligatorio de un requerimiento por el sistema. Son los adicionales que el sistema debe tener en cuenta con respecto a los requerimientos.	
<b>Principio</b>	Es la propiedad normativa de todo el sistema en el contexto en el que se realiza la arquitectura. El principio desde otra dimensión, es el PILAR. Ejemplo: Un pilar por ejemplo es la seguridad, confiabilidad, amigabilidad.	
<b>Principio</b>	Es la propiedad normativa de todo el sistema en el contexto en el que se realiza la arquitectura. El principio desde otra dimensión, es el PILAR. Ejemplo: Un pilar por ejemplo es la seguridad, confiabilidad, amigabilidad.	

### 3.1.10. Archimate: Puntos de Vista

En la realidad una organización puede llegar a tener muchos objetivos, problemas a resolver e interesados que se pueden representar mediante los puntos de vista.

Los puntos de vista son diagramas que representa una parte de la realidad que estoy modelando. Por lo tanto en Archimate tenemos las vistas que permiten modelar la realidad, mediante lenguajes simbólicos, que contienen una semántica y manejan unos conceptos.

A continuación se detallan cada uno de los puntos de vista que Archimate [7] permite modelar como herramienta para definir un concepto en diferentes contextos.

### Punto de Vista Preliminar

Es utilizado al inicio de un diseño cuando no todo tiene que ser detallado. Este punto de vista permite explicar a los no arquitectos debido a su notación sencilla, simple e intuitiva. Por ejemplo: Una nube representaría una red y las relaciones se representan con líneas simples, a excepción de las relaciones de "disparo y realización".

### Punto de Vista Organización

Representa modelos desde la perspectiva del interior de la organización. Por ejemplo: organigrama. Ver figura 3-10.

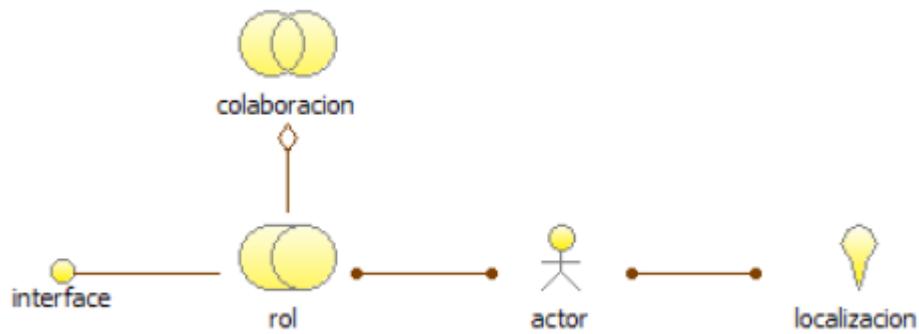


Figura 3-10.: Metamodelo Punto de Vista Organización.

### Punto de Vista Cooperación del Actor

Representa la relación de los actores entre sí y con su entorno. Ejemplo: un diagrama de contexto, conformado por: la organización, clientes, proveedores, productos o servicios. Por ejemplo: organigrama. Ver figura 3-11.

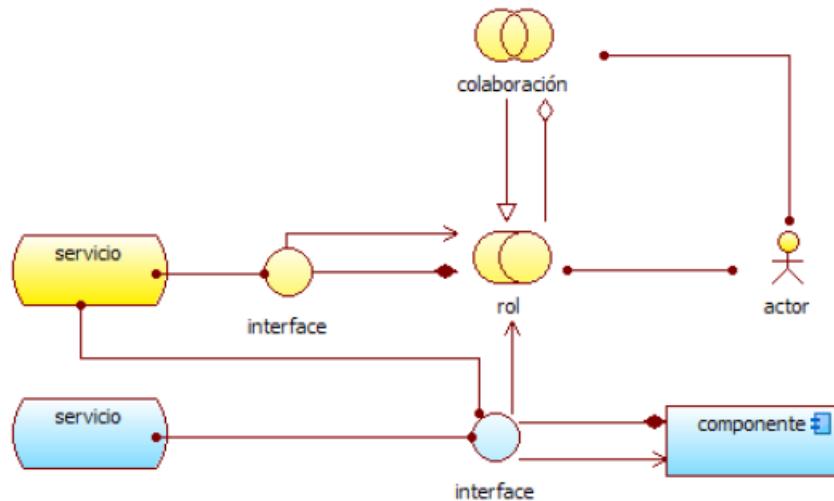


Figura 3-11.: Metamodelo Punto de Vista Cooperación del Actor.

### Punto de Vista Función del Negocio

Muestra las principales funciones primarias u operaciones generales del negocio de una organización y sus relaciones como los flujos de información, valor y otras. Ver figura 3-12.

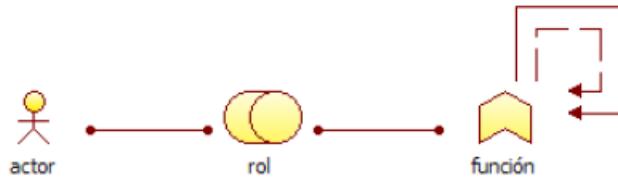


Figura 3-12.: Metamodelo Punto de Vista Función del Negocio.

### Punto de Vista Procesos de Negocio

En este punto de vista se representa la estructura de alto nivel y la composición de uno o más procesos de negocio. Por ejemplo: los servicios que ofrece un proceso en el exterior, como un proceso contribuye en la realización de productos, procesos asignados a roles, información utilizada por los procesos. Ver figura 3-13.

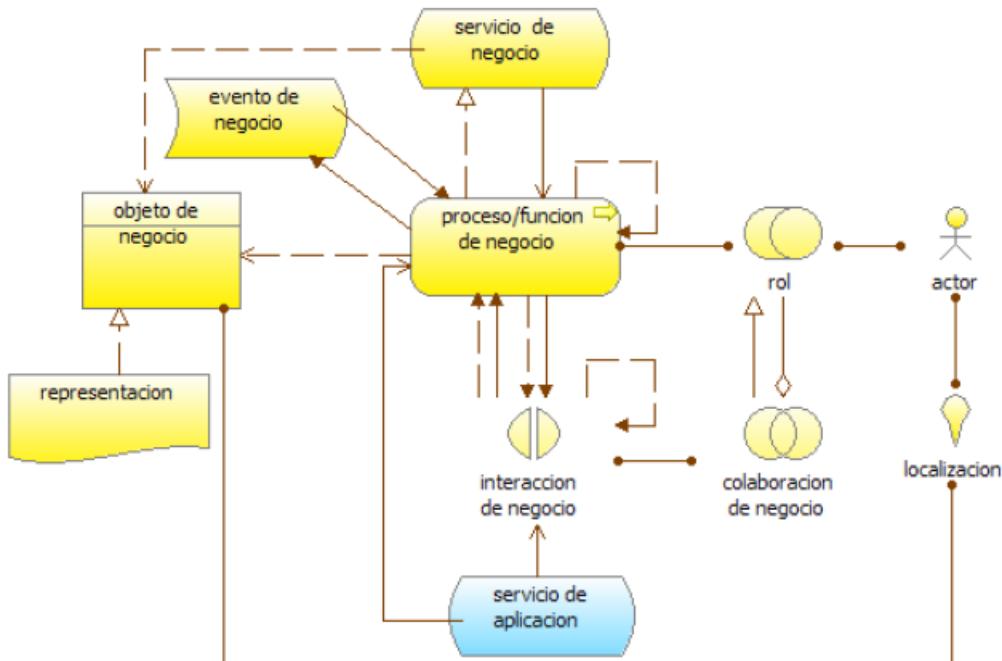


Figura 3-13.: Metamodelo Punto de Vista Procesos de Negocio.

### Punto de Vista Cooperación Procesos de Negocio

Se utiliza para representar las relaciones entre procesos y su entorno. Ejemplo: relaciones entre procesos de negocio, procesos en las funciones de negocio, procesos en la realización de servicios. Ver figura 3-14.

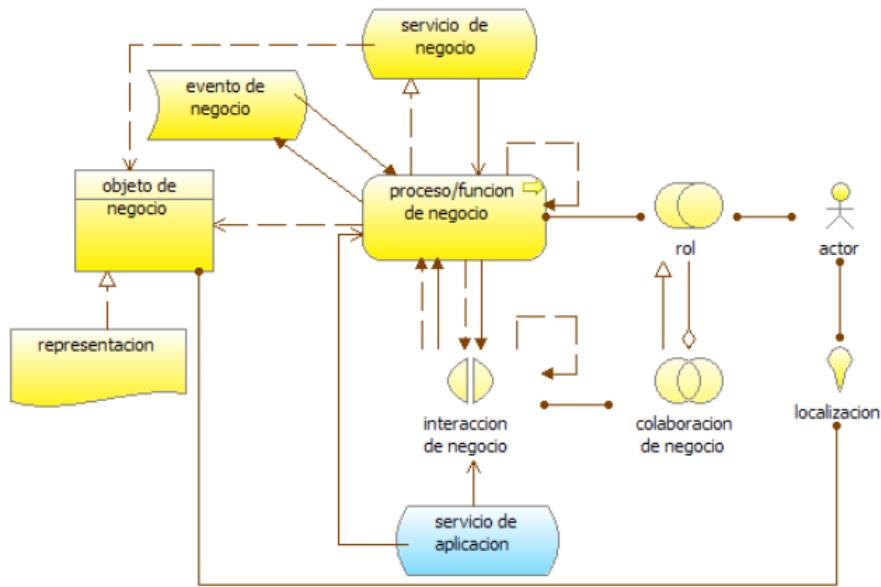


Figura 3-14.: Metamodelo Punto de Vista Cooperación Procesos de Negocio.

### Punto de Vista Producto

Este punto de vista indica el valor que los productos ofrecen a los clientes y otras partes involucradas. Representan la composición de los productos desde el punto de vista de servicios y contratos. Puede mostrar interfaces y eventos asociados. Ver figura 3-15.

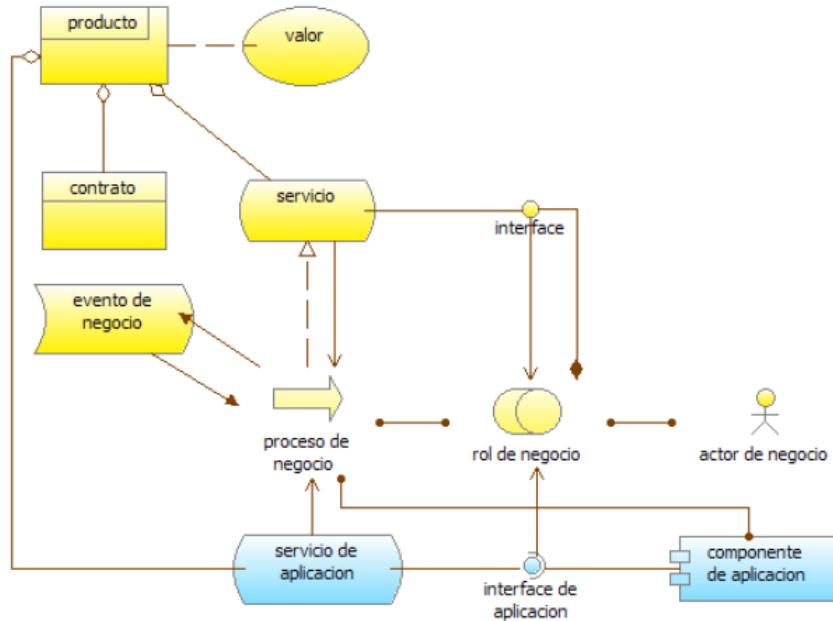


Figura 3-15.: Metamodelo Punto de Vista Producto.

### Punto de Vista de Comportamiento de Aplicación

Ver figura 3-16.

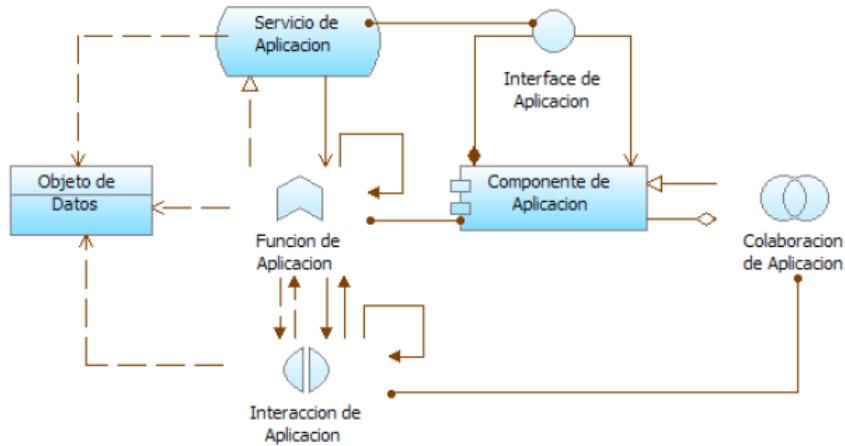


Figura 3-16.: Metamodelo Punto de Vista de Comportamiento de Aplicación.

### Punto de Vista de Cooperación de Aplicación

Aparece el concepto de localización. Se categorizan los componentes en el front office. Estos componentes deben ser diseñados por profesionales que tengan conocimientos en usabilidad, caso contrario del back office que es diseñado por arquitectos de datos. Ver figura 3-17.

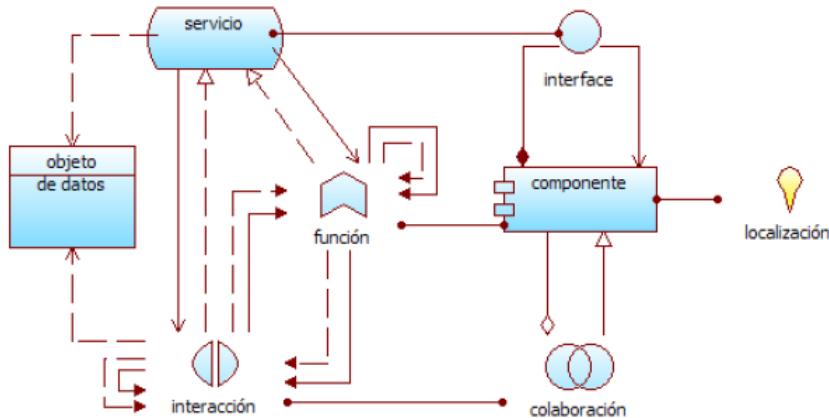


Figura 3-17.: Metamodelo Punto de Vista de Cooperación de Aplicación.

### Punto de Vista de Estructura de Aplicación

Se centra fuertemente en el componentes y unión de las interfaces. Ver figura 3-18.

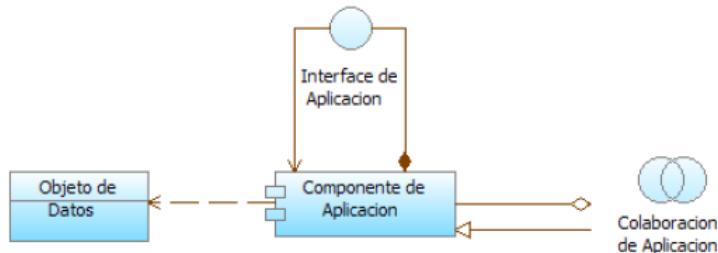


Figura 3-18.: Metamodelo Punto de Vista de Estructura de Aplicación.

#### Punto de Vista de Uso de Aplicación

Ver figura 3-19.

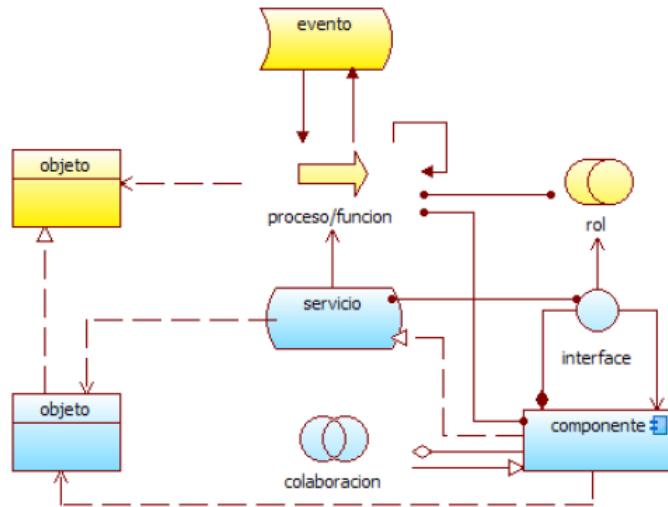


Figura 3-19.: Metamodelo Punto de Vista de Uso de Aplicación.

#### Punto de Vista de Infraestructura

Ver figura 3-20.

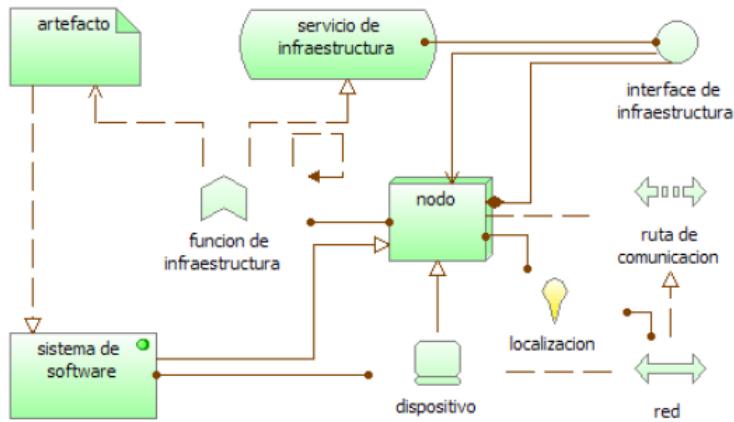


Figura 3-20.: Metamodelo Punto de Vista de Infraestructura.

#### Punto de Vista de Uso de Infraestructura

Ver figura 3-21.

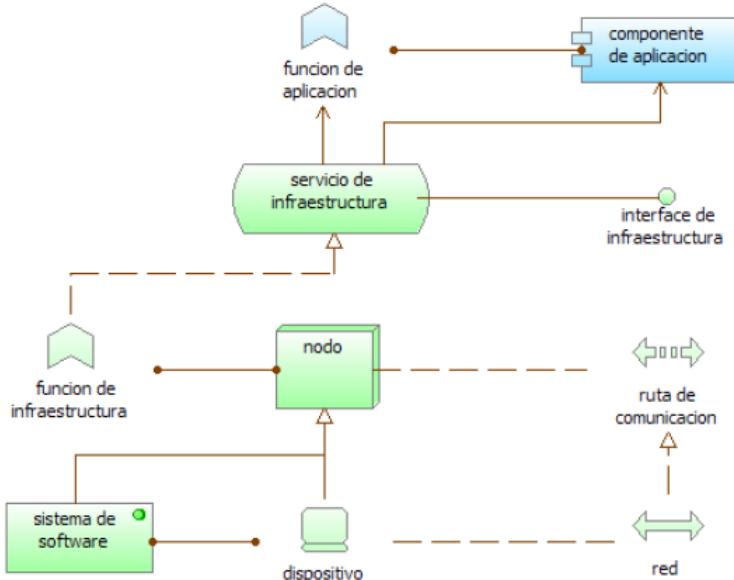
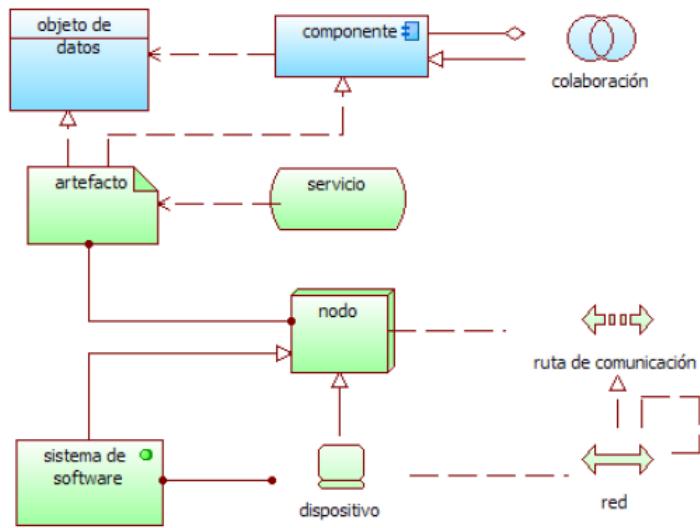


Figura 3-21.: Metamodelo Punto de Vista de Uso de Infraestructura.

#### Punto de Vista de Organización e Implementación

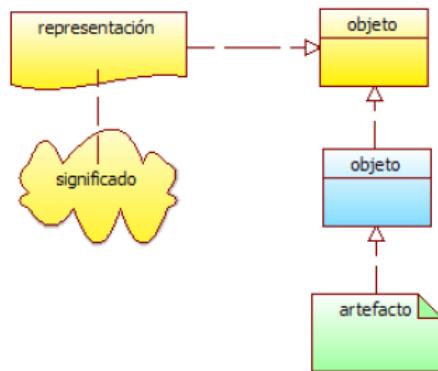
Ver figura 3-22.



**Figura 3-22.:** Metamodelo Punto de Vista de Organización e Implementación.

#### Punto de Vista de Estructura de Información

Ver figura 3-23.



**Figura 3-23.:** Metamodelo Punto de Vista de Estructura de Información.

#### Punto de Vista de Realización del Servicio

Ver figura 3-24.

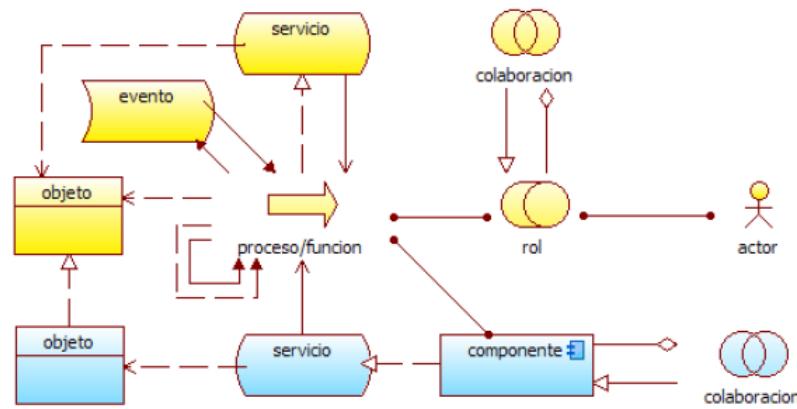


Figura 3-24.: Metamodelo Punto de Vista de Realización del Servicio.

### Punto de Vista de Interesados

Luego de haber analizado el negocio y haber encontrado en los puntos de vista del negocio, los interesados, se concretan y definen a groso modo los interesados de la arquitectura. Ver figura 3-25.

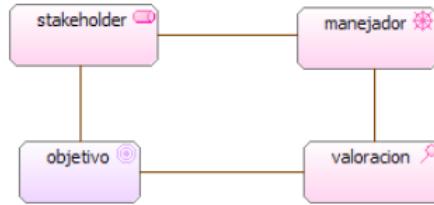


Figura 3-25.: Metamodelo Punto de Vista de Interesados.

### Punto de Vista de Realización de Objetivos

Ver figura 3-26.

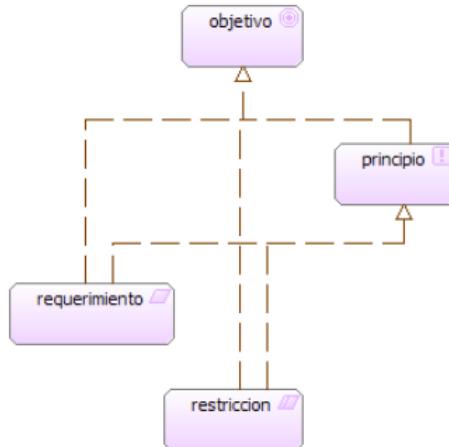
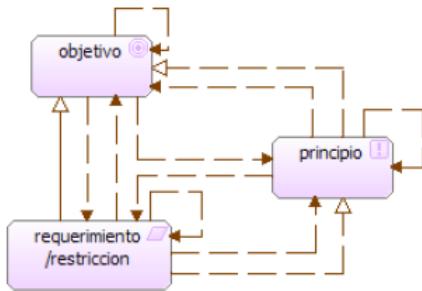


Figura 3-26.: Metamodelo Punto de Vista de Realización de Objetivos.

### Punto de Vista de Contribución

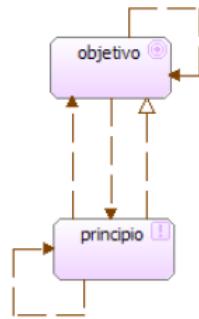
Ver figura 3-27.



**Figura 3-27.:** Metamodelo Punto de Vista de Contribución.

### Punto de Vista de Principios

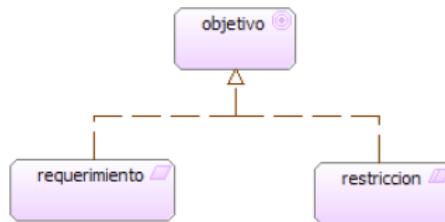
En este punto de vista solo está el objetivo y los principios. Los principios son de tipo sujeto, por ejemplo: confianza, oportunidad, etc. Ver figura 3-28.



**Figura 3-28.:** Metamodelo Punto de Vista de Principios.

### Punto de Vista de Realización de Requerimientos

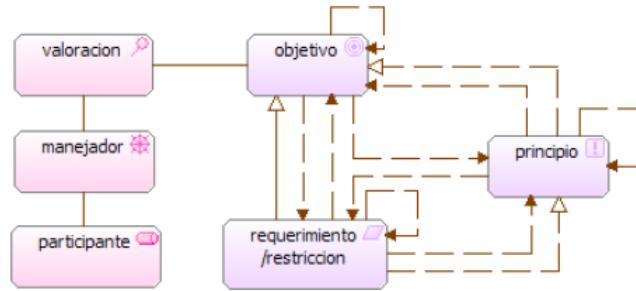
Ver figura 3-29.



**Figura 3-29.:** Metamodelo Punto de Vista de Realización de Requerimientos.

### Punto de Vista de Motivación

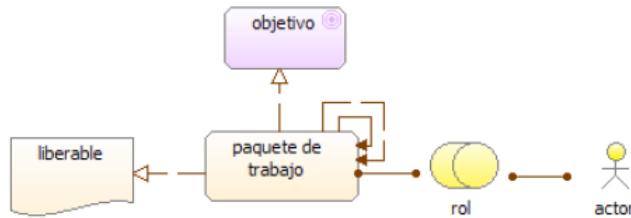
Ver figura 3-30.



**Figura 3-30.:** Metamodelo Punto de Vista de Motivación.

### Punto de Vista de Proyecto

Esta capa nos lleva al nivel de contextualización dejando de lado la abstracción. Para llegar a este nivel es importante jugar con los roles del negocio. Ver figura 3-31.



**Figura 3-31.:** Metamodelo Punto de Vista de Proyecto.

### Punto de Vista de Migración

Ver figura 3-32.



**Figura 3-32.:** Metamodelo Punto de Vista de Migración.

### Punto de Vista de Migración e Implementación

La capa de migración e implementación en el metamodelo de extensión representa el comportamiento que puede tener la aplicación o proceso que se implementa mediante un entregable asociado a una brecha. Las plateas sirven para identificar la arquitectura de transición. Ver figura 3-33.

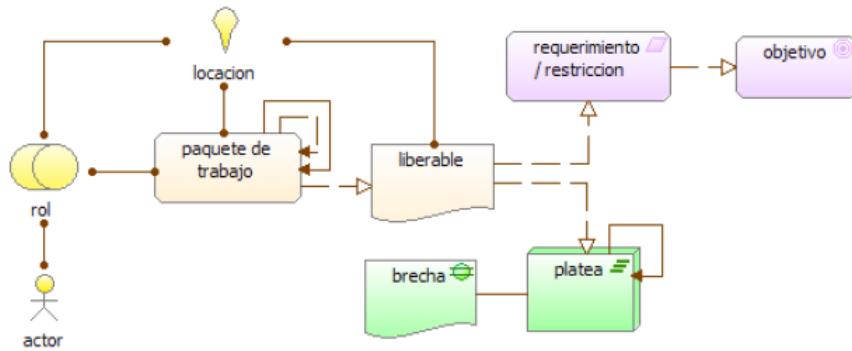


Figura 3-33.: Metamodelo Punto de Vista de Migración e Implementación.

### 3.1.11. Archimate: Extensiones de Lenguaje ArchiMate

#### Extensión de Motivación

Los conceptos principales de ArchiMate se enfocan a describir la arquitectura de los sistemas que soportan la operación de la Empresa. Lo que no alcanza a cubrir esta agrupación, son los elementos que motivan el diseño y la operación de la empresa. Estos aspectos motivacionales corresponden a la columna ”¿Por qué?” del Marco de Trabajo de Zachman.

La extensión de Motivación agrega conceptos motivacionales como ”Meta”, ”Principio”, y Requerimiento”. Se enfoca en la forma con la que la Arquitectura Empresarial se alinea a su contexto a través de los elementos motivacionales.

Un elemento motivacional se puede definir como un elemento que provee de contexto o razón que se encuentra detrás de una Arquitectura Empresarial.

#### Extensión de Implementación y Migración

La extensión de implementación y migración agrega conceptos que soportan las etapas del ADM de TOGAF que se encuentran relacionadas a la implementación y migración de una arquitectura. Las Fases son las siguientes:

- Fase E: Oportunidades y Soluciones.
- Fase F: Plan de Migración.
- Fase G: Gobierno de Implementación. En esta fase se van presentando las arquitecturas las cuales se van aprobando por un comité, esto garantiza que lo que se está diseñando se cumpla. Un ejemplo en esta fase son los contratos de niveles de servicio que se realizan entre la organización y el cliente.

Esta extensión, incluye conceptos para modelar los programas de implementación así como proyectos que soporten el programa, portafolio y la gestión del proyectos y los períodos que soportan el plan de migración.



## 3.2. Marco Conceptual

### 3.2.1. API - Interfaz de Programación de Aplicaciones

La Interfaz de Programación de Aplicaciones del inglés **Applicaton Programming Interfac (API)**, en la programación orientada a objetos es el conjunto de métodos que ofrece una librería o biblioteca para ser utilizada por otro software. [34]

### 3.2.2. Arquitectura SOA

La arquitectura orientada a servicios del inglés **Service Oriented Architecture (SOA)**, permite construir sistemas distribuidos, o sea, un conjunto de sistemas ubicados en distintas partes que colaboran entre sí y que satisfacen los requerimientos de los usuarios. [35]

#### Características principales de la arquitectura SOA

Las principales características de una arquitectura SOA [36] son:

- Los servicios deben estar definidos a través de protocolos y lenguajes estándar de forma que su uso sea independiente de la plataforma.
- Los servicios deben estar accesibles y publicados en un repositorio de servicios con el objetivo de poder monitorizar el estado de los mismos.
- Los servicios definidos deben cumplir el requisito de re-utilización.
- Los servicios encapsulan los detalles de implementación y exponen sus funcionalidades a través de una interfaz.
- Un servicio debe ser independiente y no debe depender del estado de otro servicio.
- Un servicio debe tener un nivel de granularidad o grado de complejidad en el funcionamiento.
- Los servicios pueden reorganizarse en una orquestación de servicios para representar funcionalidades más complejas o integraciones de los procesos de negocio.

### 3.2.3. Servicio Web

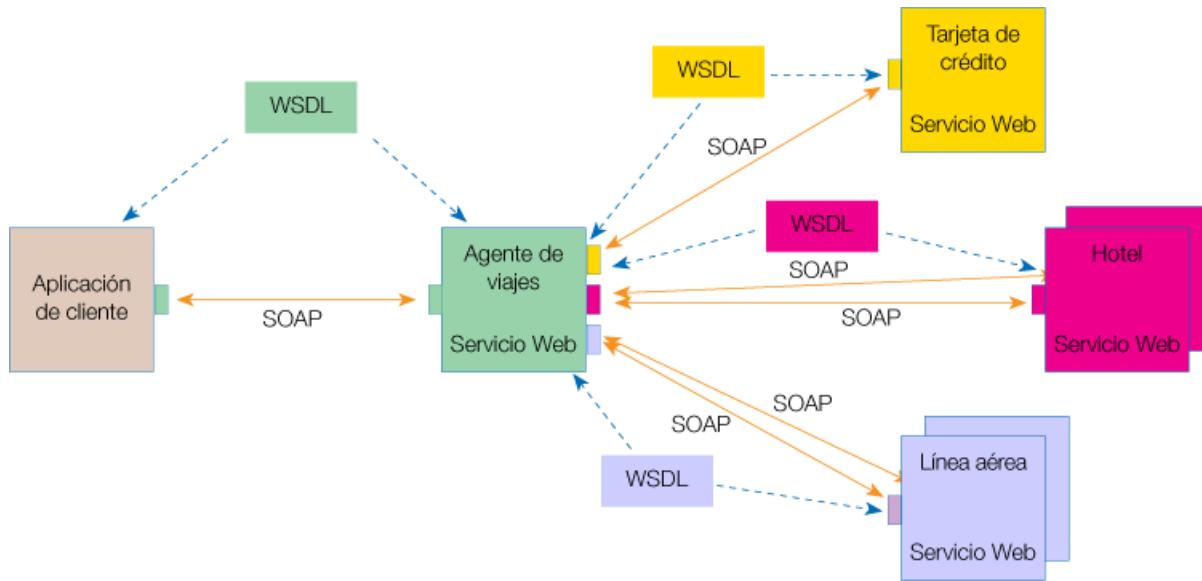
Los servicios web del inglés **Web Services (WS)**, es un software que permite la comunicación entre máquinas a través de una red. [37] Visto de esta forma también pueden ser un conjunto de aplicaciones en la red que colaboran entre sí intercambiando datos con el objetivo de ofrecer unos servicios.

Los servicios web sirven [38] para proporcionar estándares de comunicación entre aplicaciones, permitiendo ampliar el dominio de los datos, su administración y análisis, llegando a ser posible operaciones complejas.

El siguiente es un ejemplo tomado de la página W3C<sup>4</sup> (World Wide Web Consortium) donde se puede observar la interacción de los componentes entre aplicaciones y servicios web. Ver la figura 3-34.

---

<sup>4</sup>Consorcio Mundial de la Red, el número 3 es por las tres W del inglés World Wide Web Consortium (W3C)



**Figura 3-34.** Los Servicios Web en Funcionamiento. Fuente: [9].

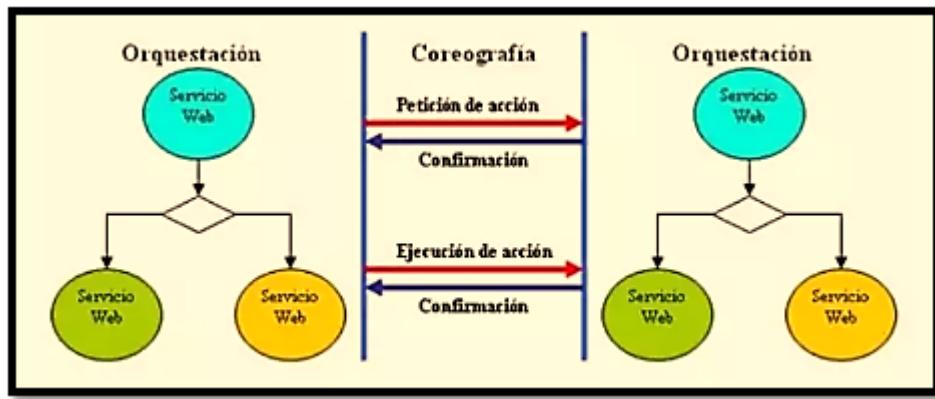
En la figura 3-34 tomada del sitio web de W3C [9] se observa un servicio web central que hace de agente de viajes. El servicio web de agentes de viajes interactúa con la aplicación del cliente, el servicio web de tarjetas de crédito, hotel y la empresa aérea mediante mensajes SOAP (basados en XML). Cuando el cliente realiza una petición al servicio web del agente este coordina y orquesta la comunicación con los otros servicios, para ofrecerle al cliente información sobre el hotel y los horarios de viaje y así este pueda realizar su compra mediante tarjetas de crédito.

