

**UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA**

**ESCUELA DE POSGRADO**

**UNIDAD DE POSGRADO DE LA FACULTAD DE INGENIERÍA DE PRODUCCIÓN  
Y SERVICIOS**



**FRAMEWORK PARA LA CONSTRUCCIÓN DE SOFTWARE EN SISTEMAS  
DE INFORMACIÓN ORIENTADOS A LA PRESTACIÓN DE SERVICIOS  
(MODELO SAAS)**

Tesis presentada por el Bachiller

**EDGAR FRANK LIZÁRRAGA UGARTE**

Para optar el Grado de Maestro en Ingeniería  
de Sistemas, con Mención en Ingeniería de  
Software.

Asesor:

Mg. Dante Velásquez Contreras

**AREQUIPA – PERÚ**

**2016**

*A mi Madre, la Sra. Presy Ugarte de Lizárraga por ser un gran ejemplo de guía, valor, coraje, dedicación, amor y porque siempre estuvo a mi lado brindándome su apoyo y sus consejos para hacer de mí una mejor persona.*

*A mi Padre, el Sr. Miguel Ángel Lizárraga Febres, porque sin él no hubiéramos podido ser los profesionales y personas que llegamos a ser.*

*A mi hermano, Miguel Ángel Lizárraga Ugarte, por ser un líder, un apoyo, ejemplo como padre y un gran hermano.*

*A mi hermano, William Robert Lizárraga Ugarte, por ser un modelo de aptitudes, conocimientos, perseverancia y un gran hermano.*

*A esta, mi extraordinaria familia.*

*Edgar Frank Lizárraga Ugarte*

## **AGRADECIMIENTOS**

*Gracias a Dios, por haberme otorgado una familia extraordinaria, quienes han creído en mí siempre, dándome ejemplo de superación, humildad, respeto y sacrificio, enseñándome a valorar todo lo que tengo.*

*Al Ingeniero Dante Velásquez y al Ingeniero Cesar Baluarte Araya, por su apoyo en la realización del presente trabajo.*

*A mis profesores de la Maestría de Ingeniería de Software por brindarme los conocimientos necesarios para dar impulso a una excelente profesión.*

*A la Srta. Helen Bellido Ocampo por su ayuda y entusiasmo en la elaboración de la presente investigación.*

*A las personas que alguna vez invirtieron parte de su tiempo para la realización de esta investigación.*

*A mis amigos.*

# ÍNDICE

LISTA DE FIGURAS.....	VIII
LISTA DE TABLAS.....	XIII
RESUMEN.....	XVI
ABSTRACT.....	XVII
INTRODUCCIÓN.....	XVIII
<b>CAPÍTULO 1: PLANTEAMIENTO TEÓRICO.....</b>	<b>1</b>
1.1. INTRODUCCIÓN.....	1
1.1.1. Enunciado del problema.....	1
1.1.2. Formulación Interrogativa del problema.....	2
1.2. ÁREA CIENTÍFICA.....	2
1.3. OBJETIVOS.....	2
1.3.1. Objetivo general.....	2
1.3.2. Objetivos específicos.....	2
1.4. HIPÓTESIS.....	3
1.5. VIABILIDAD DE LA INVESTIGACIÓN.....	3
1.5.1. Económica.....	3
1.5.2. Técnica.....	3
1.5.3. Operativa.....	4
1.6. DESCRIPCIÓN DEL PROBLEMA.....	4
1.7. JUSTIFICACIÓN.....	4
1.8. DELIMITACIONES EN LA INVESTIGACIÓN.....	5
1.9. ÁMBITO.....	6
1.9.1. Ubicación.....	6
1.9.2. Población.....	6
1.10. VARIABLES.....	6
1.10.1. Variable Independiente.....	6
1.10.2. Variable Dependiente.....	7

1.11. PLANTEAMIENTO OPERACIONAL.....	7
1.11.2. Técnicas, Instrumentos y Materiales de verificación.....	7
<b>CAPÍTULO 2: MARCO TEÓRICO.....</b>	<b>8</b>
2.1. ANTECEDENTES INVESTIGATIVOS.....	8
2.2. MARCO CONCEPTUAL.....	11
2.2.1. Framework.....	11
2.2.2. Cloud Computing.....	16
2.2.3. Modelos de Servicio en Cloud Computing.....	20
2.2.4. Software as a Service (Modelo SaaS).....	25
2.2.5. Implementación SaaS.....	33
2.2.6. Multitenancy.....	37
2.2.7. MYPE.....	38
2.2.8. Tecnología web.....	39
2.2.9. BPM y SOA.....	50
2.3. ESTADO DEL ARTE.....	53
<b>CAPÍTULO 3: DESARROLLO DEL FRAMEWORK.....</b>	<b>56</b>
3.1. INTRODUCCIÓN.....	56
3.2. ANÁLISIS DEL DOMINIO.....	57
3.2.1. Requerimientos del Framework.....	57
3.3. DESARROLLO DEL FRAMEWORK.....	59
3.3.1. Análisis, Diseño y Construcción del Framework.....	59
a. Diagrama de Casos de Uso.....	59
b. Especificación de Casos de Uso.....	60
c. Diagramas de Secuencia.....	70
d. Diagrama de Clases.....	78
e. Diagrama de Componentes.....	79
f. Diagrama de Despliegue.....	80

g. Diagrama Entidad Relación (E-R).....	81
h. Diccionario de Base de Datos.....	82
i. Selección de la Plataforma Tecnológica.....	85
j. Arquitectura del Framework.....	89
3.3.2. Pruebas del Framework.....	91
a. Especificación de casos de prueba.....	91
b. Objetivos de los casos de prueba.....	92
c. Plan de pruebas.....	92
d. Tipos de pruebas.....	92
e. Alcance.....	92
f. Ejecución de pruebas.....	93
g. Resultado de pruebas.....	106
3.4. INSTANCIA DEL FRAMEWORK.....	107
<b>CAPÍTULO 4: DESARROLLO DEL SISTEMA DE INFORMACIÓN.....</b>	<b>108</b>
4.1. INTRODUCCIÓN.....	108
4.2. DESARROLLO DEL SISTEMA DE INFORMACIÓN.....	109
4.2.1. Modelado del Negocio.....	109
4.2.2. Identificación del Problema.....	111
4.2.3. Requerimientos.....	111
4.2.4. Casos de Uso.....	117
4.2.5. Especificación de casos de uso.....	118
4.2.6. Diagramas de secuencia.....	130
4.2.7. Diagrama de clases.....	141
4.2.8. Diagrama Entidad Relación (E-R).....	142
4.2.9. Construcción del Sistema.....	143
4.2.10. Pruebas del Sistema.....	143

<b>CAPÍTULO 5: PRUEBAS Y RESULTADOS.....</b>	145
5.1. INTRODUCCIÓN.....	145
5.2. EVALUACIÓN DE EXPERTOS.....	146
5.3. PERFIL DE EXPERTOS.....	146
5.4. INTERPRETACIÓN DE EVALUACIÓN DE EXPERTOS.....	147
<b>CONCLUSIONES.....</b>	160
<b>RECOMENDACIONES.....</b>	161
<b>BIBLIOGRAFÍA.....</b>	162
<b>ANEXOS.....</b>	166
ANEXO A - MANUAL TÉCNICO DEL FRAMEWORK.....	166
ANEXO B - CÓDIGO FUENTE DEL FRAMEWORK.....	198
ANEXO C - IMPLEMENTACIÓN DEL SUBSISTEMA DE INFORMACIÓN ORIENTADO A LA PRESTACIÓN DE SERVICIOS (MODELO SAAS).....	231
ANEXO D - CUESTIONARIO.....	238
ANEXO E - GLOSARIO DE TÉRMINOS.....	241

## **LISTA DE FIGURAS**

<b>CAPÍTULO 2: MARCO TEÓRICO.....</b>	<b>8</b>
Figura 1: Puntos calientes y fríos de un Framework.....	14
Figura 2: Proceso de desarrollo del Framework.....	14
Figura 3: Actividades y artefactos durante el diseño e instancia de un Framework.....	15
Figura 4: Computación en la nube.....	19
Figura 5: Modelos de servicio - Funcionalidad.....	24
Figura 6: Interacción en los modelos de servicio.....	25
Figura 7: Implementación de SaaS - Una base de datos por cada empresa o usuario.....	34
Figura 8: Implementación de SaaS - Una base de datos con N conjunto de tablas.....	36
Figura 9: Implementación de SaaS - Una base de datos con único conjunto de tablas.....	37
Figura 10: Arquitectura Multitenancy.....	38
Figura 11: Ejemplo Introductorio Nº 1.....	40
Figura 12: Servicio.....	51
<b>CAPÍTULO 3: IMPLEMENTACIÓN DEL FRAMEWORK.....</b>	<b>56</b>
Figura 13: Diagrama de Casos de Uso del Framework.....	59
Figura 14: Diagrama de Secuencia - Utilizar el aplicativo web del Framework.....	70
Figura 15: Diagrama de Secuencia - Adaptar el repositorio de datos.....	70
Figura 16: Diagrama de Secuencia - Ingresar argumentos de configuración.....	71
Figura 17: Diagrama de Secuencia - Usar Patrón MVC.....	71
Figura 18: Diagrama de Secuencia - Usar la implementación Multitenencia.....	72
Figura 19: Diagrama de Secuencia - Interconectar componentes de software.....	72
Figura 20: Diagrama de Secuencia - Crear la arquitectura del Framework.....	73
Figura 21: Diagrama de Secuencia - Verifica los argumentos de configuración.....	73
Figura 22: Diagrama de Secuencia - Generar el código fuente inicial.....	74

Figura 23:	Diagrama de Secuencia - Utilizar el Framework.....	74
Figura 24:	Diagrama de Secuencia - Procesar solicitud.....	75
Figura 25:	Diagrama de Secuencia - Generar archivos MVC.....	76
Figura 26:	Diagrama de Secuencia - Construir software.....	77
Figura 27:	Diagrama de Secuencia - Ingresar al portal principal.....	77
Figura 28:	Diagrama de Clases del Framework.....	78
Figura 29:	Diagrama de Componentes del Framework.....	79
Figura 30:	Diagrama de Despliegue del Framework.....	80
Figura 31:	Diagrama Entidad Relación (E-R).....	81
Figura 32:	Beneficios del SaaS.....	87
Figura 33:	Arquitectura del Framework.....	89
Figura 34:	Compatibilidad web del Framework (forma automatizada).....	98
Figura 35:	Resultado Nro. 1 de Compatibilidad web del Framework (forma automatizada).....	98
Figura 36:	Resultado Nro. 2 de Compatibilidad web del Framework (forma automatizada).....	99
Figura 37:	Resultado Nro. 3 de Compatibilidad web del Framework (forma automatizada).....	99
Figura 38:	Resultado Nro. 4 de Compatibilidad web del Framework (forma automatizada).....	100
Figura 39:	Resultado Nro. 5 de Compatibilidad web del Framework (forma automatizada).....	100
Figura 40:	Resultado Nro. 6 de Compatibilidad web del Framework (forma automatizada).....	101
Figura 41:	Plan de pruebas en JMeter para módulo de Login.....	103
Figura 42:	Plan de pruebas en JMeter para módulo de registro de tenant o cliente.....	104
Figura 43:	Tablas con campo de tenant o cliente.....	106
<b>CAPÍTULO 4: DESARROLLO DEL SISTEMA DE INFORMACIÓN.....</b>		108
Figura 44:	EMV - Lista de artículos por almacén.....	110
Figura 45:	EMV - Lista de GPS y CHIP por almacén.....	110

Figura 46:	Lista de equipos GPS.....	112
Figura 47:	Lista de CHIP.....	113
Figura 48:	Diagrama BPM - Proceso actual del movimiento de almacén.....	115
Figura 49:	Diagrama BPM - Proceso propuesto para compras.....	115
Figura 50:	Diagrama BPM - Proceso propuesto para el ingreso a almacén.....	116
Figura 51:	Diagrama BPM - Proceso propuesto para la salida de almacén.....	116
Figura 52:	Diagrama de Casos de Uso del Sistema.....	117
Figura 53:	Diagrama de Secuencia - Registra empresa.....	130
Figura 54:	Diagrama de Secuencia - Login e ingreso al sistema.....	131
Figura 55:	Diagrama de Secuencia - Mantenimiento de entidades principales.....	132
Figura 56:	Diagrama de Secuencia - Registro de compras.....	133
Figura 57:	Diagrama de Secuencia - Registro de nota de ingreso por reingreso.....	134
Figura 58:	Diagrama de Secuencia - Registro de nota de ingreso por transferencia.....	135
Figura 59:	Diagrama de Secuencia - Registro de nota de ingreso por compra.....	136
Figura 60:	Diagrama de Secuencia - Registro de orden de salida.....	137
Figura 61:	Diagrama de Secuencia - Aprueba orden de salida de almacén.....	138
Figura 62:	Diagrama de Secuencia - Genera nota de salida.....	139
Figura 63:	Diagrama de Secuencia - Actualizar inventario.....	140
Figura 64:	Diagrama de Secuencia - Generar reportes.....	140
Figura 65:	Diagrama de clases.....	141
Figura 66:	Diagrama Entidad - Relación (E-R).....	142
<b>CAPÍTULO 5: PRUEBAS Y RESULTADOS.....</b>		145
Figura 67:	Framework en navegadores web.....	147
Figura 68:	Patrón de diseño MVC.....	148
Figura 69:	Utilización del modelo SaaS.....	149
Figura 70:	Creación o instancia de una aplicación.....	150
Figura 71:	Curva de aprendizaje del Framework.....	151
Figura 72:	Construcción de software en un sistema de información.....	152

Figura 73:	Agilidad en construir software.....	153
Figura 74:	Número de instancias del Framework.....	154
Figura 75:	Estructura del Framework.....	155
Figura 76:	Mantenimiento del Framework.....	156
Figura 77:	Modelo SaaS (Software as a Service).....	157
Figura 78:	Opinión sobre Frameworks.....	158
Figura 79:	Incremento del Software como Servicio.....	159
<b>ANEXOS</b>		166
<b>ANEXO A: MANUAL TÉCNICO DEL FRAMEWORK</b>		166
Figura 80:	Estructura de carpetas.....	166
Figura 81:	Estándar de Carpeta Principal.....	168
Figura 82:	Estructura de carpeta pages.....	168
Figura 83:	Estándar de Carpeta - Archivo.....	170
Figura 84:	Herramientas del Sistema de Información.....	178
Figura 85:	Carpetas - archivos que contienen a la página web.....	182
Figura 86:	Acceso al Framework.....	186
Figura 87:	Aplicativo Web - Pantalla Inicial.....	186
Figura 88:	Aplicativo Web - Argumentos de configuración.....	187
Figura 89:	Aplicativo Web - Descargas.....	188
Figura 90:	Aplicativo Web - Modelo Vista Controlador.....	188
Figura 91:	Login del software instanciado.....	189
Figura 92:	Estructura Principal del software instanciado.....	189
Figura 93:	Registro de un tenant.....	190
Figura 94:	Código Fuente del archivo de Configuración.....	191
Figura 95:	Código Fuente del archivo de Variables de Entorno.....	192
Figura 96:	Diseño de la Vista - Mproducto.....	193
Figura 97:	Código fuente de la Vista - Mproducto.....	193
Figura 98:	Código fuente del Modelo Mproducto.....	195