### Orquestación y coreografía de servicios web

**Orquestación** Se puede interpretar como las actividades de un proceso que se deben llevar a cabo en un orden al invocar servicios web o interacción entre servicios web. [39]

**Coreografía** Define las colaboraciones entre los tipos de aplicaciones, protocolos de negocio y reglas de interacción. [40]

**Orquestación y Coreografía en Servicios Web** La orquestación sucede en un dominio específico, dentro de la misma arquitectura de un servicio web único o conjunto de servicios web y la coreografía sucede entre dominios de servicios web para su interacción. [41] Ver figura 3-35.



**Figura 3-35.: Orquestación y Coreografía de Servicios Web.** Fuente: [10]

### 3.2.4. SOA y Servicios Web

El paradigma SOA es general y abstracto. Un servicio web es un software que soporta la interacción máquina a máquina a través de una red, mientras que los servicios web no son SOA o viceversa. [24]

Una arquitectura orientada a los servicios trata sobre los servicios e interfaces y la forma y orden en que son llamados de acuerdo a una orquestación. Un servicio web es un estándar de comunicación destacado para llevar a cabo o ser parte de un SOA y dependiendo del tipo de mensaje a manejar puede tratar con formatos o lenguajes de comunicación como: XML, JSON, REST, SOAP, WSDL, UDDI. [42]

### 3.2.5. Arquitectura de Microservicios

Los microservicios aún no tiene una definición formal pero se puede definir como el desarrollo de una aplicación como un conjunto de servicios pequeños donde cada uno se ejecuta en su propio proceso y su comunicación es ligera. [43]

Estos servicios son pequeños, altamente desacoplados y se centran en hacer una pequeña tarea. [44]

Los servicios son los bloques de construcción que comprenden los sistemas creados con la arquitectura de microservicios.

### 3.2.6. RabbitMQ

RabbitMQ<sup>5</sup> es una solución de código abierto para implementar productores y consumidores según las especificaciones del sistema. Es un intermediario de mensajes ligero como Vert.x<sup>6</sup> como puerta de enlace API para asignar diferentes rutas a los mensajes de los microservicios. Está escrito en Erlang que es un lenguaje extendido por su programación concurrente.

RabbitMQ actúa como Middleware de mensajería, es decir, sirve de intermediario para transmitir mensajes entre el transmisor y el receptor. Actualmente se dispone de varios plugins para su implementación, depen-

<sup>5</sup><https://www.rabbitmq.com/>

<sup>6</sup>Vert.x permite que su aplicación pueda manejar una gran cantidad de concurrencia utilizando un mínimo de recursos de su hardware. Sitio web oficial: <http://vertx.io/>

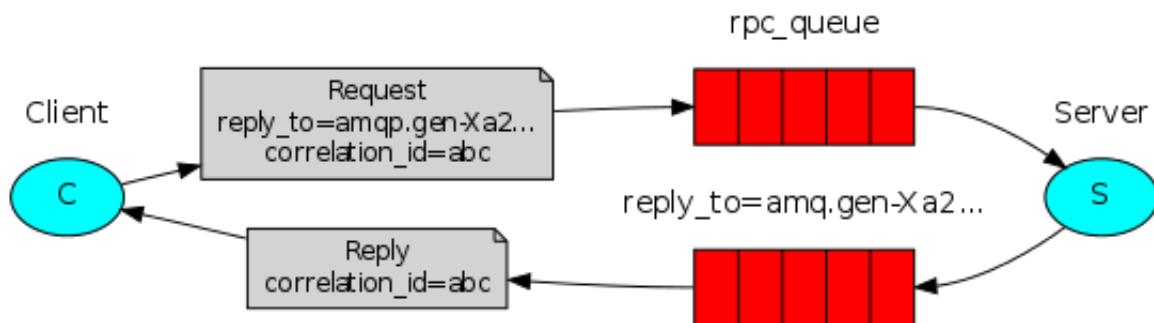
diendo del lenguaje que utilice el programador.

RabbitMQ permite que el desarrollador cree su propia configuración, pero también ofrece tres tipos de intercambio de mensajes:

- **Direct:** comunicación punto a punto copiando el mensaje en la cola que coincide con la clave del mensaje.
- **Fanout:** configuración publicación/suscripción copiando el mensaje en todas las colas que tenga conectadas.
- **Topic:** envío de mensajes según las especificaciones, como por ejemplo, mensajes de error o advertencia deben llegar a los receptores configurados.

### Patrón Llamada a Procedimiento Remoto

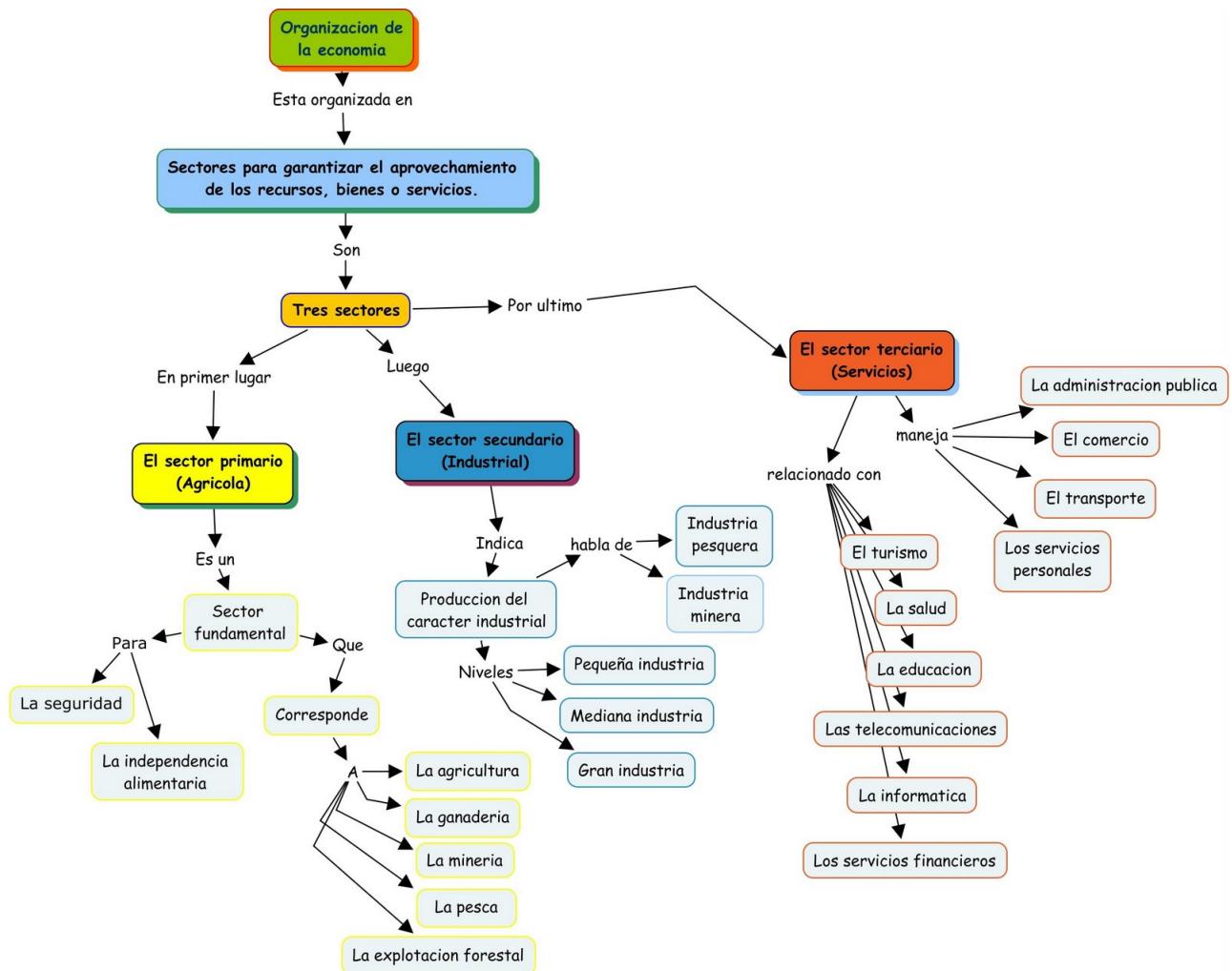
El patrón llamada a procedimiento remoto por sus siglas en inglés **Remote Procedure Call (RPC)**, es un patrón que permite crear un identificador de cola para el mensaje de solicitud y respuesta, ignorando los mensajes desconocidos sin producir error, eliminando recursos de la cola creada una vez se confirma la recepción de la respuesta a la solicitud. Ver la figura 3-36.



**Figura 3-36.:** Diagrama Patrón RPC de RabbitMQ Fuente: [11].

#### 3.2.7. Sectores Económicos en Colombia

Las actividades económicas en Colombia están clasificadas en sectores para garantizar el aprovechamiento de los recursos, bienes o servicios. A continuación en la figura 3-37 se expone brevemente cada uno de ellos:



**Figura 3-37.: Sectores Económicos en Colombia. Fuente: [12]**

- El sector primario o agrícola corresponde a la agricultura, ganadería, minería, pesca y explotación forestal. Los negocios de productos de consumo masivo pertenecen al sector de los servicios
- El sector secundario o industrial tiene tres niveles, pequeño, mediana y gran industria y habla de industria pesquera e industria minera.
- El sector terciario o de servicios está relacionado con el turismo, la salud, la educación, las telecomunicaciones, la informática, los servicios financieros y maneja la administración pública, el comercio, el transporte y los servicios personales.

En la Internet encontramos un sitio web, observatorio de la complejidad económica en inglés **The Observatory of Economic Complexity**, con información económica de todos los países del mundo<sup>7</sup>, es un excelente atlas de la economía. [45]www.gmai

<sup>7</sup>Para observar la economía de Colombia visite: <http://atlas.media.mit.edu/en/profile/country/col/>

**Parte II.**

**DESARROLLO DE LA  
INVESTIGACIÓN**

## 4. METODOLOGÍA PROPUESTA

Este capítulo no presenta un tutorial paso a paso, tampoco es un curso sobre como exponer servicios web con una tecnología específica. De lo que si se trata es de proponer y diseñar una arquitectura que permita dividir el trabajo y realizarlo en paralelo y buscar la tecnología más sencilla que en el menor tiempo nos permita desarrollar e implementar la arquitectura.

### 4.1. Arquitectura Propuesta

Esta **Arquitectura** toma elementos y relaciones de las arquitecturas existentes más aceptadas por la industria del software, los mezcla y distribuye en una nueva arquitectura que sirvan como **Propuesta** base en el desarrollo y diseño de servicios web.

#### 4.1.1. Investigación y Análisis Previo

A partir de la observación personal, corroborada por la experiencia, la mayoría de desarrollos de software no incluyen un componente de servicios por que desconocen la tecnología para hacerlo, no estudian o investigan o no ven el valor agregado de incluirlo y sus usos, piensan que se trata de un tema mas de la teoría de la computación distribuida o algún tipo de moda en los sistemas de software.

Apoyado de los resultados de las encuestas realizadas a profesionales de la industria del software (ingenieros de software, ingenieros de sistemas y otros profesionales de tecnologías), se obtuvieron los mismos problemas observados. A continuación se listan algunos de ellos:

- No se comprende el potencial de los servicios web como solución al intercambio eficiente, seguro y privado de información entre sistemas, prefieren utilizar archivos planos o accesos directos a los motores de las bases de datos.
- Debido a la gran cantidad de marcos de trabajo para crear servicios web, se dificulta y aumenta la curva de aprendizaje.
- Los proyectos de servicios web toman igual o mas cantidad de tiempo que otro tipo de proyectos complejos, por lo tanto no son rentables.
- Los profesionales no estudian e investigan como desarrollar e implementar servicios web, es algo que hacen solo hasta cuando les toca.
- Los profesionales no tienen claro la diferencia entre algunas arquitecturas y el uso de los servicios web en ellas.

Por otro lado, en la literatura revisada no se encontró una metodología que indique como desarrollar e implementar más rápido un servicio web, solo se encontraron varios temas dispersos que podrían servir para organizar mejor la forma de crearlos. La mayor cantidad de literatura está concentrada en la documentación,

ejemplos y ejercicios que se publican en los sitios oficiales que ofrecen esta tecnología.

De la revisión bibliográfica se encuentran tres usos más comunes de los servicios web:

- Se utilizan como un **componente** en la implementación de **patrones de arquitecturas de software**.
- Se utilizan como un **API** para exponer métodos, administrar solicitudes y respuestas.
- Se utilizan para intercambiar información entre componentes internos de un sistema con el exterior.

Otras fuentes de información son los sitios como: blogs, foros y wikis, creados por personas o comunidades que resuelven inconvenientes comunes que se presentan. En ninguna de las fuentes de documentación recomiendan un marco de trabajo, grupo de componentes que permitan facilitar, agilizar y mejorar el diseño, desarrollo, pruebas e implementación de servicios web.

#### 4.1.2. Servicio Web Completo

El documento IEEE Std 1471-2000, dice que:

”La **Arquitectura de Software** es la organización fundamental de un sistema encarnada en sus **componentes**, las **relaciones** entre ellos y el ambiente y los **principios** que orientan su diseño y evolución.” [46]

Acorde a la definición expuesta se propone una arquitectura con componentes y relaciones entre ellos que sirva para diseñar, **desarrollar e implementar servicio web más rápido y fácil** acorde a las tecnologías actuales.

Entonces se introduce el concepto de **Servicio Web Completo** como aquel servicio web que cumple las siguientes **funciones principales**:

- Autenticar y autorizar el acceso a la información.
- Recibir y responder información.
- Validar, verificar, cifrar y persistir información.
- Configurar y auditar el tratamiento de la información.

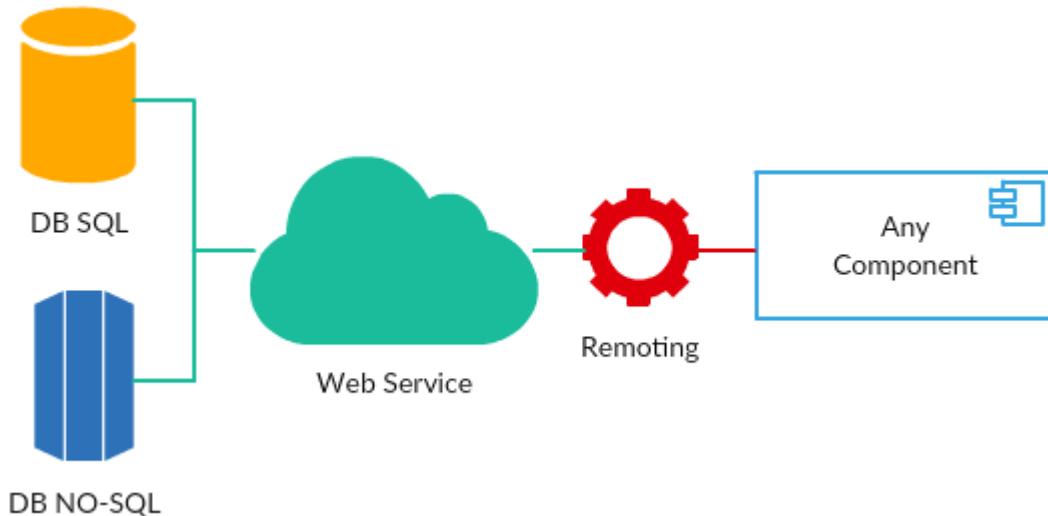
De las funciones principales se derivan las siguientes **características y cualidades** específicas de un servicio web completo:

- Administrar roles, usuarios y permisos en cada método expuesto.
- Utilizar el protocolo HTTP para poder exponer métodos GET, POST, PUT, DELETE, enviar información en la cabecera de las solicitudes y tener un conjunto de códigos de respuesta HTTP predeterminados.
- Procesar solicitudes y respuestas sin importar el lenguaje o protocolo de los mensajes de intercambio.
- Cifrar o encriptar los mensajes de intercambio.
- Comunicarse con sistemas externos para almacenar, obtener, validar o verificar información.

- Administrar y almacenar configuraciones propias del servicio web en bases de datos relacionales y no relacionales.
- Adicionar, modificar o eliminar objetos de base de datos de forma automatizada.
- Almacenar y consultar trazabilidad de las operaciones que se realizan.
- Garantizar el funcionamiento mediante pruebas unitarias sencillas y automáticas.

Para lograr un servicio web con las características anteriores dependerá en gran parte de la tecnología seleccionada para su implementación, pero antes es importante proponer una distribución de elementos que nos den una idea inicial del servicio web completo.

La figura 4-1 representa los elementos de un servicio web completo:



**Figura 4-1.:** Diagrama de Componentes Propuesto de un Servicio Web Completo.

- **WebService:** componente principal encargado de exponer métodos privados y públicos mediante el protocolo HTTP.
- **TracertDB:** base de datos no relacional que se utiliza para dejar evidencia (auditoría) de los procesos o eventos que ocurren en el servicio.
- **WebServiceDB:** base de datos relacional que se utiliza para almacenar la información del servicio y administrar las configuraciones.
- **Remoting:** componente middleware<sup>1</sup> mediante el cual interactuarán otros componentes con el servicio web.
- **Component:** este componente representa otros sistemas que consumen o interactúan con el servicio web.

Se introduce entonces el concepto de **Servicio Web Completo** al conjunto total de elementos como un todo que tienen como finalidad cumplir las funciones, características y cualidades anteriores.

<sup>1</sup>Middleware es un software que asiste a una aplicación para interactuar o comunicarse con otros software o componentes, como por ejemplo servicios web.

#### 4.1.3. Arquitectura Propuesta para un Servicio Web Completo

La arquitectura propuesta<sup>2</sup> es en esencia una combinación de patrones de arquitectura para su implementación con el objetivo de conseguir sencillez, facilidad en el desarrollo y adaptabilidad a las tecnologías actuales para su implementación.

Esta arquitectura también se basa en el ambiente y principios que orientan su diseño y evolución por lo que **hereda** las características más importantes de los patrones de arquitectura de software SOA<sup>3</sup>, ESB<sup>4</sup> y Microservices<sup>5</sup>.

A simple vista podría pensarse que se trata de una arquitectura con microservicios, pero en realidad no es así, lo que se busca es dividir su implementación en varios pasos, rápidos y sencillos de desarrollar.

##### Generalidades

1. A diferencia de un paradigma SOA la arquitectura propuesta no tiene que orquestar o coreografiar componentes ya que su manejo es asíncrono.
2. Tampoco se trata de un API de servicio web que simplemente expone métodos, ya que en lugar de utilizar las llamadas a funciones en memoria, los componentes de la arquitectura propuesta interactúan a través de interfaces de servicio. Esto pone restricciones a la introducción de componentes indeseables y filtra la funcionalidad de un componente a otro. [48]
3. La arquitectura está pensada para cumplir el principio de bajo acoplamiento y alta cohesión<sup>6</sup> consiguiendo que cualquier cambio en un módulo no tenga efecto dominó en otros módulos, la interdependencia entre módulos no exista y el módulo particular sea fácil de rehusar o probar sin necesidad de módulos dependientes.
4. En comparación con el bus de servicios empresariales (ESB) y enfoques similares donde el mecanismo de comunicación proporciona una funcionalidad sofisticada para la transformación de mensajes y la coreografía, la arquitectura propuesta utiliza las comunicaciones solo para intercambiar mensajes.

##### Descripción

La arquitectura propuesta consta de un **Servicio Web Completo** el cual puede ser consumido por clientes como: **Portales, Aplicaciones Móviles u Otros Servicios Web**.

Para ir comprendiendo esta sección de la arquitectura proponemos el siguiente ejemplo: en el contexto de las transacciones financieras se exponen servicios web que son consumidos por las aplicaciones móviles que los usuarios de las entidades financieras utilizan para realizar consultas y transacciones.

Notemos que las aplicaciones y servicios web podríamos describirlos como elementos del mismo dominio de negocio, en el caso del ejemplo, pertenecen al dominio de la entidad financiera.

---

<sup>2</sup>Ver artículo Arquitectura para Diseñar e Implementar Web Services [47]

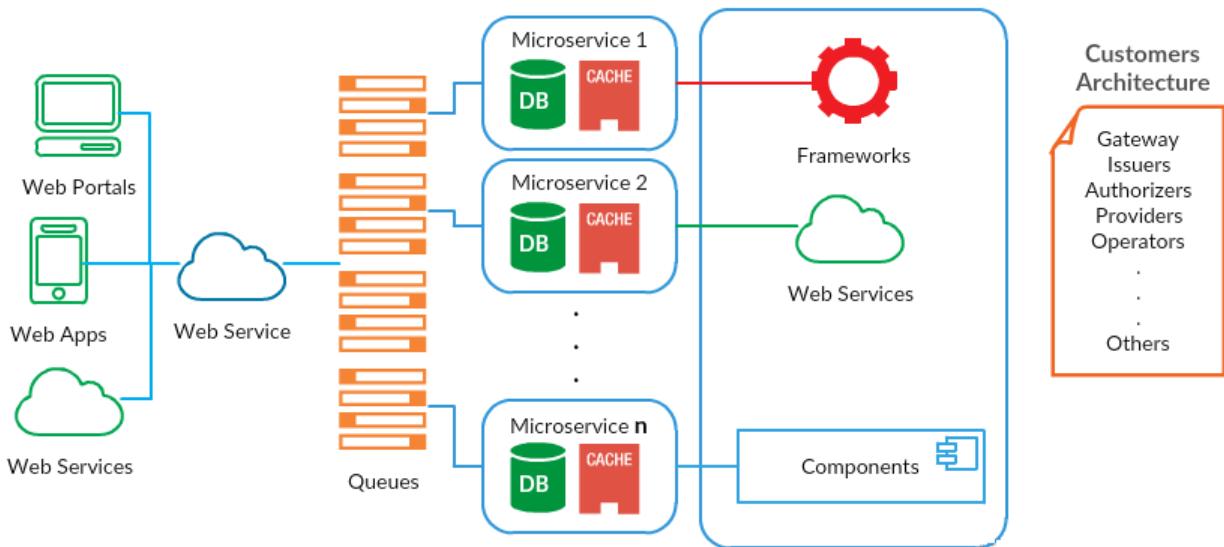
<sup>3</sup>SOA: Service Oriented Architecture

<sup>4</sup>ESB: Enterprise Service Bus

<sup>5</sup>Microservicios

<sup>6</sup>El acoplamiento está comúnmente contrastado con la cohesión. Un bajo acoplamiento normalmente se correlaciona con una alta cohesión, y viceversa. El bajo acoplamiento es frecuentemente una señal de un sistema bien estructurado y de un buen diseño de software.[49]

A continuación se presenta la arquitectura propuesta para un servicio web completo. Ver figura 4-2



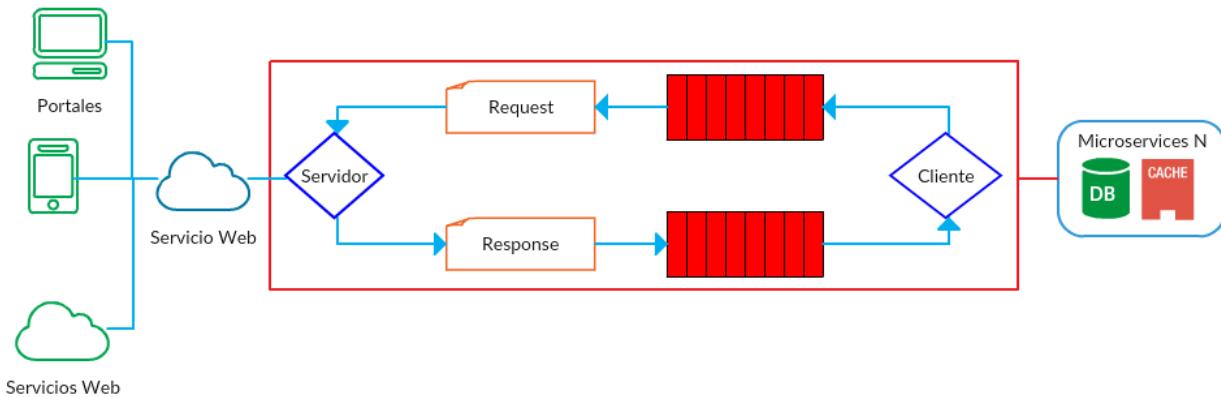
**Figura 4-2.: Arquitectura Propuesta para un Servicio Web Completo.**

- **Web Portals, Web Apps, Web Services:** son los componentes que se comunican con el componente Web Service. Su comunicación aunque puede ser inalámbrica es directa con el componente Web Service por que pertenece al mismo dominio del negocio. Estos componentes no solamente pueden representar computadoras, teléfonos inteligentes, si no también cualquier clase de dispositivo inteligente como neveras, cocinas, hornos, etc.
- **Web Service:** es el componente que se encarga de recibir y entregar mensajes, administrar la autorización y autenticación de acceso, cifrado y encriptado de la información. Sirve como intérprete, capa de seguridad, punto central para asignar la ruta correcta hacia componentes externos. Provee la facilidad de comunicación mediante el protocolo HTTP con rutas específicas de acceso. Además expone un API para su consumo.
- **Queues:** es un componente administrador de colas de trabajos que permite la interacción ordenada y organizada entre el Web Service y los componentes externos. Evita perder solicitudes y respuestas almacenándolas en una cola con un único identificador con lo cual se puede conseguir alta concurrencia.
- **Microservices:** son los componentes o unidades atómicas que tienen su propia lógica de negocio, su propio almacén de datos y manejo de caché. Son las interfaces que interactúan y procesan independiente cada componente externo.

La arquitectura se puede comunicar con componentes externos como por ejemplo: conectarse e interactuar con otros **frameworks**, otros **servicios web** y **componentes** de algún otro software.

En la figura 4-3 se observa una representación interna del funcionamiento de las colas dentro del cual se observa que recibe solicitudes (en inglés request) y entrega respuestas (en inglés response) mediante la entrada y salida de información por filas tipo FIFO<sup>7</sup> la cual va entregando al servicio web o a los microservicios la información, según estos las puedan ir procesando.

<sup>7</sup>FIFO: First In, First Out que en español significa **primero en entrar, primero en salir**



**Figura 4-3.: Arquitectura Propuesta para un Servicio Web Completo con Rabbit MQ.**

A diferencia de SOAP, incluir un manejador de cola de trabajos, **evita estar realizando orquestación y coordinación** entre el servicio y los microservicios. En este caso el servicio web, sencillamente sirve de **multiplexor** y mediante el método que consumen se sabe ya que información deberá enviar al manejador de cola de trabajos para que este se **comunique con el microservicio** correspondiente.

Recapitulando, con esta arquitectura se puede conseguir procesar información en forma paralela o concurrente gracias al servicio web y manejador de colas y disminuir drásticamente el tiempo de respuesta mediante la reutilización de datos en caché e independencia de cada microservicio.

### Ventajas

- Modularidad de los componentes o elementos que los hace fácil de mantener, rediseñar, desarrollar e implementar por separado.
- Cada componente cumple una funcionalidad específica no compartida con otros, lo que hace que sea de muy baja coherencia.
- Permite escalabilidad a futuro agregando componentes pero no implica modificar los componentes existentes.
- El trabajo se divide en equipos lo que permite que la curva de aprendizaje de la tecnología a utilizar, los desarrollos e implementaciones se trabajen en paralelo en muy poco tiempo.

Finalmente, las arquitecturas se constituyen en una solución para integrar y gestionar servicios web, **resolviendo el tema del cómo**, pero no **determinan el con qué**, que es un tema que se trata en la siguiente sección.

## 4.2. Tecnologías Seleccionadas

Actualmente hay una gran cantidad de marcos de trabajo del inglés **frameworks** para distintas plataformas y tecnologías existentes<sup>8</sup>.

<sup>8</sup> Algunos de los framework más comunes por servicios web son: JAVA (Apache Axis 2, Jersey, etc.), .NET (WCF - Windows Communication Foundation, Web API 2, etc.). Ver listados en [50, 51, 52]

Para escoger una tecnología y evitar tener que hacer un análisis comparativo, el cual se sale de los objetivos de este trabajo, se utiliza como **criterio** de selección el principio de parsimonia o **La Navaja de Ockham** [53, 54] el cual dice:

”(...)en igualdad de condiciones, la explicación más sencilla suele ser la más probable(...)”.

Hay dos grandes industrias: **Microsoft<sup>9</sup>** cuyo lenguaje más utilizado es **C#** y **Oracle<sup>10</sup>** cuyo lenguaje reconocido es **JAVA**. Ambas tecnologías tienen marcos de trabajo y componentes multiplataformas similares.

Para seleccionar la tecnología se utilizará el criterio de la Navaja de Ockham, llevado al contexto de la ingeniería de software:

”Si tenemos dos o más frameworks para construir un servicio web, el framework más simple de utilizar es preferible.”

#### 4.2.1. .NET Framework 4.5

Se selecciona como tecnología para llevar a cabo la arquitectura propuesta a **.NET Framework 4.5**. En la figura 4-4 se puede observar el conjunto de componentes, librerías y lenguajes que se pueden utilizar.

.NET Cuenta con una **documentación completa**: teoría, ejemplos, ejercicios, blogs de profesionales y la comunidad, foros con resolución de problemas y un sitio web oficial de aprendizaje en Microsoft® Virtual Academic - MVA.<sup>11</sup> lo que facilita muchísimos las cosas.

---

<sup>9</sup>Microsoft® - <https://www.microsoft.com>

<sup>10</sup>Oracle - <https://www.oracle.com>

<sup>11</sup><https://mva.microsoft.com>

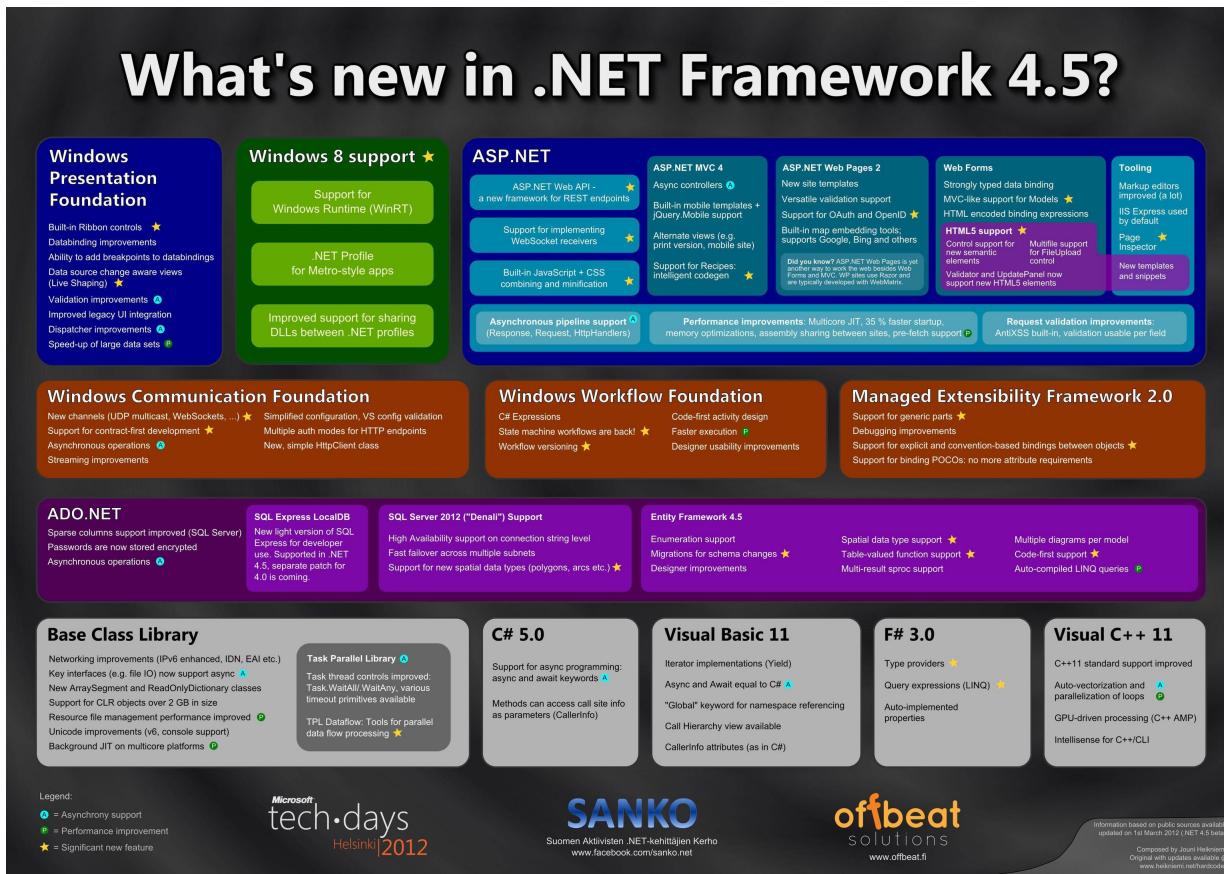


Figura 4-4.: .NET Framework 4.5. Fuente: [13].

Las versiones más recientes son la 4.6 y 5.0 traen nuevas características para aplicaciones móviles y servicios web y se pueden observar en la figura 4-5.

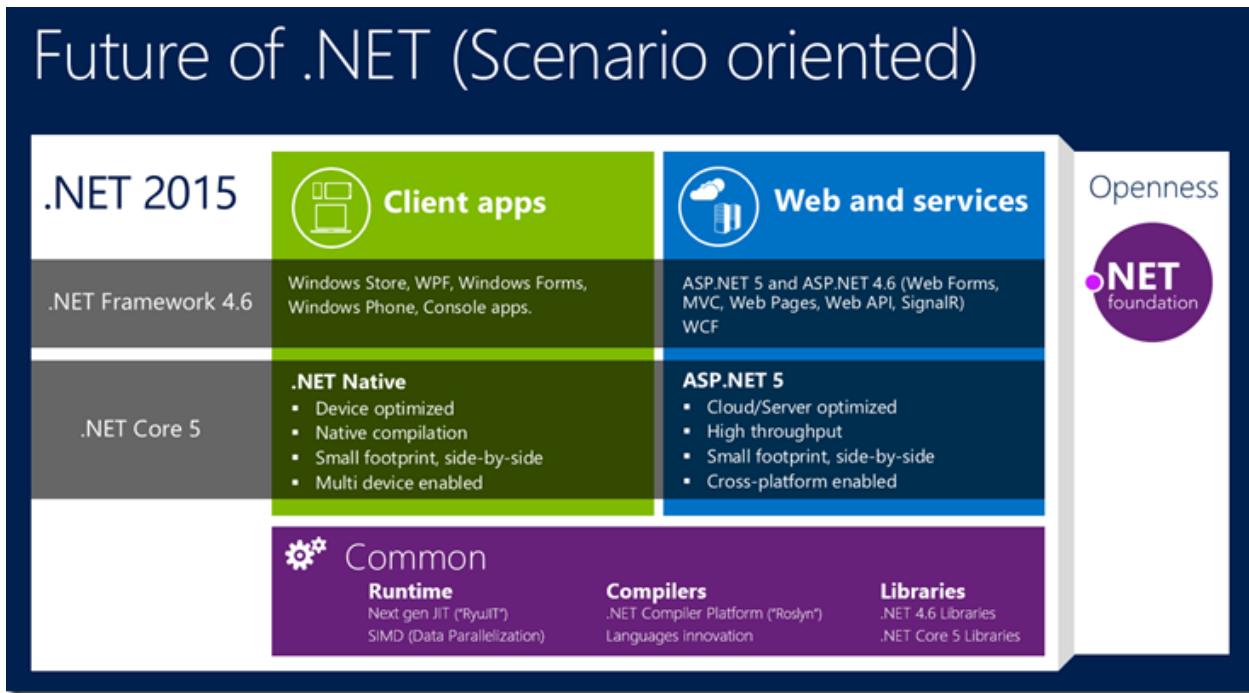


Figura 4-5.: .NET Framework 5.0. Fuente: [14]

#### Otras herramientas del universo .NET

Azure® es una plataforma de servicios integrados que ofrece a través de internet espacio en discos, escritorios virtuales, aplicaciones, hosting, almacén de datos, etc, alojados en servidores de Microsoft®. (Cloud Computing Microsoft)

Algo muy interesante que está sucediendo con Microsoft® es que esta impulsando a la comunidad a utilizar los **productos de desarrollo para crear software libre**<sup>12</sup> el cual si se quiere se puede distribuir comercialmente sin ningún tipo de contrato, acuerdo, costo o licenciamiento<sup>13</sup>.

Apoya proyectos como OWIN y Katana que permiten publicar aplicaciones y servicio web en cualquier plataforma: Windows, Linux, Android, etc.

Por otro lado, Microsoft® adquirió **Xamarin**[57] en 2016, una empresa especializada en el desarrollo de aplicaciones móviles multiplataformas<sup>14</sup>, cuyos productos ahora se ofrecen gratuitamente, lo que amplía más aun el espectro de posibilidades.

#### 4.2.2. Visual Studio 2015 Community

Para llevar a cabo el desarrollo se utiliza una de las ediciones de su herramienta de entorno de desarrollo integrado en inglés **Integrated Development Environment (IDE)** más famosa<sup>15</sup> **Visual Studio 2015**

<sup>12</sup>Recomendado leer el siguiente artículo en un blog oficial de MSDN Microsoft®. Announcing .NET 2015 Preview: A New Era for .NET [55]

<sup>13</sup>Artículo Blog Oficial MSDN Microsoft®. .NET Core is Open Source [56]

<sup>14</sup>Multiplataformas: Android, Apple, Blackberry, Windows Phone, Otras.

<sup>15</sup>Artículo Blog Oficial MSDN Microsoft®. .NET Core is Open Source [56]

**Edición Community**<sup>16</sup> 100 % gratuita.

Con esta herramienta IDE se pueden crear desde aplicaciones de consola, escritorio, web y móviles, hasta servicios web, componentes, librerías, entre otros proyectos que se pueden publicar y utilizar en la nube, lo que lo hace una propuesta realmente atractiva e interesante.

#### 4.2.3. Visual Studio Online

Para administrar el desarrollo se utiliza el portal **Visual Studio Online**<sup>17</sup> el cual se puede enlazar en Visual Studio a través de una cuenta de correo. Permite administrar el desarrollo de aplicaciones.

Algo interesante de este portal es que permite administrar el ciclo de vida de los proyectos de software con metodología como: desarrollo ágil, SCRUM, CMMI entre otras.

Otros servicios que ofrece son: Herramientas para repositorio de información, informes y pruebas de carga.

#### 4.2.4. ASP .NET - Web API 2

”(...)ASP.NET Web API es un framework que facilita la creación de servicios HTTP disponibles para una amplia variedad de clientes, entre los que se incluyen navegadores y dispositivos móviles(...)”[59]

**Web API 2** pertenece a la familia .NET y se utilizará para implementar el servicio web en la arquitectura propuesta. Es muy rápido y sencillo de aprender<sup>18</sup>. La documentación explica muy bien en capítulos, temas específicos. (Ver documentación oficial de Web API 2 [59])

Algunos de estos temas son:

- Crear y exponer métodos.
- Verbos HTTP de acceso u operación de los métodos (Get, Put, Post, Delete)<sup>19</sup>.
- Configuración de las rutas de acceso.
- Administración de seguridad: autenticación y autorización.
- Serialización, cifrado y encriptación de la información.
- Trabajar con datos y clientes móviles.
- Administración de mensajes de respuesta HTTP.
- Manejo de errores y trazabilidad.
- Pruebas unitarias.

Dos de las ventajas más importantes de este framework es que se puede estructurar los mensajes con lenguajes **XML o JSON**. La otra ventaja es que Web API 2 utiliza el **estándar REST** el cual no almacena un estado entre peticiones, no consume memoria del servidor, por lo que a **más clientes no será necesario más hardware**, lo que permite ganar **escalabilidad**.

<sup>16</sup>Sitio web Oficial de Visual Studio Community[58] <https://www.visualstudio.com/products/visual-studio-community-vs>

<sup>17</sup>Sitio oficial de Visual Studio Online: <https://www.visualstudio.com/es-es/products/what-is-visual-studio-online-vs.aspx>

<sup>18</sup>Sitio Web Oficial de Aprendizaje de Web API 2 [60]

<sup>19</sup>Get, Put, Post, Delete corresponden en español a: Obtener, Poner (de insertar o crear), Enviar (de actualizar o modificar) y Eliminar

#### 4.2.5. PowerShell

Microsoft ha incluido en sus sistemas operativos más recientes una consola de sistema más avanzada que la línea de comandos, la cual ha llamado Microsoft® PowerShell® [61]. Ver figura 4-6

The screenshot shows the Windows PowerShell ISE interface. At the top, there's a menu bar with File, Edit, View, Debug, Help. Below the menu is a toolbar with various icons. The main area has two tabs: Untitled1.ps1 and DownloadAllHanselminutesPodcasts.ps1. The Untitled1.ps1 tab is active, displaying the following PowerShell script:

```

1 cd "C:\users\scottha\desktop\Hanselminutes Complete Download"
2 [Environment]::CurrentDirectory=(Get-Location -PSProvider FileSystem)
3 $a = ([xml](new-object net.webclient).downloadstring("http://feeds.f
4 $a.rss.channel.item | foreach{
5     $url = New-Object System.Uri($_.enclosure.url)
6     $file = $url.Segments[-1]
7     $url
8     if (!(test-path $file))
9     {
10         (New-Object System.Net.WebClient).DownloadFile($url, $file)
11     }
12 }
13 }
14

```

Below the script, there are two panes showing command history. The top pane shows:

```
PS C:\Users\scottha\Desktop\Hanselminutes Complete Download> cd ..
```

The bottom pane shows:

```
PS C:\Users\scottha\Desktop>
> cd '.\Hanselminutes Complete Download'
```

At the bottom right of the interface, it says Ln 1 Col 39 and has a character count of 18.

Figura 4-6.: Microsoft® PowerShell® ISE Fuente: [15]

PowerShell® es una herramienta orientada a la automatización de tareas [62] para administradores de sistemas, incluso llegando a igualarse a la programación PERL de UNIX para poder controlar todo en el sistema y realizar tareas administrativas de forma muy rápida, sencilla y segura.

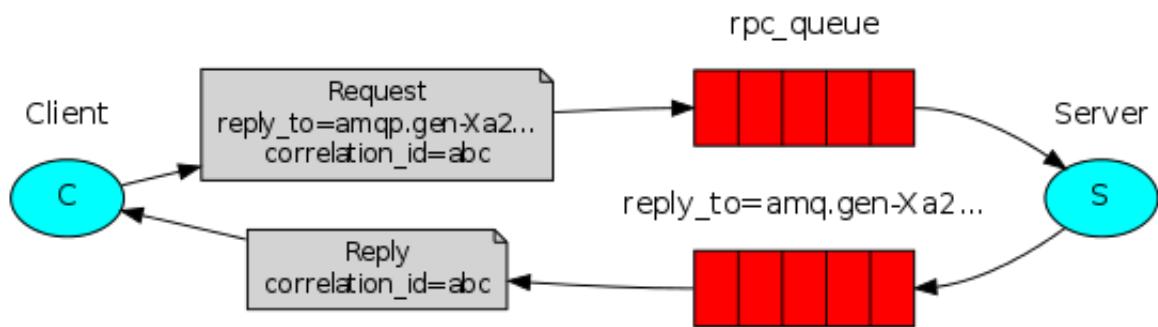
PowerShell® tiene una gran cantidad de elementos [63] con los que se puede interactuar como: variables, operadores, comparadores, comentarios y muchísimos más elementos utilizados en la programación de script.

Se ha seleccionado PowerShell® porque permite interpretar comandos orientados a objetos, por ejemplo leer una librería y utilizar sus métodos sin necesidad de crear entidades que representen la estructura de los datos que queremos entregar o recibir y además se pueden utilizar directamente los métodos sin necesidad de crear una instancia.

Con PowerShell® podemos implementar los elementos "Microservices" de la arquitectura propuesta y automatizar las pruebas unitarias ahorrando tiempo de implementación, disminuyendo la complejidad de programación, haciéndolas más fácil de entender y mantener.

#### 4.2.6. Rabbit MQ

Para llevar a cabo el componente "**Queues**" de la arquitectura propuesta se ha seleccionado RabbitMQ como administrador de colas de trabajo. RabbitMQ se utilizará en modo llamada de procedimientos remotos en inglés **Remote Procedure Call (RPC)** el cual asigna un identificador correlacional único a cada solicitud y respuesta desde o hacia el componente **Queues** lo que permite manejar un gran número de tareas en cola evitando retransmisiones o perdida de la información. Ver figura 4-7



**Figura 4-7.: Rabbit MQ Utilizado como RPC** Fuente: [11]

#### 4.2.7. Dot Net Nuke

**Dot Net Nuke (DNN)** [64] es un gestor de contenidos en inglés **Content Management System (CMS)** que se puede emplear para crear webs, sistemas de intranet y extranet, tiendas virtuales, portales web e incluso servicios web.

Para evitar **reiventar la rueda** y conseguir las características y cualidades deseadas para la arquitectura propuesta se ha seleccionado este CMS que ofrece muchas ventajas como disminuir el tiempo de desarrollo e implementación debido a varias características deseadas, algunas de ellas son:

- **Administradores:** A través de un portal web desde un navegador se puede administrar el sitio web, los portales, módulos, usuarios, roles y permisos necesarios.
- **Autenticación y Autorización:** DNN cuenta con un administrador para los roles, usuarios y permisos. También tiene un administrador para los módulos y los permisos por rol, usuario o portal asociados. En cada módulo se puede administrar los permisos de escritura, lectura, edición y borrado de información.
- **Actualización y Mejoras:** DNN cuenta con un administrador de módulos que permite instalarlos mediante un archivo ZIP. También permite realizar actualizaciones y mejoras en los módulos, llevando un control de versiones. Ideal para instalar y actualizar el módulo de servicio web.
- **Trazabilidad de Eventos:** DNN cuenta con un administrador de eventos que nos permite dejar ver los procesos que se realizan, ideal para ver las solicitudes y respuestas que el servicio recibe y responde.

- **Despliegue e Implementación:** DNN se instala en Internet Information Server que es un servicio del sistema operativo Windows que publica sitios y aplicaciones web para que sean accedidas a través de la red a la que se encuentre conectado.

En la siguiente sección podremos ver como se usan estas tecnologías en el desarrollo e implementación, aplicadas a una parte de la solución de un ejercicio de transformación digital en un negocio de productos de consumo masivo.

# 5. EJERCICIO APLICACIÓN

En este capítulo se mostrarán técnicas de desarrollo e implementación de un servicio web completo como parte de la solución que se plantea en el ejercicio de arquitectura empresarial.

## 5.1. Arquitectura Empresarial

El ejercicio de modelamiento de arquitectura empresarial busca hacer una transformación digital de un negocio de productos de consumo masivo mediante una iteración del ciclo ADM de Archimate omitiendo la fase G y H correspondientes a la gobernabilidad y administración de la arquitectura que para efectos del alcance ejercicios no se incluyen.

A cambio de los entregables formales de cada fase se presentan los puntos de vista mediante diagramas que representan una parte de la realidad que estoy modelando. Estas vistas que permiten modelar la realidad, mediante lenguajes simbólicos, que contienen una semántica y manejan unos conceptos.

### 5.1.1. Fase Preliminar

En esta fase se identifica y describe el negocio que deseamos transformar, el problema que se presenta y la solución deseada.

#### Modelo de Negocio Canvas

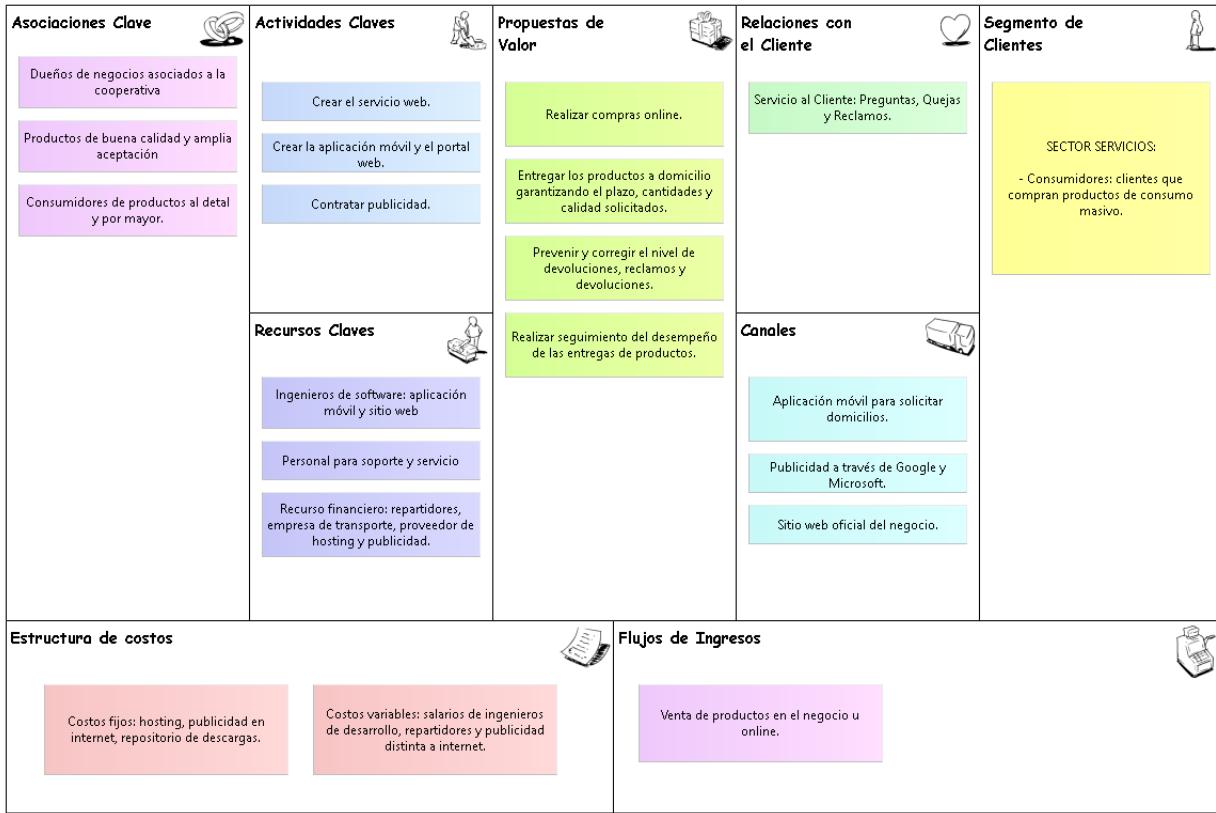
Con este modelo podremos describir el negocio que deseamos tener una vez realizada la transformación digital. Ver figura 5-1. (Para mas detalle ver anexo B)

**Segmento de Clientes** El negocio a transformar pertenece al sector de servicios, específicamente al de productos de consumo masivo. Este negocio tiene un segmento de clientes que son los consumidores o cliente que compran los productos.

**Propuestas de Valor** Se proponen cuatro propuestas de valor: permitir a los consumidores realizar compras online, entregar los productos a domicilio garantizando el plazo, calidad y cantidad acordados, prevenir y corregir el nivel de devoluciones o reclamos y realizar seguimiento del desempeño de las entregas de los productos.

**Relaciones con el Cliente** La relación se llevará a cabo mediante el servicio de atención de preguntas, quejas y reclamos.

**Canales** Son varios los canales de servicio que se utilizarán para llegar a los clientes. Para la publicidad se podrá avisos en Google y Microsoft. La consulta de los productos, domicilio y novedades se realizará a través de sitio web o la aplicación móvil.



**Figura 5-1.: BMS de una Cooperativa con Negocios de Productos de Consumo Masivo. (Para mas detalle ver anexo B)**

**Actividades Claves** Se requiere llevar a cabo:

- Crear el servicio web.
- Crear la aplicación y el portal web.
- Contratar servidores de hosting.
- Contratar publicidad.

**Recursos Claves** Los recursos necesarios son:

- Ingenieros de software para construir la aplicación móvil, el portal web y el servicio web.
- Personal para soporte y servicio.
- Recursos financieros para pagar honorarios, servicios y proveedores.

**Asociaciones Clave** Es importante mantener vínculos estrechos u asociaciones con:

- Consumidor o cliente del negocio.

- Repartidores.
- Dueño del negocio.

**Flujos de Ingresos** El dinero se obtendrá de la venta de productos en el negocio ya sea que el consumir los adquiera en el negocio u online.

### Estructura de Costos

- Costos fijos: internet, teléfono, hosting, publicidad en internet, repositorio de descargas.
- Costos variables: salario de ingenieros, repartidores.

### Negocio de Consumo Masivo

Existe actualmente en Bogotá - Colombia, negocios (tiendas), cooperativas, cadenas de supermercados de productos de consumo masivo como por ejemplo: Éxito, Surtimax, Cooratiendas, etc., que han visto la **oportunidad de mejorar e innovar** el servicio prestado a sus clientes mediante la **venta y entrega de productos a domicilio apoyados de las últimas tecnologías**.

Los negocios de una cooperativa son los más interesados en compartir una visión, misión, objetivos organizacionales y un manejo de imagen e identidad corporativa que los unifique e identifique.

A continuación conoceremos como está organizada una cooperativa de negocios de productos de consumo masivo.

**Organigrama** El organigrama más común de una cooperativa de negocios de productos de consumo masivo se puede ver Ver en la figura **5-2**.

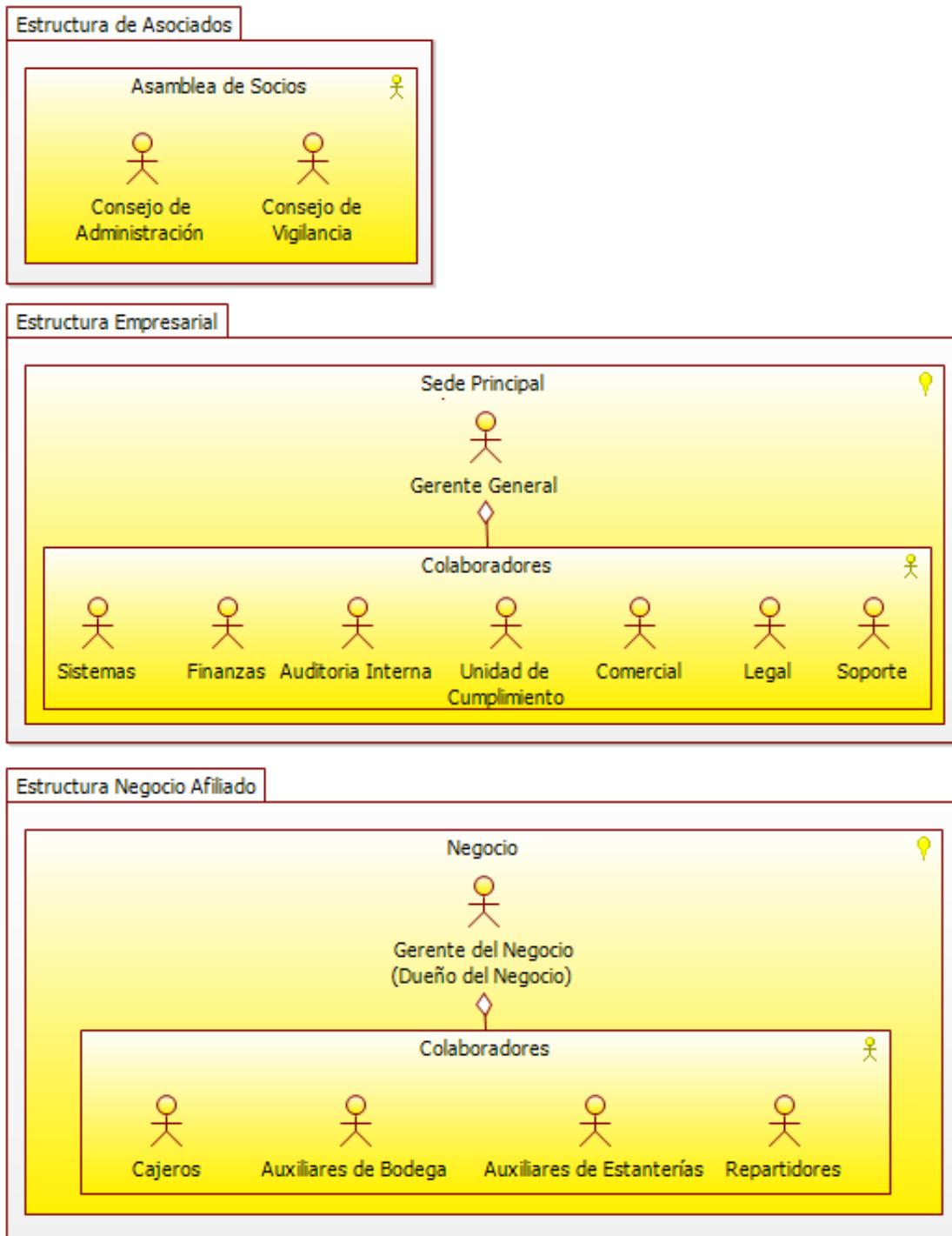


Figura 5-2.: Organigrama Cooperativa de un Negocio de Productos de Consumo Masivo.

La cooperativa está dividida en tres estructuras organizativas: asociados, empresarial y negocios afiliados.

- **Estructura de Asociados:** El máximo organismo de representación es la asamblea de socios conformada por un grupo de socios, el consejo de administración y el consejo de vigilancia. Ubicados en la sede principal.

- **Estructura Empresarial:** Está representada por el Gerente General quien tiene a su cargo un equipo de colaboradores de distintas áreas: Auditoria Interna, Sistemas, Finanzas, Comercial, Soporte, Legal, Unidad de Cumplimiento. Todos ubicados en una sede principal.
- **Estructura Negocio Afiliado:** Son las personas naturales (dueños) que han arrendado un local y se han afiliado a la cooperativa, la cual, en calidad de préstamo les surte el negocio. El dueño del negocio o afiliado cumple la función de gerente del negocio y tiene a su cargo colaboradores en el establecimiento: Cajeros, Auxiliares de Bodega, Auxiliares de Estanterías y Repartidores.

**Misión** Generamos valor a través de productos de consumo de buena calidad transformando los procesos de nuestros clientes y comunidad en general mediante la innovación tecnológica.

**Visión** Se una de las cadenas de supermercados más extensa y exitosa en Colombia para el 2020 mediante innovaciones tecnológicas confiables.

### Objetivos

- Facilitar la vida a nuestros consumidores por medio de la compra de productos online.
- Incrementar el nivel de satisfacción de nuestros consumidores, por medio de la entrega de productos a domicilio que cumplan los parámetros de calidad establecidos, dentro de los plazos y cantidades acordadas.
- Reducir el nivel de devoluciones, reclamos y atrasos en la entrega a domicilio de productos por medio de acciones preventivas, correctivas y un excelente servicio al cliente.

**Objetivo Estratégico** Aumentar los beneficios y fidelización de los consumidores, mediante las ventas online y la entrega de productos de calidad a domicilio.

**Inconveniente Actual** Un negocio como por ejemplo: Cooratiendas no cuentan con ventas online y tampoco con el servicio de entrega a domicilio.

**Solución Pensada** Permitir a los clientes comprar productos mediante portales web o aplicaciones móviles y escoger la dirección donde desea que sean entregados a domicilio.

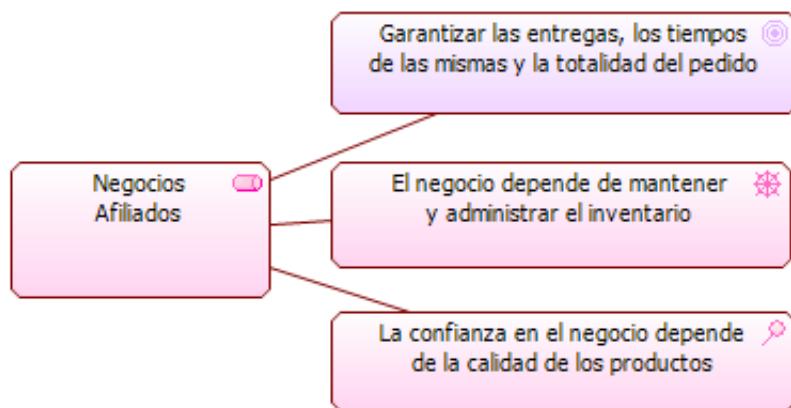
### 5.1.2. Fase A: Visión de la Arquitectura

#### Punto de Vista Motivacional

Se identifica el negocio afiliado a la cooperativa como el **interesado** en llevar a cabo los **objetivos** propuestos: Garantizar las entregas, los tiempos de las mismas y la totalidad del pedido.

Para lograrlo se debe seguir un lineamiento **conductor**: el negocio depende de mantener y administrar el inventario.

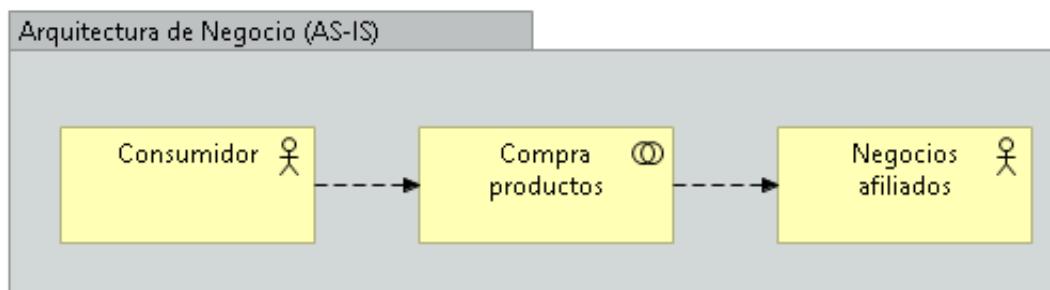
Para garantizar este lineamiento se realiza una **valoración**: la confianza en el negocio depende de la calidad de los productos. Ver figura 5-3.



**Figura 5-3.:** Punto de Vista Motivacional de un Negocios de Productos de Consumo Masivo.

#### Arquitectura Actual (AS-IS)

El estado actual muestra el negocio antes de la transformación. En la figura 5-4 se puede apreciar que el consumidor compra productos en los negocios afiliados a la cooperativa y no se tiene más novedad e innovación alguna.

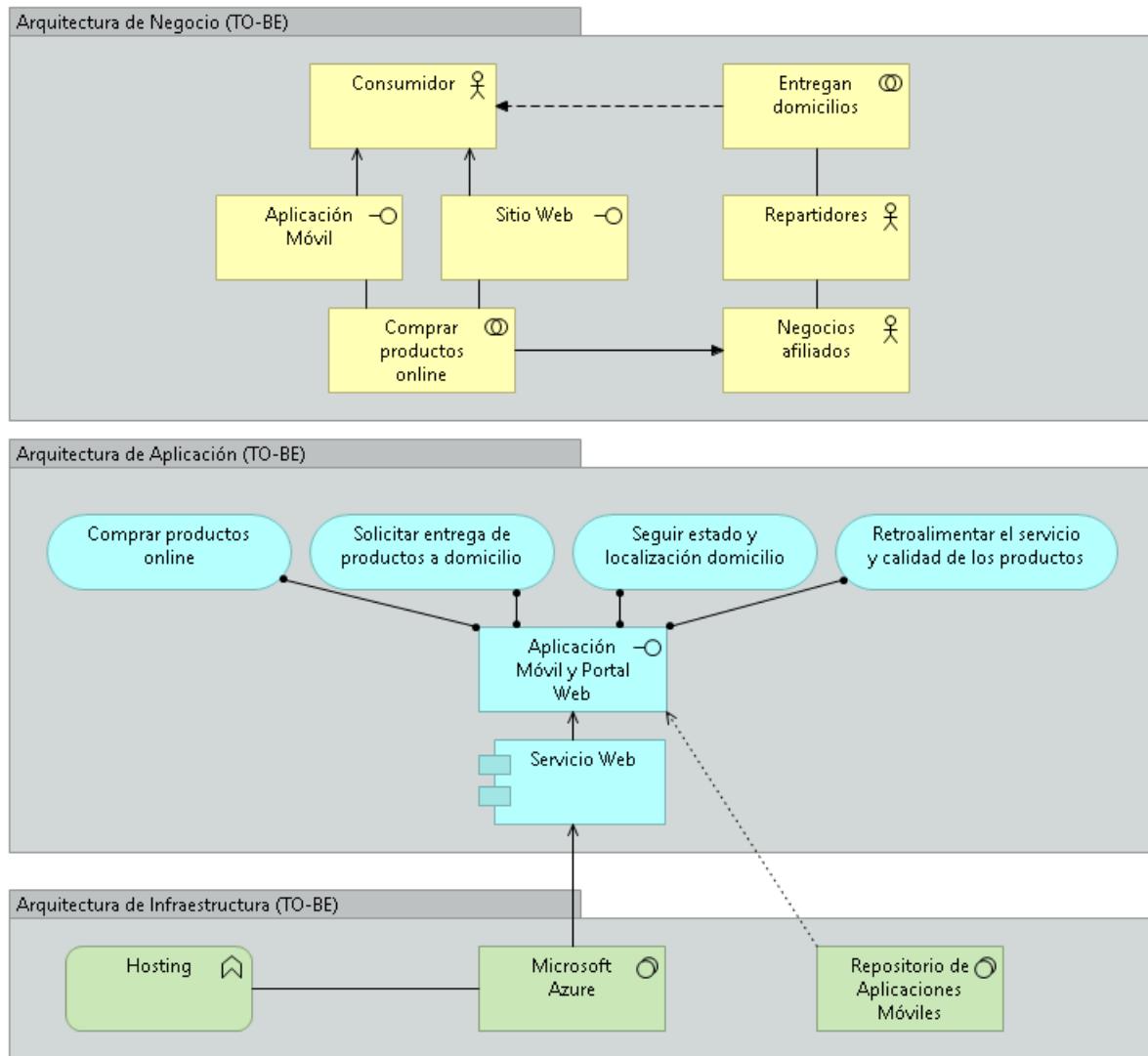


**Figura 5-4.:** Estado Actual (AS-IS) de un Negocio de Productos de Consumo Masivo.

#### Arquitectura Futura (TO-BE)

El estado futuro muestra el negocio después de la transformación. En la Ver figura 5-5 se aprecian tres capas: la de negocio, aplicación e infraestructura, el cual propone que el consumidor pueda realizar compras online

mediante la aplicación móvil o un portal web y que sus productos se entreguen a domicilio.



**Figura 5-5.: Arquitectura del Negocio de Domicilios Futura (TO-BE).**

Para esto se requiere de un componente como un servicio web que sea consumido por una portal y aplicación móvil y que ofrezca los servicios de compra de productos online, solicitar la entrega de productos a domicilio, hacer seguimiento del estado y localización del domicilio y retroalimentar el servicio y calidad de los productos, acorde al punto de vista motivacional.

La infraestructura necesaria será un hosting para el portal y el servicio web y un repositorio para almacenar y permitir descargar e instalar las aplicaciones móviles.

### Objetivos SMART

A partir de lo misional para llevar a cabo la transformación de la arquitectura del AS-IS al TO-BE se propone el siguiente objetivo inteligente u objetivo smart.

**Aumentar el nivel de re-compra de productos 3 veces mediante la venta de productos online y entrega a domicilio, midiendo la periodicidad de compra mensualmente durante el 2017.**

### 5.1.3. Fase B: Arquitectura del Negocio

En esta etapa tenemos como objetivo identificar el negocio y comprender los requerimientos que no son claros frente a la arquitectura. Para tal fin se utilizarán los siguientes puntos de vista:

#### Punto de Vista de Cooperación de Actor

En este punto de vista se identifica un consumidor cuyo papel es el de ser cliente del negocio que mediante una interacción con los productos ubicados en las estanterías de los negocios podrá realizar las compras a través de un portal web o aplicación móvil. Estos dos últimos utilizarán un Web Service para poder vender los productos a domicilio. Ver figura 5-6.

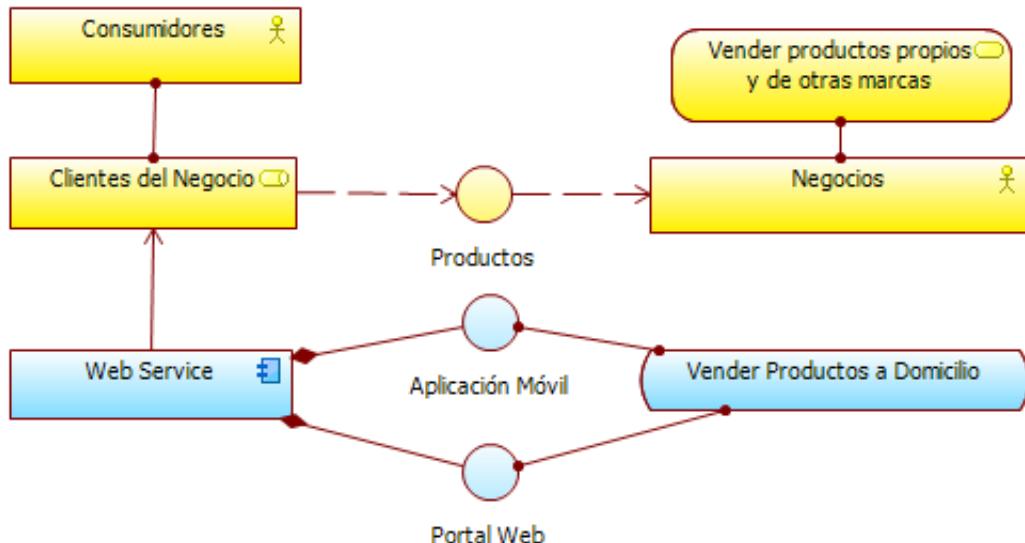
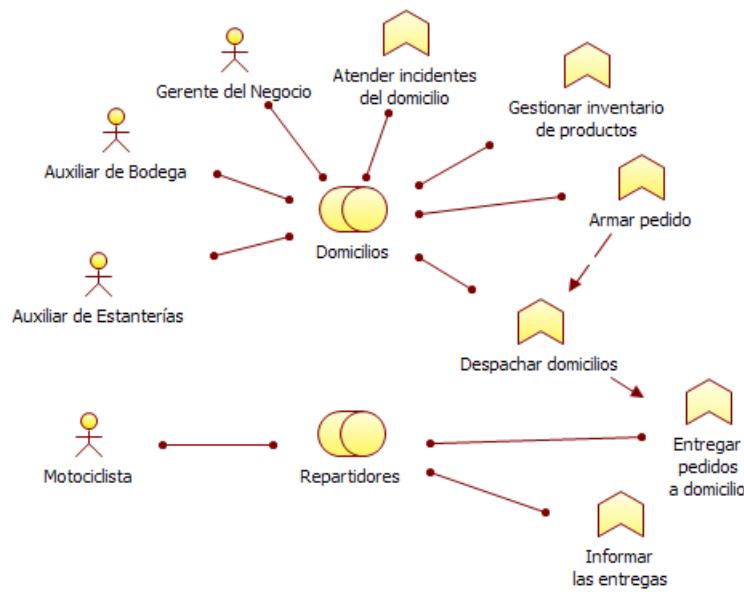


Figura 5-6.: Punto de Vista de Cooperación de Actor.