Figura 99:	Código fuente del controlador - Mproducto.....	196
Figura 100:	Registro de opción de menú.....	197
<b>ANEXO C: IMPLEMENTACIÓN DEL SUBSISTEMA DE INFORMACIÓN ORIENTADO A LA PRESTACIÓN DE SERVICIOS (MODELO SAAS).....</b>		231
Figura 101:	Login del sistema de información instanciado.....	231
Figura 102:	Listado de productos de un primer cliente.....	232
Figura 103:	Listado de productos de un segundo cliente.....	232
Figura 104:	Registro de compras.....	233
Figura 105:	Selección de productos.....	233
Figura 106:	Listado de compras.....	234
Figura 107:	Nueva nota de ingreso por compra.....	234
Figura 108:	Impresión de nota de ingreso.....	235
Figura 109:	Orden de salida por operación interna.....	235
Figura 110:	Listado de órdenes de salida sin aprobar.....	236
Figura 111:	Ingreso de productos por transferencia de almacén.....	236
Figura 112:	Consulta del movimiento de los productos.....	237

## **LISTA DE TABLAS**

<b>CAPÍTULO 1: PLANTEAMIENTO TEÓRICO.....</b>	<b>1</b>
Tabla 1:     Viabilidad económica.....	3
<b>CAPÍTULO 3: IMPLEMENTACIÓN DEL FRAMEWORK.....</b>	<b>56</b>
Tabla 2:     Caso de Uso - Utilizar el aplicativo web del Framework.....	60
Tabla 3:     Caso de Uso - Adaptar el repositorio de datos.....	61
Tabla 4:     Caso de Uso - Ingresar argumentos de configuración.....	62
Tabla 5:     Caso de Uso - Usar Patrón MVC.....	62
Tabla 6:     Caso de Uso - Usar la implementación Multitenencia.....	63
Tabla 7:     Caso de Uso - Interconectar componentes.....	64
Tabla 8:     Caso de Uso - Crear la arquitectura del Framework.....	64
Tabla 9:     Caso de Uso - Verifica los argumentos de configuración.....	65
Tabla 10:    Caso de Uso - Generar el código fuente inicial.....	66
Tabla 11:    Caso de Uso - Utilizar el Framework.....	66
Tabla 12:    Caso de Uso - Procesar solicitud.....	67
Tabla 13:    Caso de Uso - Generar archivos MVC.....	68
Tabla 14:    Caso de Uso - Construir software.....	68
Tabla 15:    Caso de Uso - Ingresar al portal principal.....	69
Tabla 16:    Base de Datos - Tabla AuditoriaMaestros.....	82
Tabla 17:    Base de Datos - Tabla Mestado.....	82
Tabla 18:    Base de Datos - Tabla Mmenu.....	82
Tabla 19:    Base de Datos - Tabla Mperfil.....	82
Tabla 20:    Base de Datos - Tabla Mtenant.....	83
Tabla 21:    Base de Datos - Tabla Musuario.....	83
Tabla 22:    Base de Datos - Tabla MusuarioMenu.....	83
Tabla 23:    Base de Datos - Claves Primarias.....	83
Tabla 24:    Base de Datos - Claves Foráneas.....	84

Tabla 25:	Base de Datos - Procedimientos almacenados.....	84
Tabla 26:	Cuadro comparativo de Gestores de Base de Datos.....	87
Tabla 27:	Prueba unitaria - Ejecución de script de base de datos.....	93
Tabla 28:	Prueba unitaria - Crear recursos para la instancia del Framework.....	94
Tabla 29:	Prueba unitaria - Creación de archivos MVC.....	95
Tabla 30:	Prueba unitaria - Ingresar al sistema instanciado.....	96
Tabla 31:	Prueba unitaria - Registrar un tenant o cliente.....	96
Tabla 32:	Compatibilidad web del Framework (forma manual).....	97
Tabla 33:	Pruebas de concurrencia y carga a módulo de Login.....	103
Tabla 34:	Pruebas de concurrencia y carga a módulo de registro de tenant o cliente.....	105
Tabla 35:	Características mínimas y recomendadas del equipo para utilizar el Framework.....	107
<b>CAPÍTULO 4: IMPLEMENTACIÓN DEL SISTEMA DE INFORMACIÓN.....</b>		108
Tabla 36:	Caso de Uso - Registrar empresa.....	118
Tabla 37:	Caso de Uso - Login - Ingreso al Sistema.....	118
Tabla 38:	Caso de Uso - Mantenimiento de usuarios.....	119
Tabla 39:	Caso de Uso - Mantenimiento de productos.....	120
Tabla 40:	Caso de Uso - Registro de compras.....	121
Tabla 41:	Caso de Uso - Registro de nota de ingreso por reingreso.....	122
Tabla 42:	Caso de Uso - Registro de nota de ingreso por transferencia.....	123
Tabla 43:	Caso de Uso - Registro de nota de ingreso por compra.....	124
Tabla 44:	Caso de Uso - Registro de orden de salida por operación interna.....	125
Tabla 45:	Caso de Uso - Registro de orden de salida por transferencia.....	126
Tabla 46:	Caso de Uso - Aprobación de orden de salida de almacén.....	127
Tabla 47:	Caso de Uso - Genera nota de salida.....	128
Tabla 48:	Caso de Uso - Actualizar inventario.....	128
Tabla 49:	Caso de Uso - Generar reportes.....	129
Tabla 50:	Prueba unitaria - verificación de datos.....	144

<b>CAPÍTULO 5: PRUEBAS Y RESULTADOS.....</b>	145
Tabla 51: Framework en navegadores web.....	147
Tabla 52: Patrón de diseño MVC.....	148
Tabla 53: Utilización del modelo SaaS.....	149
Tabla 54: Creación o instancia de una aplicación.....	150
Tabla 55: Curva de aprendizaje del Framework.....	151
Tabla 56: Construcción de software en un sistema de información.....	152
Tabla 57: Agilidad en construir software.....	153
Tabla 58: Número de instancias del Framework.....	154
Tabla 59: Estructura del Framework.....	155
Tabla 60: Mantenimiento del Framework.....	156
Tabla 61: Modelo SaaS (Software as a service).....	157
Tabla 62: Opinión sobre Frameworks.....	158
Tabla 63: Incremento del Software como Servicio.....	159
<b>ANEXO A: MANUAL TÉCNICO DEL FRAMEWORK.....</b>	166
Tabla 64: Descripción de Carpetas del Framework.....	167
Tabla 65: Definición de carpetas en pages.....	170
Tabla 66: Descripción de métodos de la clase Generador.....	172
Tabla 67: Descripción de métodos de la clase Modelo.....	173
Tabla 68: Descripción de métodos de la clase Vista.....	174
Tabla 69: Descripción de métodos de la clase mssql.....	175
Tabla 70: Descripción de métodos de la clase Mmenu.....	175
Tabla 71: Descripción de métodos de la clase Musuario.....	176
Tabla 72: Descripción de Función de la clase Mreporte.....	176
Tabla 73: Descripción de Funciones de archivo _cabecera.js.....	178
Tabla 74: Descripción de constantes para acceder a Base de Datos.....	179
Tabla 75: Descripción de constantes principales del Framework.....	181
Tabla 76: Descripción de constantes creadas por el desarrollador.....	182
Tabla 77: Nombrar a tipos de datos.....	184

## **RESUMEN**

La presente investigación se basa en el desarrollo de un FRAMEWORK PARA LA CONSTRUCCIÓN DE SISTEMAS DE INFORMACIÓN ORIENTADOS A LA PRESTACIÓN DE SERVICIOS (modelo SaaS) específicamente para brindar un marco de trabajo que agilice la construcción de software orientado al modelo de distribución SaaS (del inglés, Software as a Service).

El desarrollo del Framework se enfoca en el patrón de diseño Modelo - Vista - Controlador, el cual se encarga de separar el acceso a los datos, la presentación y la lógica de negocios en módulos, archivos o extensiones independientes, logrando así, bastante orden y modularidad en la construcción de software, sin embargo, lo que realmente hace que el Framework sirva para la construcción de Sistemas orientados al modelo SaaS, es la utilización de la arquitectura Multitenant (multipropietario), la cual una sola instancia del sistema o aplicación aparece en el servidor, siendo utilizada por múltiples clientes, organizaciones o propietarios (tenant) desde una conexión a Internet, adicionalmente, para el desarrollo del Framework se utilizaron diversas tecnologías web como el lenguaje de programación PHP, el gestor de base de datos SQL Server y el servidor web Apache. El desarrollar el Framework, implica tener como resultado a una estructura de carpetas, librerías y archivos organizados, además de una arquitectura de software para su utilización.

Finalizando el desarrollo del Framework, se instancia un Subsistema de Información Logístico, con la finalidad de comprobar la utilización de tal, en donde dicho subsistema se encarga básicamente de llevar un seguimiento y control de los artículos; demostrando la utilización del Subsistema de Información orientado a la prestación de servicios y la construcción de tal a través del Framework desarrollado. Así mismo, se realizan las pruebas del Framework por expertos en base al empleo de técnicas e instrumentos, tales como: la entrevista, encuesta y cuestionario.

Por último, se concluye que, con el desarrollo y uso del Framework se logra construir software en Sistemas de Información orientados al modelo de distribución SaaS.

## **ABSTRACT**

This research is based on the development of a FRAMEWORK FOR BUILDING INFORMATION SYSTEMS AIMED TO PROVIDE SERVICES (SaaS model) specifically to provide a framework to speed up construction oriented software distribution model SaaS (Software as a Service).

The development of the Framework focuses on the design pattern Model - View - Controller, which is responsible for separating the data access, presentation and business logic modules, files or separate extensions, thus giving enough order and modularity in building software, however, what really makes the Framework serve to build oriented SaaS model systems is the use of multi-tenant architecture (multi-owner), which a single instance of the system or application appears in the server being used by multiple clients, organizations or owners (tenant) from an Internet connection, in addition to the development of the Framework various web technologies such as PHP programming language, the database manager SQL server database and used Apache web server. Developing the Framework implies result to a folder structure, libraries and files organized, along with a software architecture for use.

It finalizing development Framework, a Logistics Information Subsystem is instantiated, in order to verify the use of such, wherein said subsystem is basically responsible for keeping track and control items; demonstrating the use of Information Subsystem oriented service delivery and construction of such through the Framework developed. Also, testing Framework by experts based on the use of techniques and instruments, such as: interview, quiz and questionary.

Finally, we conclude that, with the development and use of software built Framework is achieved in Information Systems oriented SaaS distribution model.

## **INTRODUCCIÓN**

El Software como Servicio (en inglés Software as a Service, SaaS) es una nueva manera de entregar aplicaciones basadas en las tecnologías de Internet. Una aplicación SaaS está siempre disponible, cuando un usuario tiene acceso a Internet. De acuerdo a sus necesidades, el usuario (u organización) elige una aplicación SaaS (NIEDRÍTIS, 2013), utilizando sus servicios, funcionalidades y estructura previamente definida dentro de los alcances de la entidad que entrega la aplicación, por ejemplo: sistemas de ventas, compras, financieros, logísticos, ERP, CRM, entre otros. Software as a Service, viene siendo un modelo de distribución de software que pretende reducir costos y mejorar la productividad dentro de una organización, empresa o usuario, siendo estas últimas las ventajas más representativas dentro del modelo de distribución. Es así que, hoy en día, existe un gran aumento de aplicaciones web orientados al modelo SaaS, generando una mayor demanda para construir software orientado a prestar servicios a través del modelo SaaS.

En el proceso de desarrollo de software en aplicaciones o sistemas de información, es importante tener en claro un análisis correcto de los requerimientos, diseño de tales, la construcción del software y su posterior mantenimiento; sin embargo, se ha notado que, no se cuenta con arquitecturas y/o estructuras definidas capaces de soportar la construcción de software orientado al modelo SaaS y que existen marcos de trabajo en el mercado capaces de otorgar buenas funcionalidades en referencia a construir software orientado a aplicaciones web clásicas, ocasionando que, la entidad que construya software deba realizar nuevos métodos, procesos y funcionalidades respecto a construir aplicaciones o sistemas de información orientados a SaaS, debido a ello, resulta necesario desarrollar y brindar un Framework o marco de trabajo para la construcción de software en sistemas de información orientados a la prestación de servicios (modelo SaaS), enfocando su desarrollo en el patrón de diseño MVC (Modelo - Vista - Controlador), arquitectura Multitenant (multipropietario) y tecnología web.

Para el desarrollo del Framework se ha tomado como base, la metodología RUP, facilitando un análisis de requerimientos que se encuentran detallados en el análisis del dominio respectivo, así mismo, se realizó el diseño, en donde se culmina con una Arquitectura de

Software que engloba y muestra la forma de cómo ha sido modelado u organizado el Framework para agilizar la construcción de software y facilitar su posterior mantenimiento.

Luego, se realiza la construcción del Framework, visualizando a detalle el diseño y la lógica de negocio a la cual apunta dicho Framework. Finalmente se lleva a cabo las pruebas, especificando los casos y el plan de pruebas a ser implementado.

Para verificar la puesta en marcha del Framework, se desarrolla un Subsistema de Información, pues, para la presente investigación se toma en cuenta el proceso de negocio logístico en Empresas relacionadas con el control y seguimiento vehicular, las cuales son llamadas EMV (Empresa de Monitoreo Vehicular), con la finalidad de evidenciar que el Framework pueda ayudar y agilizar la construcción del sistema orientado a SaaS. Por último, se realiza una evaluación por expertos a través de un cuestionario, dando resultados favorables y alcanzando los objetivos planteados en la investigación.

Es así que, la tesis se desarrolla y organiza como sigue: se presenta un planteamiento teórico en el Capítulo 1 colocando los objetivos, hipótesis, variables, indicadores, viabilidad económica, táctica y operativa, entre otros; todo esto, previo al abordaje del marco teórico presentado en el Capítulo 2, donde se muestra; trabajos relacionados y realizados anticipadamente a la presente investigación, así mismo se explica los principales recursos teóricos que llevarán a obtener mayor conocimiento sobre los temas necesarios de conocer y por ende, llegar al objetivo del tema de tesis, finalmente se hace un estudio pormenorizado de los avances o relaciones directas con el trabajo en mención gracias al estado de arte. En el Capítulo 3 se desarrolla el Framework para la construcción de software en sistemas de información orientados a la prestación de servicios, teniendo como etapa inicial un análisis de los requerimientos hasta las pruebas necesarias del Framework, conllevando al uso de este. El Capítulo 4 muestra el desarrollo del Subsistema de Información, en donde se utiliza el Framework para la construcción de software en el subsistema de control de inventarios, considerado como proceso de negocio en las EMV (Empresa de Monitoreo Vehicular), como MYPE a tratar. Por último, el Capítulo 5 muestra las pruebas y resultados que se obtienen en cuanto al desarrollo y la utilización del Framework.

# **Capítulo 1**

## **Planteamiento Teórico**

---

---

### **1.1. INTRODUCCIÓN**

#### **1.1.1. Enunciado del problema**

Framework para la construcción de software en sistemas de información orientados a la prestación de servicios (Modelo SaaS).

### **1.1.2. Formulación Interrogativa del problema**

Mediante el uso de patrones de diseño, tecnologías web y el modelo de distribución SaaS ¿será posible desarrollar un Framework para construir software en sistemas de información orientados a brindar servicios a través de Internet?

## **1.2. ÁREA CIENTÍFICA**

**Área:** Ingeniería de Software.

**Línea:** Web.

**Tipo de investigación:** Aplicada.

**Nivel de investigación:** Descriptiva.

## **1.3. OBJETIVOS**

### **1.3.1. Objetivo General**

Desarrollar un Framework para construir software en sistemas de información orientados a la prestación de servicios.

### **1.3.2. Objetivos Específicos**

- Agilizar la construcción de software en sistemas de información orientados a la prestación de servicios.
- Utilizar la propiedad Multitenant del modelo SaaS y el patrón de diseño Modelo Vista Controlador.
- Diseñar la estructura principal que tomará el Framework para conocer la interacción de sus componentes internos en cuanto a: carpetas, librerías, clases y estándares de programación.

- Demostrar la construcción de software en Sistemas de Información orientados a la prestación de servicios a través de un subsistema de Control de Inventarios en la unidad de negocio de logística en una MYPE.

## 1.4. HIPÓTESIS

A través del desarrollo y utilización de un Framework, es probable que; se utilice para la construcción de software en Sistemas de Información orientados a la prestación de servicios aplicando el modelo SaaS.

## 1.5. VIABILIDAD DE LA INVESTIGACIÓN

### 1.5.1. Económica

Recursos	Costo unitario	Cantidad	Unidad	Costo Total
Internet	S/. 50.00	6	Meses	S/. 300.00
Memoria USB 4 GB	S/. 60.00	1	Unidad	S/. 60.00
Impresora	S/. 300.00	1	Unidad	S/. 300.00
Papel	S/. 9.00	1.5	Millar	S/. 27.00
Computadora	S/. 1200.00	1	Unidad	S/. 1200.00
Otros	S/. 100.00	1	Varios	S/. 100.00
<b>Total:</b>				S/. 1987.00

Tabla 1: Viabilidad económica.

Fuente: El Autor.

Económicamente es factible porque se cuenta con los recursos necesarios para que el proyecto se mantenga y siga adelante.

### 1.5.2. Técnica

Técnicamente es completamente factible el desarrollo del Framework, dado que se requiere de tecnología y herramientas que son gratuitas, lo que no implica la compra de nuevos productos de desarrollo, y además se dispone del conocimiento para desarrollar este tipo de software.

### **1.5.3. Operativa**

Para que el proyecto sea viable operativamente, se debe contar con el suficiente apoyo por parte de la dirección y usuarios de la empresa donde se vaya a utilizar el sistema de información orientado a la prestación de servicios instanciado por el Framework. Así mismo, se debe contar con el apoyo de usuarios expertos para realizar las pruebas del Framework. Por lo tanto, desde el punto de vista operativo es completamente factible y totalmente favorable para el proyecto.

## **1.6. DESCRIPCIÓN DEL PROBLEMA**

Para la construcción de software en Sistemas de Información orientados a brindar servicios, no se cuenta con una idea clara o qué recursos utilizar en cuanto a: modelos, tecnologías, estructura, entre otros capaces de llegar al objetivo; además, se hace confusa, desordenada y lenta, debido a que las entidades que construyen el software, utilizan diferentes métodos, estándares o marcos de trabajo sin tener en cuenta un orden o una estructura en la que se pueda aprovechar las potencialidades de diferentes tecnologías web, por lo que conlleva a un resultado lento e inflexible en cuanto a la construcción y su posterior mantenimiento del sistema.

## **1.7. JUSTIFICACIÓN**

- Se presenta el Framework como punto de partida para agilizar la construcción de software en Sistemas de Información orientados a la prestación de servicios, utilizando tecnologías web, patrones de diseño y la propiedad multitenencia del modelo de distribución SaaS.
- Organizar la construcción de software en Sistemas de Información orientados a prestar servicios, a través del manejo del Framework, en cuanto a: estándares de programación, estructura de componentes definidos, agilidad de desarrollo y facilidad en un mantenimiento posterior.

- Es importante realizar la debida investigación sobre el modelo de distribución SaaS y su propiedad Multitenencia, ya que es el eje principal para que un sistema de información pueda prestar servicios a través del Internet.
- Es necesario, puesto que, con el uso del Framework se deberá construir software en Sistemas de Información orientados a la prestación de servicios, haciendo que, el programador, área o equipo de desarrollo tenga mayor prioridad por la lógica del negocio.
- La utilización del Subsistema de Información, podrá brindar un aporte a cualquier otra MYPE cuya necesidad sea el manejo, control, seguimiento y administración de su información en el control de inventarios dentro de su proceso de negocio logístico; colaborando así, con el desarrollo de las MYPE, debido a que según FAEDPYME (2014), el 61.1% de las micro y pequeñas empresas no usa ninguna herramienta de las Tecnologías de Información y Comunicación (TIC).

## **1.8. DELIMITACIONES EN LA INVESTIGACIÓN**

- Para efectos de construcción del Framework e instancia del Sistema de Información, se concebirá software libre en el lado de programación y software propietario en el lado de base de datos, bajo la infraestructura tecnológica de Microsoft Windows.
- Para la construcción del Framework, solamente se tomará en cuenta la propiedad Multitenencia del modelo de distribución SaaS.
- Las principales operaciones (inserción, actualización, listado, eliminación) del Framework y los Sistemas de Información instanciados se llevarán a cabo por medio de procedimientos almacenados.
- Para efecto de demostrar la construcción de software en Sistemas de Información orientados a la prestación de servicios, se tomará en cuenta el subsistema de control de inventarios en el proceso de negocio logístico en micro y pequeñas empresas como son las Empresas de Monitoreo Vehicular.

## **1.9. ÁMBITO**

### **1.9.1. Ubicación**

Framework desarrollado para que pueda ser utilizado por cualquier persona, equipo de desarrollo o empresa que requiera construir software en aplicaciones orientadas a la prestación de servicios. El estudio e instancia de un Sistema de Información se realizará en MYPEs de Arequipa - Perú.

### **1.9.2. Población**

**a. Universo:** El Framework podrá ser utilizado por toda aquella entidad que requiera instanciar un Sistema de Información. La aplicación instanciada se podrá utilizar en toda aquella empresa que considere implantar un sistema de información para el manejo, control y seguimiento logístico. Tal aplicación podrá ser utilizada por medio de un navegador web y a través de internet. En referencia a lo expuesto, el universo es el conjunto de Empresas de Monitoreo Vehicular encargadas del seguimiento, monitoreo y control de unidades de transporte terrestre.

**b. Muestra:** El tipo de muestra a utilizar es el de No Probabilístico, tomando como muestra a 2 Empresas de Monitoreo Vehicular (EMV).

## **1.10. VARIABLES**

### **1.10.1. Variable Independiente**

- Framework

#### **Indicadores**

- Facilidad de uso.
- Curva de aprendizaje.
- Reutilización.

### **1.10.2. Variable Dependiente**

- Construcción de software en Sistemas de Información orientados a la prestación de servicios.

#### **Indicadores**

- Agilidad en construir software.
- Facilidad en el mantenimiento.

## **1.11. PLANTEAMIENTO OPERACIONAL**

### **1.11.1. Técnicas, Instrumentos y Materiales de Verificación**

- **Técnicas**
  - Entrevista.
  - Encuesta.
- **Instrumentos**
  - Cuestionario.

# **Capítulo 2**

## **Marco Teórico**

---

---

### **2.1. ANTECEDENTES INVESTIGATIVOS**

**Título:** Customização de interfaces web para clientes de software como serviço multitenant.

**Autor:** Alexandre Michetti Manduca

**Publicación:** Instituto de Ciências Matemáticas e de Computação, Ciências de Computação e Matemática Computacional, São Carlos, 2014.

## **Resumen**

A adoção de Software como Serviço (do inglês, Software as a Service ou simplesmente SaaS) está em expansão em todo o mundo, alavancada pelas muitas vantagens que esse modelo de distribuição de software oferece tanto para os provedores desses serviços quanto para seus clientes. Em SaaS, o provedor do serviço também é responsável pelo seu desenvolvimento e execução, o que permite a esses provedores fazerem escolhas sobre a arquitetura de seus sistemas visando diminuir a complexidade e os custos relacionados ao seu desenvolvimento e operação. Nesse contexto, um padrão de arquitetura freqüentemente utilizado é o Multitenant, que torna uma mesma instância do software capaz de servir a múltiplos clientes (tenants) simultaneamente. No entanto, sistemas que utilizam Multitenancy enfrentam uma série de desafios, principalmente no que se refere à flexibilidade em atender os requisitos específicos de cada cliente na customização de processos, de fluxos e regras de negócio, e de interfaces com o usuário. Especificamente no problema de customização de interfaces, embora existam trabalhos na literatura relacionados à customização de interfaces Web, e existam implementações de mercado de mecanismos para possibilitar essas customizações, esses trabalhos e mecanismos ou não são projetados especificamente para sistemas Multitenant ou, quando são, não têm sua arquitetura publicada e seu código fonte disponíveis para serem reutilizados. Assim, esta pesquisa investigou uma alternativa para a customização de interfaces Web no contexto de Software como Serviço Multitenant. Como resultado, foi inicialmente definido e implementado um mecanismo não intrusivo para o desenvolvimento de aplicações Multitenant chamado DORMT (Domain-Based Shared-Database Multitenancy), sobre o qual foi possível construir um mecanismo para a customização de interfaces Web, chamado MHT (Multitenant Hierarchical Themes), baseado no padrão MVC (Model-View-Controller) e nos conceito de temas e de hierarquias (MICHETTI MANDUCA, 2014).

**Título:** Desarrollo de Software Orientado a la Prestación de Servicios

**Autor:** Víctor Alejandro Álvarez Contreras

**Publicación:** Universidad de San Carlos de Guatemala, Facultad de Ingeniería, Escuela de Ingeniería en Ciencias y Sistemas, Guatemala, noviembre de 2009.

## **Resumen**

La tecnología de la información avanza aceleradamente y junto con ella, la cantidad de proveedores de servicios de informática aumenta cada vez más, lo que ha provocado que se busquen nuevas estrategias para mantenerse competitivos en un mercado ya saturado.

El uso de Internet a nivel global ha aumentado exponencialmente y esto ha permitido que los paradigmas de desarrollo de software evolucionen del clásico enfoque de desarrollo de aplicaciones específicas a terceros y se desarrolle aplicaciones flexibles y versátiles, que le permitan a personas y empresas hacer uso de sistemas que satisfagan cualquier tipo de necesidad desde cualquier parte del mundo.

Los avances tecnológicos dan la oportunidad de que las aplicaciones ofrecidas a través de Internet puedan ser tan robustas como las aplicaciones instaladas en una máquina, por lo que las empresas de desarrollo de software han optado por ofrecer sus sistemas, a través de paquetes de servicios integrales, cambiando la estrategia de venta de licencias a través de un producto empaquetado, a una venta de licencias a través de usuarios de Internet que hacen uso de las aplicaciones a través de un explorador o browser, y cuentan con ventajas adicionales como la capacidad de hacer uso de las aplicaciones desde cualquier lugar con conexión de Internet, disponibilidad de los sistemas las 24 horas del día los 7 días de la semana, bajos precios en comparación a licencias de productos empaquetados, personalización avanzada para cubrir necesidades específicas de cada negocio, mejoras periódicas de los sistemas, soporte técnico de alta calidad , entre otras (ÁLVAREZ CONTRERAS, 2009).

**Título:** Modelo tecnológico de integración de servicios para la mype peruana

**Autores:** Yamakawa, Peter; Del Castillo, Carlos; Baldeón, Johan; Espinoza, Luis Miguel; Granda, Juan Carlos; Vega, Lidia.

**Publicación:** Universidad ESAN, 2010. Tecnología de la Información / Microempresas / Pequeñas Empresas / Modelos / Competitividad/ Política y Estrategia Empresarial HD 62.7 Y35 ISBN 978-9972-622-83-0.

### **Resumen**

La Comisión Económica para América Latina y el Caribe (Cepal) ha comprobado que la globalización y las tecnologías de información y comunicación (TIC) ejercen un fuerte impacto sobre la composición de la actividad económica y los patrones de interacción social (Cepal, 2000); en especial, según el punto de vista de los autores,

sobre la competitividad empresarial. En este campo, el Perú se ubicó en el puesto 83 del ranking del Informe Global de Competitividad 2008-2009 (World Economic Forum, 2009) y mostró una mejora de tres posiciones con relación al estudio del periodo 2007-2008. Las principales mejoras registradas en el informe corresponden a los pilares de estabilidad macroeconómica, eficiencia del mercado de bienes y eficiencia del mercado laboral. Igualmente, se mantienen como fortalezas los pilares relacionados con la complejidad del mercado financiero, tamaño de este y mayor desarrollo empresarial. Sin embargo, continúan como principales debilidades los aspectos relacionados con instituciones, infraestructura, salud, educación primaria y superior, capacitación, preparación tecnológica e innovación. En el ámbito empresarial, esas debilidades son más acusadas en el segmento de las micro y las pequeñas empresas (MYPE) que en el de las medianas o las grandes empresas. En este contexto resulta pertinente analizar el marco establecido para fortalecer la competitividad de la MYPE en el Perú para, a partir de estas premisas, formular una iniciativa basada en tecnologías actuales, las cuales puedan servir de plataforma eficaz para desarrollar este marco y contribuir a una más eficiente inserción de la MYPE en la economía global. (ESAN, 2010).