#### Punto de Vista de Función de Negocio

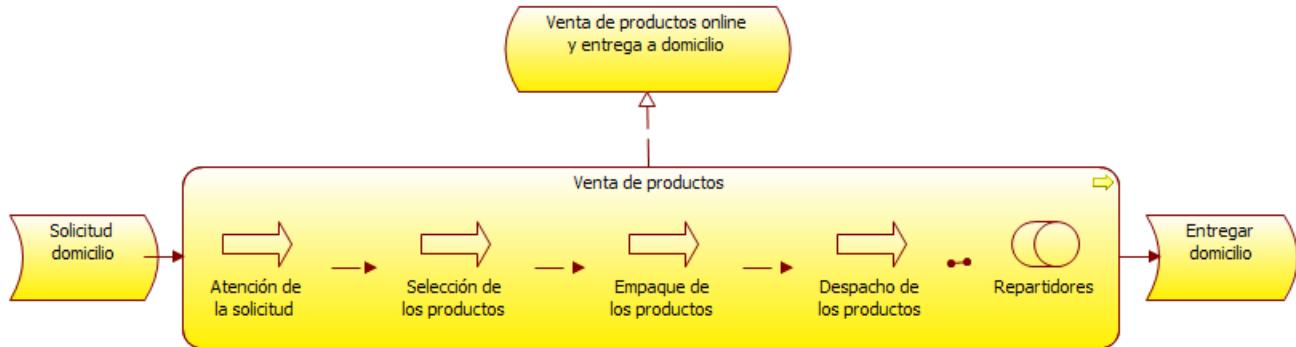
Este punto de vista identifica varios actores: gerente, auxiliares, repartidores y las funciones que cumplen el negocio, como por ejemplo: armar pedido, despachar domicilios, entregar e informar domicilios. Ver figura 5-7.



**Figura 5-7.:** Punto de Vista de Función de Negocio.

#### Punto de Vista de Procesos de Negocio

Esta vista tiene como entrada la solicitud del domicilio la cual conlleva a realizar los siguientes pasos: atención de la solicitud, selección de los productos, empaque de los productos, despacho de los productos. Su salida es la entrega del domicilio. Ver figura 5-8.



**Figura 5-8.:** Punto de Vista de Procesos de Negocio.

#### Punto de Vista de Cooperación de Procesos de Negocio

Los anteriores procesos tienen como función la venta de productos propia del negocio como se aprecia en la figura 5-9.

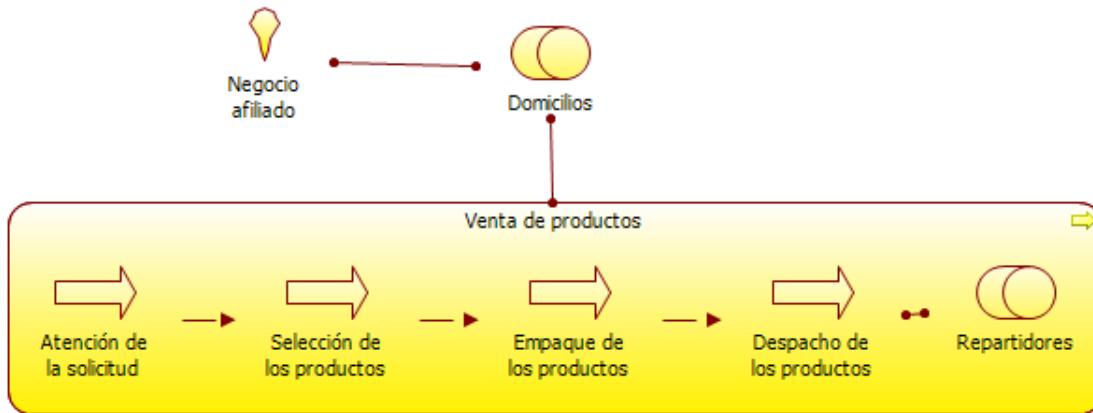


Figura 5-9.: Punto de Vista de Cooperación de Procesos de Negocio.

### Punto de Vista de Producto

El producto del negocio, no el producto de consumo masivo, es la **Venta Online** que me ofrece un valor particular: ayuda a hacer más rápido el trabajo y favorece al cliente. Ver figura 5-10.

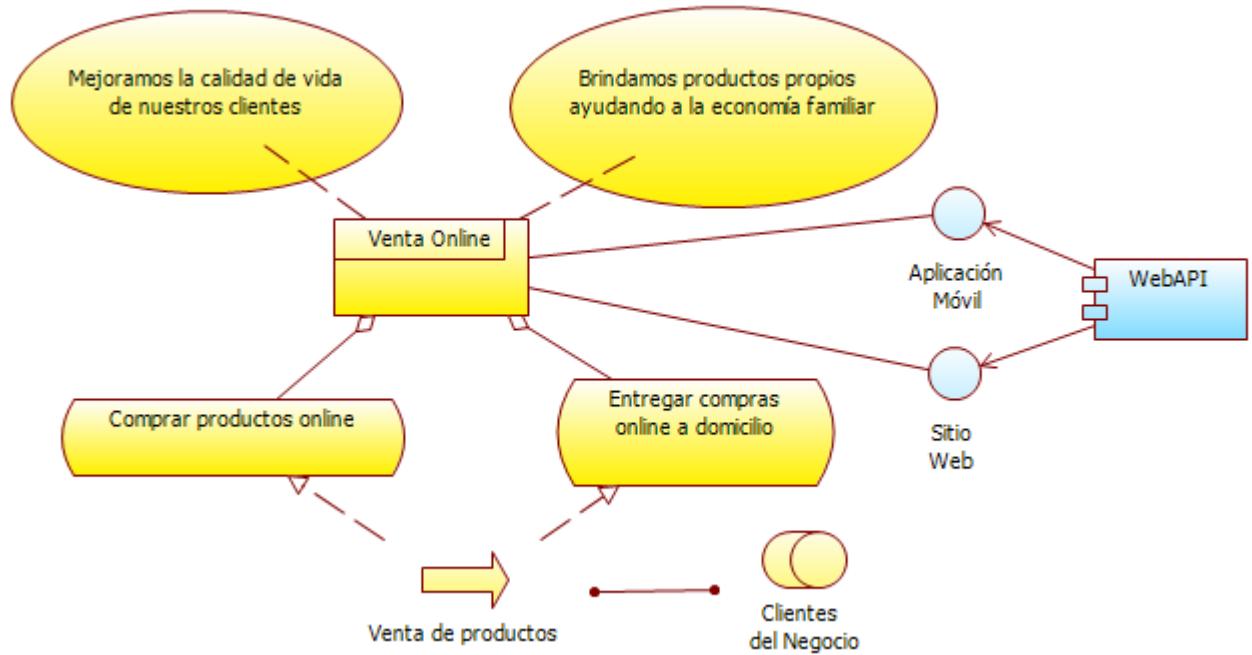


Figura 5-10.: Punto de Vista de Producto.

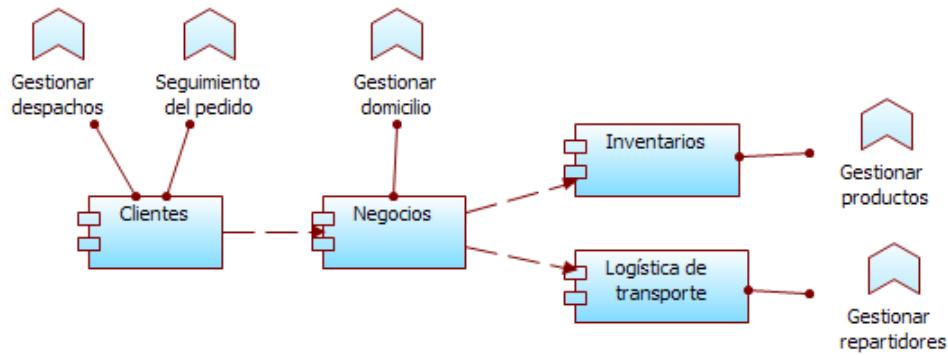
#### 5.1.4. Fase C: Arquitectura de Datos y Aplicación

En esta fase identificaremos los datos más importantes de las aplicaciones para llevar a cabo la transformación del negocio.

### Punto de Vista de Comportamiento de Aplicación

Los componentes o módulos de clientes, negocio prestarán los servicios para gestionar despachos, seguimiento del pedido y la gestión del domicilio de las compras online realizadas.

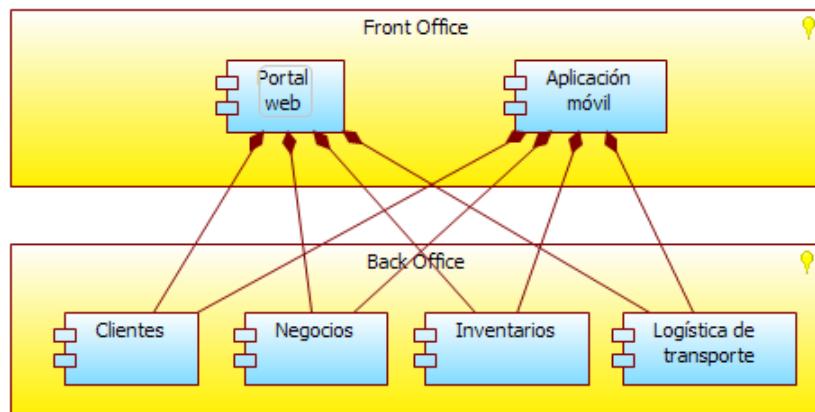
Se observan dos módulos importantes que colaboran con este proceso que son los inventarios: encargados de gestionar productos y la logística de transporte: gestionan los repartidores. Ver figura 5-11.



**Figura 5-11.:** Punto de Vista de Comportamiento de Aplicación.

### Punto de Vista de Cooperación de la Aplicación

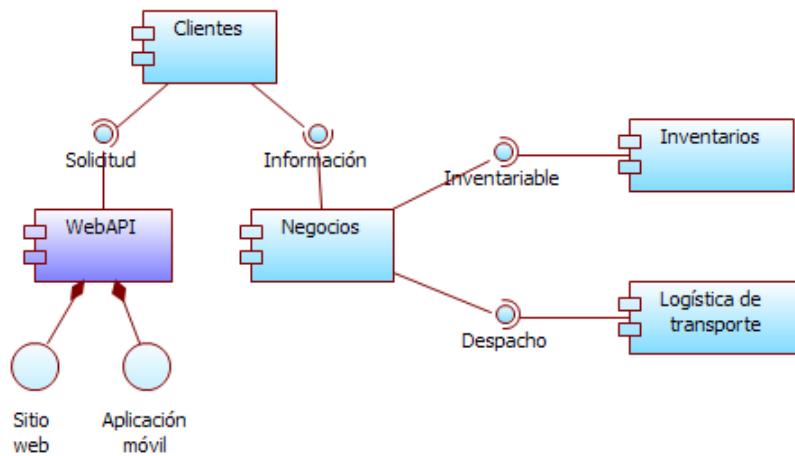
En el front office encontramos el portal y la aplicación móvil. En el back office están los módulos: clientes, negocios, inventarios y logística de transporte. Ver figura 5-12.



**Figura 5-12.:** Punto de Vista de Cooperación de la Aplicación.

### Punto de Vista de Estructura de la Aplicación

Acá aparece un componente y es el del servicio web nombrado WebAPI. Este componente recibe solicitudes y responde a los sitios web y aplicaciones móviles. Se puede observar que este componente se conecta para interactuar con el de clientes y este con el de los negocios. El componente de negocios interactúa con los componentes o módulos de inventarios y logística de transporte. Ver figura 5-13.

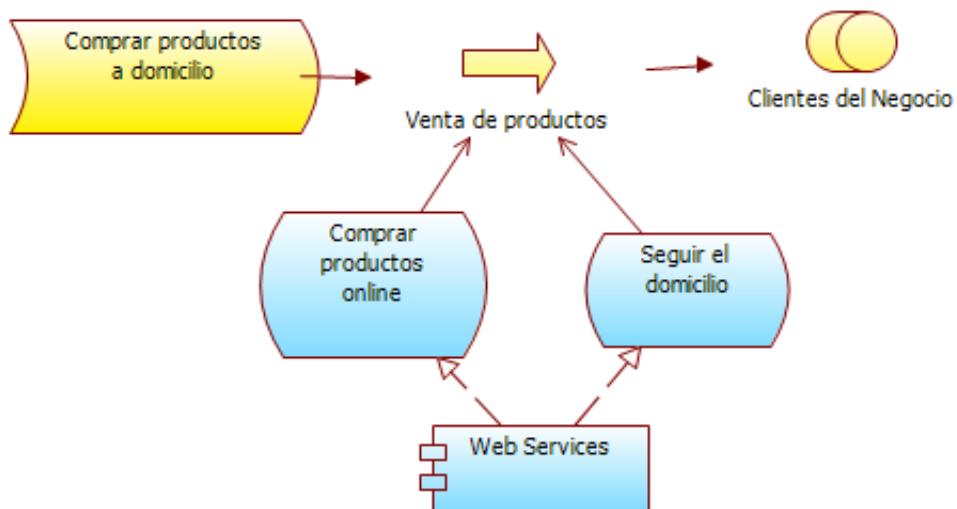


**Figura 5-13.:** Punto de Vista de Estructura de la Aplicación.

### Punto de Vista de Uso de la Aplicación

Conecta la capa de negocio con la de la aplicación.

Los servicios que ofrece el componente Web Services son el de comprar productos online y seguir el domicilio. Lo anterior con tal de cumplir la función “Venta de productos” como respuesta al servicio “Comprar productos a domicilio” que podría realizar cada cliente del negocio. Ver figura 5-14.



**Figura 5-14.:** Punto de Vista de Uso de la Aplicación.

### 5.1.5. Fase D: Arquitectura de la Infraestructura

Una tercera arquitectura es la de tecnología o infraestructura y es la que me va a soportar mi aplicación. En esta capa se identifican los servidores, sistemas operativos, conexiones, ubicaciones y datos.

#### Punto de Vista de Infraestructura

Se identifican tres ubicaciones: la de los clientes, la nube (WAN) y el cloud hosting. En los clientes tenemos las computadoras y dispositivos inteligentes (En inglés SmartPhone).

Los clientes del negocio con sus dispositivos mediante internet (Cloud Hosting) se comunican con los servidores de hosting y de aplicaciones móviles. Ver figura 5-15.

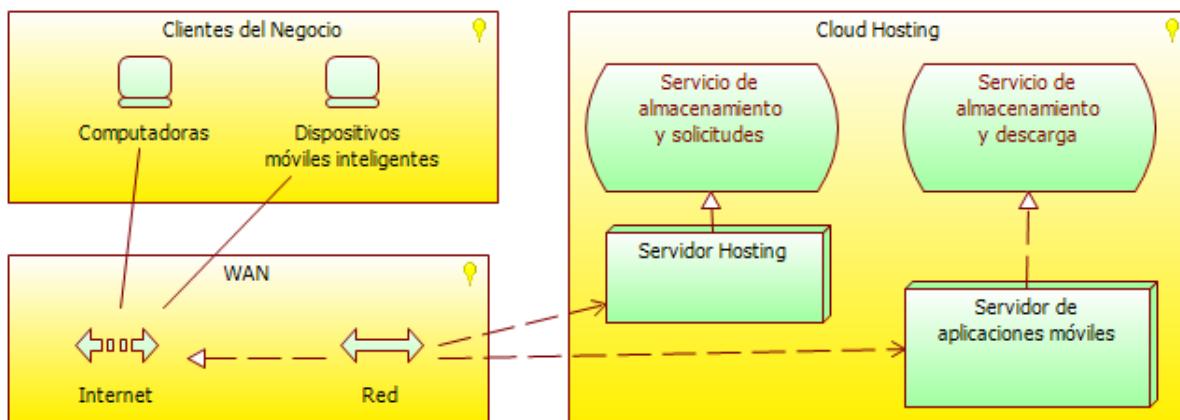


Figura 5-15.: Punto de Vista de Infraestructura.

#### Punto de Vista de Uso de Infraestructura

La infraestructura propone almacenar y ejecutar todos los componentes en un mismo servidor hosting. Ver figura 5-16.

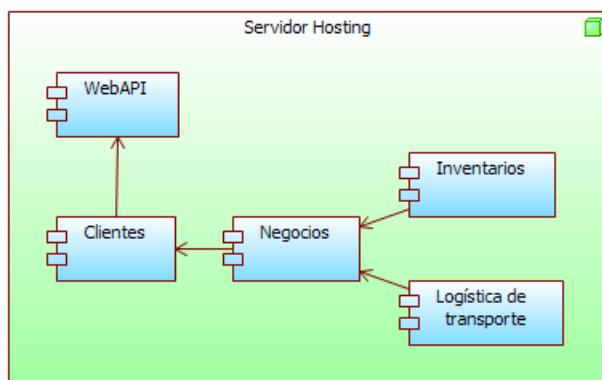
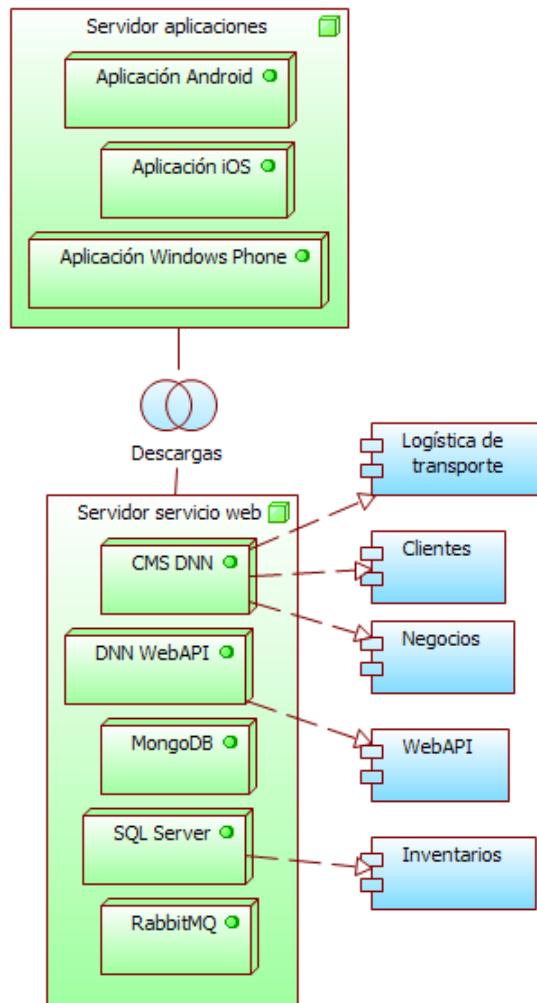


Figura 5-16.: Punto de Vista de Uso de Infraestructura.

### Punto de Vista de Implementación en la Organización

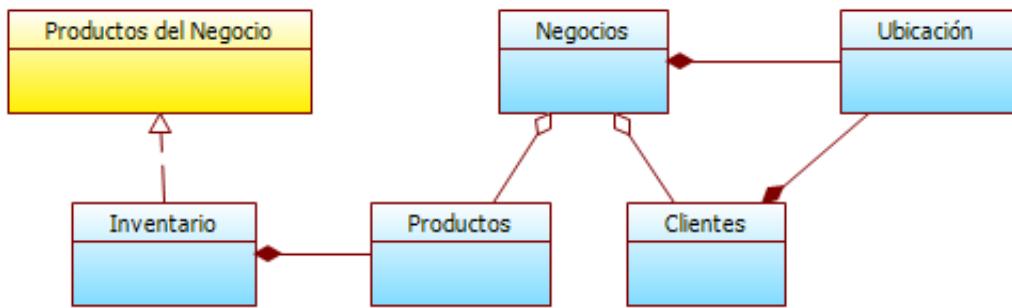
Una implementación real requiere de un servidor que tenga los componentes más específicos. A demás se puede observar como cada componente de infraestructura puede tener ninguno, uno o varios componentes de aplicación. Ver figura 5-17.



**Figura 5-17.:** Punto de Vista de Implementación en la Organización.

### Punto de Vista de Estructura de Información

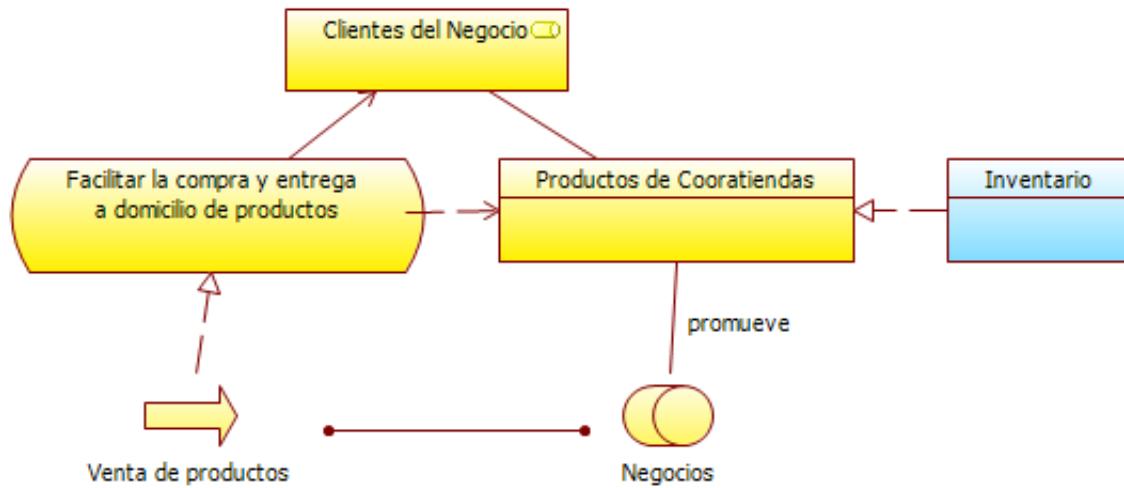
Se representan los objetos de base de datos y en este caso: productos de negocio sería el entregable. Ver figura 5-18.



**Figura 5-18.:** Punto de Vista de Estructura de Información.

#### Punto de Vista de Realización del Servicio

Este punto de vista facilita la compra y entrega a domicilio de productos. Ver figura 5-19.



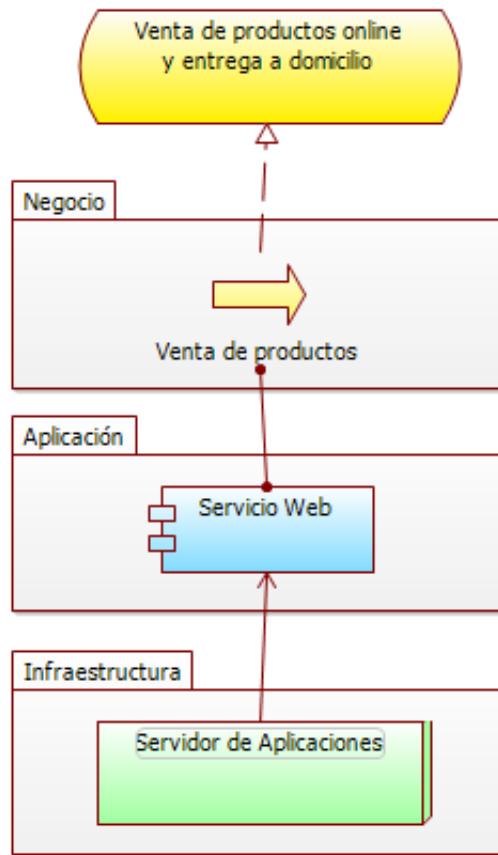
**Figura 5-19.:** Punto de Vista de Realización del Servicio.

#### Punto de Vista de Capas

Se pueden visualizar las tres capas de la infraestructura:

- La capa de negocio con su función más representativa la de Venta de productos.
- La capa de aplicación con su componente más representativo el servicio web.
- La capa de infraestructura con su nodo más representativo el servidor de aplicaciones.

Estas capas se pueden apreciar en la figura 5-20.



**Figura 5-20.: Punto de Vista de Capas.**

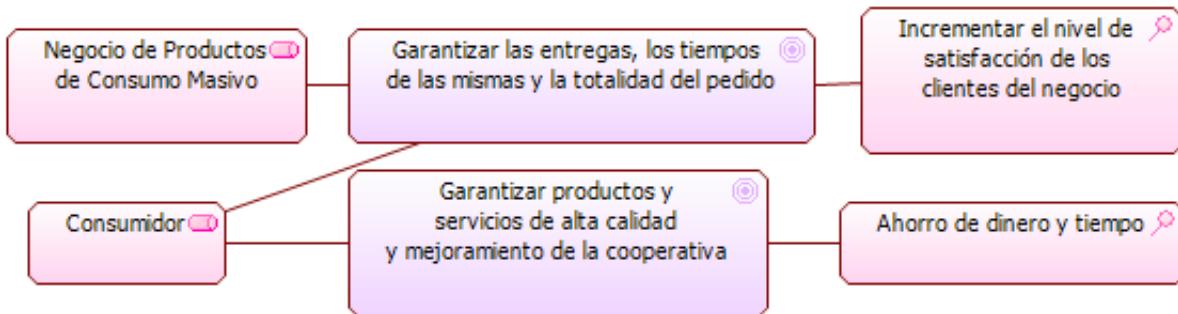
### 5.1.6. Fase E: Oportunidades y Soluciones

Ya habiendo realizado una aproximación de la capa de negocio, aplicación e infraestructura se analiza otra capa de lenguajes que es la de motivación y migración de la arquitectura.

El nivel motivacional se centra en los objetivos organizacionales a partir de los cuales se infieren los requerimientos. El idioma es organizacional ya que uno de los errores clásicos en esta capa es que se piense mucho en el terreno de software y acá en un lenguaje organizacional.

#### Punto de Vista de Interesados

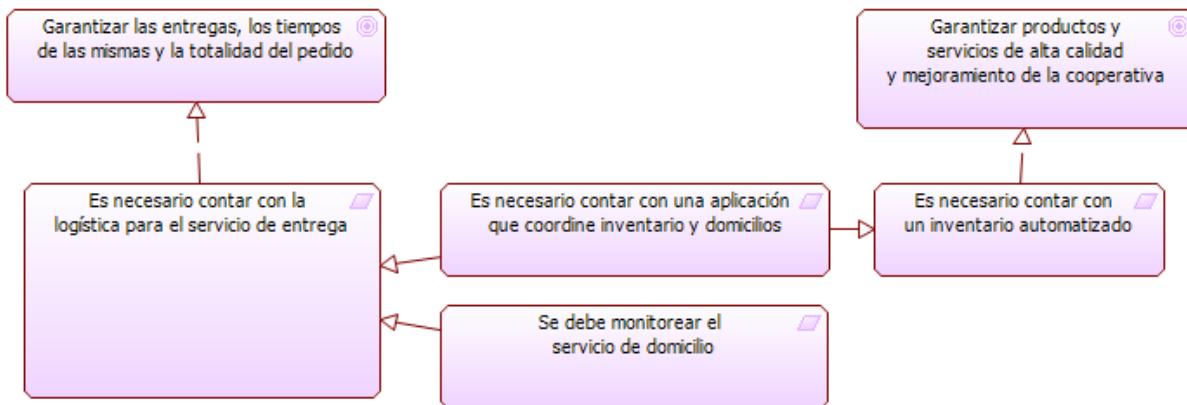
El consumidor y el negocio de productos de consumo masivo son dos de los interesados en garantizar las entregas, los tiempo y la calidad de los mismos, buscando incrementar el nivel de satisfacción de los clientes y ahorro de tiempo y dinero. Ver figura 5-21.



**Figura 5-21.:** Punto de Vista de Implicados.

#### Punto de Vista de Realización de Objetivos

En la figura 5-22 se puede apreciar los siguiente requerimiento:



**Figura 5-22.:** Punto de Vista de Realización de Objetivos.

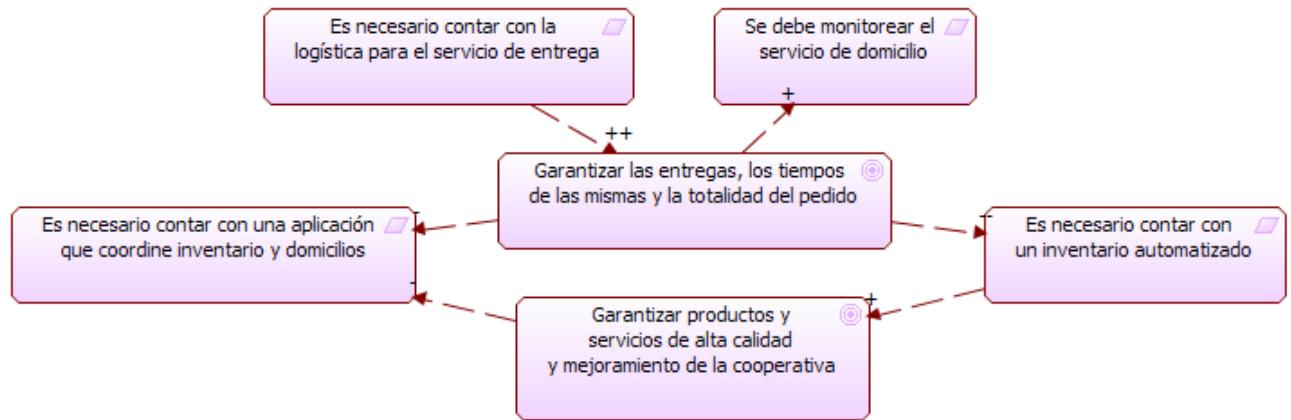
- Es necesario contar con la logística para el servicio de entrega.
- Es necesario contar con una aplicación móvil
- Se debe monitorear el servicio de domicilio
- Es necesario contar con un inventario automatizado

También se observan manejadores (en inglés Drivers):

- Garantizar las entregas, los tiempos de las mismas y la totalidad del pedido.
- Garantizar productos y servicios de alta calidad y mejoramiento de la cooperativa.

### Punto de Vista de Contribución de Objetivos

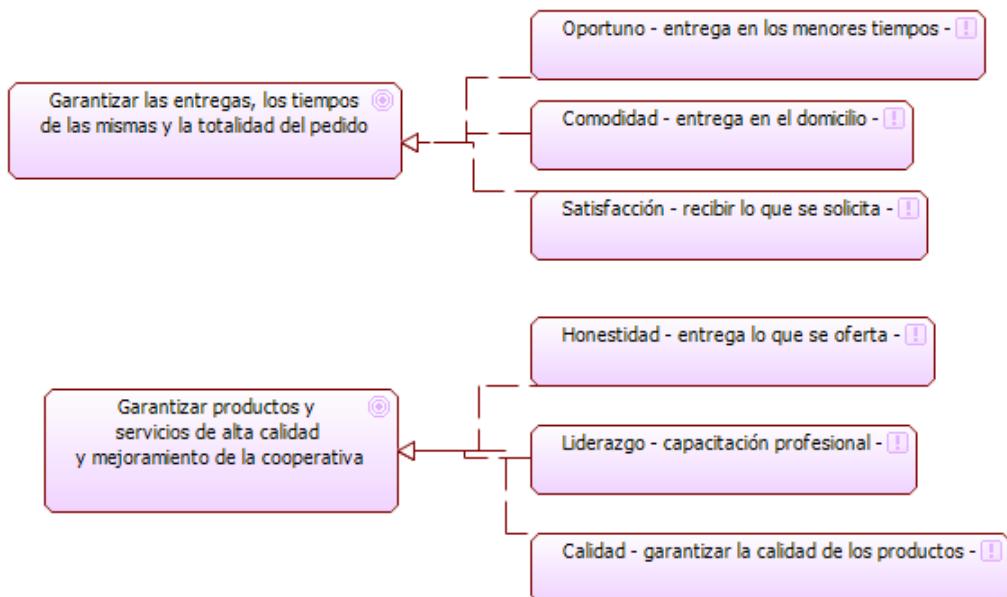
Se puede observar que influye demasiado y positivamente garantizar las entregas, los tiempos y la totalidad del pedido en los manejadores que proponen que es necesario contar con una logística y monitorear el servicio a domicilio. No se aprecian influencias negativas. Ver figura 5-23.



**Figura 5-23.:** Punto de Vista de Contribución de Objetivos.

### Punto de Vista de Principios

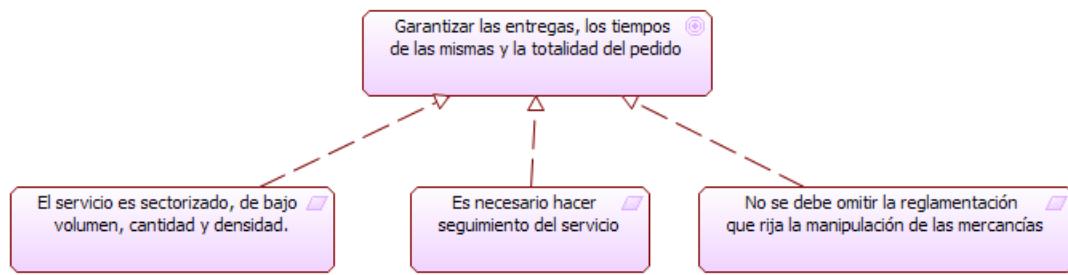
Los siguientes son los principales principios asociados a cada manejador de la arquitectura empresarial. Ver figura 5-24.



**Figura 5-24.:** Punto de Vista de Principios.

**Punto de Vista de Realización de Requerimientos**

Llevar a cabo los requerimientos realizan el manejador presentado en la figura 5-25.



**Figura 5-25.:** Punto de Vista de Realización de Requerimientos.

### 5.1.7. Fase F: Planear la Migración

Los siguientes conceptos dicen cómo evolucionan el siguiente conocimiento, determinan cómo progresan e indican el horizonte hacia donde vamos o que deseamos alcanzar.

- **Paquete de trabajo:** Es una serie de acciones diseñadas para conseguir un único objetivo del cual sale el liberable. Es una gran abstracción de alto nivel reuniendo un concepto más grande como una actividad.
- **Liberable:** Tiene forma sinusoidal formado por mesetas y entre ellas las brechas.
- **Meseta:** Son los hitos a alcanzar.
- **Brecha:** Son los obstáculos o problemas a resolver.

#### Punto de Vista del Proyecto

En este punto de vista se propone como objetivo del proyecto, el objetivo smart propuesto en la fase A de Visión de Arquitectura, el cual dice: “Aumentar el nivel de re-compra de productos 3 veces mediante la venta de productos online y entrega a domicilio, midiendo la periodicidad de compra mensualmente durante el 2017.” Ver figura 5-26.

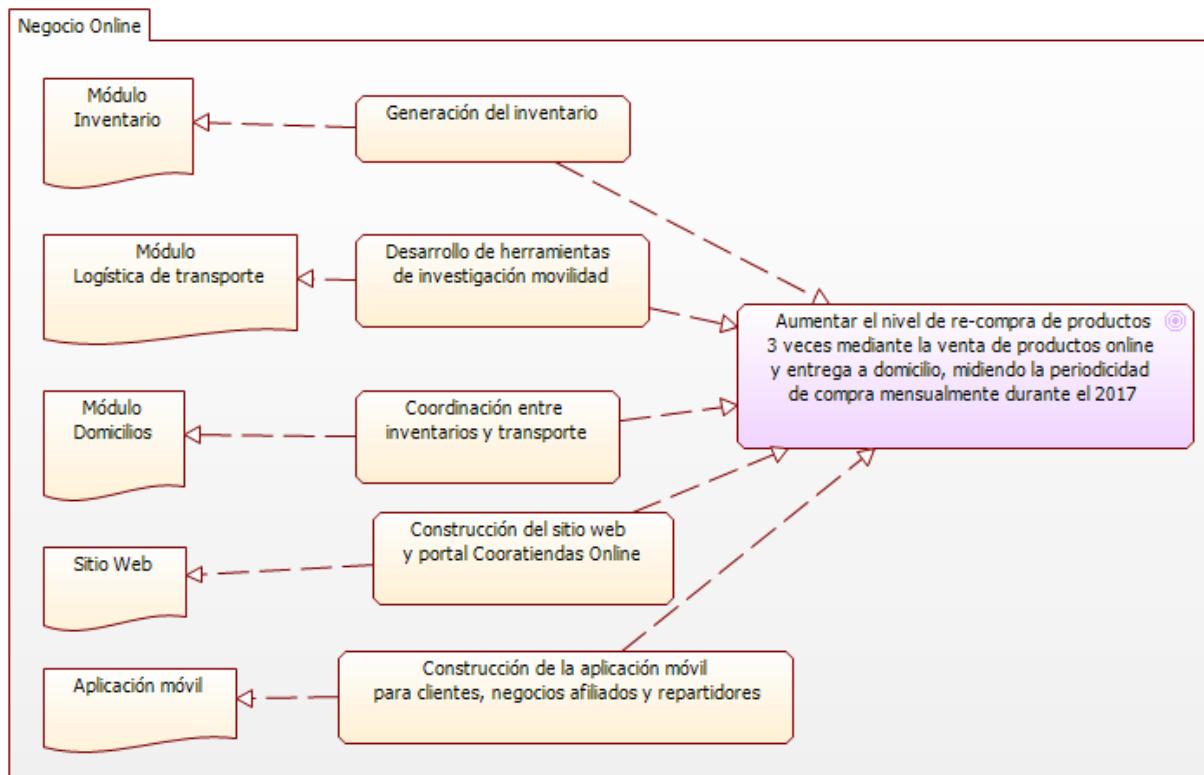


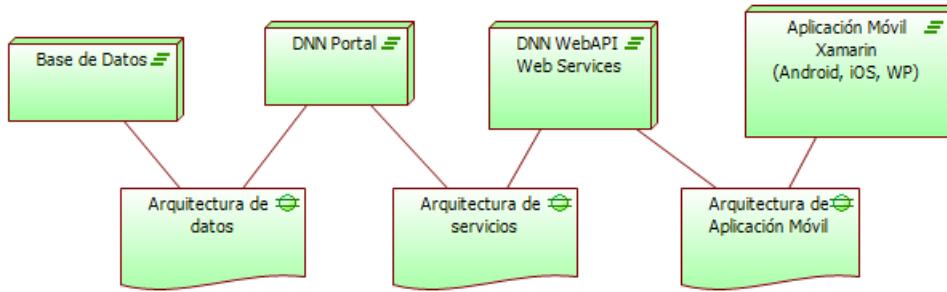
Figura 5-26.: Punto de Vista del Proyecto.

#### Punto de Vista de la Migración

Partiendo de la base de datos para llegar a tener un portal DNN se debe haber diseñado o propuesto la arquitectura de datos a seguir. Para llegar a crear el servicio web debe haberse diseñado la arquitectura de

servicios.

Finalmente para llegar a tener la aplicación o el portal web se debe haber diseñado la arquitectura de la aplicación móvil. Ver figura 5-27.



**Figura 5-27.: Punto de Vista de la Migración.**

#### Punto de Vista de la Implementación de la Migración

Finalmente, en esta vista se presenta una dirección URL como el lugar o ubicación de acceso al portal web de la solución con el objetivo de cumplir o alcanzar el objetivo estratégico mediante el escalamiento de meseta en meseta de los diagramas de migración habiendo superado las brechas. Todo lo anterior nos lleva a construir un portal web y una aplicación móvil que ofrezcan el servicio de venta online a domicilio. Ver figura 5-28.

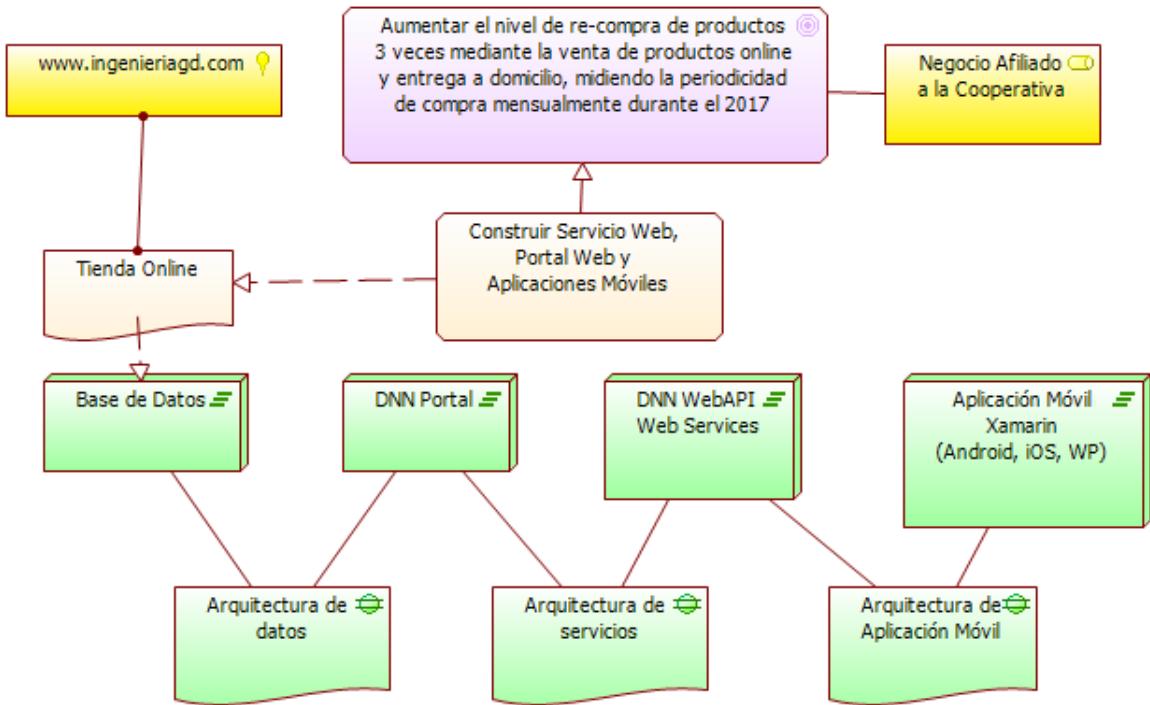


Figura 5-28.: Punto de Vista de la Implementación de la Migración.

## 5.2. Desarrollo e Implementación

A partir de los puntos de vista de la arquitectura de datos y aplicación (fase C) y de la infraestructura (fase D) del ciclo ADM en el ejercicio de arquitectura empresarial de la sección anterior, se presenta las técnicas que permiten desarrollar e implementar un servicio web completo.

En esta sección se presentan porciones de código utilizado en la creación del servicio web completo, presentar y explicar todo el código no tiene valor agregado al objetivo de este documento.

En este documento para el desarrollo e implementación del servicio web se plantean los siguientes alcances:

- Utilizar las tecnologías seleccionadas para su desarrollo e implementación.
- Presentar la forma de exponer métodos que realizan las cuatro operaciones básicas de lectura, creación, actualización y eliminación.
- Validar y verificar parámetros de entrada a los métodos.
- Autorizar y autenticar el acceso a los métodos.
- Crear una prueba unitaria sobre alguno de los métodos.

### 5.2.1. Creando el Servicio Web

El servicio web se desarrolla utilizando Web API 2. Luego de varias pruebas se llega a la conclusión que es mejor utilizar Web API a través del CMS Dot Net Nuke quien tiene integrada esta librería pero además cubre temas de administración, autenticación, seguridad y otros que ahoran drásticamente tiempo de desarrollo e implementación.

Lo primero antes de comenzar es **importante tener instalado Dot Net Nuke 7.4.2<sup>1</sup>** que permite administrar a través de un portal los módulos<sup>2</sup>.

Es importante mencionar que Dot Net Nuke corre sobre Internet Information Server [65] en plataformas con sistemas operativos Windows pero también puede correr sobre OWIN [66] en otras plataformas con código abierto como el sistema operativo Linux.

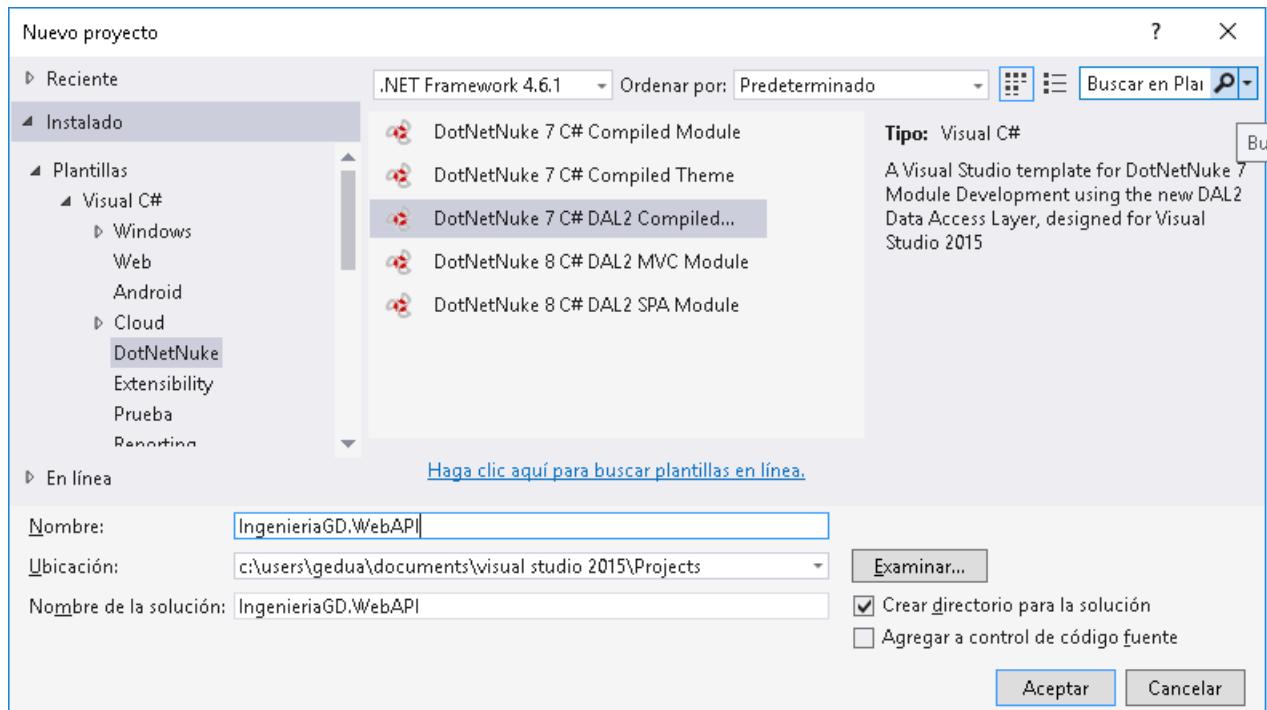
Una vez instalado y corriendo Dot Net Nuke se procede a crear el instalador del módulo de servicio web. Para esto utilizamos Visual Studio, creando un proyecto de tipo DNN. Ver figura 5-29. Para que aparezca este tipo de proyecto en Visual Studio es necesario descargar e instalar las plantillas<sup>3</sup> de DNN.

---

<sup>1</sup>Puede aprender a instalar DNN con la documentación del sitio web oficial: <http://www.dnnsoftware.com> o visitando: <http://www.dnnsoftware.com/community/download/manuals>

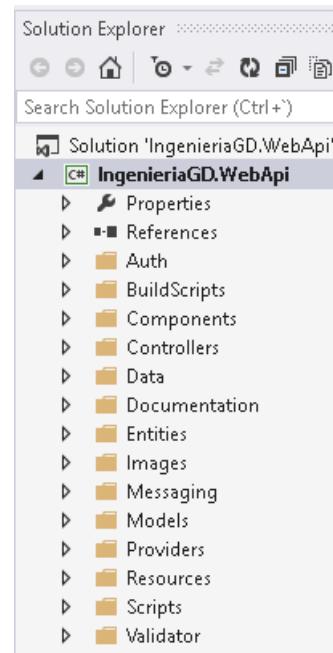
<sup>2</sup>Administración de módulos: instalación, actualización, modificación, versionamiento y desinstalación.

<sup>3</sup>Descargar y utilizar la plantilla para crear módulos DNN en Visual Studio es muy sencillo, para aprender puede visitar: <https://visualstudiogallery.msdn.microsoft.com/bdd506ef-d5c3-4274-bf1d-9e673fb23484>



**Figura 5-29.:** Creando en Visual Studio un Proyecto de Módulo DNN. 2016

Una vez creado el proyecto en Visual Studio se debe asegurar de crear la siguiente estructura (Ver figura 5-30) donde se pueden ver las carpetas recomendadas que debe tener el proyecto.



**Figura 5-30.:** Estructura Recomendada Proyecto Módulo DNN para el Servicio Web.

- **Auth:** Contiene las clases de autorización y autenticación.
- **Components:** Contiene las clases que configuran las rutas de acceso al servicio web.
- **Documentation:** Contiene la documentación del módulo.
- **Controllers:** Contiene las clases con los controladores y dentro de estos los métodos que se exponen en el servicio.
- **Entities:** Contiene todos los modelos de datos y clases generados automáticamente para interactuar con la base de datos.
- **Models:** Contiene las clases que son necesarias crear a partir de las clases generadas automáticamente por Entity Framework.
- **Providers:** Contiene los archivos de script de base de datos de instalación y desinstalación del módulo.
- **Scripts:** Contiene los archivos de script de PowerShell que se ejecutan al instalar y desinstalar el módulo.
- **Validator:** Contiene las clases de validación de clases entidad.

El resto de carpetas no mencionadas en el listado las crea automáticamente la plantilla del proyecto DNN y cumplen una función específica en la creación del módulo que no viene al caso mencionar pero que puede consultarse en el sitio web oficial del creador de la plantilla<sup>4</sup>.

Se detallará la mejor práctica para configurar el enrutamiento dentro de la documentación de Web API 2 y como exponer métodos, recomendados para el servicio web propuesto.

### 5.2.2. Ruta de Acceso o Enrutamiento

La ruta o URI para acceder al servicio web se recomienda configurarla como muestra en el código 5.1

En la carpeta **Components** se debe crear una clase **WebApiConfiguration.cs**. Esta clase debe heredar de **IServiceRouteMapper**. Basta con un constructor y un método implementado desde la clase heredada.

El método público **RegisterRouter** es el encargado de mapear el acceso al servicio web. El nombre del servicio se puede configurar en la línea 12. Por lo tanto para poder acceder a los métodos del controlador del servicio web instalado en DNN la sintaxis es la siguiente:

**[Ruta acceso a DNN]\Desktop\[Nombre del servicio]\[Controlador]\[Método]**

Ejemplo: `http://www.ingenieriad.net/DesktopModules/WebServiceCompleted/UserRegister/Post`  
 Notemos que en la línea 14 encontramos el orden en el que queremos que quede configurado el acceso a cada controlador (**controller**), cada método (**action**) y los parámetros que deseemos enviar (**id**), como se verá en el siguiente párrafo.

**Listing 5.1: WebApiConfiguration.cs**

```

1   public class WebApiConfiguration : IServiceRouteMapper
2   {
3       static WebApiConfiguration()
4       {
5           GlobalConfiguration.Configuration.Services.Replace(typeof(IHttpActionInvoker),
6                                                       new WebApiControllerActionInvoker());
7       }
8   }
  
```

<sup>4</sup>Plantilla DNN: <https://visualstudiogallery.msdn.microsoft.com/bdd506ef-d5c3-4274-bf1d-9e673fb23484>

```

6         GlobalConfiguration.Configuration.MessageHandlers.Add(new
7             LoggingTraceHandler());
8     }
9
10    public void RegisterRoutes(IMapRoute mapRouteManager)
11    {
12        mapRouteManager.MapHttpRoute(
13            "Mobility",
14            "default",
15            "{controller}/{action}/{id}",
16            new
17            {
18                action = RouteParameter.Optional,
19                id = RouteParameter.Optional
20            },
21            new[] { "IngenieriaGD.DNN.Modules.WebApi.Controllers" });
22    }

```

---

### 5.2.3. Métodos

Antes de comenzar la implementación del método se tuvo en cuenta los siguientes pasos cuando se consume un método:

1. Enviar la solicitud (URL, tipo de método, encabezado de autorización, parámetros).
2. Obtener tipo de método: GET, PUT, POST o DELETE.
3. Verificar Autenticación.
4. Verificar Autorización.
5. Validar y Verificar Parámetros de Entrada (Si es GET van en la URL, si es de otro tipo van en el BODY (no importa el formato del mensaje: XML o JSON)).
6. Realizar los procesos propios del método.
7. Controlar las excepciones.
8. Retornar una respuesta que indique si la respuesta a la solicitud fue exitosa o si falló ya sea por falta de un parámetro, dirección incorrecta, problema interno del servidor, autorización o autenticación no superadas o por cualquier otro tipo de excepción. Las respuestas se deben entregar en códigos HTTP Response Code.

Una vez configurada la ruta de acceso a los controladores y métodos para crear un controlador que contenga los métodos basta con crear una clase en la carpeta **Controllers** y heredar la clase de **DnnApiController** de la cual no hay que implementar nada.

En el código de la clase **UserController.cs** (Ver código 5.2) se observa como se ha creado un método de tipo **HttpResponseMessage** con nombre de método **SessionLogin** el cual se encarga de recibir como parámetro la información del dispositivo de tipo **Device**.

**Listing 5.2: UserController.cs**

```

1   public class UserController : DnnApiController

```

```

2      {
3          public HttpResponseMessage SessionLogin([FromBody]Device deviceInfo)
4          {
5          }
6      }

```

---

### Asignar Verbo a los Métodos

Cada método puede tener su correspondiente verbo: GET, POST, PUT o DELETE.

- **GET:** Es un método que puede recibir o no parámetros para “obtener” o devolver información.
- **PUT:** Es un método que puede recibir o no parámetros para “poner”, crear o insertar información.
- **POST:** Es un método que puede recibir o no parámetros para .<sup>enviar</sup>, modificar o actualizar información.
- **DELETE:** Es un método que puede recibir o no parámetros para .<sup>eliminar</sup>.<sup>o</sup> borrar información.

Para indicar que tipo de método es el método expuesto basta con agregar un decorador con el atributo del tipo de verbo del método. Por ejemplo, si el método es de tipo GET basta con colocar encima del método **[HttpGet]**. Para el método del ejemplo (Ver código 5.3) se agregaría **[HttpPost]**.

**Listing 5.3: UserController.cs**

```

1  [HttpPost]
2  public class UserController : DnnApiController
3  {
4      public HttpResponseMessage SessionLogin([FromBody]Device deviceInfo)
5      {
6      }
7  }

```

---

Con lo anterior podemos garantizar que el método quede expuesto y que además cumpla una operación de acuerdo al verbo asignado.

### Autenticación o Acceso a Métodos

A cada usuario registrado en el portal DNN se le puede configurar si está o no autenticado para acceder al portal. Este permiso de autenticación aplica en los métodos que tengan un atributo específico. Los atributos se ubican encima de la firma del método, fuera de este.

- Para autenticar un **Acceso Anónimo** se utiliza el atributo **[DnnAuthorize(AllowAnonymous=true)]**.
- Para autenticar a **Usuario Administrador** se utiliza el atributo **[DnnAuthorize(RequireHost=true)]**. (Ver código 5.4)
- Para autenticar a **Un Grupo de Usuario que Pertenezcan al Mismo ROL** se utiliza el atributo **[DnnAuthorize(StaticRoles = ".^ppsRol")]**.

Como se aprecia, es sencillo validar el acceso mediante la autenticación del usuario que realiza la petición a un método. El mismo framework de Web API junto con DNN se encarga de entregar una respuesta en caso de que la autenticación sea fallida, ahorrando tiempo de codificación y pruebas.

#### Listing 5.4: UserController.cs

```

1   [HttpPost]
2   [DnnAuthorize(RequireHost=true)]
3   public class UserController : DnnApiController
4   {
5       // ...
6   }

```

### Respuesta de los Métodos

Esta es tal vez la parte más sencilla de desarrollar ya que luego de pasar por cada una de las líneas del código del método, ejecutando la funcionalidad del método, si llega hasta la línea de respuesta es por que el resultado ha sido exitoso.

Las respuestas se basan en los códigos de respuesta del protocolo HTTP. Por citar algunos ejemplos, si la respuesta es exitosa el código devuelto será un 200, si los parámetros en la solicitud no vienen o están incompletos el código de respuesta será un 400. Se recomienda repasar en el marco teórico los códigos de respuesta del protocolo HTTP.

La respuesta se da utilizando se puede observar en la línea 9 del código 5.5. **Request** es un método del ApiController de Web API que se usa en conjunto con un método de extensión **CreateResponse** de la clase de extensiones HttpRequestMessageExtensions de Web API.

Si en la respuesta se desea devolver información, en el método **CreateResponse** se puede pasar como parámetro el objeto (Ejemplo: ObjectInfo) o variable que se desea enviar como respuesta a la solicitud.

#### Listing 5.5: UserController.cs

```

1   [HttpPost]
2   [DnnAuthorize(RequireHost=true)]
3   public class UserController : DnnApiController
4   {
5       public HttpResponseMessage SessionLogin([FromBody]Device deviceInfo)
6       {
7           // ....
8
9           return this.Request.CreateResponse(HttpStatusCode.OK, ObjectInfo);
10      }
11  }

```

El objeto (ObjectInfo) devuelto en la respuesta puede llegar a contener información que la aplicación móvil o quien haya consumido el método, requiera. El formato de visualización de su respuesta dependerá del parámetro de la cabecera **Content-Type**<sup>5</sup> si es **application/json** o **application/xml**. El siguiente código

<sup>5</sup>Este parámetro permite que Web API a través de DNN entregue el formato deseado por el usuario evitando drásticamente tener que realizar desarrollos o componentes que devuelvan formatos específicos.

5.6 muestra la respuesta en formato JSON.

#### Listing 5.6: ResponseJSON

```

1 {
2   "ObjectInfo": {
3     "Name": "Gabriel Duarte",
4     "SessionLoginDate": "2016-05-17T20:42:50.4565217Z"
5 }
```

### 5.2.4. Seguridad Autorización de Solicituds

El servicio web utiliza el protocolo HTTP esto significa que cualquier dispositivo o software que utilice este protocolo puede consumir el servicio y obtener una respuesta, enviando una solicitud sencilla como la del siguiente ejemplo:

**URI:** http:\\host\\service\\operationName HTTP\\1.1

**Verb:** POST

**Content-Type:** application\\json; charset=utf-8

**Bode:** “data”：“Aus Lore ipsum lorem aus Lore ipsum lorem”

Partiendo de las siguientes preguntas:

- ¿Qué dispositivo realiza la solicitud?
- ¿Qué aplicación realiza la solicitud?
- ¿Qué usuario de la aplicación está realizando la solicitud?
- ¿El usuario ha concedido autorización a la aplicación para solicitar la información?
- ¿Los datos de la solicitud han sido manipulados por un tercero, mientras estaban en tránsito hacia el servidor?
- ¿Esta solicitud ya había sido procesada?

Para aplicar seguridad al servicio web, la información de acceso (autenticación) y los permisos sobre los métodos (autorización) deben enviarse en el encabezado de la solicitud.

Además para evitar vulnerabilidades en el servicio, como el robo o captura de información sensible, la información de cada uno de los campos debe ir cifrada.

Finalmente con el objetivo de cumplir las recomendaciones de estándares como PCI-DSS [67] y recomendaciones como las de OWASP TOP 10 [68] se firma el encabezado enviado con el objetivo de verificar su validez.

Resumiendo, los métodos del servicio podrán ser consumidos si y sólo si:

- El encabezado lleva el campo: “Authorization”.
- La información concatenada en el campo: “Authorization” va cifrada o concatenada según se defina.
- Se firma el campo: “Authorization” mediante la encriptación del mismo con algún sistema criptográfico indicado o definido.

### Formato de la Cabecera Authorization

La estructura del campo **Authorization**<sup>6</sup> en la cabecera es una cadena de texto (string) separada por comas compuesta de elementos clave/valor. La parte a la izquierda del igual corresponde a la clave, y la que se encuentra a la derecha es el valor.

**Parámetros en la cabecera Authorization** El nombre que se le da a cada clave en la cabecera debe ser distinto al que se sugiere por seguridad. Además debe ser definido por cada organización de acuerdo a sus políticas. A continuación se describen cada par de clave/valor a enviar en el campo **Authorization** de la cabecera

- **app:** Identifica la aplicación desde la cual se realiza la solicitud. Este valor se genera al momento de registrar la aplicación en el servicio web. Es requerido para todas las operaciones hacia el servicio. Ejemplo: app="67cf6bcd82fc4a5ead4c"
- **user:** Identifica el usuario que realiza la solicitud. Este valor se genera al momento de registrar el usuario en el servicio web. Ejemplo. user="67cf6bcd82fc4a5ead4c"
- **nonce:** Valor único que la aplicación debe generar para cada solicitud. El servicio web utilizará este valor para determinar si la solicitud ha sido enviada varias veces. Se puede utilizar cualquier implementación que produzca una cadena alfanumérica aleatoria y única para cada petición. Es requerido en todas las operaciones del servicio. Ejemplo: nonce="L4bKDTwdLe2H0"
- **timestamp:** Corresponde al número de milisegundos transcurridos desde el primero de enero de 1970 a las 00:00:00 hasta la fecha y hora en que se genera la solicitud (tiempo UTC). El servicio web rechazará las solicitudes que se hayan creado demasiado lejos en el pasado. Es requerido en todas las operaciones del servicio. Ejemplo: timestamp="1363621133230"
- **sign:** Contiene un valor que se genera mediante la aplicación del algoritmo de reducción SHA512 a todos los demás parámetros de la solicitud. El propósito de esta firma es: Verificar que la solicitud no ha sido modificada en tránsito; Verificar la autenticidad de la aplicación que envía la solicitud; Verificar que el usuario es quien inicia la solicitud en la aplicación; Más adelante en este documento se describe el proceso de generación de una firma.

Una solicitud autorizada es aquella que en la cabecera en el campo “Authorization” lleva concatenados los parámetros de autorización definidos y firmados. El siguiente es un ejemplo de una solicitud autorizada:

**Uri:** POST http:\\host\\service\\operationName HTTP\\1.1

**Content-Type:** application\\json; charset=utf-8

**Authorization:** app="67cf6bcd82fc4a5ead4c",user="67cf6bcd82fc4a5ead4c",nonce="L4bKDTwdLe2H0", timestamp="1363621133230",sign="2f5f47257648e1d2322b4990c3030a00fff8843c"

**Proceso de generación de la firma** El proceso de generación de la firma requiere de definir un sistema criptográfico: MD5, SHA1, SHA256, SHA512. El más recomendado es SHA512 pero depende de las políticas de seguridad que usted requiera. Para este ejemplo se utilizará SHA1.

---

<sup>6</sup>Cada solicitud HTTP puede llevar en el encabezado un campo de nombre: Authorization el cual va acompañado de un valor

La generación de la firma consiste en tomar los valores de cada parámetro en la solicitud, sin incluir el parámetro **sign** y concatenarlos en una cadena de texto y luego calcular el SHA1.

Para el ejemplo anterior la cadena concatenada sería:

”67cf6bcd82fc4a5ead4c67cf6bcd82fc4a5ead4cL4bKDTwdLe2H01363621133230”

El SHA1 de esa cadena es:

”2f5f47257648e1d2322b4990c3030a00fff8843c”

Por lo tanto esta firma deberá ser igual a la que viene en el parámetro **sign** del campo **Authorization** en la cabecera de la solicitud HTTP.

**Código de Autorización de un Método** Para validar los parámetros en el encabezado de la solicitud HTTP es importante crear una clase que herede de **AuthorizeAttributeBase** e implemente el método **IsAuthorized** en el cual se puede realizar la validación de los parámetros. (Ver código 5.7)

La validación se puede implementar tan sencillo como sobre escribir el método **IsAuthorized** el cual recibe como parámetro un objeto de tipo **AuthFilterContext** el cual contiene el encabezado de la solicitud. Ver línea 4 del código y con los comentarios son dicientes para comprender lo que se está haciendo.

**Buenas prácticas:** En el código se utilizan las siguientes técnicas:

- Las cadenas de texto en el código que representen títulos o nombres claves de campos o “keys” se almacenan en un archivo de recursos, nunca se “queman” o escriben como cadenas de texto directamente en el código. Para acceder al archivo de recursos se utiliza el código: **Resources.[Key del Recurso]**<sup>7</sup>
- Se utiliza mucho LINQ<sup>8</sup> para ahorrar grandes cantidades de código y hacer mas fácil de mantener el código<sup>9</sup>.
- Se manejan y controlan todo tipo de excepciones, cada una con su correspondiente respuesta (texto del mensaje almacenado en archivos de recursos).

**Listing 5.7: ApiAuthorizeAttribute.cs**

```

1  public sealed class ApiAuthorizeAttribute : AuthorizeAttributeBase
2  {
3      public override bool IsAuthorized(AuthFilterContext filterContext)
4      {
5          try
6          {
7              Inumerable<string> authorizationHeader;
8              //El nombre de la clase se guarda en un recurso "Resources.
9              AuthorizationHeaderKey"
10             if (!filterContext.ActionContext.Request.Headers.TryGetValues(Resources.
11                 AuthorizationHeaderKey, out authorizationHeader)
12                 || authorizationHeader == null)
13             {
14                 return false;
15             }
16         }
17     }
18 }
```

<sup>7</sup>Ver como crear un archivo de recursos en: [https://msdn.microsoft.com/es-es/library/7zxb70x7\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/7zxb70x7(v=vs.110).aspx)

<sup>8</sup>Todo acerca de LINQ en: <https://msdn.microsoft.com/es-co/library/bb397926.aspx>

<sup>9</sup>Ver ejemplos de LINQ en: <https://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b>

```
11             || !authorizationHeader.Any())
12         {
13             filterContext.AuthFailureMessage = ((ErrorResponse)ResponseCode.
14                 MissingAuthHeader).Message;
15             return false;
16         }
17
18         filterContext.ActionContext.Request.Properties.Add(Resources.
19             AuthorizationHeaderKey, authHeader);
20         IAuthValidatorFluent validator = null;
21
22         if (validator != null)
23         {
24             //Los parametros de la solicitud se convierten en formato JSON para
25             //luego poder validarlos
26             string parameters = string.Empty;
27
28             if (filterContext.ActionContext.Request.Method != HttpMethod.Get)
29             {
30                 parameters = filterContext.ActionContext.Request.GetBody();
31             }
32
33             if (filterContext.ActionContext.Request.Method == HttpMethod.Get)
34             {
35                 IEnumerable<KeyValuePair<string, object>> parametersList =
36                     filterContext.ActionContext.Request.GetQueryNameValuePairs()
37                     .Select(x => new KeyValuePair<string, object>(x.Key, x.Value));
38                 Dictionary<string, object> dictionaryParameters = parametersList
39                     .ToDictionary(x => x.Key, x => x.Value);
40                 parameters = dictionaryParameters.Count <= 0 ? string.Empty :
41                     dictionaryParametersToJson();
42             }
43
44             //Se llaman los metodos del objeto IAuthValidatorFluent declarado con
45             //anterioridad
46             validator
47                 //Establece los parametros contra los que se debe validar la firma
48                 .WhereSignMatchWith(parameters)
49                 //Establece la fecha de la operacion contra la que se hace la
50                 //verificacion de expiracion
51                 .NotExpired()
52                 //Establece que un numero arbitrario sea utilizado solo una vez
53                 //en cada solicitud
54                 .EnsureNonce()
55                 //Genera el objeto que permite llevar a cabo la validacion
56                 .Build()
57                 //Genera una excepcion si la validacion de los parametros no es
58                 //exitosa
59                 .ThrowIfNotValid();
60
61             return true;
62         }
63
64         filterContext.AuthFailureMessage = ((ErrorResponse)ResponseCode.
65             InvalidAuthHeader).Message;
66         return false;
67     }
68     catch (OperationException exception)
69     {
70         filterContext.AuthFailureMessage = new ErrorResponse(ResponseCode.
```

```

59     InternalServerError, exception.Message).Message;
60     LogWriter.AddError(filterContext.AuthFailureMessage, exception);
61     DnnEventLogMsg.LogMsg(ResponseCode.InternalServerError.ToString(),
62     exception.Message);
63     return false;
64   }
65   catch (Exception exception)
66   {
67     filterContext.AuthFailureMessage = new ErrorResponse(ResponseCode.
68     InternalServerError, exception.Message).Message;
69     LogWriter.AddError(filterContext.AuthFailureMessage, exception);
70     DnnEventLogMsg.LogMsg(ResponseCode.InternalServerError.ToString(),
71     exception.Message);
72     return false;
73   }
74 }
```

### 5.2.5. Patrón Fluent

El código 5.7 utiliza un patrón de validaciones titulado **Interfaz Fluent** se usa para hacer el código más legible y fácil de mantener, el cual ha sido desarrollada por varios profesionales en la industria del software y cada uno publica y permite descargar la librería de este componente.

La validación "Fluent" consiste en crear una interfaz `IAuthValidatorFluent` la cual define un contrato con las propiedades y métodos a implementar donde se requiera realizar validaciones.