## **2.2. MARCO CONCEPTUAL**

### **2.2.1. Framework**

El concepto Framework (GUTIÉRREZ, 2009) se emplea en muchos ámbitos del desarrollo de sistemas, no solo en el ámbito de aplicaciones Web. Podemos encontrar tipos de Framework para el desarrollo de aplicaciones médicas, de visión por computador, para el desarrollo de juegos, y para cualquier otro ámbito. En general, con el término Framework, nos estamos refiriendo a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un Framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Según RIEHLE (2000), un Framework o marco de trabajo modela un dominio específico o un aspecto importante de la misma. Representan el dominio como diseño abstracto, que consta de clases abstractas (o interfaces). El diseño abstracto es más que un conjunto de clases, porque define cómo se permite instanciar las clases que colaboran entre sí en tiempo de ejecución. Efectivamente, actúa como un esqueleto o un andamio, que determina cómo los objetos marco se relacionan entre sí.

En el desarrollo de software basado en un Framework, las aplicaciones (o sistemas) se convierten en extensiones del marco de aplicación. Se reutiliza la arquitectura de software de este tipo particular de aplicación tal como se define por el marco de aplicación y sus marcos de dominio. Existe un número de ventajas claves que se ganan al utilizar un Framework de aplicación.

La primera ventaja técnica es que provee diseño y código reutilizable. Así también, los sistemas basados en un Framework son fáciles de mantener, porque la clave del diseño y la implementación se encuentran localizados en un solo sitio, el Framework.

Las primeras ventajas de diseño y código reutilizable en cuanto al negocio son: mayor productividad del desarrollador y el corto tiempo de lanzamiento al mercado de nuevas aplicaciones.

### **a. Desarrollo de un Framework**

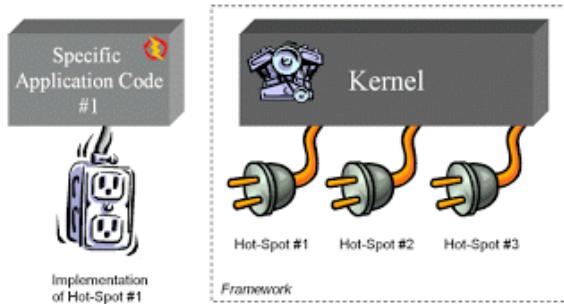
Las tres etapas principales del desarrollo del Framework colocadas en LIZÁRRAGA (2011) son: análisis del dominio, diseño del Framework, y la instancia del Framework.

Para completar los requerimientos, sirven las experiencias previamente publicadas, los sistemas de software similar y/o existente, las experiencias personales, y los estándares considerados. Durante el análisis del dominio, los puntos calientes y los puntos congelados se destapan parcialmente.

RODRÍGUEZ (2007) indica que, los puntos flexibles de un Framework se llaman puntos calientes (hotspots), y son las clases o los métodos abstractos que deben ser implementados o puestos en ejecución. Los tipos de Framework no son ejecutables, para generar un ejecutable, uno debe "especificar en una instancia" el Framework (llámese especificar en una instancia, al hecho de producir y completar un objeto llenando con valores en lugar de variables en una clase), poniendo el código específico de la aplicación en ejecución para cada Hotspots, los que, una vez que son especificados en una instancia, son usados por el Framework, especialmente aquellas clases que usan el callback o repetición de la llamada (acto de repetir la autentificación del número de usuario en caso de re conexión) . En esta repetición de la llamada o callback, el código del usuario del servicio declara que desea ser llamado en la ocurrencia, un determinado evento. Entonces, el código del proveedor del servicio realiza la repetición de la llamada o callback con el código del usuario del servicio al momento de ocurrir ese determinado evento. Por esta razón, en primera instancia, el Framework se caracteriza a veces como " el viejo código que llama al nuevo código."

Algunas de las características del Framework no son mutables ni tampoco pueden ser alteradas fácilmente. Estos puntos inmutables constituyen el núcleo o kernel de él, también llamados como los puntos congelados o frozen-spots del Framework. A diferencia de los puntos calientes o hot-spots, los puntos congelados o inmutables son los pedacitos del código puestos en ejecución ya dentro del Framework que llaman a uno o más puntos calientes proporcionados por el ejecutor. El núcleo o Kernel será la constante y presentará siempre la parte de cada instancia del Framework. Se puede pensar en un Framework como si fuese un motor, que requiere potencia. A diferencia de un motor tradicional, un motor del Framework tiene muchas entradas de potencia, donde cada una de estas entradas de potencia es un Hot-spot del Framework, y cada uno de ellos debe ser accionado para que el motor funcione. Los generadores de potencia son el código específico de la aplicación que se debe enchufar a los Hot-spot. El

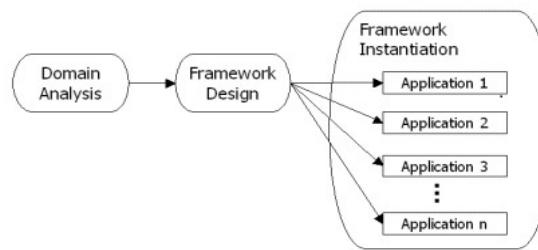
código agregado de la aplicación será utilizado por el código kernel del Framework. El motor no correrá hasta que todos los enchufes estén conectados. Esta metáfora se ilustra en la Figura 1.



*Figura 1: Puntos calientes y fríos de un Framework.*

Fuente: [http://www.tesis.uchile.cl/tesis/uchile/2007/rodriguez\\_pr/sources/rodriguez\\_pr.pdf](http://www.tesis.uchile.cl/tesis/uchile/2007/rodriguez_pr/sources/rodriguez_pr.pdf)

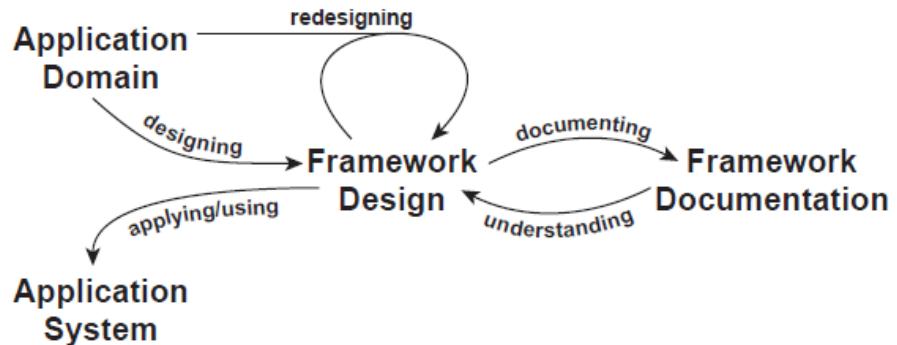
RODRÍGUEZ (2007) indica también que, la fase del diseño del Framework define las abstracciones de éste. Se modelan los puntos calientes y los puntos congelados (utilizando por ejemplo diagramas de UML), y la extensión y la flexibilidad propuesta en el análisis del dominio se esbozan en líneas generales. En la fase de instancia, los puntos calientes del Framework son implementados, generando un software del sistema. Es importante observar que cada una de estas aplicaciones tendrá los puntos congelados del Framework en común.



*Figura 2: Proceso de desarrollo del Framework.*

Fuente: [http://www.tesis.uchile.cl/tesis/uchile/2007/rodriguez\\_pr/sources/rodriguez\\_pr.pdf](http://www.tesis.uchile.cl/tesis/uchile/2007/rodriguez_pr/sources/rodriguez_pr.pdf)

Finalmente, es importante el diseño y la documentación en un Framework (RIEHLÉ, 2000), y se puede mostrar mediante la siguiente figura:



*Figura 3: Actividades y artefactos durante el diseño e instancia de un Framework.*

Fuente: Riehle 2000.

La Figura 3, muestra cuatro artefactos: dominio de la aplicación, aplicación del sistema, diseño y documentación del Framework. Tal diagrama muestra la diferencia entre artefactos y actividades, por ejemplo, el diseño del Framework es un artefacto, mientras el proceso de "diseñando" es una actividad. De todos los artefactos, solamente la documentación es tangible en su totalidad. El dominio de la aplicación, aplicación del sistema y el diseño del Framework son artefactos parcial o totalmente abstractos.

Es importante distinguir el diseño y la documentación de un Framework. El diseño del Framework contiene todas las ideas y conceptos implicados en un Framework. La documentación del Framework muestra todas esas ideas y conceptos de manera explícita, pero nunca se permite revelar todo en su detalle. Rara vez, la documentación describe aquellos aspectos del diseño que son más importantes para desarrolladores que conocer quien tratará de entender el Framework.

Los artefactos son relacionados a través de las actividades. Diseñando un sistema en un dominio de aplicación, conduce a realizar un diseño de Framework. Documentando el diseño, conduce a la documentación del Framework. Leyendo la documentación y trabajando con el Framework, conduce a entender el diseño del Framework. Aplicando el Framework, conduce a instanciar una aplicación.

### **2.2.2. Cloud Computing**

El *Cloud Computing* o Computación en la nube (MARROQUÍN, 2010), es un concepto que se puede definir como una tecnología que ofrece servicios a través de una plataforma de *Internet*, de tal forma que todo lo que puede ofrecer un sistema informático se ofrece como servicio, al cual los usuarios acceden sin que tengan conocimientos sobre la gestión de los diversos recursos que utilizan. Los programas están en los servidores en línea y se pueden acceder a los servicios a través de *Internet*.

MARROQUIN (2010), muestra que hoy en día, *Cloud Computing* ha tenido una evolución desde hace varios años:

- En la década de los sesenta, existían los *Mainframes*, los cuales eran computadoras centrales, grandes, muy costosas y pesadas, usadas principalmente por una gran compañía para el procesamiento de una gran cantidad de datos.
- En la década de los ochenta, se inició la fabricación de computadoras personales a gran escala, empezando con esto la descentralización, ya que anteriormente con los *Mainframe* se tenía todo centralizado.
- En la década de los noventa, se inició con el esquema de Cliente - Servidor, en el cual algunas funciones están del lado del cliente y otras del lado del servidor; en este esquema los equipos clientes, es decir, los equipos que forman parte de la red, envían una solicitud al servidor mediante su dirección IP y el puerto que está reservado para un servicio en particular que se ejecuta en el servidor, el cual recibe la solicitud y responde con la dirección IP de equipo cliente y su respectivo puerto.
- Alrededor del año 2005, gracias a nuevas tecnologías e infraestructuras, el concepto de computación en la nube se empezó a nombrar a gran escala por proveedores de *Internet*, como *Google*, *Amazon AWS* y otros que construyeron su propia infraestructura de la cual surge una nueva

arquitectura. Otras empresas que ofrecen estos servicios son: *Azure* de *Microsoft*, *Rackspace*, entre otras.

WIKIPEDIA (2015), invita a leer y conocer que, el concepto fundamental de la entrega de los recursos informáticos a través de una red global tiene sus raíces en los años sesenta. La idea de una "red de computadoras intergaláctico" fue introducido en los años sesenta por JCR Licklider, su visión era que todo el mundo pudiese estar interconectado y poder acceder a los programas y datos desde cualquier lugar, explicó Margaret Lewis, directora de marketing de producto de AMD. "Es una visión que se parece mucho a lo que llamamos cloud computing".

Otros expertos atribuyen el concepto científico de la computación en nube a John McCarthy, quien propuso la idea de la computación como un servicio público, de forma similar a las empresas de servicios que se remontan a los años sesenta. John McCarthy, 1960: "Algún día la computación podrá ser organizada como un servicio público".

Desde los años sesenta, la computación en nube se ha desarrollado a lo largo de una serie de líneas. La Web 2.0 es la evolución más reciente. Sin embargo, como Internet no empezó a ofrecer ancho de banda significativo hasta los años noventa, la computación en la nube ha sufrido algo así como un desarrollo tardío. Uno de los primeros hitos de la computación en nube es la llegada de Salesforce.com en 1999, que fue pionero en el concepto de la entrega de aplicaciones empresariales a través de una página web simple. La firma de servicios allanó el camino para que tanto especialistas como empresas tradicionales de software pudiesen publicar sus aplicaciones a través de Internet.

El siguiente desarrollo fue Amazon Web Services en 2002, que prevé un conjunto de servicios basados en la nube, incluyendo almacenamiento, computación e incluso la inteligencia humana a través del Amazon Mechanical Turk. Posteriormente en 2006, Amazon lanzó su Elastic Compute Cloud (EC2) como un servicio comercial que permite a las

pequeñas empresas y los particulares alquilar equipos en los que se ejecuten sus propias aplicaciones informáticas. "Amazon EC2/S3 fue el que ofreció primero servicios de infraestructura en la nube totalmente accesibles", dijo Jeremy Allaire, CEO de Brightcove, que proporciona su plataforma SaaS de vídeo en línea a las estaciones de televisión de Reino Unido y periódicos. George Gilder, 2006: "El PC de escritorio está muerto. Bienvenido a la nube de Internet, donde un número enorme de instalaciones a lo largo de todo el planeta almacenarán todos los datos que usted podrá usar alguna vez en su vida".

Otro hito importante se produjo en 2009, cuando Google entre otros, empezaron a ofrecer aplicaciones basadas en navegador. Servicios, como Google Apps. "La contribución más importante a la computación en nube ha sido la aparición de "aplicaciones asesinas" de los gigantes de tecnología como Microsoft y Google. Cuando dichas compañías llevan a cabo sus servicios de una manera que resulta segura y sencilla para el consumidor, el efecto 'pasar la pelota' en sí, crea un sentimiento de mayor aceptación de los servicios online", dijo Dan Germain, jefe de la oficina de tecnología en IT proveedor de servicios Cobweb Solutions.

Otro de los factores clave que han permitido evolucionar a la computación en la nube según el británico y pionero en computación en la nube Jamie Turner, han sido las tecnologías de virtualización, el desarrollo del universal de alta velocidad de ancho de banda, y normas universales de interoperabilidad de software. Turner añadió: "A medida que la computación en nube se extiende, su alcance va más allá de un puñado de usuarios de Google Docs. Sólo podemos empezar a imaginar su ámbito de aplicación y alcance. Casi cualquier cosa puede ser utilizado en la nube".