**Listing 5.8: IAuthValidatorFluent.cs**

```

1  public interface IAuthValidatorFluent
2  {
3    /// <summary>
4    /// Establece la fecha de la operacion contra la que se hace la verificacion de
5    /// expiracion
6    /// </summary>
7    /// <returns>Instancia de <see cref="IAuthValidatorFluent"/>.</returns>
8    IAuthValidatorFluent NotExpired();
9
10   /// <summary>
11   /// Establece que un numero arbitrario sea utilizado solo una vez por cada
12   /// solicitud
13   /// </summary>
14   /// <returns>Instancia de <see cref="IAuthValidatorFluent"/>.</returns>
15   IAuthValidatorFluent EnsureNonce();
16
17   /// <summary>
18   /// Establece los parametros contra los que se debe validar la firma
19   /// </summary>
20   /// <param name="parameters">Parametros contra los que se debe validar la firma
21   /// </param>
22   /// <returns>
23   /// Instancia de <see cref="IAuthValidatorFluent" />.
24   /// </returns>
25   IAuthValidatorFluent WhereSignMatchWith(params string[] parameters);
26
27   /// <summary>
```

```

25     /// Establece los controles de acceso contra los que se debe validar los
26     /// permisos
27     /// </summary>
28     /// <param name="accessValidation">Reglas de acceso de la petición.</param>
29     /// <returns>
30     /// Instancia de <see cref="IAuthValidatorFluent" />.
31     /// </returns>
32     IAAuthValidatorFluent WhereCanAccess(AccessRule accessValidation);
33
34     /// <summary>
35     /// Valida el Pin
36     /// </summary>
37     /// <param name="checkPin"><c>true</c> si se requiere validar Pin</param>
38     /// <returns>
39     /// Instancia de <see cref="IAuthValidatorFluent" />.
40     /// </returns>
41     IAAuthValidatorFluent WhereCheckPin(bool checkPin);
42
43     /// <summary>
44     /// Genera el objeto que permite llevar a cabo la validación
45     /// </summary>
46     /// <returns>Instancia de <see cref="AuthValidator"/> con los valores de
47     /// validación configurados</returns>
48     AuthValidator Build();
49 }

```

Luego de definir la interfaz se implementa en la clase AuthValidatorFluent que hereda de la interfaz IAAuthValidatorFluent como se explica en los comentarios del código 5.9. Una vez implementado ya se puede utilizar para validar uno o varios métodos donde se requiera como se observó en la clase ApiAuthorizeAttribute en la línea 38 del código 5.7.

**Listing 5.9: AuthValidatorFluent.cs**

```

1     /// <summary>
2     /// Implementar el validador de autenticación fluent.
3     /// </summary>
4     public class AuthValidatorFluent : IAAuthValidatorFluent
5     {
6         #region Fields
7
8         /// <summary>
9         /// Almacena la instancia sobre la que se lleva a cabo la validación de
10        /// parámetros.
11        /// </summary>
12        private readonly AuthValidator transaction;
13
14        /// <summary>
15        /// Reglas de acceso aplicadas.
16        /// </summary>
17        private AuthValidationRule accessRules = 0;
18
19        #endregion
20
21        #region AuthValidatorFluent
22
23        /// <summary>
24        /// Inicializa una nueva instancia de la clase <see cref="AuthValidatorFluent"
25        /// />.
26        /// </summary>

```

```
25     /// <param name="transaction">Instancia sobre la que se lleva a cabo la
26     /// validacion de parametros.</param>
27     public AuthValidatorFluent(AuthValidator transaction)
28     {
29         transaction.ThrowExceptionIfTrue<ArgumentException>(transaction == null, "
30             transaction");
31         this.transaction = transaction;
32     }
33
34     #endregion
35
36     /// <summary>
37     /// Establece la fecha de la operacion contra la que se hace la verificacion de
38     /// expiracion.
39     /// </summary>
40     /// <returns>
41     /// Instancia de <see cref="IAuthValidatorFluent" />.
42     /// </returns>
43     public IAuthValidatorFluent NotExpired()
44     {
45         this.accessRules |= AuthValidationRule.NotExpired;
46         return this;
47     }
48
49     /// <summary>
50     /// Establece que un numero arbitrario sea utilizado solo una vez por cada
51     /// solicitud
52     /// </summary>
53     /// <returns>
54     /// Instancia de <see cref="IAuthValidatorFluent" />.
55     /// </returns>
56     public IAuthValidatorFluent EnsureNonce()
57     {
58         this.accessRules |= AuthValidationRule.EnsureNonce;
59         return this;
60     }
61
62     /// <summary>
63     /// Establece los parametros contra los que se debe validar la firma.
64     /// </summary>
65     /// <param name="parameters">Parametros contra los que se debe validar la firma
66     /// .</param>
67     /// <returns>
68     /// Instancia de <see cref="IAuthValidatorFluent" />.
69     /// </returns>
70     public IAuthValidatorFluent WhereSignMatchWith(params string[] parameters)
71     {
72         this.accessRules |= AuthValidationRule.SignMatch;
73         this.transaction.Parameters = parameters;
74         return this;
75     }
76
77     /// <summary>
78     /// Establece los controles de acceso contra los que se debe validar los
79     /// permisos.
80     /// </summary>
81     /// <param name="accessValidation">Reglas de acceso de la peticion.</param>
82     /// <returns>
83     /// Instancia de <see cref="IAuthValidatorFluent" />.
84     /// </returns>
85     public IAuthValidatorFluent WhereCanAccess(AccessRule accessValidation)
```

```

80         {
81             this.accessRules |= AuthValidationRule.CanAccess;
82             this.transaction.AccessRules = accessValidation;
83             return this;
84         }
85
86         /// <summary>
87         /// Valida el Pin.
88         /// </summary>
89         /// <param name="checkPin"><c>true</c> si se requiere validar Pin</param>
90         /// <returns>
91         /// Instancia de <see cref="IAuthValidatorFluent" />.
92         /// </returns>
93         public IAuthValidatorFluent WhereCheckPin(bool checkPin)
94     {
95         if (checkPin)
96     {
97         this.accessRules |= AuthValidationRule.CheckPin;
98     }
99
100        return this;
101    }
102
103    /// <summary>
104    /// Genera el objeto que permite llevar a cabo la validacion.
105    /// </summary>
106    /// <returns>
107    /// Instancia de <see cref="AuthValidator" /> con los valores de validacion
108    /// configurados.
109    /// </returns>
110    public AuthValidator Build()
111    {
112        this.transaction.ValidationRule = this.accessRules;
113        return this.transaction;
114    }

```

### 5.2.6. Parámetros de los Métodos

Implementando el método para iniciar sesión que lo consume la aplicación móvil propuesta en la arquitectura desarrollada, esta envía como parámetro información relacionada con el dispositivo para poder controlar quien realiza la solicitud, independiente de si el método es consumido por un usuario autorizado o si supera la autorización del encabezado.

El siguiente código 5.10 presenta la clase que representa la entidad u objeto del dispositivo.

**Listing 5.10: Device.cs**

```

1  public class Device
2  {
3      [DataMember(IsRequired = false)]
4      public string Brand
5      {
6          get;
7          set;
8      }
9

```

```

10     [DataMember]
11     public string DeviceCode
12     {
13         get;
14         set;
15     }
16
17     [DataMember(IsRequired = false)]
18     public string Model
19     {
20         get;
21         set;
22     }
23 }
```

Con este objeto Device puedo esperar como parámetros de un método de inicio de sesión información, como se muestra en la firma del método del siguiente código 5.11.

**Listing 5.11:** SessionLogin.cs

```

1 [HttpPost]
2 [AllowAnonymous]
3 public HttpResponseMessage SessionLogin([FromBody]Device deviceInfo)
4 {
5     // ...
6 }
```

Se observa que acompañando a los parámetros de acuerdo a como lo sugiere la teoría de Web API 2 [60] se puede indicar si los parámetros vienen en el cuerpo (Body en inglés) de la solicitud utilizando el atributo **[FromBody]** o vienen en la URL acompañando la solicitud para lo cual se debe utilizar el atributo **FromUri**.

En el código se utiliza el atributo **[FromBody]** ya que en el cuerpo de la solicitud viene la información del dispositivo, representada por un JSON 5.12

**Listing 5.12:** BodyJSONSessionLogin.cs

```

1 {
2     "Platform": "Android",
3     "DeviceCode": "91001",
4     "Brand": "Samsung",
5 }
```

## Validación de los Parámetros

Para validar los métodos se propone utilizar la siguiente técnica:

Se crea una clase que almacenará las validaciones a realizar sobre los métodos, la cual heredará del objeto **TypeValidator** del componente NValidator que se descarga de Nuget. El **TypeValidator** recibe como tipo de dato el objeto a validar en este caso el mismo parámetro del método expuesto. Por ejemplo: la clase **Device**. Ver línea 1 del código 5.13.

**Listing 5.13:** DeviceValidator.cs

```
1 internal class DeviceValidator : TypeValidator<Device>
```

```

2 {
3     internal DeviceValidator()
4     {
5         this.RuleFor(dev => dev.Platform)
6             .NotNull()
7             .NotEmpty()
8             .AllWithMessage(dev => ResponseCode.InvalidPlatform.GetDescription());
9
10        this.RuleFor(dev => dev.DeviceCode)
11            .NotNull()
12            .Length(1, 100)
13            .AllWithMessage(dev => ResponseCode.InvalidDeviceCode.GetDescription());
14    }
15 }

```

En el constructor de la clase validadora, definimos una serie de reglas mediante métodos de extensión que **NValidator** nos ofrece.

Se puede validar si la propiedad del objeto que estoy validando viene vacía (.NotEmpty()), nula (.NotNull()), se encuentra en el rango de una longitud definida (.Length(1, 100)), si es mayor, menor o igual a un determinado número, etc.

Esta validación se instancia dentro del método expuesto. Ver línea 3 del código 5.11

**Listing 5.14:** SessionLogin.cs

```

1 public HttpResponseMessage SessionLogin([FromUri]DeviceInfo deviceInfo)
2 {
3     // Continua con las siguientes líneas si se superan las reglas de validación sobre el
4     // objeto que se le ha pasado
5     new DeviceValidator().ThrowExceptionIfNotValid(deviceInfo, ResponseCode.
6         InvalidDeviceInfo);
7 }

```

Acompañando a la clase de validación que se ha instanciado está un método de extensión **ThrowExceptionIfNotValid** que utiliza las reglas de la clase validadora para generar o no una excepción y permitir continuar o no con las siguientes líneas del código del método. Los comentarios en el código permiten aclarar que parámetros recibe y dentro del código se puede observar como maneja la excepción si las reglas de validación del objeto a validar fueron superadas. Ver código 5.15.

**Listing 5.15:** ThrowExceptionIfNotValid.cs

```

1 /// <summary>
2 /// Lanza una excepción si la validación falla.
3 /// </summary>
4 /// <typeparam name="T">Tipo de objeto a validar</typeparam>
5 /// <param name="validator">Validador a usar.</param>
6 /// <param name="validationObject">Objeto a validar.</param>
7 /// <param name="errorCode">Código de error a utilizar si la validación falla.</param>
8 /// <exception cref="OperationException">Excepción generada si la validación falla.</
9 exception>
10 public static void ThrowExceptionIfNotValid<T>(this TypeValidator<T> validator, T
11 validationObject, ResponseCode errorCode)

```

```

10 {
11     if (!validator.IsValid(validationObject))
12     {
13         throw new OperationException(new ErrorResponse(
14             errorCode,
15             string.Join(Resources.ValidationMessageSeparator, validator.
16                 GetValidationResult(validationObject).Select(item => item.Message).
17                 Distinct())));
18     }
19 }
```

---

### 5.2.7. CRUD de Métodos Expuestos

A continuación se muestra el desarrollo de una parte de la solución del ejercicio de arquitectura empresarial. Este desarrollo utiliza las técnicas expuestas en las secciones anteriores.

Las aplicaciones móviles, portales web y en general cualquier software o dispositivo que consuman el servicio de movilidad requieren de poder registrarse como una aplicación. El registro garantiza que se pueda identificar como confiable la aplicación que consume otras solicitudes.

Para conseguir esto se **creó una controladora de aplicaciones** en la cual se codificaron métodos que permiten crear, solicitar, modificar o eliminar información de las aplicaciones, como se verá a continuación.

Nota: El código dentro de la controladora se divide por cada método para poder ir dando una reseña de lo que hace. El código completo de la clase se puede observar completo en el Anexo F

#### Controladora de Aplicaciones

En la carpeta **Controllers** se creó la clase **ClientsController.cs** (Ver encabezado y herencia de la clase en el código 5.16) la cual contiene cuatro métodos.

**Listing 5.16:** AppController.cs

```

1 public class AppController : DnnApiController
2 {
3     // Aca: Lineas de codigo de los metodos
4 }
```

---

#### Método GET sin parámetro

El siguiente es un fragmento de código (Ver código 5.17) de la clase que representa el método que devuelve información sin necesidad de pasar parámetros. Nótese los atributos de autenticación y autorización, las validaciones y la respuesta. El método devuelve información que extrae de la base de datos.

**Listing 5.17:** AppController.cs

```

1 /// <summary>
2 /// Obtiene la informacion de todas las aplicaciones .
3 /// </summary>
4 /// <returns>Resultado de la operacion</returns>
5 public HttpResponseMessage Get()
```

---

```

6  {
7      I Enumerable<AppInfo> appsInfo = AppRepository.Get();
8      return this.Request.CreateResponse(HttpStatusCode.OK, appsInfo.ToList());
9 }

```

---

### Método GET con parámetro

El método que se muestra en el código 5.18 recibe parámetros individuales que vienen en la URL de la solicitud. Nótese los atributos de autenticación y autorización, las validaciones y la respuesta. El método devuelve información que extrae de la base de datos y la filtra mediante LINQ.

**Listing 5.18: AppController.cs**

```

1 /// <summary>
2 /// Obtiene la informacion de la aplicacion a partir del identificador.
3 /// </summary>
4 /// <param name="appId">Identificador de la aplicacion.</param>
5 /// <returns>Resultado de la operacion</returns>
6 public HttpResponseMessage Get(Guid appId)
7 {
8     appId.ThrowOperationIfTrue(!appId.ValidGuid(), ResponseCode.
9         InvalidApplicationGuid);
10    AppInfo appInfo = AppRepository.Get(appId);
11    return this.Request.CreateResponse(HttpStatusCode.OK, appInfo);
12 }

```

---

### Método PUT

El método que se muestra en el código 5.19 recibe varios parámetros que están representados por la clase **AppInfo** los cuales vienen en el **Body** de la solicitud. Nótese los atributos de autenticación y autorización, las validaciones y la respuesta. El método envía la información de los parámetros a la base de datos para almacenarla.

**Listing 5.19: AppController.cs**

```

1 /// <summary>
2 /// Actualiza la informacion de la aplicacion.
3 /// </summary>
4 /// <param name="request">Informacion de la aplicacion.</param>
5 /// <returns>Resultado de la operacion</returns>
6 public HttpResponseMessage Put([FromBody]AppInfo request)
7 {
8     new AppValidator().ThrowExceptionIfNotValid(request, ResponseCode.
9         InvalidApplicationInfo);
10    AppRepository.Update(request);
11    return this.Request.CreateResponse(HttpStatusCode.OK);
12 }

```

---

### Método POST

El método que se muestra en el código 5.20 recibe varios parámetros que están representados por la clase **AppInfo** los cuales vienen en el **Body** de la solicitud. Nótese los atributos de autenticación y autorización,

las validaciones, la actualización de los datos a modificar y la respuesta. El método envía la nueva información a la base de datos para actualizarla.

#### Listing 5.20: AppController.cs

```

1  /// <summary>
2  /// Registra una nueva aplicacion.
3  /// </summary>
4  /// <param name="request">Informacion de la aplicacion.</param>
5  /// <returns>Resultado de la operacion</returns>
6  /// <remarks>Como el CAST del body se hace a traves de los que ofrece WebApi
7  /// si se envia un GUID invalido el CAST lo asumira como Empty y se creara con GUID
8  /// autogenerado.</remarks>
9  public HttpResponseMessage Post([FromBody]AppInfo request)
10 {
11     new AppValidator().ThrowExceptionIfNotValid(request, ResponseCode.
12         InvalidApplicationInfo);
12     Guid appId = AppRepository.Save(request);
13     return this.Request.CreateResponse(HttpStatusCode.Created, appId);
14 }
```

#### Método DELETE

El método que se muestra en el código 5.21 recibe un único parámetro que representa el dato a eliminar. Este parámetro viene en la URL de la solicitud. Nótese los atributos de autenticación y autorización, las validaciones y la respuesta. El método invoca el método de eliminación de la base de datos para eliminar el registro cuyo identificador se pasó como parámetro.

#### Listing 5.21: AppController.cs

```

1  /// <summary>
2  /// Marca la informacion de un emisor especifico como deshabilitado.
3  /// </summary>
4  /// <param name="appId">Identificador unico que identifica a la aplicacion.</param>
5  /// <returns>Response con el estado de la peticion.</returns>
6  public HttpResponseMessage Delete(Guid appId)
7 {
8     AppRepository.Delete(appId);
9     return this.Request.CreateResponse(HttpStatusCode.OK);
10 }
```

#### 5.2.8. Seguimiento, Trazabilidad o Auditoría

DNN ofrece una librería de componentes para escribir en la base de datos los eventos que ocurren en el portal. Esta librería la podemos utilizar para escribir los eventos que suceden en el servicio como las solicitudes, las respuestas, la cabecera de la solicitud, los parámetros de la solicitud, por fecha y categoría.

A continuación se muestra en el código 5.22 las líneas que documentan la forma de crear un evento, seleccionando el tipo de evento y el mensaje a almacenar. El resto lo hace la plataforma DNN.

#### Listing 5.22: DnnEventLogMsg

```
1 //Se instancian dos objetos: el controlador de eventos.
2 EventLogController eventLog = new EventLogController();
3
4 //Entidad con la informacion del evento.
5 LogInfo logInfo = new LogInfo();
6
7 //Se obtienen los datos del portal donde se instaló y esta ejecutando el servicio web.
8 PortalSettings ps = PortalController.Instance.GetCurrentPortalSettings();
9
10 //Se obtiene la informacion del usuario que ha sido autorizado y autenticado.
11 UserInfo userInfo = UserController.Instance.GetCurrentUserInfo();
12
13 //Se llena la infomacion del evento.
14 logInfo.LogUserID = userInfo.UserID;
15 logInfo.LogPortalID = ps.PortalId;
16 logInfo.LogUserName = userInfo.DisplayName;
17
18 //La siguiente linea nos permite escoger el tipo de operacion que se puede utilizar para
19 //registrar eventos.
20 //Algunas operaciones en eventos pueden ser: Excepcion, Item creado, modificado,
21 //eliminado, Alerta de administrador
22 //alerta de usuario registrado (HOST_ALERT), operacion realizada, fallida, excepcion de
23 //seguridad.
24 logInfo.LogTypeKey = EventLogController.EventLogType.HOST_ALERT.ToString();
25
26 //Finalmente se agrega el log al visor de eventos (VIEWER EVENT)
27 eventLog.AddLog(logInfo);
```

---

Se puede observar en la figura 5-31 como observa una captura de pantalla del visor de eventos de DNN, allí se notará que se puede ver la fecha en la que ocurre el evento, el tipo de evento, el nombre de usuario que estaba ejecutando el evento, el WebSite desde el cual fue ejecutado y un resumen del inconveniente. Uno puede hacer clic sobre cada fila y allí se abrirá mas información como la que se aprecia en la figura 5-31.

Event Viewer					
	Date	Log Type	Username	Website	Summary
█	5/24/2016 8:13:56 PM	Login - Superuser	dnnhost	IGD.FullWebServices\$P 1	
█	5/24/2016 8:13:52 PM	Login Failure	dnnhost	IGD.FullWebServices\$P 1	
█	5/24/2016 8:13:30 PM	Application Started			
█	5/24/2016 8:13:30 PM	Web Server Updated			Server Updated DESKTOP-74BJTF4 IISAppName /LM/W3SVC/1/ROOT/dnndev.me Last
█	5/21/2016 2:02:51 PM	Host Alert	SuperUser Account	IGD.FullWebServices\$Message	Status Code 200 ReasonPhrase OK Version 1.1 Content System.Net.Http
<pre>Message: StatusCode: 200, ReasonPhrase: 'OK', Version: 1.1, Content: System.Net.Http.ObjectContent`1 [[System.Collections.Generic.List`1[[IGD.FullWebServices.Models.HomeDeliveryInfo, IGD.FullWebServices, Version=1.0.0.25223, Culture=neutral, PublicKeyToken=null]]], mscorelib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089]], Headers: {     Content-Type: application/json; charset=utf-8 }</pre>					

Figura 5-31.: Visor de Eventos de las Operaciones en el Portal de DNN

Y en la figura 5-32 se puede ver un cuadro de convenciones de las operaciones que representa cada evento.

Color Coding Legend					
█ Exception	█ Admin Alert	█ Operation Success			
█ Item Created	█ Host Alert	█ Operation Failure			
█ Item Updated	█ Security Exception	█ General Admin Operation			
█ Item Deleted					

Figura 5-32.: Convenciones de los Tipos de Evento Utilizados en el Visor de Eventos de DNN

### 5.2.9. Pruebas Unitarias

Las pruebas unitarias automatizadas son un paso importante dentro del desarrollo por lo que este proyecto no será esta la excepción a la regla. Si bien existen frameworks para realizar pruebas unitarias de forma intuitiva que se pueden utilizar en cualquier proyecto creado dentro de Visual Studio el mayor inconveniente es que hay que crear la data de pruebas y la data de respuesta que se espera, a demás de crear las clases que representan las entidades y su estructura de datos (parámetros) que se van a pasar en los proyectos que se van a probar.

Lo que se plantea en el párrafo anterior es que las pruebas llevan tiempo y esfuerzo importante mientras se declaran e instancian clases, solo por nombrar uno de los factores que dificultan las pruebas. Este problema

ya está resuelto en PowerShell por que allí solo se carga la clase y todo el trabajo ya está realizado.

Las siguientes líneas de código (comandos) en PowerShell presentan solo una parte de las pruebas unitarias a la controladora de aplicaciones utilizada como ejemplo en el desarrollo de ejemplos en esta sección. Ver código 5.23.

**Listing 5.23: TestAppController**

```

1 Describe 'Validaciones EndPoint' {
2
3     It 'El servicio esta disponible' {
4         $ComputerName = (New-Object -TypeName System.Uri -ArgumentList $Script:Config.
5             ServiceEndPoint).Host
6         (Test-NetConnection -CommonTCPPort HTTP -ComputerName $ComputerName).
7             PingSucceeded | Should Be $true
8     }
9
10    It 'Invocar al servicio sin especificar una clase Controller debe generar un error
11        404 (Not Found)'{
12        (Invoke-Controller -InputObject @{$Uri=$Script:Config.ServiceEndPoint;Method='Get
13        '}).StatusCode | Should Be 404
14    }
15
16    Describe 'Validaciones AppController' {
17        $AppControllerUri = '{0}/{1}' -f $Script:Config.ServiceEndPoint, 'App'
18        $AuthorizedHeader = New-BasicAuthHeader
19        $UnauthorizedHeader = New-BasicAuthHeader -Username (Get-RandomString) -Password (
20            Get-RandomString)
21
22        Context 'Listar aplicaciones' {
23
24            It 'Obtener la lista de todas las aplicaciones con un usuario autorizado' {
25                $Response = Invoke-Controller -InputObject @{$Uri=$AppControllerUri;Method='
26                    Get';Headers=$AuthorizedHeader}
27                $Response.StatusCode | Should Be 200
28                $Response.Content | Should Not BeNullOrEmpty
29                {$Script:AppList = ($Response.Content | ConvertFrom-Json)} | Should Not
30                    Throw
31                {$Response.Content | Assert-Schema -SchemaFileName 'AppInfoSchema.json' -
32                    AsArray} | Should Not Throw
33            }
34
35            It 'Obtener la lista de todas las aplicaciones sin usuario y clave debe generar
36                un error 401 (Unauthorized)' {
37                (Invoke-Controller -InputObject @{$Uri=$AppControllerUri;Method='Get';Headers
38                    =$null}).StatusCode | Should Be 401
39            }
40
41            It 'Obtener la informacion de cada aplicacion con un usuario autorizado' {
42                $Script:AppList | ForEach-Object {
43                    $AppUri = '{0}/Get?appId={1}', -f $AppControllerUri, $_.ApplicationGuid
44                    (Invoke-Controller -InputObject @{$Uri=$AppUri;Method='Get';Headers=
45                        $AuthorizedHeader}).StatusCode | Should Be 200
46                    $Script:LastAppUri = $AppUri
47                }
48            }
49        }
50    }
51
52 }
```

```

41 Context 'Crear aplicaciones' {
42     $AppGuid = [System.Guid]::.NewGuid().ToString('N')
43     $AppName = 'App demo Pester'
44
45     It 'Crear una aplicacion con datos correctos utilizando un usuario no autorizado
        debe generar un error 401 (Unauthorized)' {
46         $Body = (@{ApplicationGuid=$AppGuid;Name=$AppName}) | ConvertTo-Json
47         (Invoke-Controller -InputObject @{Uri=$AppControllerUri;Method='Post';
48             Headers=$UnauthorizedHeader;Body=$Body}).StatusCode | Should Be 401
49     }
50
51     It 'Crear una aplicacion con el mismo nombre de una que ya exista debe generar
        el error 409 (Conflict)' {
52         $Body = (@{ApplicationGuid=$AppGuid;Name=$AppName}) | ConvertTo-Json
53         (Invoke-Controller -InputObject @{Uri=$AppControllerUri;Method='Post';
54             Headers=$AuthorizedHeader;Body=$Body}).StatusCode | Should Be 409
55     }
56
57     It 'Crear una aplicacion solo con el nombre utilizando un usuario autorizado' {
58         $RandomName = '{0} {1}' -f $AppName, (Get-RandomString -Length 10)
59         $Body = (@{Name=$AppName}) | ConvertTo-Json
60         (Invoke-Controller -InputObject @{Uri=$AppControllerUri;Method='Post';
61             Headers=$AuthorizedHeader;Body=$Body}).StatusCode | Should Be 201
62     }
63 }

```

Analizando alguna de las líneas del código anterior: las líneas 1 y 13 representan la agrupación de casos de pruebas.

La línea 1 tiene las pruebas para las validaciones de punto final (EndPoint) que validan la disponibilidad del servicio, la controladora y el método. La línea 13 tiene las pruebas para validar los métodos de la controladora de aplicaciones.

Las líneas 3 y 8 con el inicio de cada caso de prueba para los puntos final. Nótese que para la primera prueba para identificar si el servicio está disponible se realiza una prueba de conexión pasando la dirección del servicio y esperando a que sea respondido un ping (ver línea de código 5).

En la línea 9 para el otro caso de prueba se llama el servicio y uno de sus métodos sin encabezado de autenticación en la solicitud de forma tal que se espera (Should Be) que el servicio mediante el protocolo HTTP responda el código 404, haciendo que la prueba sea válida por que efectivamente genera un error.

De igual forma para el segundo grupo de pruebas que valida los métodos de la controladora, observamos que en las líneas de código 22, 29 y 35 para esos casos de prueba, se esperan respuestas para identificar si efectivamente el servicio responde de acuerdo al caso para marcar la prueba como válida o inválida.

Lo único que se hace en cada una de las pruebas es variar los parámetros que se pasan en la URL, encabezado y body de la solicitud a fin de crear un caso exitoso o provocar fallas intencionales y así comparar que efectivamente están devolviendo el código esperado como se presentó en el código anterior.

Nota: el código completo de la prueba a la controladora de aplicaciones lo encontrará en el anexo G

# **6. RECOLECCIÓN, ANÁLISIS Y PRESENTACIÓN DE INFORMACIÓN**

## **6.1. Recolección y Ordenamiento de la Información**

La planeación de la recopilación de datos primarios se realiza mediante la encuesta como enfoque de investigación. Las encuestas fueron enviadas a un grupo de profesionales en ciencias de la computación, ingeniería de sistemas, ingeniería de software y otras ingenierías.

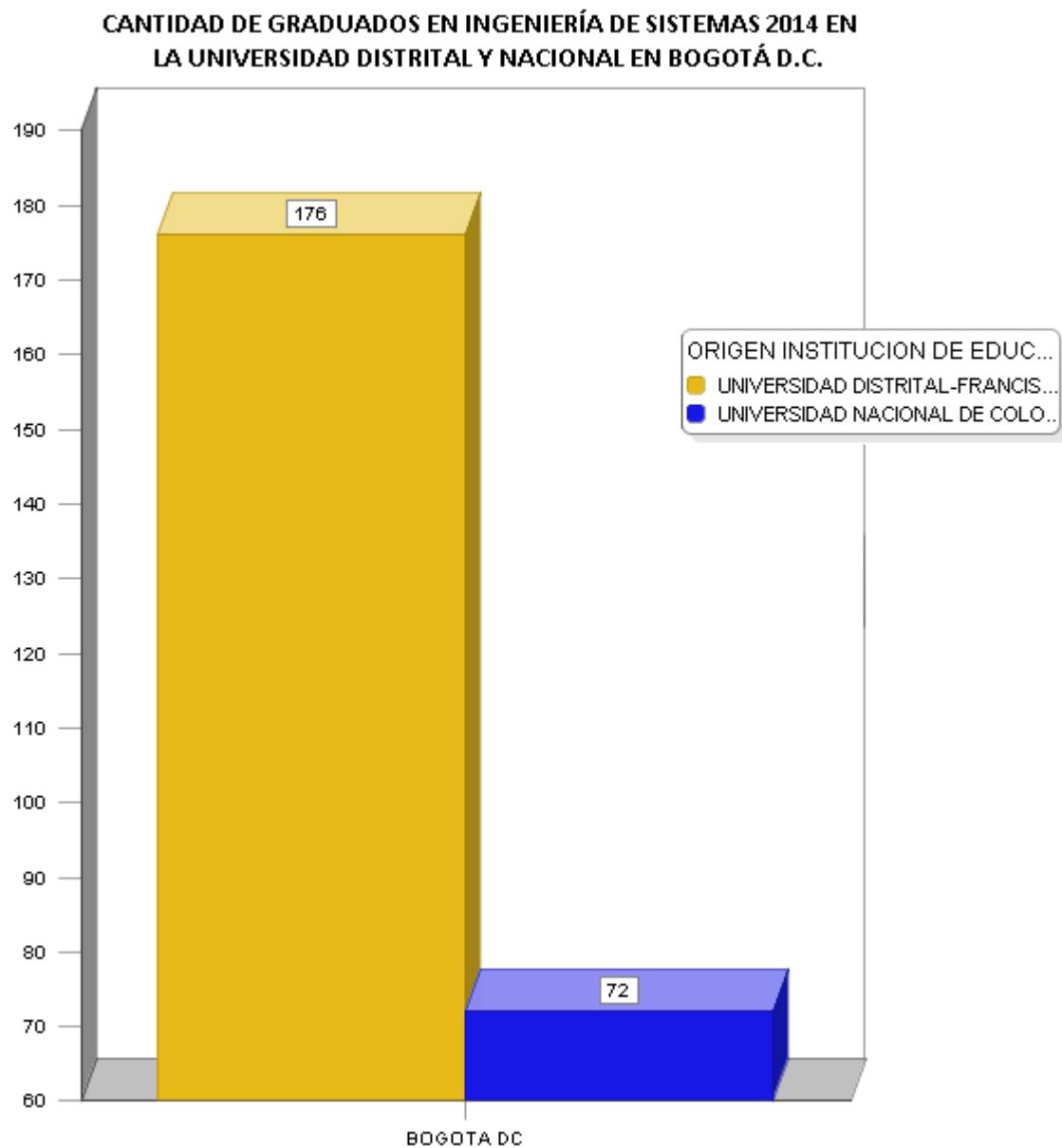
Se utilizó como método de contacto el correo electrónico e instrumento de investigación cuestionarios creados con la aplicación web “Formularios de Google” con preguntas de selección múltiple con única respuesta y algunas preguntas abiertas con el fin de conocer las tecnologías de desarrollo de servicios web utilizadas.

El método utilizado es el **Muestreo Probabilístico** en el que cada elemento de la población tiene la misma probabilidad para ser seleccionado en la muestra.

### **6.1.1. Población**

Para calcular la muestra se tomo como población a la **cantidad de graduados en ingeniería de sistemas en la Universidad Distrital y Universidad Nacional en Bogotá en el 2014**

La estadística se obtuvo del portal web del **“Observatorio Laboral del Ministerio de Educación en Colombia”** con un total de 248 estudiantes graduados de los cuales 176 son de la Universidad Distrital y 72 de la Universidad Nacional de Colombia. [16]. Ver la figura **6-1**



**Figura 6-1.:** Cantidad de Graduados en Ingeniería de Sistemas en la Universidad Nacional y Distrital en Bogotá en el 2014. Fuente: Observatorio Laboral del Ministerio de Educación. [16]

### 6.1.2. Muestra

Esta medición se realizara mediante encuestas de una muestra de la población. Se pretende medir la **aceptación, uso y tiempos empleados en el aprendizaje y desarrollo de servicios web**.

N: Tamaño de la población o máximo número de encuestados.

N = 248

Se ha decidido aceptar un error máximo del 10 % y un nivel de confianza del 90 %, para obtener el tamaño

de la muestra.

NC: Nivel de confiabilidad. NC = 90 %

e: Error máximo permitido redondeado

$$e = 10\%$$

Z: Constante que depende del nivel de confiabilidad.

$$Z = 1.65 \text{ para un nivel de confiabilidad del } 90\%$$

La razón de que esta p aparezca en la fórmula es que cuando una población es muy uniforme, la convergencia a una población normal es más precisa, lo que permite reducir el tamaño de muestra.

P: Proporción de individuos que poseen en la población la característica de estudio.

$$P = 0.5$$

Q: Varianza de la proporción de individuos que no poseen esa característica, es decir, es 1-p.

$$Q = 0.5$$

n = el tamaño de la muestra o número de encuestas que vamos a hacer.

$$n_0 = \frac{Z^2 * PQ}{e^2} = \frac{1,65^2 * (0,5 * 0,5)}{0,10^2} \quad (6-1)$$

$$n_0 = 68,06 \quad (6-2)$$

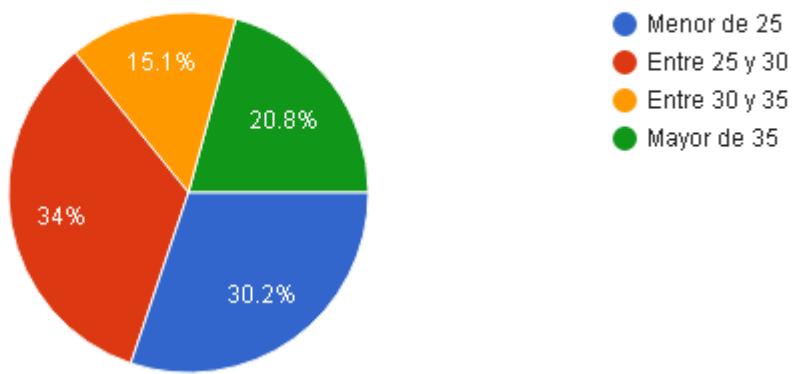
$$n = \frac{n_0}{1 + \frac{n_0 - 1}{N}} = \frac{68,06}{1 + \frac{68,06 - 1}{230}} \quad (6-3)$$

$$n = 53,57 \quad (6-4)$$

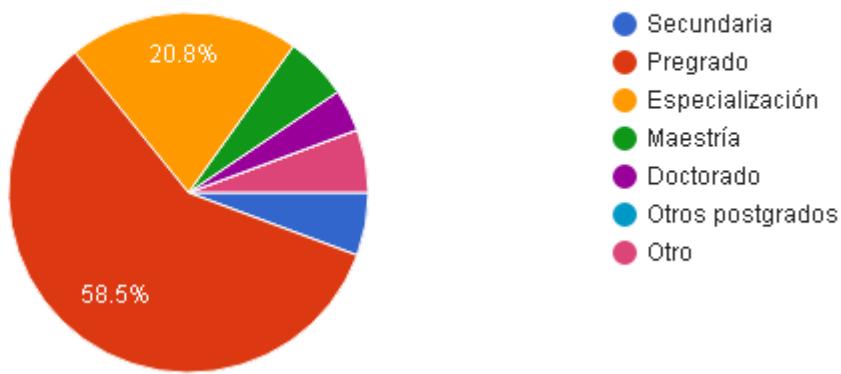
**El tamaño de la población de la muestra a encuestar, tomando la parte entera del cálculo, es de 53 egresados.**

### 6.1.3. Presentación de los Resultados

A continuación se presentan las preguntas, las opciones de respuesta y los gráficos con el porcentaje totalizado por respuesta de los 53 profesionales seleccionados como muestra a aplicar la encuesta. Las encuesta aplicada se puede ver en el anexo E.