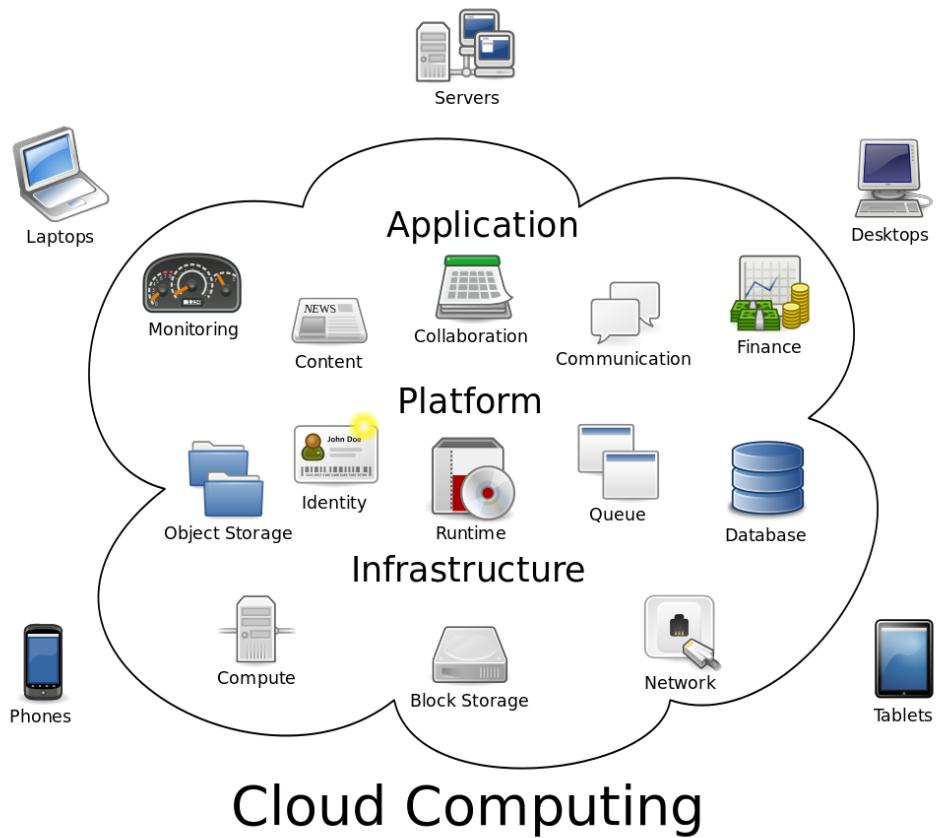


Figura 4: Computación en la nube.

Fuente: [https://commons.wikimedia.org/wiki/File:Cloud\\_computing-es.svg](https://commons.wikimedia.org/wiki/File:Cloud_computing-es.svg)

Una de las principales diferencias del *Cloud Computing* consiste en que no es necesario conocer la infraestructura y funcionamiento que está detrás, ya que todo esto pasa a ser parte de la nube en la cual las aplicaciones y servicios pueden crecer y funcionar rápido.

Entre las ventajas del *Cloud Computing* se tienen (MARROQUÍN, 2010):

- Acceso a la información y a los servicios desde cualquier lugar.
- Es escalable, por ejemplo, una empresa pequeña con un costo accesible puede comprar un pequeño grupo de licencias y una multinacional comprará un grupo mayor de licencias. Ambas podrán compartir los mismos servicios.

- Toda la capacidad de procesamiento y respaldo de la información, estará a cargo de la empresa proveedora del servicio; con esto, el cliente no tendrá que tener equipos instalados en forma local para que se ocupen de dichas tareas.
- Ahorro en *hardware*, ya que no se tendrá que preocupar por la compra de equipo ni en cambios por equipos obsoletos, ya que para esto se encargará la empresa proveedora del servicio.
- Algunos servicios son gratuitos y otros con costo, según las necesidades del usuario.
- Implementación rápida de *software* el cual ya ha sido probado por varios usuarios.

Entre las desventajas tenemos:

- La privacidad de la información, ya que todos estos datos están en manos de terceras personas.
- Uno depende de que el proveedor de servicios tenga una buena política de seguridad y de la realización de copias de seguridad en forma periódica.
- Dependencia de una conexión a *Internet*.

### **2.2.3. Modelos de Servicio en Cloud Computing**

#### **a. SaaS - Software as a Service**

PEREA (2011), muestra que, el usuario que opte por este servicio cloud podrá hacer uso de las aplicaciones que contrate al correspondiente proveedor. Un ejemplo puede ser una MYPE (micro y pequeña empresa) que contrate una aplicación de correo electrónico para sus 10 empleados. La aplicación no podrá ser modificada por la MYPE ni sus usuarios a excepción de posibles configuraciones de usuario o personalizaciones que le

permite el proveedor. La aplicación se encontrará alojada en las infraestructuras Cloud del proveedor y el usuario no tendrá ningún control sobre las mismas.

El software como servicio (en inglés *Software as a Service*, SaaS) se encuentra en la capa más alta y se caracteriza por ser una aplicación completa ofrecida como un servicio, en-demanda, vía multitenencia que significa una sola instancia del software que corre en la infraestructura del proveedor y sirve a múltiples organizaciones de clientes. El ejemplo de SaaS conocido más ampliamente es Salesforce.com, pero ahora ya hay muchos más, incluyendo las Google Apps que ofrecen servicios básicos de negocio como el e-mail. Por supuesto, la aplicación multitenencia de Salesforce.com ha constituido el mejor ejemplo de cómputo en nube durante unos cuantos años. Por otro lado, como muchos otros jugadores en el negocio del cómputo en nube, Salesforce.com ahora opera en más de una capa de la nube con su Force.com, que ya está en servicio, y que consiste en un ambiente de desarrollo de una aplicación compañera (“companion application”), o plataforma como un servicio. Otro ejemplo es la plataforma MS Office como servicio SaaS con su denominación de Microsoft Office 365, que incluye versiones online de la mayoría de las aplicaciones de esta suite ofimática de Microsoft. (WIKIPEDIA, 2015).

SaaS (URUEÑA et al., 2012) es consistente en la entrega de aplicaciones como servicio, siendo un modelo de despliegue de software mediante el cual el proveedor ofrece licencias de su aplicación a los clientes para su uso como un servicio bajo demanda. Los proveedores de los servicios SaaS pueden tener instalada la aplicación en sus propios servidores web (permitiendo a los clientes acceder, por ejemplo, mediante un navegador web), o descargar el software en los sistemas del contratante del servicio. En este último caso, se produciría la desactivación de la aplicación una vez finalice el servicio o expire el contrato de licencia de uso. La solución de *cloud computing* de *Software as a Service* puede estar orientada a distintos tipos de clientes según su condición: Usuarios particulares (servicios de

ofimática en cloud, redes sociales, cuentas de correo electrónico) y usuarios profesionales (CRM, ERP).

### **b. PaaS - Platform as a Service**

PEREA (2011) muestra en este caso que, el usuario estará contratando un servicio que le permite alojar y desarrollar sus propias aplicaciones (desarrollos propios o licencias adquiridas) en una plataforma que dispone de herramientas de desarrollo para que el usuario pueda elaborar una solución. En este modelo el proveedor ofrece el uso de su plataforma que a su vez se encuentra alojada en sus infraestructuras. Por lo que el usuario no tiene control sobre la plataforma ni las infraestructuras pero si sobre sus aplicaciones.

La capa del medio, que es la plataforma como servicio (en inglés *Platform as a Service*, PaaS), es la encapsulación de una abstracción de un ambiente de desarrollo y el empaquetamiento de una serie de módulos o complementos que proporcionan, normalmente, una funcionalidad horizontal (persistencia de datos, autenticación, mensajería, etc.). De esta forma, un arquetipo de plataforma como servicio podría consistir en un entorno conteniendo una pila básica de sistemas, componentes o API pre configurado y listo para integrarse sobre una tecnología concreta de desarrollo (por ejemplo, un sistema Linux, un servidor web, y un ambiente de programación como Perl o Ruby). Las ofertas de PaaS pueden dar servicio a todas las fases del ciclo de desarrollo y pruebas del software, o pueden estar especializadas en cualquier área en particular, tal como la administración del contenido.

Los ejemplos comerciales incluyen Google App Engine, que sirve aplicaciones de la infraestructura Google, y también Windows Azure, de Microsoft, una plataforma en la nube que permite el desarrollo y ejecución de aplicaciones codificadas en varios lenguajes y tecnologías como .NET, Java y PHP. Servicios PaaS tales como estos permiten gran

flexibilidad, pero puede ser restringida por las capacidades que están disponibles a través del proveedor (WIKIPEDIA, 2015).

PaaS (URUEÑA et al., 2012) consistente en la entrega, como un servicio, de un conjunto de plataformas informáticas orientadas al desarrollo, testeo, despliegue, hosting y mantenimiento de los sistemas operativos y aplicaciones propias del cliente. Las principales características asociadas al *Platform as a Service* como solución *cloud* se exponen a continuación:

- Facilita el despliegue de las aplicaciones del cliente, sin el coste y la complejidad derivados de la compra y gestión del hardware y de las capas de software asociadas.
- Ofrece a través de redes de servicio IP todos los requisitos necesarios para crear y entregar servicios y aplicaciones web.

### **c. IaaS - Infrastructure as a Service**

PEREA (2011), indica que, en este modelo el usuario estará contratando únicamente las infraestructuras tecnológicas (capacidad de procesamiento, de almacenamiento y / o de comunicaciones).

La infraestructura como servicio se encuentra en la capa inferior y es un medio de entregar almacenamiento básico y capacidades de cómputo como servicios estandarizados en la red. Servidores, sistemas de almacenamiento, conexiones, enrutadores, y otros sistemas se concentran (por ejemplo, a través de la tecnología de virtualización) para manejar tipos específicos de cargas de trabajo, desde procesamiento en lotes (“batch”) hasta aumento de servidor/almacenamiento durante las cargas pico. El ejemplo comercial mejor conocido es Amazon Web Services, cuyos servicios EC2 y S3 ofrecen cómputo y servicios de almacenamiento esenciales (respectivamente). Otro ejemplo es Joyent cuyo producto principal es una línea de servidores virtualizados, que proveen una infraestructura en demanda altamente escalable para manejar sitios Web, incluyendo

aplicaciones Web complejas escritas en Python, Ruby, PHP, y Java. (WIKIPEDIA, 2015).

IaaS (URUEÑA et al., 2012) consistente en poner a disposición del cliente el uso de la infraestructura informática (capacidad de computación, espacio de disco y bases de datos entre otros) como un servicio. Los clientes que optan por este tipo de familia *cloud* en vez de adquirir o dotarse directamente de recursos como pueden ser los servidores, el espacio del centro de datos o los equipos de red optan por la tercerización, en busca de un ahorro en la inversión en sistemas TI. Con esta tercerización, las facturas asociadas a este tipo de servicios se calculan en base a la cantidad de recursos consumidos por el cliente, basándose así en el modelo de pago por uso.

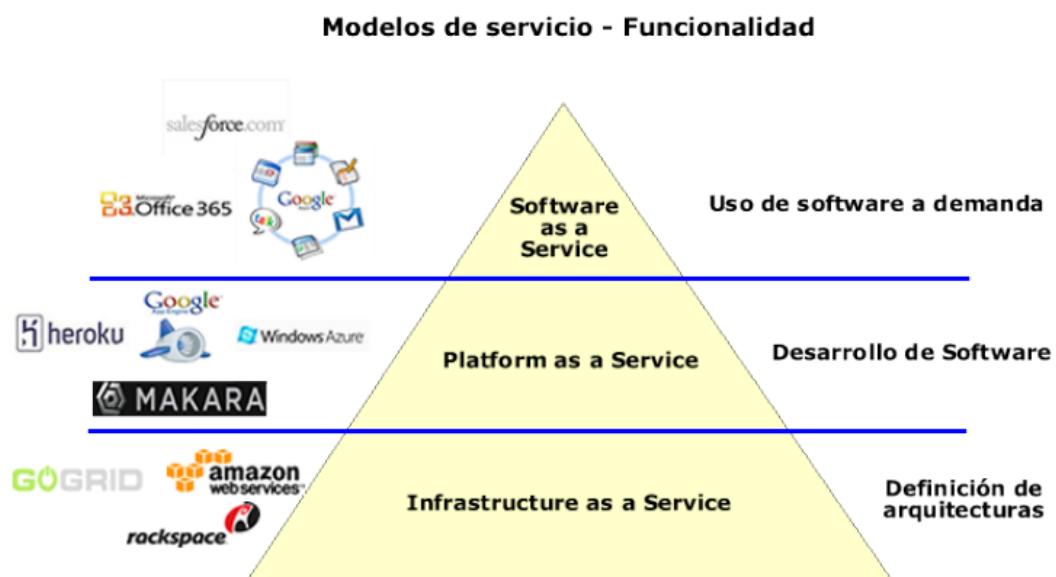
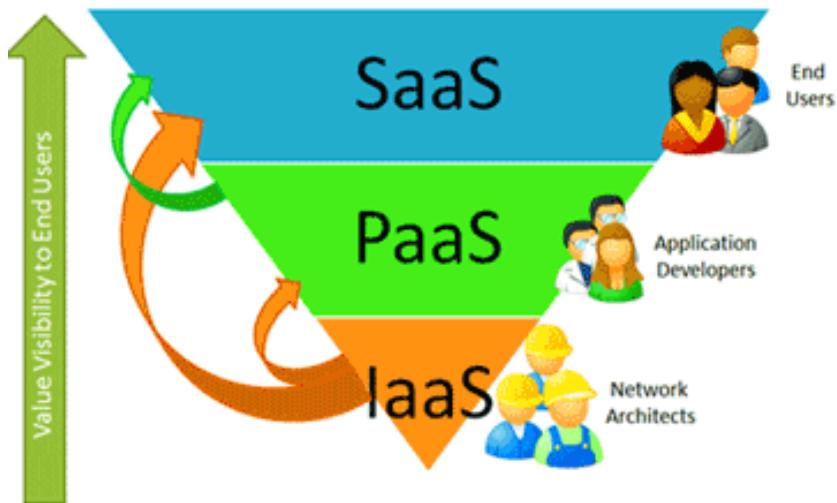


Figura 5: Modelos de servicio - Funcionalidad.