**Edad** (53 respuestas)

**Figura 6-2.**: Porcentajes de Edad de los Profesionales Encuestados.

**Nivel Académico** (53 respuestas)

**Figura 6-3.**: Porcentajes por Nivel Académico de los Profesionales Encuestados.

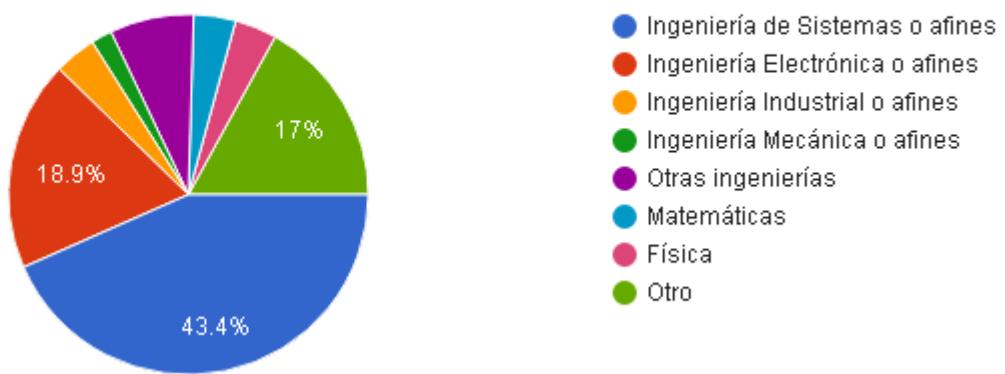
**Profesión** (58 respuestas)

Figura 6-4.: Porcentajes por Profesión de los Profesionales Encuestados.

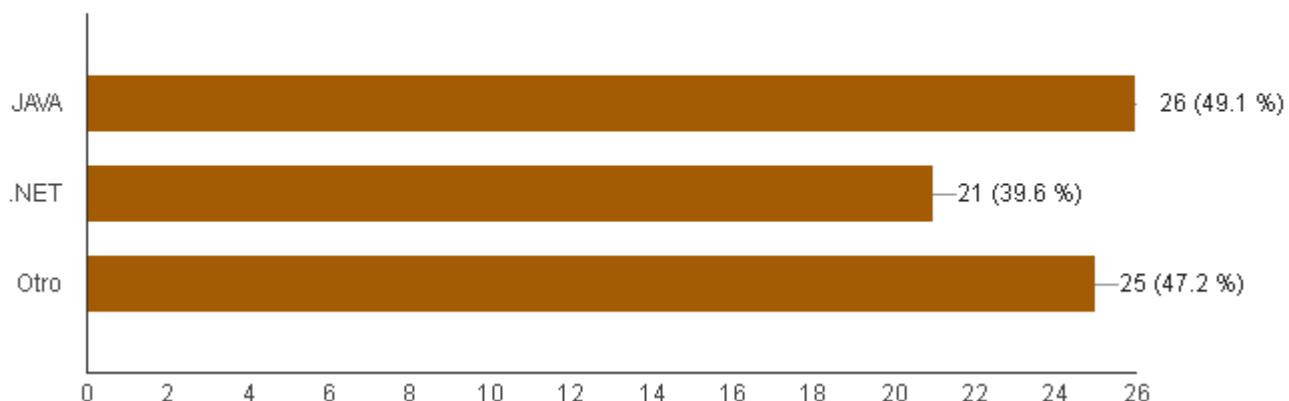
**Plataformas que realmente utiliza** (58 respuestas)

Figura 6-5.: Porcentajes de Plataformas de Desarrollo de Software Utilizadas por los Profesionales Encuestados.

**Ha diseñado, desarrollado o programado componentes o librerías?**  
(58 respuestas)

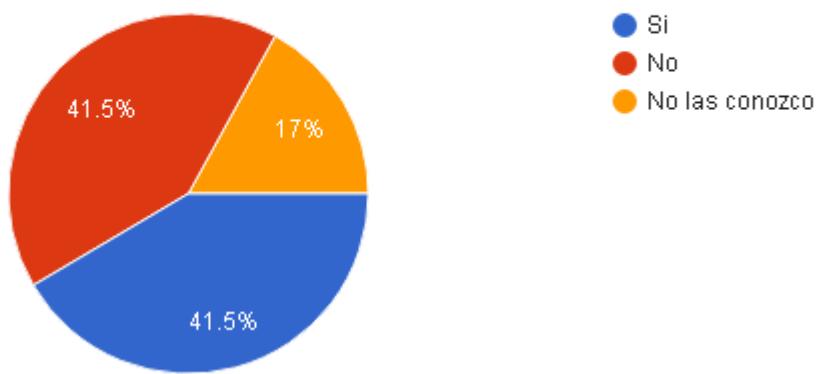


Figura 6-6.: Porcentajes de Profesionales Encuestados Que Han Creado Componentes.

**Ha diseñado, desarrollado o programado aplicaciones de consola?**  
(58 respuestas)

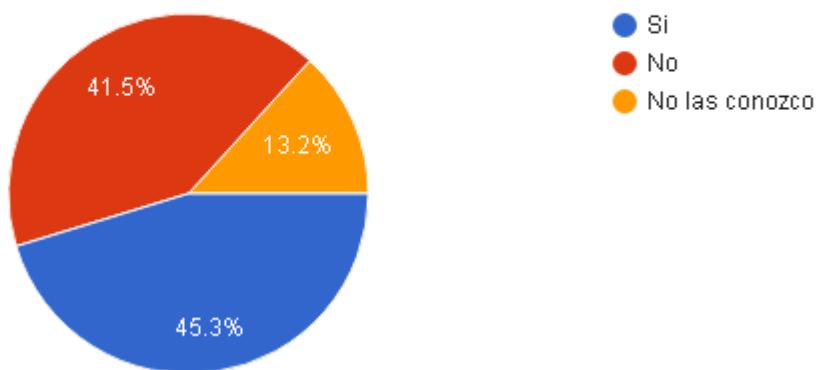


Figura 6-7.: Porcentajes de Profesionales Encuestados Que Han Creado Aplicaciones de Consola.

**Ha diseñado, desarrollado o programado aplicaciones de escritorio?**  
(53 respuestas)

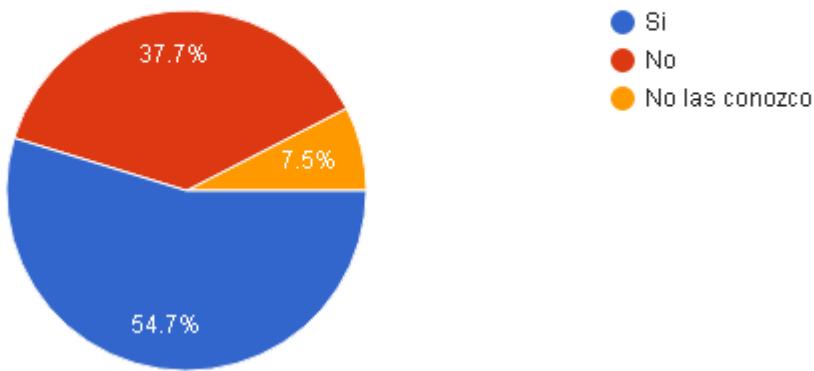


Figura 6-8.: Porcentajes de Profesionales Encuestados Que Han Creado Aplicaciones de Escritorio.

**Ha diseñado, desarrollado o programado aplicaciones web (páginas web con una lógica de negocio)?**  
(53 respuestas)

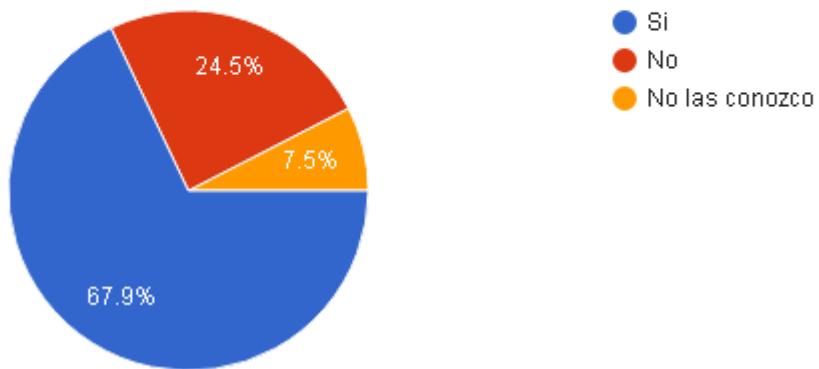
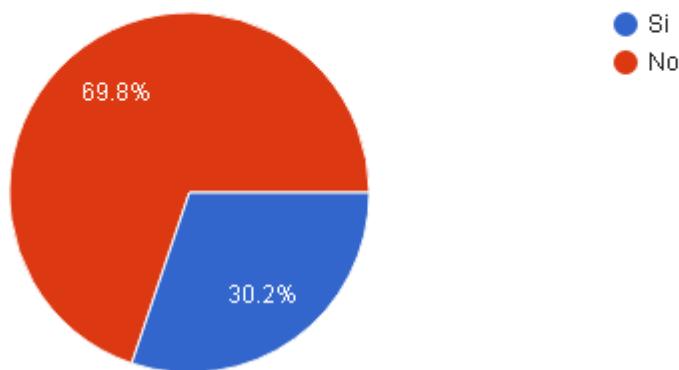


Figura 6-9.: Porcentajes de Profesionales Encuestados Que Han Creado Portales Web.

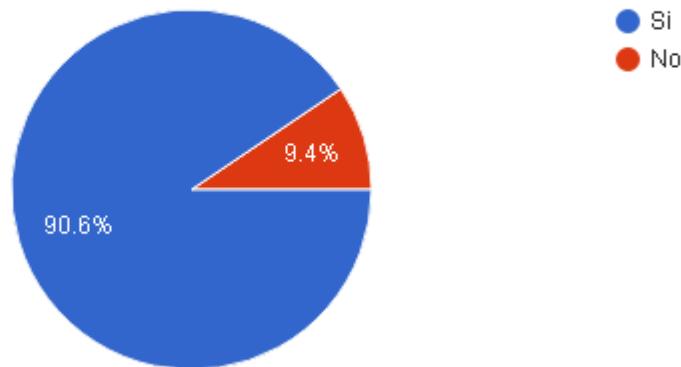
**Conoce la diferencia entre servicio web (webservice), arquitectura orientada a servicios SOA y bus de servicios empresariales EBS?**

(53 respuestas)

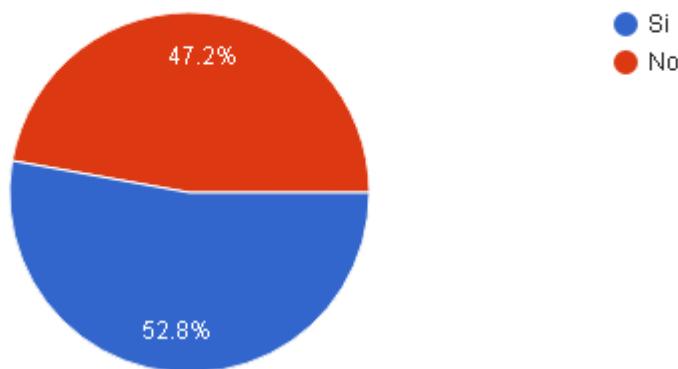


**Figura 6-10.: Porcentajes de Profesionales Encuestados Que Conocen La Diferencia de Conceptos entre SOA, WS y ESB.**

**Ha consumido o utilizado servicios web? (53 respuestas)**



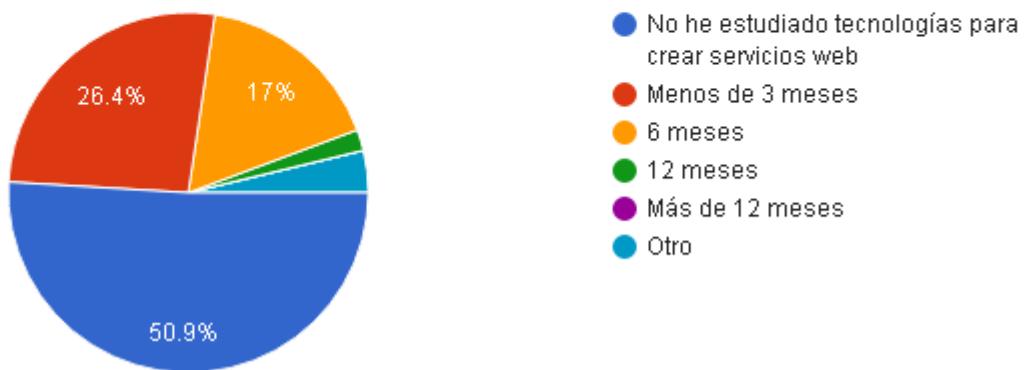
**Figura 6-11.: Porcentajes de Profesionales Encuestados Que Han Consumido Servicios Web.**

**Ha creado, desarrollado o programado servicios web? (53 respuestas)**

**Figura 6-12.**: Porcentajes de Profesionales Encuestados Que Han Creado Servicios Web.

**Sí ha creado servicios web, indique la cantidad de tiempo que le tomó aprender las tecnologías, frameworks, componentes, librerías o plugins para desarrollar o programar servicios web.**

(53 respuestas)



**Figura 6-13.**: Porcentajes de Tiempo Empleado en Aprender Alguna Tecnología de Servicio Web por los Profesionales Encuestados.

**Sí ha creado servicios web, indique la cantidad de tiempo que les está tomando o les llevó a usted o su equipo desarrollar o programar el proyecto más largo de servicios web.**

(53 respuestas)

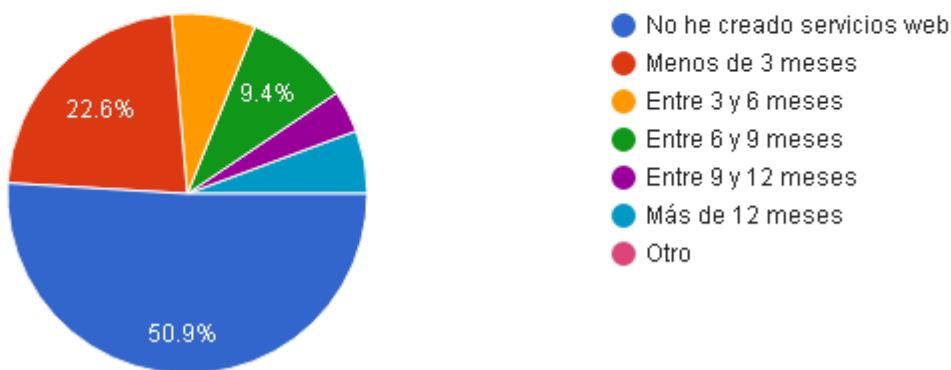


Figura 6-14.: Porcentajes de Tiempo Empleado en Crear Servicio Web por los Profesionales Encuestados.

**Considera usted que es viable económicoamente incluir en un proyecto de software el diseño y desarrollo de servicios web como parte de la solución?**

(53 respuestas)

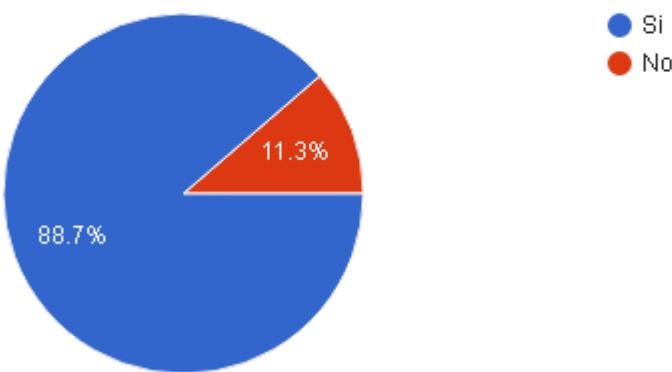


Figura 6-15.: Porcentajes de Profesionales Encuestados Que Consideran Viable Económicamente Incluir un Servicio Web en Proyectos de Software.

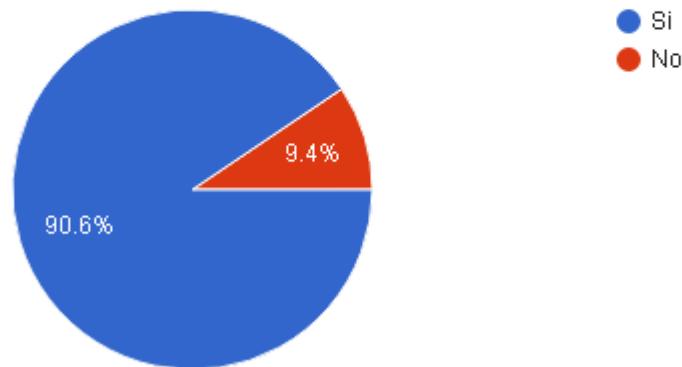
**Los costos de un proyecto de software con servicios web son:** (53 respuestas)



**Figura 6-16.**: Porcentajes de Nivel de Costos de un Proyecto con Servicio Web Considerado por los Profesionales Encuestados.

**Incluiría usted en un proyecto de software el diseño y desarrollo de servicios web como parte de la solución?**

(53 respuestas)



**Figura 6-17.**: Porcentajes de Profesionales Encuestados Que Incluirían Servicios Web en sus Proyectos de Software.

**Indique el grado de rechazo o aceptación personal o su negocio con respecto a los servicios web.**

(53 respuestas)

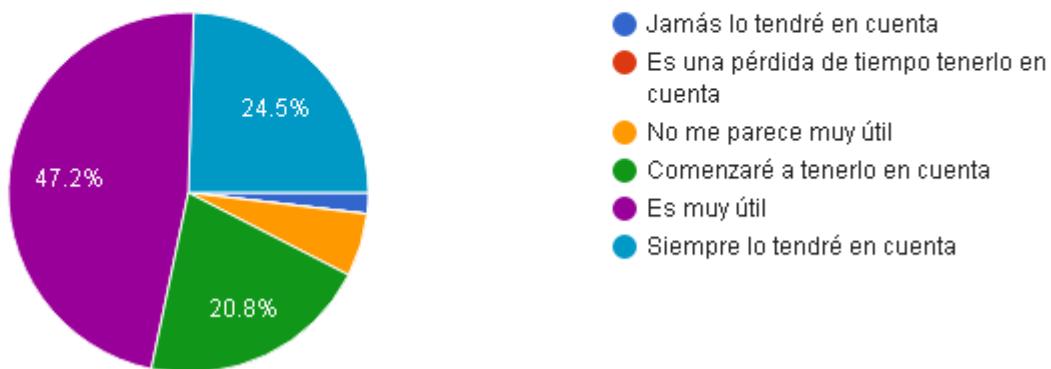


Figura 6-18.: Porcentajes de Aceptación Personal de Servicios Web de los Profesionales Encuestados.

**Indique el grado de rechazo o aceptación de los profesionales de software que lo rodean o la empresa en la que trabaja con respecto a los servicios web.**

(53 respuestas)

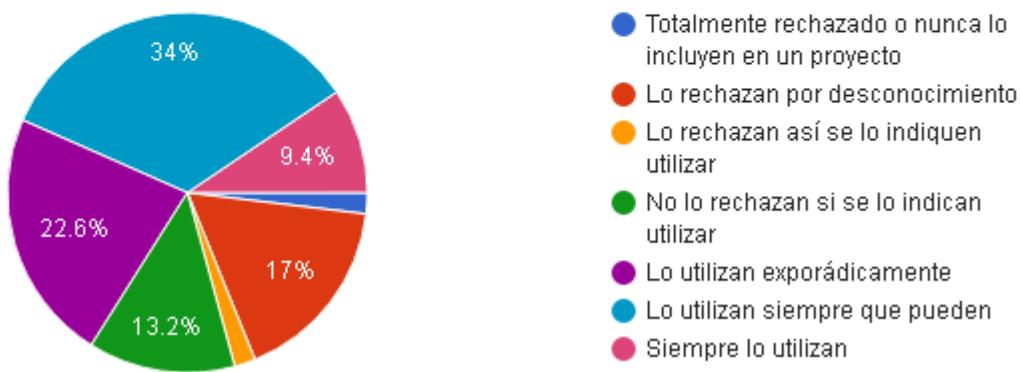


Figura 6-19.: Porcentajes de Aceptación de Servicios Web de las Personas Que Rodean a los Profesionales Encuestados.

## 6.2. Presentación y Análisis de los Resultados

La encuesta realizada permite obtener información acerca del conocimiento, uso, aceptación e implementación de servicios web, como soporte y argumento en el desarrollo del proyecto.

Esta información se utiliza para apoyar la observación realizada en el problema de investigación. Los resultados de la encuesta también permiten comparar el tiempo que tardan los encuestados contra el tiempo de la arquitectura propuesta en desarrollar e implementar un servicio web. La encuesta nos permite corroborar conclusiones que surgen del desarrollo del documento como se presenta en el cierre de la investigación.

**Parte III.**

## **CIERRE DE LA INVESTIGACIÓN**

## 7. RESULTADOS Y DISCUSIÓN

- El desarrollo del proyecto responde a la pregunta de formulación ya que se logra proponer una arquitectura que permite implementar servicio web más rápido y sencillo, cumpliendo las características y cualidades deseadas utilizando una tecnología específica.
- Esta arquitectura es altamente escalable y mantenable lo que da una gran ventaja al hacer crecer la cantidad de servicios que puede atender el servicio web.
- Se comprueba la hipótesis ya que luego de ordenar y distribuir los componentes de la arquitectura utilizando las técnicas y métodos en la tecnología seleccionada, se disminuyó el tiempo y complejidad de diseño, desarrollo e implementación de un servicio web.
- El tiempo de desarrollo e implementación comparado con la información obtenida en las encuestas se logró reducir en una relación 10:1.
- Se entrega el diseño de la arquitectura para un servicio web completo y se propone que sea utilizado sin importar la tecnología que se utilice para su desarrollo e implementación, debido a que son componentes totalmente independientes que simplemente interactúan entre sí, basta con definir los contratos que les permita interactuar.
- La tecnología seleccionada, .NET 4.5, Web API 2, PowerShell, Dot Net Nuke 7.4.2 y RabbitMQ además de ser sencilla, rápida de entender y aprender nos permite dar un valor agregado al servicio web creado en frentes como la autenticación, autorización, seguridad, serialización, manejo de excepciones y la aplicación de patrones en unas sencillas líneas de código.
- Comparando los tiempos de respuesta del servicio web completo implementado los tiempos de respuesta fueron al rededor de 500milisegundos, casi 3 veces menores que los tiempos de respuesta entregados por otros servicios web en proyectos open source disponibles en internet, dejándole al lector su comprobación.
- Como valor agregado al trabajo se presenta un método de autorización para garantizar los dispositivos o sistemas que consuman los servicios.
- Crear métodos y exponer métodos resulta tan sencillo como escribir una simple función.
- El código completo de la aplicación se puede descargar de la URL **www.ingenieriad.net** bajo licencia **Creative Commons** en la sección de proyectos personales del autor.

# 8. CONCLUSIONES

## 8.0.1. Conclusiones

- Actualmente existen una gran cantidad de aplicaciones de consola, escritorio, web y móviles que aún no permiten intercambiar información. Es importante que los sistemas actuales expongan un API para comunicarse entre si, evitando, por ejemplo, tener una conexión directa a un motor de base de datos, distribuir archivos planos.
- Utilizar un servicio web con las características del servicio web completo presentado, garantiza evitar riesgos de robo de información sensible y disminuir las vulnerabilidades del sistema.
- La arquitectura propuesta en la metodología no solamente se podrá implementar con las tecnologías seleccionadas, también se podrán utilizar otras como las de Oracle, por lo tanto se invita a que prueben descubrir marcos de trabajo y componentes que faciliten su desarrollo e implementación.
- Se lograron reducir drásticamente los tiempos de desarrollo de implementación del servicio web completo, incluso reducir los tiempos de respuesta a las solicitudes.
- Utilizar el modelo o patrón RPC de RabbitMQ tiene como consecuencia añadir una complejidad innecesaria a la depuración. En vez de simplificar el software, RPC, puede resultar en código spaghetti difícil de mantener.
- Utilizar un lenguaje de modelamiento es útil para realizar un análisis de cada uno de los elementos que conforman una organización, modelo que luego me permitirá llegar a cualquiera de sus capas, en este caso, la capa de aplicación y desde allí realizar una implementación casi que automatizada de acuerdo a las necesidades o problemas detectados o a resolver, en una organización.
- En la arquitectura empresarial en las fases de arquitectura de negocio, aplicación e infraestructura no hay que llevar el detalle a un nivel algorítmico, se tiende a confundir el proceso con el algoritmo.
- Con Archimate no se dejan vacíos en la descripción de componentes, sin necesidad de lanzar código o instrumentación.
- La documentación es síntoma de no saber escribir, por eso a nivel de código hay que utilizar descripciones en lenguaje natural para poder identificar muy rápido los componentes y que el mismo código de programación sea tan descriptivo como sea posible.
- Es importante cada vez que se adquiera un nuevo conocimiento apropiarse del discurso propio del mismo.
- El desarrollo de la arquitectura empresarial con Archimate puede llevarse a cabo comprometiéndose con cosas pequeñas, progresar y hacer entregas (metodologías de desarrollo rápido, ej: Scrum), esto mejora la métrica comenzando a ver evidencias de entregables, así sean pequeños.
- Un diagrama solamente modela un área del conocimiento, un punto de vista de Archimate le permite a uno modelar diferentes áreas del conocimiento.

- Archimate es fuerte conceptualmente porque trata sobre el negocio en diferentes áreas (software, alta gerencia, objetivos estratégicos del negocio, etc.).
- Archimate es un lenguaje que aborda la problemática desde las perspectivas mucho más amplias, por lo tanto no se puede comparar con lenguajes como el UML. El área de UML es el área de diseño de bajo nivel, el área de Archimate es el área de alto nivel.
- En las áreas de sistemas y afines, toda área de conocimiento se puede abrir con una llave de dos componentes: parte conceptual o de conocimiento y el lenguaje para expresar todo esto. Un ejemplo histórico es Dmitri Mendeléyev quien al crear la tabla periódica generó un lenguaje. Permitió observar un patrón y así determinar que había algún elemento que aún no se había descubierto. Archimate, sin ser expertos, lo entendemos porque vienen de la lógica humana. Lo mismo sucedió con el lenguaje organizacional porque permite tener una propuesta conceptual muy rica.

### **8.0.2. Aportes originales**

- Se deben utilizar frameworks o componentes que la industria acepta con el fin de disminuir tiempos de desarrollo, pruebas e implementación. No es necesario reinventar la rueda creando componentes que ya existen. Lo que si es recomendable es utilizarlos bien validando su procedencia, los comentarios de la industria, potenciales vulnerabilidades y la calidad de su documentación que facilite su uso.
- Las organizaciones deben tener claro sus objetivos y estrategias organizacionales, involucrar todos los elementos, teniendo como elemento común un lenguaje de diseño (arquitectura empresarial), así podrán generar modelar sus procesos, que mediante metamodelos podrían utilizar con la arquitectura propuesta y plantillas generadoras de código automático para crear en muy poco tiempo, con muy pocos recursos humanos y económicos, sistemas de información que además de intercambiar información de forma segura y compatible, ayuden a alcanzar los objetivos estratégicos de la organización.

# Bibliografía

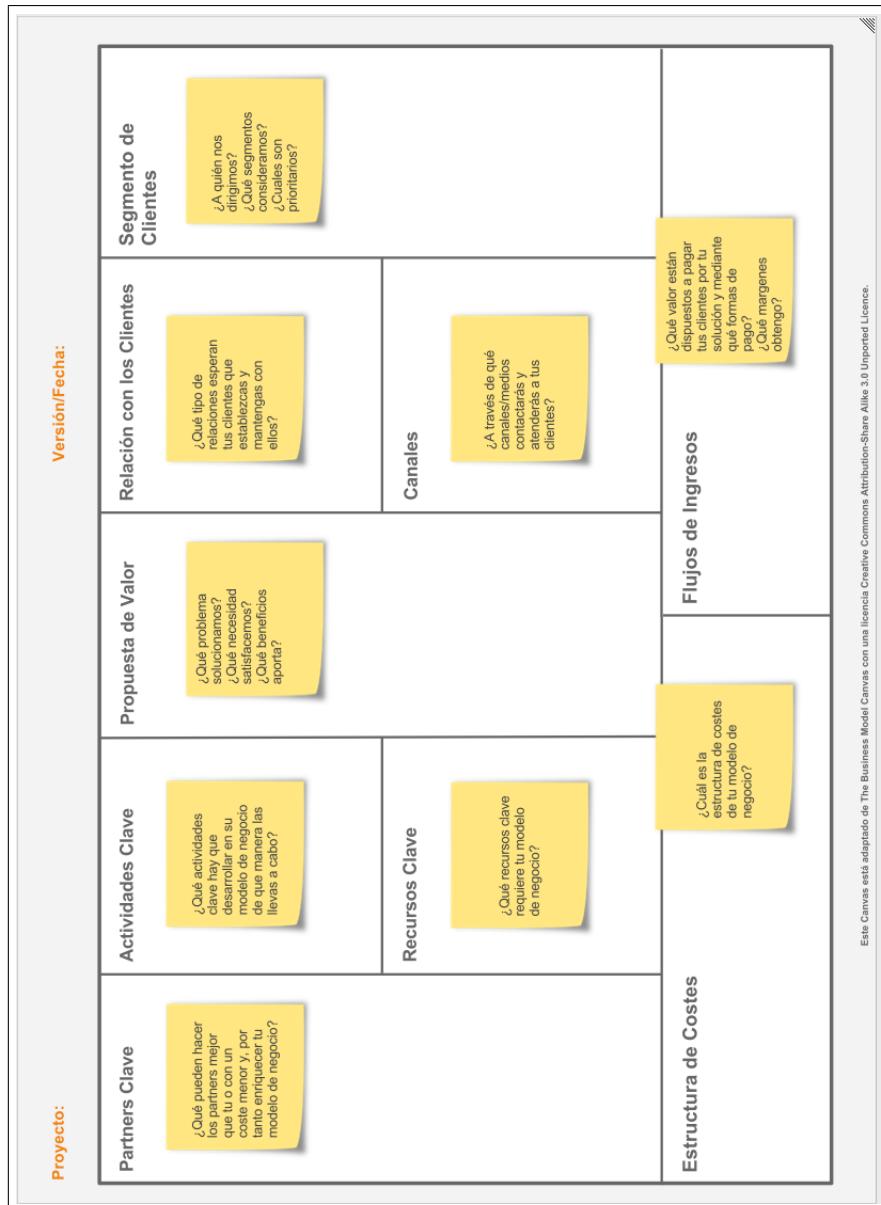
- [1] CodeJOBS, “¿Qué es el modelo cliente-servidor? Networking - Aprende a Programar - Codejobs,” 2015. [Online]. Available: <https://www.codejobs.biz/es/blog/2014/11/05/que-es-el-modelo-cliente-servidor-networking>
- [2] Alfsan, “Arquitectura de n capas,” 2012. [Online]. Available: <http://iutll-abdd.blogspot.com.co/2012/05/arquitectura-de-n-capas.html>
- [3] S. Moreno Saiz, “Estudio de Arquitecturas Software para Servicios de Internet de las Cosas,” 2015. [Online]. Available: <http://oa.upm.es/37339/>
- [4] T. Rademarkers, *Open Source ESBs in Action*, 2009.
- [5] Strategyzer, “Business Model Canvas,” 2016. [Online]. Available: <http://www.businessmodelgeneration.com/canvas/bmc>
- [6] A. Villalpando, “Arquitectura Empresarial, Architect, ArchiMate y TOGAF,” 2014. [Online]. Available: <http://togaf-architect-archimate.blogspot.com.co/>
- [7] The Open Group, *ArchiMate® 2.1 Specification Motivation Extension*, 2013. [Online]. Available: <http://pubs.opengroup.org/architecture/archimate2-doc/chap10.html{\#}{\-\}Toc371945259>
- [8] A. Villalpando, “Arquitectura Empresarial, Architect, ArchiMate y TOGAF: ArchiMate y su Integración con el ADM,” 2014. [Online]. Available: <http://togaf-architect-archimate.blogspot.com.co/2014/03/archimate-elrol-del-estandar-de.html>
- [9] W3C, “Guía Breve de Servicios Web,” 2012. [Online]. Available: <http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb>
- [10] Daniel\_lt, “Orquestacion y Coreografia de Servicios Web,” 2012. [Online]. Available: [http://es.slideshare.net/daniel{\\\_}lt/orquestacion-y-coreografia-de-servicios-web](http://es.slideshare.net/daniel{\_}lt/orquestacion-y-coreografia-de-servicios-web)
- [11] RabbitMQ, “RabbitMQ tutorial - Remote procedure call (RPC),” 2016. [Online]. Available: <http://www.rabbitmq.com/tutorials/tutorial-six-dotnet.html>
- [12] WebNode, “Sectores Económicos de Colombia,” 2015. [Online]. Available: <http://cciassocialespoliticaeconomia.webnode.es/contactanos/sociales/noveno/>
- [13] A. Ahani, “WhatsIsNewWCF4.5,” 2011. [Online]. Available: <http://blog.csharplearners.com/tag/wcf/>
- [14] C. de la Torre, “What is .NET Core 5 and ASP.NET 5 within .NET 2015 Preview — Cesar de la Torre [Microsoft] – BLOG,” 2014. [Online]. Available: <https://blogs.msdn.microsoft.com/cesardelatorre/2014/11/18/what-is-net-core-5-and-asp-net-5-within-net-2015-preview/>
- [15] S. Hanselman, “Download Podcasts with Powershell - Scott Hanselman,” 2009. [Online]. Available: <http://www.hanselman.com/blog/DownloadPodcastsWithPowershell.aspx>
- [16] C. Ministerio de Educación Nacional, “Observatorio Laboral para la Educación,” 2016. [Online]. Available: <http://www.graduadoscolombia.edu.co/html/1732/w3-propertyvalue-36267.html>

- [17] Significados, “Know how,” 2016. [Online]. Available: <http://www.significados.com/know-how/>
- [18] H. Bernardis, E. Bernardis, M. Berón, D. Riesco, P. Henriques, and M. J. Pereira, “Técnicas y estrategias para comprender procesos de negocios especificados en WS-BPEL,” 2015. [Online]. Available: <http://bibliotecadigital.ipb.pt/handle/10198/11886>
- [19] C. Calero, *Calidad del producto y proceso software*, 2nd ed., 2010.
- [20] L. C. Escalante, “El patrón de arquitectura n-capas con orientación al dominio como solución en el diseño de aplicaciones empresariales.” pp. 59–66, 2 2016. [Online]. Available: <http://revistas.ucv.edu.pe/index.php/RTD/article/view/679>
- [21] S. Guru, “Metodologías Ágiles,” *Software Guru*, 2006. [Online]. Available: [http://www.ozarate.net/articulos/arquitectura{\\\_}sw{\\\_}sg{\\\_}2006.pdf](http://www.ozarate.net/articulos/arquitectura{\_}sw{\_}sg{\_}2006.pdf)
- [22] B. M. Márquez Avendaño, *Implementación de un reconocedor de voz gratuito a el sistema de ayuda a invíidentes Dos-Vox en español*, 2004. [Online]. Available: [http://catarina.udlap.mx/u{\\\_}dl{\\\_}a/tales/documentos/lis/marquez{\\\_}a{\\\_}bm/capitulo5.pdf](http://catarina.udlap.mx/u{\_}dl{\_}a/tales/documentos/lis/marquez{\_}a{\_}bm/capitulo5.pdf)
- [23] Oracle, “Distributed Multitiered Applications - The Java EE 6 Tutorial.” [Online]. Available: <http://docs.oracle.com/javaee/6/tutorial/doc/bnaay.html>
- [24] A. Manso, “SOA y Web Services,” 2016. [Online]. Available: <http://soawebs.blogspot.com.co/2011/01/soa-y-web-services.html>
- [25] J. A. Zachman, “A framework for information systems architecture,” *IBM Systems Journal*, vol. 26, no. 3, pp. 276–292, 1987.
- [26] B. Bellman and K. Griesi, “Enterprise architecture advances in technical communication,” in *2015 IEEE International Professional Communication Conference (IPCC)*, 2015, pp. 1–5.
- [27] O. Lengerke, “Arquitectura empresarial, El camino hacia un gobierno integrado,” *Cio@Gov*, no. 2, 2013.
- [28] D. F. Ruíz Sanchez, “Diseño de Arquitectura Empresarial en el Sector Educativo Colegio Bogotá,” Ph.D. dissertation, 2014. [Online]. Available: <http://repository.ucatolica.edu.co/jspui/bitstream/10983/1691/1/TrabajodeGradoArquitecturaEmpresarial.pdf>
- [29] A. d. Choachí, “Documento Modelo Arquitectura Voto Electrónico Municipio Choachí,” Ph.D. dissertation, 2011. [Online]. Available: <http://pegasus.javeriana.edu.co/{}PA111-01-eVoto/docs/documentofinaldearquitecturaempresarialconvalidacion.pdf>
- [30] J. Guerrero, “Archimate: lenguaje para modelamiento de la arquitectura empresarial.” [Online]. Available: <http://es.slideshare.net/jdelgadog17/archimate-lenguaje-para-modelamiento-de-la-arquitectura-empresarial>
- [31] T. O. Group, “The Open Group,” 2016. [Online]. Available: <http://www.opengroup.org/>
- [32] O. G. TOGAF, *TOGAF 9.1*, 2011. [Online]. Available: <http://pubs.opengroup.org/architecture/togaf9-doc/arch/index.html>
- [33] D. Emery and R. Hilliard, “Every architecture description needs a framework: Expressing architecture frameworks using ISO/IEC 42010,” in *Software Architecture, 2009 European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on*, 9 2009, pp. 31–40.