Fuente: <http://www.cloudcomputingla.com/2010/08/saas-paas-e-iaas.html>



*Figura 6: Interacción en los modelos de servicio.*

Fuente: <http://prcerda.blogspot.com/2012/02/2012-el-ano-clave-para-el-cloud.html>

#### **2.2.4. Software as a Service (Modelo SaaS)**

ÁLVAREZ (2009), indica que es un modelo de desarrollo de software orientado a la prestación de servicios, también conocido como modelo SaaS, es una forma relativamente nueva de vender y distribuir software. Le permite a una persona individual o a una pequeña o mediana empresa, hacer uso de aplicaciones de software y servicios de tecnología avanzados a cambio de una cuota periódica o de modo gratuito. La clave de este modelo es el enfoque que se le da a una aplicación de software, ya que es tratada como un servicio y no como un producto. Por lo tanto, los clientes no pagan por adquirir el software, sino por hacer uso de él. Cabe mencionar que las aplicaciones de software no dejan de ser productos en este modelo, ya que el proveedor de servicios es propietario de los productos de software, pero alquila las licencias de sus productos en forma de servicios a sus clientes. Las aplicaciones de software ofrecidas bajo este modelo también reciben el nombre de "aplicaciones de software bajo demanda". De forma general, este modelo consiste en brindar a usuarios, accesos a una aplicación de software (como una contabilidad) utilizando un protocolo estándar como el protocolo HTTP, con ayuda de un explorador o a través de aplicaciones cliente con propósitos específicos, y brindar servicios a los usuarios a través de estos accesos dependiendo de un

paquete específico o una cuota seleccionada. Este tipo de servicios también pueden integrarse a aplicaciones propias de una empresa para manejar procesos de negocio si esto fuera requerido, siendo esta integración generalmente, a través de servicios Web o Aplicaciones REST.

Algunas de las características asociadas a este modelo incluyen:

- El proveedor de servicio de aplicaciones es dueño y maneja por completo, de forma independiente o a través de otra empresa, las aplicaciones de software que ofrece.
- El proveedor es dueño de toda la infraestructura necesaria para soportar las aplicaciones de software.
- El proveedor pone la información y las aplicaciones de software a disponibilidad del cliente a través de un explorador e Internet o un "*thin client*" (aplicación cliente "delgada" que utiliza la información de un servidor al que se encuentra conectada para realizar sus operaciones).
- El proveedor cobra una cuota periódica por uso de la aplicación que puede ser mensual, semestral, anual, etc. (no aplicable para el presente proyecto).

Las aplicaciones de software distribuidas bajo este modelo deben cumplir con ciertas características clave descritas a continuación:

- Las aplicaciones deben estar disponibles comercialmente, ser accedidas y administradas a través de una plataforma en red.
- Sus actividades o funciones deben ser administradas en localidades centrales en vez de ser administradas desde las localidades de cada cliente, permitiéndole a estos acceder a las aplicaciones remotamente a través de Internet.
- La arquitectura debe permitir que una instancia de la aplicación corra en un servidor y permita que varios clientes accedan a la misma

concurrentemente, al contrario del software tradicional que debe correr un número de instancias igual al número de clientes a utilizarlo.

- Las aplicaciones deben permitir ser actualizadas centralizadamente lo que elimina la necesidad de bajar parches o actualizaciones posteriores.

#### **a. Modelo Tradicional vs Modelo SaaS**

ÁLVAREZ (2009) indica que, el modelo tradicional de desarrollo de software consiste en la adquisición de una aplicación de software por parte de un cliente a través de una licencia. Esta licencia le permite a un cliente hacer uso de funcionalidades específicas de la aplicación, dependiendo del tipo de licencia adquirido. En este modelo una aplicación de software es tomada como un producto y no como un servicio ya que esta es vendida como un paquete con licencia por lo que el cliente es el dueño de la aplicación y puede hacer uso de ella en la forma que lo desee siempre y cuando se mantenga dentro de los límites definidos por la misma. Dentro de este modelo es el cliente el que se encarga de mantener la infraestructura necesaria para soportar la aplicación. Este modelo es muy usado por empresas grandes, ya que las necesidades tecnológicas en sus procesos de negocio resultan ser más específicas que las de una MYPE y son estas quienes generalmente cuentan con el capital necesario para pagar una aplicación hecha a la medida y mantener la infraestructura que la soporte.

#### **b. Similitudes y diferencias**

Al momento de realizar una comparación entre ambos modelos de negocio, podemos darnos cuenta que son pocas las similitudes entre ellos, ya que la diferencia de enfoques aumenta la brecha en la forma de funcionar de cada uno. Entre las similitudes identificadas entre ambos modelos se pueden describir las siguientes:

- Ambos modelos hacen uso de licencias.

- Los modelos ofrecen a los clientes funcionalidades específicas de una aplicación de software de acuerdo a una presentación adquirida. Esto puede observarse claramente en ambos enfoques si tomamos como ejemplo de un modelo tradicional, las distintas ediciones que ofrece Microsoft para su Sistema Operativo Windows XP (Home, Professional), y como ejemplo de un modelo SaaS, las distintas versiones de la contabilidad en línea Servicont que ofrece el sitio <http://www.nubox.com/home.html> (Servicont Plus, Servicont Premium).
- El cliente es el dueño de la información manejada en las aplicaciones desarrolladas bajo ambos modelos. Actualmente existe la falsa creencia de que en el modelo SaaS es el proveedor de servicios de aplicación y el propietario de la información, lo que ha generado desconfianza por parte de los clientes de utilizar este tipo de servicios. Cabe aclarar que el proveedor de servicios es únicamente propietario del espacio físico en el cual se almacena la información, más no de la información como tal, lo que debería quedar bien definido en los términos de contratación del servicio que el proveedor establezca con el cliente.
- A pesar de que ambos modelos manejen licencias existe diferencia en la forma de ofrecer las mismas, ya que, en el modelo tradicional, el cliente adquiere la licencia al momento de comprar el producto y se vuelve propietario de ella, mientras que en el modelo SaaS es la empresa proveedora del servicio la propietaria absoluta de las licencias, que son únicamente rentadas por el cliente durante el tiempo que éste pague la cuota periódica correspondiente.

Al hablar de las diferencias que existen entre ambos modelos de desarrollo se pueden mencionar las siguientes:

- En el modelo tradicional el cliente realiza un solo pago para adquirir un producto de software mientras que en el modelo SaaS, el cliente paga una cuota periódica (mensual, semestral, anual, etc., no aplicable para la

presente tesis) para hacer uso de los servicios que ofrece la aplicación proporcionada.

- Los costos de una aplicación desarrollada bajo el modelo tradicional son altos la mayoría de las veces, mientras que los costos de utilizar una aplicación en el modelo SaaS son mucho más accesibles.
- Las actualizaciones de aplicaciones desarrolladas bajo el modelo tradicional suelen tener costos muy elevados, mientras que para el cliente las actualizaciones de las aplicaciones desarrolladas bajo el modelo SaaS son transparentes en costos.
- En el modelo tradicional, el cliente es el dueño de la licencia del producto de software, mientras que en el modelo SaaS, el cliente alquila la licencia del producto de software, propiedad del proveedor, bajo la forma de un servicio.
- Las aplicaciones en el modelo SaaS son desarrolladas exclusivamente en plataforma Web, mientras que las aplicaciones en el modelo tradicional pueden ser desarrolladas en plataforma Web o escritorio. Esto se debe a que el acceso a las aplicaciones en el modelo SaaS se realiza vía Internet a través de un explorador (Internet Explorer, Mozilla Firefox, etc.), mientras que las aplicaciones en el modelo tradicional pueden ser accedidas ya sea por un explorador si se encuentran desarrolladas en una plataforma Web y almacenadas en un servidor remoto, o localmente, a través de su instalación en el equipo a utilizar si fuesen desarrolladas en plataforma de Escritorio.
- Al momento de adquirir una aplicación desarrollada bajo el modelo tradicional, es el cliente el que debe poseer la infraestructura que soporte la misma, mientras que en el modelo SaaS, es el proveedor de servicios de aplicación el que posee la infraestructura necesaria, y el cliente solo hace uso de la aplicación a través de un explorador e Internet.

### **c. Ventajas del Modelo SaaS**

Dentro de las ventajas que el cliente obtiene al utilizar los servicios de aplicaciones desarrolladas bajo el modelo SaaS, se pueden mencionar las siguientes:

- En MYPES y de recién comienzo, se reduce la necesidad de una inversión inicial fuerte en aplicaciones de software e infraestructura que soporte las mismas, permitiendo a las empresas utilizar herramientas avanzadas a precios moderados.
- El cliente no debe preocuparse por la instalación e integración de su equipo y las aplicaciones de software a utilizar.
- No se necesita poseer un departamento de informática con personal calificado que se encargue de dar mantenimiento a las aplicaciones.
- Las aplicaciones desarrolladas bajo el modelo SaaS poseen los mismos beneficios que un sistema desarrollado bajo el modelo tradicional, pero a un menor costo.
- Por una cuota periódica, los clientes pueden hacer uso de un servicio que incluya alojamiento de la aplicación de software e información manejada en ella, mantenimiento de la aplicación, soporte técnico, actualizaciones periódicas, utilización de servicios avanzados proporcionados por las aplicaciones, capacitación a usuarios, uso de licencias, infraestructura de soporte, seguridad y comunicación entre sucursales. Gracias a esto se reduce de manera importante la carga administrativa y de gestión de una empresa (cuota periódica que no se aplica para el presente proyecto).
- Se reduce el costo de utilizar un producto no deseado, ya que el cliente puede probar una aplicación durante un tiempo y si la aplicación no satisface sus necesidades de negocio, el cliente puede simplemente cambiar a otro proveedor, a diferencia de los productos desarrollados

bajo el modelo tradicional, en los que se paga el valor total del producto, antes de poder evaluarlo.

- La empresa puede trabajar con información en tiempo real desde cualquiera de sus sucursales.

También existen ventajas por parte del proveedor de servicios al utilizar el modelo de desarrollo SaaS como modelo de negocios. Entre éstas se pueden incluir:

- Se reducen los costos de distribución del software a los clientes ya que las aplicaciones se encuentran centralizadas en los servidores del proveedor y utilizadas únicamente a través del explorador.
- Se reducen los costos de mantenimiento, actualización y soporte ya que la empresa proveedora se asegura de que todos sus clientes estén utilizando la misma aplicación de software con las mismas actualizaciones, por lo que los servicios de soporte son especializados y se necesita menor cantidad de recursos para su administración.
- Mejora la atención al cliente ya que la resolución de errores es rápida debido a que se hace de forma centralizada sin necesidad de crear parches y aplicarlos en todos los clientes, pues todas las actualizaciones se hacen directamente sobre el servidor.
- Elimina el problema de que los clientes compartan ilegalmente el software con otros y la piratería.
- El hecho que haya muchos clientes utilizando la aplicación al mismo tiempo haciendo uso de diferentes funcionalidades permite que se puedan realizar estudios para optimizar las aplicaciones de una forma más sencilla.

#### **d. Desventajas del Modelo SaaS**

El modelo SaaS puede presentar algunas desventajas para el cliente, las cuales serán descritas a continuación:

- Para hacer uso de aplicaciones de software desarrolladas bajo el modelo SaaS, es necesario poseer una conexión de Internet de buena calidad ya que la mayoría de transacciones realizadas en estas aplicaciones requieren de una aceptable cantidad de banda ancha para que la velocidad de respuesta en el lado del cliente pueda ser comparable con una aplicación de escritorio. Además, si la conexión de Internet falla, no se puede seguir haciendo uso de las aplicaciones, por lo que se depende completamente del proveedor de servicio de Internet.
- Temor de almacenar la información de la empresa en un lugar remoto fuera de la misma ya que esta información puede ser confidencial. Este factor se convierte en un problema importante al momento de comercializar una aplicación desarrollada bajo el modelo SaaS.
- La empresa cliente depende de la supervivencia de la empresa proveedora para poder seguir haciendo uso de las aplicaciones y mantener su información segura. El riesgo puede disminuir si se pacta con la empresa proveedora una política de envío periódico de backups de la información para que esta pueda ser utilizada por otras aplicaciones ante cualquier eventualidad.

El modelo SaaS puede presentar algunas desventajas para el proveedor, las cuales serán descritas a continuación:

- Para prestar este tipo de servicios es necesario realizar inversiones fuertes en infraestructura que garantice calidad y seguridad de las aplicaciones ofrecidas.

- Los costos de call center, soporte técnico, personal administrativo, etc. son altos ya que se requiere de una gran cantidad de empleados para poder atender una mayor cantidad de clientes.
- Aumenta la dificultad de crear tarifas que se adapten a la utilización de recursos de la mayoría de clientes.

### **2.2.5. Implementación SAAS**

Según MORENO (2014), las aplicaciones como servicio tienen una característica que hace que el modelo sea especialmente eficiente: el multitenancy. Esta es la propiedad que permite ofrecer la misma aplicación a muchos usuarios y así distribuir el coste de la infraestructura y del mantenimiento entre todos. Técnicamente no se trata solo de ofrecer la misma aplicación, sino de realizar una aplicación que permita con una sola instancia de la aplicación y una sola base de datos o mejor dicho un único de tablas relacionadas, dar servicio a todos sus clientes. Sin embargo, desde el punto de vista de la BBDD hay otras dos implementaciones "multitenancy" donde el tratamiento de los datos es diferente y aunque se acercan más al antiguo modelo ASP se venden como SaaS. Todas tienen sus ventajas y sus inconvenientes, no son peores ni mejores, cumplen con la condición de que dan soporte a muchos clientes, por tal razón se las diferenciará a continuación.

#### **a. Una base de datos por cada empresa o usuario**

Es decir, tenemos una misma instancia de aplicación para dar servicio a todos los usuarios, pero una BBDD por cada empresa. Esta es la opción que está más lejos del modelo SaaS y más cerca al modelo ASP. Las tablas no necesitan ningún atributo para diferenciar si los datos pertenecen a un cliente a otro, pero si es necesario tener una estructura de datos que determine qué base de datos le corresponde a cada cliente.

### **Las ventajas son:**

- Un Motor de BBDD dedicado para cada usuario, por tanto, no afectan los datos y accesos de otros clientes.
- Posibilidad de tener una máquina dedicada para los usuarios.

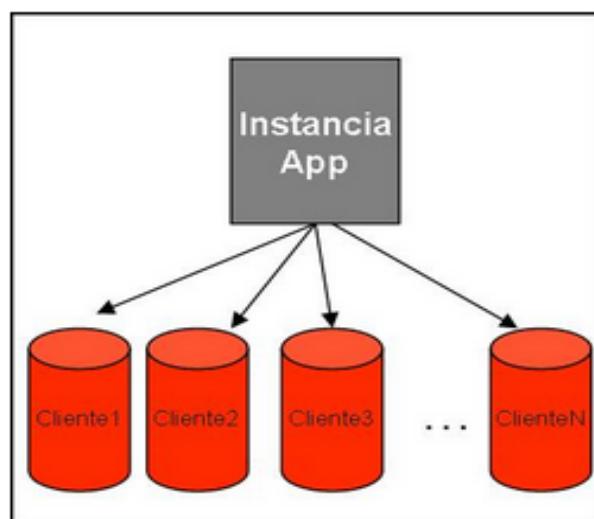
### **Las desventajas son:**

#### **▪ Desde el punto de vista del proveedor:**

- Más recursos humanos. Mantenimiento tedioso y largo ya que cualquier modificación en el modelo de datos (tablas) hay que replicarla en todas las BBDD.
- Más recursos hardware. A partir de un cierto número de clientes necesitarás más máquinas para albergar las BBDD.

#### **▪ Desde el punto de vista del cliente:**

- Más expuestos a errores. La replicación masiva da lugar a errores y puede afectar a los datos de tu aplicación.
- Precio alto. Más recursos, más mantenimiento requiere, más gasto y esto incidirá en el precio.



*Figura 7: Implementación de SaaS - Una base de datos por cada empresa o usuario.*

Fuente: <http://www.saasmania.com/blog/2010/02/04/%C2%BFcomo-se-hace-saas/>

## **b. Una base de datos con N conjunto de tablas**

Es decir, tenemos una misma instancia de aplicación para dar servicio a todos los usuarios y una sola base de datos con tantos conjuntos de tablas como clientes haya. Esta opción se acerca más al modelo SaaS, aunque sigue siendo un dolor su mantenimiento. Las tablas tampoco necesitan un atributo para diferenciar si los datos pertenecen a un cliente a otro, y de igual forma necesitan una estructura de datos que determine qué conjunto de tablas (esquema) pertenece a cada cliente.

### **Las ventajas son:**

- Menos inversión de hardware que la opción anterior.
- Menos mantenimiento del hardware.
- Precio menor que la opción anterior.

### **Las desventajas son:**

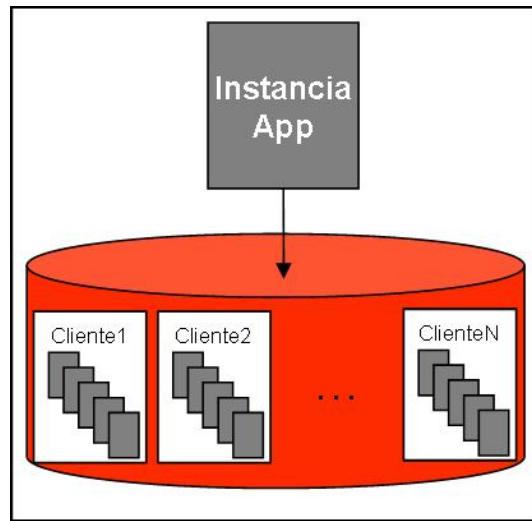
#### **▪ Desde el punto de vista del proveedor:**

- Más recursos humanos. Mantenimiento tedioso y largo ya que cualquier modificación en el modelo de datos (tablas) hay que replicarla en todos los conjuntos de TABLAS.
- En caso de fallo del motor de BBDD no puede dar servicio a ningún cliente.

#### **▪ Desde el punto de vista del cliente:**

- Probabilidad de pérdida de rendimiento. Se está compartiendo los mismos recursos hardware y el mismo motor de BBDD para todos los usuarios.
- Más expuesto a errores. La replicación masiva da lugares a errores y puede afectar a los datos de su aplicación.

- Precio alto. Más recursos humanos requiere más gasto y esto incidirá en el precio.



*Figura 8: Implementación de SaaS - Una base de datos con N conjunto de tablas.*

Fuente: <http://www.saasmania.com/blog/2010/02/04/%C2%BFcomo-se-hace-saas/>

### c. Una base de datos con ÚNICO conjunto de tablas

Es decir, tenemos una misma instancia de aplicación y una sola base de datos con un único conjunto de tablas para dar servicio a todos los usuarios.

El mayor problema que presenta es la **complejidad** de la aplicación y la estructura de datos. Las tablas necesitarán un atributo para diferenciar si los datos pertenecen a un cliente u otro.

#### **Las ventajas son:**

- Menos recurso de hardware que las anteriores.
- Mantenimiento menos tedioso que evita la probabilidad de error en las actualizaciones de la estructura de datos.
- Mantenimiento más corto que permite tener la aplicación disponible mucho más tiempo.

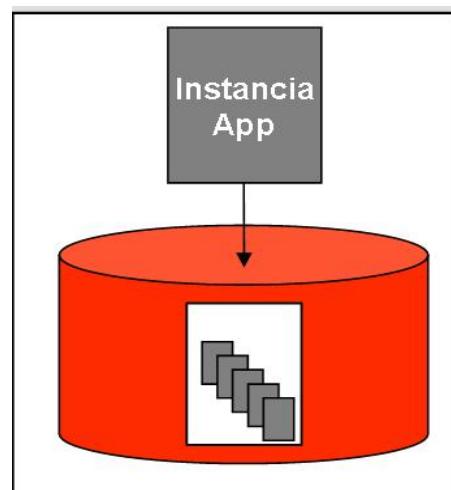
### **Las desventajas son:**

#### **▪ Desde el punto de vista del proveedor:**

- Aplicación más compleja porque debe determinar el acceso a los datos correctos de cada cliente.
- En caso que falle el motor de base de datos no se podrá realizar el servicio a ningún cliente.

#### **▪ Desde el punto de vista del cliente:**

- Probabilidad de pérdida de rendimiento. Se está compartiendo los mismos recursos hardware y el mismo motor de BBDD para todos los usuarios.
- La complejidad de la aplicación puede dar lugar a errores.



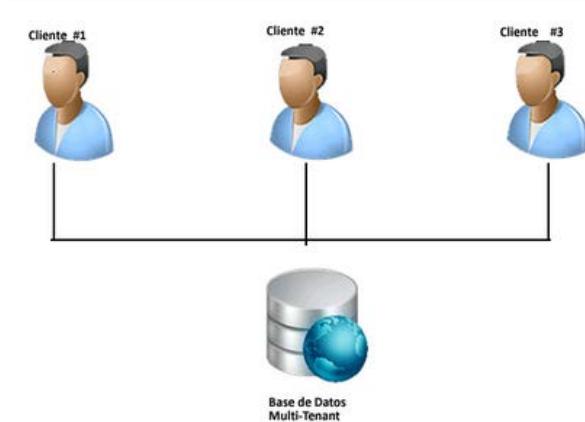
*Figura 9: Implementación de SaaS - Una base de datos con único conjunto de tablas.*

Fuente: <http://www.saasmania.com/blog/2010/02/04/%C2%BFcomo-se-hace-saas/>

#### **2.2.6. Multitenancy**

Las arquitecturas Multitenant (multipropietario) son cada vez más utilizadas entre los proveedores de SaaS. En un entorno Multitenant, todos los clientes y sus usuarios consumen el servicio desde la misma plataforma tecnológica, el intercambio de todos los componentes de la tecnología incluyendo el modelo de datos, servidores y las capas de base de datos. La arquitectura Multitenant se

refiere a un principio en la arquitectura de software donde una única instancia del software se ejecuta en un servidor y al servicio a múltiples clientes (los inquilinos). Este tipo de arquitecturas se denominan Multitenant o Multipropietario, en las que sobre un único recurso operan múltiples usuarios que son dueños, por así decirlo, del mismo (GRAVITAR, 2010).



*Figura 10: Arquitectura Multitenancy*

Fuente: [http://www.gravitar.biz/index.php/tecnologia\\_negocios/arquitecturas-multi-tenant/](http://www.gravitar.biz/index.php/tecnologia_negocios/arquitecturas-multi-tenant/)

### 2.2.7. MYPE

Según SUNAT (2015), la Micro y Pequeña Empresa (MYPE) es la unidad económica constituida por una persona natural o jurídica, bajo cualquier forma de organización o gestión empresarial contemplada en la legislación vigente, que tiene como objeto desarrollar actividades de extracción, transformación, producción, comercialización de bienes o prestación de servicios.

El Decreto Legislativo N° 1086 ([DLES - 2008]) tiene por objetivo la promoción de la competitividad, formalización y desarrollo de las Micro y Pequeñas Empresas para la ampliación del mercado interno y externo de estas, en el marco del proceso de promoción del empleo, inclusión social y formalización de la economía, para el acceso progresivo al empleo en condiciones de dignidad y suficiencia.

Las MYPE, constituyen uno de los pilares de la economía nacional porque además de generar el autoempleo, promueven la competitividad y formalización de la economía, redundando en el crecimiento y desarrollo del país; de ahí el interés del Estado de promover a estas pequeñas unidades económicas con capitales pequeños dedicadas a la extracción, transformación, producción o comercialización para que se desarrolle.

Las MYPE se caracterizan por tener:

▪ **Microempresa**

- ✓ Nro. de trabajadores: De uno (01) hasta diez (10) trabajadores inclusive.
- ✓ Ventas anuales: Hasta el monto máximo de 150 UIT (\*).

▪ **Pequeña empresa**

- ✓ Nro. de trabajadores: De once (11) hasta cien (100) trabajadores inclusive.
- ✓ Ventas anuales: Hasta el monto máximo de 1,700 UIT (\*).

(\*) La UIT para el año 2015 es de S/ 3,850 nuevos soles. Para mayor información ver <http://www.sunat.gob.pe/indicestasas/uit.html>

### **2.2.8. Tecnología Web**

La Tecnología Web permite el desarrollo de aplicaciones distribuidas basadas en el modelo Cliente/Servidor. Las aplicaciones web suponen un importante cambio de enfoque con respecto al desarrollo de aplicaciones tradicionales. Su principal característica consiste en que la comunicación con el usuario se establece utilizando páginas web, que se pueden visualizar desde un navegador que se esté ejecutando en cualquier ordenador conectado a la red. Otra característica importante, consiste en que el código de la aplicación se puede ejecutar en el cliente, en el servidor o distribuirse entre ambos. Además, debido al gran volumen de información que se maneja, las aplicaciones web suelen utilizar una Base de Datos para organizar y facilitar el acceso a la información.

### a. PHP

PHP (acrónimo recursivo de *PHP: Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

Bien, pero ¿qué significa realmente? Un ejemplo nos aclarará las cosas:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Ejemplo</title>
  </head>
  <body>

    <?php
      echo "¡Hola, soy un script de PHP!";
    ?>

  </body>
</html>
```

Figura 11: Ejemplo Introductorio Nº 1

Fuente: <http://docs.php.net/manual/es/intro-whatis.php>

En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), las páginas de PHP contienen HTML con código incrustado que hace "algo" (en este caso, mostrar "¡Hola, soy un script de PHP!). El código de PHP está encerrado entre las etiquetas especiales de comienzo y final `<?php` y `?>` que permiten entrar y salir del "modo PHP".

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no

hay manera de que los usuarios puedan saber qué se tiene debajo de la manga.

Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. En unas pocas horas podrá empezar a escribir sus primeros scripts. Aunque el desarrollo de PHP está centrado en la programación de scripts del lado del servidor, se puede utilizar para muchas otras cosas.

Este lenguaje de programación está preparado para realizar muchos tipos de aplicaciones web gracias a la extensa librería de funciones con la que está dotado. La librería de funciones cubre desde cálculos matemáticos complejos hasta tratamiento de conexiones de red, por poner dos ejemplos (PHP, 2015).

Sus características más resaltantes son:

- **Simplicidad:** Se les debe permitir a los programadores generar código productivamente en el menor tiempo posible.
- **Adaptabilidad:** PHP utiliza dos sintaxis: una es de procedimiento, el otro es orientado a objetos. Cada una de estas sintaxis hace posible la aplicación de las mismas funcionalidades, pero dirigidas a un público diferente. La sintaxis de procedimiento se utiliza comúnmente por los webmasters y procesamiento de datos de especialistas que trabajan en la interfaz de usuario de una aplicación. La segunda sintaxis, Programación orientada a objetos, es a propósito similar a Java y C #, con el fin de reducir los costes de formación y fomentar la migración a PHP.
- **Portabilidad:** PHP está disponible en todos los principales sistemas operativos. El enfoque técnico de PHP es idéntico a una máquina virtual Java (JVM). Sólo tienes que instalar PHP en una estación cliente o servidor, y la aplicación funcionará de inmediato, sin tener que recompilar, no importa qué sistema operativo se usa. PHP está disponible

en todos los principales sistemas operativos. El enfoque técnico de PHP es idéntico a una máquina virtual Java (JVM). Sólo tienes que instalar PHP en una estación cliente o servidor, y la aplicación funcionará de inmediato, sin tener que recompilar, no importa qué sistema operativo se usa.

- **Durabilidad:** La durabilidad de la tecnología de información depende principalmente de su número de usuarios. PHP es utilizado por más de 4.500.000 desarrolladores de todo el mundo. Más de 20 millones de sitios web utilizan esta tecnología. Además, el código de fuente abierto y los miembros de esta comunidad Open Source en la fundación Apache garantizar la durabilidad de PHP.

#### b. Smarty

Es un motor de plantillas para PHP. Más específicamente, esta herramienta facilita la manera de separar la aplicación lógica y el contenido en la presentación. La mejor descripción está en una situación donde la aplicación del programador y la plantilla del diseñador juegan diferentes roles, o en la mayoría de los casos no la misma persona. Uno de los objetivos de diseño es facilitar la separación del código de la aplicación con la de presentación.

Normalmente, el código de la aplicación contiene la lógica de negocio de su solicitud, por escrito y mantenerse en el código PHP. Este código es gestionado por los programadores.

La presentación es la forma en que su contenido se presenta al usuario final, que es escrita y mantenida en los archivos de plantilla. Las plantillas son mantenidos por los diseñadores de la plantilla. En su función más básica, el código de la aplicación recoge el contenido, le asigna a la plantilla de motores y la muestra. El código de aplicación no tiene ninguna preocupación cómo este contenido se presentará en la plantilla. El diseñador de la plantilla es responsable de la presentación.

Esto normalmente implica cosas como etiquetas HTML, hojas de estilo y otras herramientas proporcionadas por el motor de plantillas. Este paradigma tiene varias finalidades:

- 1.-** Los diseñadores no pueden romper el código de aplicación. Ellos pueden meterse con las plantillas todo lo que quieran, pero el código permanece intacto. El código será más estricto, más seguro y más fácil de mantener.
- 2.-** Los errores en las plantillas se limitan a las rutinas de manejo de errores Smartys, haciéndolos tan simple e intuitiva como sea posible para el diseñador.
- 3.-** Con la presentación en su propia capa, los diseñadores pueden modificar o rediseñar completamente desde cero, todo ello sin la intervención del programador.
- 4.-** Los programadores no son de jugar con las plantillas. Se puede ir sobre el mantenimiento del código de la aplicación, cambiando el contenido de modo de su adquisición, lo que hace nuevas reglas de negocio, etc., sin alterar la capa de presentación. (LIZÁRRAGA, 2011).

### c. XAJAX

Ajax es una tecnología que utiliza a su vez otra combinación de tecnologías, como XML y Javascript, para realizar peticiones de contenido o computación de servidor sin tener que recargar la página en la que está el usuario.

Es una tecnología que permite una nueva gama de aplicaciones interactivas en la web, mucho más ricas y rápidas, dado que no precisamos recargar todo el contenido de una página para realizar peticiones al servidor. XAJAX, es una clase realizada con PHP que nos permite utilizar Ajax, combinado con PHP, para la creación de aplicaciones interactivas, de una manera muy simplificada.

Con xajax podemos fácilmente ejecutar funciones PHP, que se ejecutan en el servidor, cuando el usuario realiza acciones en la página. Luego, los resultados de esas funciones PHP se producen en la misma página, sin que se tenga que recargarse. Xajax es un producto Open Source gratuito y compatible con los navegadores más comunes, como Firefox, u otros navegadores basados en Mozilla, Internet Explorer, Opera, etc. (LIZÁRRAGA, 2011).

#### d. JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

El código JavaScript se encierra entre etiquetas <script> y se incluye en cualquier parte del documento. Aunque es correcto incluir cualquier bloque de código en cualquier zona de la página, se recomienda definir el código JavaScript dentro de la cabecera del documento (dentro de la etiqueta <head>):

#### **Nombre de Archivo: prueba.html**

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio
documento</title>
<script type="text/javascript">
function prueba (){
alert("Mensaje 1");
alert("Mensaje 2");
}
```

```

</script>
</head>
<body>
<p>Un párrafo de texto.</p>
</body>
</html>