- [34] W. Wang and M. W. Godfrey, "Detecting API usage obstacles: A study of iOS and Android developer questions," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 5 2013, pp. 61–64. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6624006>
- [35] M. Colan, "Service-Oriented Architecture expands the vision of web services, Part 1," 4 2004. [Online]. Available: <http://www.ibm.com/developerworks/library/ws-soaintro/index.html>
- [36] M. Diaz Rosales, "Arquitectura Orientada a Servicios," 2008. [Online]. Available: <http://faquinones.comze.com/educativos/eai/EAISOA.pdf>
- [37] W3C, "Web Services Glossary." [Online]. Available: <https://www.w3.org/TR/ws-gloss/>
- [38] D. Booth, "Web Services Architecture," 2003. [Online]. Available: <https://www.w3.org/TR/ws-arch/>
- [39] S. Meng and F. Arbab, "Web services choreography and orchestration in Reo and constraint automata," in *Proceedings of the 2007 ACM symposium on Applied computing - SAC '07*. New York, New York, USA: ACM Press, 3 2007, p. 346. [Online]. Available: <http://dl.acm.org.bdigital.udistrital.edu.co:8080/citation.cfm?id=1244002.1244085>
- [40] D. N. d. Bibliotecas, J. E. GIRONDO P., J. A. GUZMÁN L., and D. A. OVALLE, "Técnicas de inteligencia artificial aplicadas a la coreografía de servicios web," 7 2009. [Online]. Available: <http://www.bdigital.unal.edu.co/15413/1/10020-18198-1-PB.pdf>
- [41] K. Benghazi, M. Noguera, C. Rodríguez-Domínguez, A. B. Pelegrina, and J. L. Garrido, "Real-time web services orchestration and choreography," pp. 142–153, 6 2010. [Online]. Available: <http://dl.acm.org.bdigital.udistrital.edu.co:8080/citation.cfm?id=1866939.1866952>
- [42] IBM, "IBM developerWorks en español : Introducción a SOA y servicios web," 3 2007. [Online]. Available: <http://www.ibm.com/developerworks/ssa/webservices/newto/>
- [43] J. Lewis, "Microservices," 2014. [Online]. Available: <http://martinfowler.com/articles/microservices.html>
- [44] S. Newman, "Chapter 1: Microservices," in *Building Microservices*, 2015, pp. 1–11. [Online]. Available: <https://www.google.hr/books?hl=en&lr={\&}id=jjl4BgAAQBAJ{\&}pgis=1{\textbackslash}textbackslash>  
[http://shop.oreilly.com/product/0636920033158.do?cmp=af-code-books-video-product{\\\_.}cj{\\\_.}0636920033158{\\\_.}7739078](http://shop.oreilly.com/product/0636920033158.do?cmp=af-code-books-video-product{\_.}cj{\_.}0636920033158{\_.}7739078)
- [45] A. Simoes, "OEC: The Observatory of Economic Complexity," 2016. [Online]. Available: <http://atlas.media.mit.edu/en/>
- [46] IEEE, "IEEE SA - 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems," 2000. [Online]. Available: <https://standards.ieee.org/findstds/standard/1471-2000.html>
- [47] G. E. D. Vega, "Arquitectura para diseñar e implementar Web Services," pp. 8–18, 3 2016. [Online]. Available: <http://revistas.udistrital.edu.co/ojs/index.php/tia/article/view/9811>
- [48] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a Smart City Internet of Things Platform with Microservice Architecture," in *2015 3rd International Conference on Future Internet of Things and Cloud*. IEEE, 8 2015, pp. 25–30. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7300793>
- [49] C. Cuesta, "Principio de Acoplamiento y Cohesión," 2007. [Online]. Available: [http://kybele.escet.urjc.es/docencia/IS5/2007-2008/Material/{\%}5BIS5-2007-08{\%}5DT2{\\\_.}Recopilacion.pdf](http://kybele.escet.urjc.es/docencia/IS5/2007-2008/Material/{\%}5BIS5-2007-08{\%}5DT2{\_.}Recopilacion.pdf)

- [50] Wikipedia, “List of web service frameworks.”
- [51] L. Jie, *Web Services: European Conference, ECOWS 2004, Erfurt, Germany, September 27-30, 2004, Proceedings*. Springer, 2004. [Online]. Available: <https://books.google.com/books?id=FYX0BwAAQBAJ&pgis=1>
- [52] Microsoft, “Wikipedia .NET Framework,” 2016. [Online]. Available: <https://www.microsoft.com/net>
- [53] D. Cambridge, *The Cambridge Dictionary of Philosophy*. Cambridge University Press.
- [54] J.-P. Margot, *El Nombre de la Rosa o Los Infortunios de la Razón*, 1988. [Online]. Available: <http://www.bdigital.unal.edu.co/24563/1/21752-74480-1-PB.pdf>
- [55] T. .NET, “Announcing .NET 2015 Preview: A New Era for .NET — .NET Blog,” 2014. [Online]. Available: <https://www.visualstudio.com/products/visual-studio-community-vs>
- [56] I. Landwerth, “.NET Core is Open Source — .NET Blog,” 2014. [Online]. Available: <https://blogs.msdn.microsoft.com/dotnet/2014/11/12/net-core-is-open-source/>
- [57] X. Microsoft®, “Mobile App Development & App Creation Software - Xamarin,” 2016. [Online]. Available: <https://www.xamarin.com/>
- [58] Microsoft, “Visual Studio Community 2015,” 2015. [Online]. Available: <https://www.visualstudio.com/products/visual-studio-community-vs>
- [59] ——, “ASP.NET Web API 2.” [Online]. Available: [https://msdn.microsoft.com/es-es/library/dn448365\(v=vs.118\).aspx](https://msdn.microsoft.com/es-es/library/dn448365(v=vs.118).aspx)
- [60] A. .NET, “ASP.NET Web API — The ASP.NET Site.” [Online]. Available: <http://www.asp.net/web-api>
- [61] Microsoft, “Microsoft PowerShell,” 2016. [Online]. Available: <https://msdn.microsoft.com/en-us/powershell>
- [62] K. Spilker, “New book: Windows PowerShell Step by Step, Third Edition — Microsoft Press blog,” 2015. [Online]. Available: [https://blogs.msdn.microsoft.com/microsoft\\_press/2015/10/15/new-book-windows-powershell-step-by-step-third-edition/](https://blogs.msdn.microsoft.com/microsoft_press/2015/10/15/new-book-windows-powershell-step-by-step-third-edition/)
- [63] M. TechNET, “Usar Windows PowerShell,” 2010. [Online]. Available: <https://technet.microsoft.com/es-es/library/cc482998.aspx>
- [64] DNN, “DotNetNuke en Español,” 2016. [Online]. Available: <http://dotnetnuke.com.es/>
- [65] Microsoft, “Servidor web (IIS),” 2013. [Online]. Available: [https://technet.microsoft.com/es-es/library/cc753433\(v=ws.10\).aspx](https://technet.microsoft.com/es-es/library/cc753433(v=ws.10).aspx)
- [66] C. Carrillo, “Owin y Katana,” 2013. [Online]. Available: <https://blogs.msdn.microsoft.com/esmsdn/2013/12/17/owin-y-katana-dos-nuevas-palabras-que-empezaran-a-sonar-mucho/>
- [67] PCI, “PCI DSS v3.0,” 2016. [Online]. Available: <https://es.pcisecuritystandards.org/minisite/en/>
- [68] OWASP, “Top 10 2013-Top 10 - OWASP,” 2013. [Online]. Available: <https://www.owasp.org/index.php/Top10-2013-Top10>

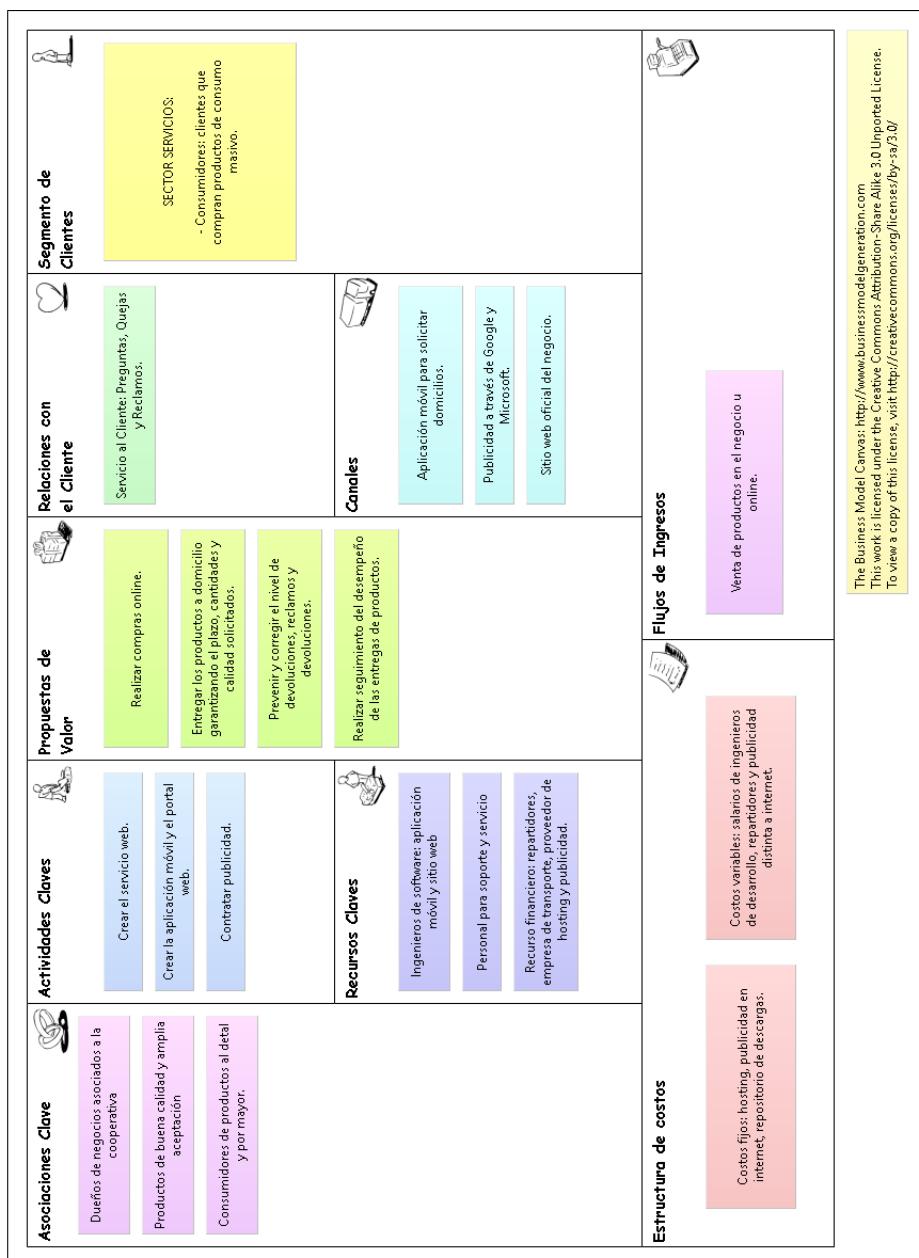
## A. Anexo: Diagrama Modelo Business Model Canvas - BMC



**Figura A-1.: Diagrama Modelo BMC.** Fuente: [5].

Este Canvas está adaptado de The Business Model Canvas con una licencia Creative Commons Attribution-Share Alike 3.0 Unported Licence.

## B. Anexo: BMS de una Cooperativa con Negocios de Productos de Consumo Masivo



**Figura B-1.: BMS de una Cooperativa con Negocios de Productos de Consumo Masivo.**

## C. Anexo: Sectores Económicos en Colombia

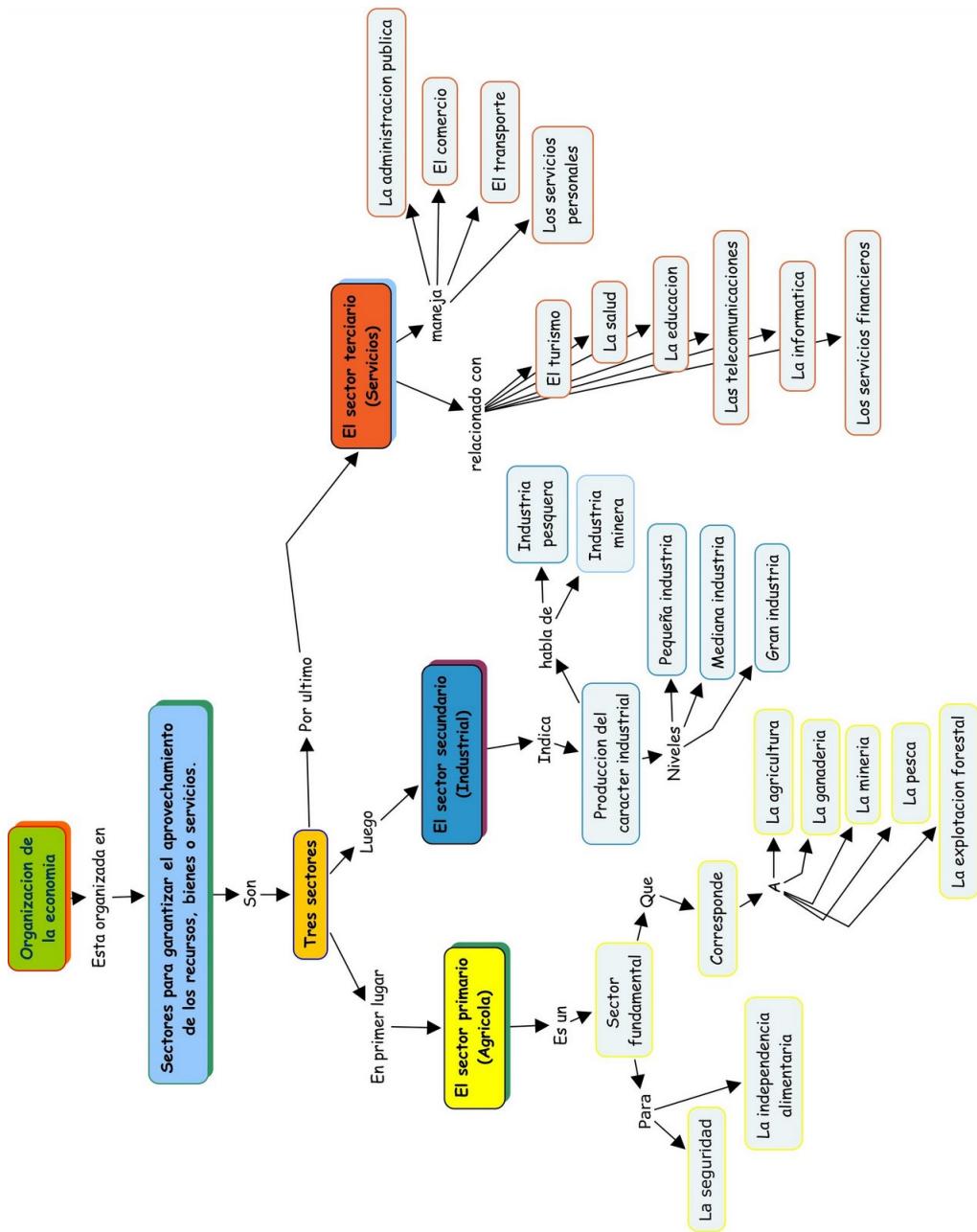


Figura C-1.: Sectores Económicos en Colombia. Fuente: [12]

## D. Anexo: .NET Framework 4.5



Figura D-1.: .NET Framework 4.5. Fuente: [13]

## E. Anexo: Encuesta Aceptación, Uso, Aprendizaje y Desarrollo de Servicios Web

**Aceptación, Uso, Aprendizaje y Desarrollo de Servicios Web**

Este formulario pretende recolectar información sobre servicios web partiendo de su apreciación, experiencia y conocimientos personales, laborales y el entorno que le rodea (su empresa, negocio, comunidad, universidad, etc).

**Nombres y Apellidos** \*

Texto de respuesta breve

**Código de** ▲

Aplica solo para estudiantes de la especialización en ingeniería de software de la Universidad Distrital.

Texto de respuesta breve

E ▲ \*

1. Menor de 25  
2. Entre 25 y 30  
3. Entre 30 y 35  
4. Mayor de 35

**Año finalización** ▲

El año en el que finalizó sus estudios universitarios o profesionales (pregrado). Si no tiene ningún estudio universitario deje en blanco esta casilla.

Texto de respuesta breve

**Figura E-1.:** Formulario Encuesta: Datos del Encuestado

<b>Nivel</b>		
<input type="radio"/> Secundaria		
<input type="radio"/> Pregrado		
<input type="radio"/> Especialización		
<input type="radio"/> Maestría		
<input type="radio"/> Doctorado		
<input type="radio"/> Otros postgrados		
<input type="radio"/> Otra...		
<b>Año finalización último nivel</b>		
Texto de respuesta breve		
<b>Profesi</b>		
<input type="radio"/> Ingeniería de Sistemas o afines		
<input type="radio"/> Ingeniería Electrónica o afines		
<input type="radio"/> Ingeniería Industrial o afines		
<input type="radio"/> Ingeniería Mecánica o afines		
<input type="radio"/> Otras ingenierías		
<input type="radio"/> Matemáticas		
<input type="radio"/> Física		
<input type="radio"/> Otra...		

**Figura E-2.:** Formulario Encuesta: Nivel Académico del Encuestado

**Plataformas que realmente utiliza \***

Múltiple respuesta

JAVA  
 .NET  
 Otra...

**Marcos de trabajo (frameworks), componentes, librerías que realmente utiliza \***

Escriba los frameworks, componentes, librerías o plugins que conoce y realmente utilizó o utiliza en las plataformas de su preferencia.

Texto de respuesta largo

**Ha diseñado, desarrollado o programado componentes o librerías? \***

Si  
 No  
 No las conozco

**Ha diseñado, desarrollado o programado aplicaciones de consola? \***

Si  
 No  
 No las conozco

**Figura E-3.: Formulario Encuesta: Conocimientos del Encuestado Primera Parte**

Ha diseñado, desarrollado o programado aplicaciones de escritorio? \*

Si

No

No las conozco

Ha diseñado, desarrollado o programado aplicaciones web (páginas web con una lógica de negocio)? \*

Si

No

No las conozco

Conoce la diferencia entre servicio web (webservice), arquitectura orientada a servicios SOA y bus de servicios empresariales EBS? \*

Si

No

**Figura E-4.: Formulario Encuesta: Conocimientos del Encuestado Segunda Parte**

Ha consumido o utilizado servicios  \*

Esta pregunta no hace referencia a crear, desarrollar o programar.

Si  
 No

Ha creado, desarrollado o programado servicios  \*

Si  
 No

Nombre las plataformas, frameworks, componentes, librerías, plugins utilizados para crear servicios web.

Deje en blanco esta casilla si aún no conoce o recuerda nada al respecto.

Texto de respuesta largo

**Figura E-5.: Formulario Encuesta: Usos y creación de Servicios Web del Encuestado**

Si ha creado servicios web, indique la cantidad de tiempo que le tomó aprender las tecnologías, frameworks, componentes, librerías o plugins para desarrollar o programar servicios web. \*

Si no ha creado servicios web deje esta pregunta en blanco.

No he estudiado tecnologías para crear servicios web

Menos de 3 meses

6 meses

12 meses

Más de 12 meses

Otra...

Si ha creado servicios web, indique la cantidad de tiempo que les está tomando o les llevó a usted o su equipo desarrollar o programar el proyecto más largo de servicios web. \*

No responda esta pregunta si usted no ha creado o participado aún en un proyecto para crear un servicio web.

No he creado servicios web

Menos de 3 meses

Entre 3 y 6 meses

Entre 6 y 9 meses

Entre 9 y 12 meses

Más de 12 meses

Otra...

Figura E-6.: Formulario Encuesta: Tiempos de Aprendizaje y creación de Servicios Web del Encuestado

Considera usted que es viable económicamente incluir en un proyecto de software el diseño y desarrollo de servicios web como parte de la solución? \*

Si  
 No

Los costos de un proyecto de software con servicios web \* ▲ \*

Muy altos con sobre costos  
 Altos  
 Semejantes a los de otros proyectos sin servicios web  
 Bajos  
 Muy bajos

Incluiría usted en un proyecto de software el diseño y desarrollo de servicios web como parte de la solución? \*

Si  
 No

**Figura E-7.: Formulario Encuesta: Viabilidad de los Servicios Web para el Encuestado**

Indique el grado de rechazo o aceptación personal o su negocio con respecto a los servicios web. \*

- Jamás lo tendré en cuenta
- Es una pérdida de tiempo tenerlo en cuenta
- No me parece muy útil
- Comenzaré a tenerlo en cuenta
- Es muy útil
- Siempre lo tendré en cuenta

Indique el grado de rechazo o aceptación de los profesionales de software que lo rodean o la empresa en la que trabaja con respecto a los servicios web. \*

- Totalmente rechazado o nunca lo incluyen en un proyecto
- Lo rechazan por desconocimiento
- Lo rechazan así se lo indiquen utilizar
- No lo rechazan si se lo indican utilizar
- Lo utilizan esporádicamente
- Lo utilizan siempre que pueden
- Siempre lo utilizan

Opinión abierta sobre lo que quiera aportar con respecto a los servicios web \*

Texto de respuesta largo

Figura E-8.: Formulario Encuesta: Aceptación de los Servicios Web por el Encuestado

## F. Anexo: Código Controladora de Aplicaciones

Listing F.1: AppController.cs

```
1 namespace IGD.FullWebServices.Controllers
2 {
3     #region using
4
5     using DotNetNuke.Web.Api;
6     using System;
7     using System.Collections.Generic;
8     using System.Linq;
9     using System.Net;
10    using System.Net.Http;
11    using System.Web.Http;
12
13    #endregion
14
15    /// <summary>
16    /// Expone las operaciones HTTP para administrar aplicaciones.
17    /// </summary>
18    public class AppController : DnnApiController
19    {
20        /// <summary>
21        /// Obtiene la informacion de la aplicacion a partir del identificador.
22        /// </summary>
23        /// <param name="appId">Identificador de la aplicacion.</param>
24        /// <returns>Resultado de la operacion</returns>
25        public HttpResponseMessage Get(Guid appId)
26        {
27            appId.ThrowOperationExceptionIfTrue(!appId.ValidGuid(), ResponseCode.
28                InvalidApplicationGuid);
29            AppInfo appInfo = AppRepository.Get(appId);
30            return this.Request.CreateResponse(HttpStatusCode.OK, appInfo);
31        }
32
33        /// <summary>
34        /// Obtiene la informacion de todas las aplicaciones.
35        /// </summary>
36        /// <returns>Resultado de la operacion</returns>
37        public HttpResponseMessage Get()
38        {
39            IEnumerable<AppInfo> appsInfo = AppRepository.Get();
40            return this.Request.CreateResponse(HttpStatusCode.OK, appsInfo.ToList());
41        }
42
43        /// <summary>
44        /// Registra una nueva aplicacion.
45        /// </summary>
46        /// <param name="request">Informacion de la aplicacion.</param>
47        /// <returns>Resultado de la operacion</returns>
48        /// <remarks>Como el CAST del body se hace a traves de los que ofrece WebApi
49        /// si se envia un GUID invalido el CAST lo asumira como Empty y se creara con
50        /// GUID
```

```
49     /// autogenerado.</remarks>
50     public HttpResponseMessage Post([FromBody]AppInfo request)
51     {
52         new AppValidator().ThrowExceptionIfNotValid(request, ResponseCode.
53             InvalidApplicationInfo);
53         Guid appId = AppRepository.Save(request);
54         return this.Request.CreateResponse(HttpStatusCode.Created, appId);
55     }
56
57     /// <summary>
58     /// Actualiza la informacion de la aplicacion.
59     /// </summary>
60     /// <param name="request">Informacion de la aplicacion.</param>
61     /// <returns>Resultado de la operacion</returns>
62     public HttpResponseMessage Put([FromBody]AppInfo request)
63     {
64         new AppValidator().ThrowExceptionIfNotValid(request, ResponseCode.
65             InvalidApplicationInfo);
65         AppRepository.Update(request);
66         return this.Request.CreateResponse(HttpStatusCode.OK);
67     }
68
69     /// <summary>
70     /// Marca la informacion de un emisor especifico como desabilitado.
71     /// </summary>
72     /// <param name="appId">Identificador unico que identifica a la aplicacion.</
72     param>
73     /// <returns>Response con el estado de la peticion.</returns>
74     public HttpResponseMessage Delete(Guid appId)
75     {
76         AppRepository.Delete(appId);
77         return this.Request.CreateResponse(HttpStatusCode.OK);
78     }
79 }
80 }
```

## G. Anexo: Código y Comandos de PowerShell

### Prueba Unitaria Automatizada de la Controladora de Aplicaciones

Listing G.1: PruebaUnitariaControladoraAplicaciones

```
1 $Script:Config = Import-LocalizedData -FileName 'Config.psd1',
2 Import-Module '.\Functions' -Force
3
4 Describe 'Validaciones EndPoint' {
5
6     It 'El servicio esta disponible' {
7         $ComputerName = (New-Object -TypeName System.Uri -ArgumentList $Script:Config.
8             ServiceEndPoint).Host
9         (Test-NetConnection -CommonTCPPort HTTP -ComputerName $ComputerName).
10            PingSucceeded | Should Be $true
11    }
12
13    It 'Invocar al servicio sin especificar una clase Controller debe generar un error
14        404 (Not Found)'{
15        (Invoke-Controller -InputObject @{$Uri=$Script:Config.ServiceEndPoint;Method='Get
16            '}).StatusCode | Should Be 404
17    }
18
19 }
20
21 Describe 'Validaciones AppController' {
22
23     $AppControllerUri = '{0}/{1}' -f $Script:Config.ServiceEndPoint, 'App'
24     $AuthorizedHeader = New-BasicAuthHeader
25     $UnauthorizedHeader = New-BasicAuthHeader -Username (Get-RandomString) -Password (
26         Get-RandomString)
27
28     Context 'Listar aplicaciones' {
29
30         It 'Obtener la lista de todas las aplicaciones con un usuario autorizado' {
31             $Response = Invoke-Controller -InputObject @{$Uri=$AppControllerUri;Method='
32                 Get';Headers=$AuthorizedHeader}
33             $Response.StatusCode | Should Be 200
34             $Response.Content | Should Not BeNullOrEmpty
35             {$Script:AppList = ($Response.Content | ConvertFrom-Json)} | Should Not
36                 Throw
37             {$Response.Content | Assert-Schema -SchemaFileName 'AppInfoSchema.json' -
38                 AsArray} | Should Not Throw
39     }
40
41     It 'Obtener la lista de todas las aplicaciones con un usuario no autorizado debe
42         generar un error 401 (Unauthorized)' {
43         (Invoke-Controller -InputObject @{$Uri=$AppControllerUri;Method='Get';Headers
44             ='$UnauthorizedHeader}).StatusCode | Should Be 401
45     }
46 }
```

```

35     It 'Obtener la lista de todas las aplicaciones sin usuario y clave debe generar
36         un error 401 (Unauthorized)' {
37         (Invoke-Controller -InputObject @{$Uri=$AppControllerUri;Method='Get';Headers=
38             =$null}).StatusCode | Should Be 401
39     }
40
41     It 'Obtener la informacion de cada aplicacion con un usuario autorizado' {
42         $Script:AppList | ForEach-Object {
43             $AppUri = '{0}/Get?appId={1}' -f $AppControllerUri, $_.ApplicationGuid
44             (Invoke-Controller -InputObject @{$Uri=$AppUri;Method='Get';Headers=
45                 $AuthorizedHeader}).StatusCode | Should Be 200
46             $Script>LastAppUri = $AppUri
47         }
48     }
49
50     It 'Obtener la informacion de una aplicacion existente con un usuario no
51         autorizado debe generar un error 401 (Unauthorized)' {
52         (Invoke-Controller -InputObject @{$Uri=$Script>LastAppUri;Method='Get';
53             Headers=$UnauthorizedHeader}).StatusCode | Should Be 401
54     }
55
56     It 'Obtener la informacion de una aplicacion no existente debe generar un error
57         404 (Not Found)' {
58         $FakeAppUri = '{0}/Get?appId={1}' -f $AppControllerUri, ([System.Guid]::
59             NewGuid()).ToString('N')
60         (Invoke-Controller -InputObject @{$Uri=$Script>LastAppUri;Method='Get';
61             Headers=$AuthorizedHeader}).StatusCode | Should Be 404
62     }
63
64     It 'Obtener la informacion de una aplicacion con identificador mal formado (no
65         GUID) debe generar un error 404 (Not Found)' {
66         $FakeAppUri = '{0}/Get?appId={1}' -f $AppControllerUri, (Get-RandomString)
67         (Invoke-Controller -InputObject @{$Uri=$Script>LastAppUri;Method='Get';
68             Headers=$AuthorizedHeader}).StatusCode | Should Be 404
69     }
70
71     It 'Obtener la informacion de una aplicacion sin enviar el identificador de la
72         aplicacion debe generar un error 400 (Bad Request)' {
73         $FakeAppUri = '{0}/Get?appId={1}' -f $AppControllerUri, [string]::Empty
74         (Invoke-Controller -InputObject @{$Uri=$Script>LastAppUri;Method='Get';
75             Headers=$AuthorizedHeader}).StatusCode | Should Be 400
76     }
77
78     Context 'Crear aplicaciones' {
79         $AppGuid = [System.Guid]::NewGuid().ToString('N')
80         $AppName = 'App demo Pester'
81
82         It 'Crear una aplicacion con datos correctos utilizando un usuario no autorizado
83             debe generar un error 401 (Unauthorized)' {
84             $Body = (@{ApplicationGuid=$AppGuid;Name=$AppName}) | ConvertTo-Json
85             (Invoke-Controller -InputObject @{$Uri=$AppControllerUri;Method='Post';
86                 Headers=$UnauthorizedHeader;Body=$Body}).StatusCode | Should Be 401
87         }
88
89         It 'Crear una aplicacion con datos correctos utilizando un usuario autorizado' {
90             $Body = (@{ApplicationGuid=$AppGuid;Name=$AppName}) | ConvertTo-Json
91             (Invoke-Controller -InputObject @{$Uri=$AppControllerUri;Method='Post';
92                 Headers=$AuthorizedHeader;Body=$Body}).StatusCode | Should Be 201
93         }
94     }

```

```
81     It 'Crear una aplicacion con el mismo nombre de una que ya existe debe generar
82         el error 409 (Conflict)' {
83         $Body = (@{ApplicationGuid=$AppGuid;Name=$AppName})| ConvertTo-Json
84         (Invoke-Controller -InputObject @{Uri=$AppControllerUri;Method='Post';
85             Headers=$AuthorizedHeader;Body=$Body}).StatusCode | Should Be 409
86     }
87
88     It 'Crear una aplicacion solo con el nombre utilizando un usuario autorizado' {
89         $RandomName = '{0} {1}' -f $AppName, (Get-RandomString -Length 10)
90         $Body = (@{Name=$AppName})| ConvertTo-Json
91         (Invoke-Controller -InputObject @{Uri=$AppControllerUri;Method='Post';
92             Headers=$AuthorizedHeader;Body=$Body}).StatusCode | Should Be 201
93     }
94
95     It 'Crear una aplicacion con un identificador (ApplicationGuid) que ya existe
96         debe generar un error 409 (Conflict)' {
97         $RandomName = '{0} {1}' -f $AppName, (Get-RandomString -Length 10)
98         $Body = (@{ApplicationGuid=$AppGuid;Name=$RandomName})| ConvertTo-Json
99         (Invoke-Controller -InputObject @{Uri=$AppControllerUri;Method='Post';
100             Headers=$AuthorizedHeader;Body=$Body}).StatusCode | Should Be 409
101     }
102
103     It 'Crear una aplicacion con un identificador (ApplicationGuid) mal formado (no
104         GUID) debe generar un error 400 (Bad Request)' {
105         $RandomName = '{0} {1}' -f $AppName, (Get-RandomString -Length 10)
106         $Body = (@{ApplicationGuid=(Get-RandomString);Name=$RandomName})| ConvertTo-
107             Json
108         (Invoke-Controller -InputObject @{Uri=$AppControllerUri;Method='Post';
109             Headers=$AuthorizedHeader;Body=$Body}).StatusCode | Should Be 400
110     }
111 }
```

---