```

El trato de javascript es diferente en archivos con extensión "HTML" de los que tienen extensión "tpl", esto se puede notar al utilizar funciones JavaScript, debido a que el código JavaScript es interpretado de diferente manera en motores de plantillas como Smarty, por ejemplo.

### **Nombre de Archivo: prueba.tpl**

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio
documento</title>
<script type="text/javascript">
function prueba (){{ldelim}
alert("Mensaje 1");
alert("Mensaje 2");
{rdelim}
</script>
</head>
<body>
<p>Un párrafo de texto.</p>
</body>
</html>

```

El anterior archivo (prueba.tpl), la abertura y cierre de la función está hecho por etiquetas como {{ldelim}} y {{rdelim}} debido a que se trabaja con Smarty. La recomendación sería para estos casos, trabajar con archivos que se encuentren en otra carpeta, haciendo más fácil el entendimiento del código y sin cambiar la estructura del código JavaScript, esto se realiza definiendo el JavaScript en un archivo externo.

Las instrucciones JavaScript se pueden incluir en un archivo externo de tipo JavaScript que los documentos XHTML enlazan mediante la etiqueta <script>. Se pueden crear todos los archivos JavaScript que sean necesarios y cada documento XHTML puede enlazar tantos archivos JavaScript como necesite.

### Ejemplo:

#### Nombre de Archivo: **codigo.js**

```
alert ("Un mensaje de prueba");
```

#### Nombre de Archivo: **prueba.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio
documento</title>
<script type="text/javascript" src="codigo.js"></script>
</head>
<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

Además del atributo type, este método requiere definir el atributo src, que es el que indica la URL correspondiente al archivo JavaScript que se quiere enlazar. Cada etiqueta <script> solamente puede enlazar un único archivo, pero en una misma página se pueden incluir tantas etiquetas <script> como sean necesarias (LIZÁRRAGA, 2011).

## e. Hojas de estilo

CSS (*Cascading Style Sheets*, Hojas de estilo en Cascada), es una tecnología que nos permite crear páginas web de una manera más exacta. Gracias a las

CSS somos mucho más dueños de los resultados finales de la página, pudiendo hacer muchas cosas que no se podía hacer utilizando solamente HTML, como incluir márgenes, tipos de letra, fondos, colores, etc. Una de las características más potentes de la programación con hojas de estilo consiste en definir los estilos de todo un sitio web. Esto se consigue creando un archivo donde tan sólo colocamos las declaraciones de estilos de la página y enlazando todas las páginas del sitio con ese archivo. De este modo, todas las páginas comparten una misma declaración de estilos y, por tanto, si la cambiamos, cambiarán todas las páginas. El proceso para incluir estilos con un fichero externo, sería:

**1.- Creamos el fichero con la declaración de estilos:** Es un fichero de texto normal, que puede tener cualquier extensión, aunque le podemos asignar la extensión .css para aclararnos qué tipo de archivo es. El texto que debemos incluir debe ser escrito exclusivamente en sintaxis CSS, es un poco distinta que la sintaxis que utilizamos dentro del atributo style. Sería erróneo incluir código HTML en este archivo: etiquetas y demás. Podemos ver un ejemplo a continuación:

```
P {  
    font-size : 12pt;  
    font-family : arial,helvetica;  
    font-weight : normal;  
}  
  
H1 {  
    font-size : 36pt;  
    font-family : verdana,arial;  
    text-decoration : underline;  
    text-align : center;  
    background-color : Teal;  
}  
  
BODY {  
    background-color : #006600;  
    font-family : arial;  
    color : White;  
}
```

**2.- Enlazamos la página web con la hoja de estilos:** Para ello vamos a colocar la etiqueta <LINK> con los atributos:

- **rel:** "STYLESTHEET" indicando que el enlace es con una hoja de estilo.
- **type:** "text/css" porque el archivo es de texto, en sintaxis CSS.
- **href:** "estilos.css" indica el nombre del fichero fuente de los estilos.

Veamos una página web entera que enlaza con la declaración de estilos anterior:

```
<html>
<head>
<link rel="STYLESTHEET" type="text/css" href="estilos.css">
<title>Página que lee estilos</title>
</head>
<body>
<h1>Página que lee estilos</h1>
<p> Esta página tiene en la cabecera la etiqueta necesaria para enlazar con la hoja de estilos.
</p>
</body>
</html>
```

El trabajo del diseñador web siempre está limitado por las posibilidades de los navegadores que utilizan los usuarios para acceder a sus páginas. Por este motivo es imprescindible conocer el soporte de CSS en cada uno de los navegadores más utilizados del mercado.

Internamente los navegadores están divididos en varios componentes. La parte del navegador que se encarga de interpretar el código HTML y CSS para mostrar las páginas se denomina motor (LIZÁRRAGA, 2011).

#### f. Servidor Apache

Es el servidor web hecho por excelencia, su flexibilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. El servidor Apache es un servidor web de código abierto

para plataformas Linux, Microsoft Windows o Macintosh, que implementa el protocolo HTTP. Los programadores de aplicaciones web a veces utilizan una versión local de Apache con el fin de pre visualizar y probar código mientras éste es desarrollado. A continuación, se muestran ciertas características del servidor web Apache:

- ✓ Apache es uno de los mejores servidores de Webs utilizados en la red internet desde hace mucho tiempo. Por lo que este servidor es uno de los mayores triunfos del software libre.
- ✓ Es un servidor de web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos (HTTP 1.1). Implementa los últimos protocolos, aunque se base en el HTTP / 1.1.
- ✓ Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo y con la API de programación de módulos.
- ✓ Soporte para los lenguajes Perl, Python, y PHP.
- ✓ Permite la configuración de mensajes de errores personalizados y negociación de contenido.
- ✓ Permite autenticación de base de datos basada en SGBD. (LIZÁRRAGA, 2011).

#### **g. Servidor IIS**

Los Servicios de Internet Information Services (IIS) proporcionan las herramientas y funciones necesarias para administrar de forma sencilla un servidor Web seguro. Internet Information Services, es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del *Option Pack* para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS. Será de gran importancia para poder construir

reportes con SQL SERVER, ya que los reportes se encuentran en arquitectura web y estos necesitan de un servidor propio, el cual es Internet Information Services (IIS) (LIZÁRRAGA, 2011).

### **2.2.9. BPM y SOA**

Business Process Management (GARIMELLA, 2008) es un conjunto de métodos, herramientas y tecnologías utilizados para diseñar, representar, analizar y controlar procesos de negocio operacionales. BPM es un enfoque centrado en los procesos para mejorar el rendimiento que combina las tecnologías de la información con metodologías de proceso y gobierno. BPM es una colaboración entre personas de negocio y tecnólogos para fomentar procesos de negocio efectivos, ágiles y transparentes. BPM abarca personas, sistemas, funciones, negocios, clientes, proveedores y socios. BPM combina métodos ya probados y establecidos de gestión de procesos con una nueva clase de herramientas de software empresarial. Ha posibilitado adelantos muy importantes en cuanto a la velocidad y agilidad con que las organizaciones mejoran el rendimiento de negocio. Con BPM:

- ✓ Los directores de negocio pueden, de forma más directa, medir, controlar y responder a todos los aspectos y elementos de sus procesos operacionales.
- ✓ Los directores de tecnologías de la información pueden aplicar sus habilidades y recursos de forma más directa en las operaciones de negocio.
- ✓ La dirección y los empleados de la organización pueden alinear mejor sus esfuerzos y mejorar la productividad y el rendimiento personal.
- ✓ La empresa, como un todo, puede responder de forma más rápida a cambios y desafíos a la hora de cumplir sus fines y objetivos.

BPM y SOA (BERNAL, 2015) son utilizadas conjuntamente en la industria para lograr alineación de TI al negocio, existen enfoques y metodologías para lograr este objetivo. Para poder llevar a cabo la descomposición del proceso, es necesario tener presentes algunos conceptos básicos. Es importante mencionar

que este tipo de metodología es sólo una propuesta para la industria y puede ser adaptada dependiendo de nuestras implementaciones en cada proyecto.

El objetivo principal es mostrar una pequeña parte del paradigma de orientación a servicios: Modelamiento de Servicios, mediante la descomposición de un proceso de negocio en servicios candidatos.

### Orientación a servicios

“La orientación a servicios es un paradigma de diseño destinado a la creación de unidades lógicas de solución que son formadas individualmente con el propósito de que puedan ser utilizadas colectiva y repetidamente en la realización de una estrategia de solución específica.”

Como se puede observar en la Figura 12, se conceptualiza la orientación a servicios como una capa superior a la orientación a objetos, en la cual se hace uso de metodologías, tecnologías y notaciones para diseñar unidades de solución soportándose en la orientación a objetos, la cual a su vez utiliza herramientas tecnológicas de más bajo nivel para dar soporte técnico.



Figura 12: Servicio.

Fuente: <http://www.baware.com.mx/es/index.php/blog/54-descomposicion-de-procesos-de-negocio-en-servicios-soa-paradigma-de-orientacion-a-servicios>

## **Proceso de negocio**

El proceso como tal, es un conjunto de actividades coordinadas que combinan y utilizan recursos y capacidades con el fin de producir un resultado que directa o indirectamente crea valor para los clientes o los interesados.

El proceso de negocio, es un conjunto de tareas y actividades que permiten lograr un objetivo organizacional específico.

Se puede observar que la diferencia entre un proceso y un proceso de negocio es el enfoque organizacional que posee el último de estos.

## **Servicio**

Es una unidad lógica de solución a la cual se le ha aplicado la orientación a servicios y se le han aplicado los principios del paradigma orientado a objetos y que distinguen a un servicio de un componente o una aplicación.

## **Tipos de Lógica**

Es importante también, identificar los tipos de lógica que ayudarán a clasificar un servicio, a continuación, se describen los cuatro tipos de lógica:

- a. Negocio: Lógica que expresa capacidades de negocio. Esta es la lógica de más alto nivel y es generalmente expresada mediante BPM, BPMN (notación para BPM) o modelos de datos.
- b. No Agnóstica: Lógica específica. Este tipo de lógica es utilizada para realizar tareas concretas y por esta razón tiende a no ser reutilizable.
- c. Agnóstica: Lógica genérica. Este tipo de lógica permite una fácil reutilización.
- d. Utilidad: Lógica que realiza tareas más técnicas y utilizando recursos tecnológicos. Este tipo de lógica es la de más bajo nivel se utiliza para realizar tareas como persistencia, acceso a datos, mensajería, seguridad,

manejo de memoria y recursos computacionales, etc. Es altamente reutilizable.

## 2.3. ESTADO DEL ARTE

El objetivo de esta sección es dar a conocer una visión general sobre las arquitecturas o Frameworks diseñados para construir software en sistemas de información orientados a la prestación de servicios.

NIEDRĪTIS (2013) muestra su tesis (*Architecture for multi-tenant adaptive information systems*), teniendo como objetivo la construcción de una arquitectura para el desarrollo de sistemas de información web (WIS - web information systems) basados en el enfoque de SaaS garantizando las posibilidades de adaptación del WIS de acuerdo a las necesidades de los usuarios individuales. Propone una arquitectura para sistemas de información adaptadas al multi tenant (MTAIS - Multi Tenant Adaptive Informations Systems) que está planeado para soportar el uso de sistemas de información web en diferentes organizaciones, en la misma área de negocio y con similares procesos de negocio, implementando técnicas de perfiles en la capa de datos, modelos de presentación, interfaz de usuario, adaptación de la interfaz de usuario de acuerdo a las acciones de los usuarios; sin presentar un marco de trabajo definido para construir los sistemas de información web (WIS), sino que, al contrario, la arquitectura se basa en los usuarios finales que se encuentran en diferentes organizaciones, adaptando tal arquitectura a dichos usuarios finales.

SALEH (et al., 2012), propone un Framework para migrar las aplicaciones web tradicionales en aplicaciones SaaS, aplicando dentro de su proceso de migración, una serie de pasos que conlleva a aplicar: un módulo de autenticación dedicado que es usado para mapear el usuario hacia una tabla colocando el Tenant - ID, cambios en términos de la personalización del software, arquitectura de la base de datos y aislamiento de clientes o propietarios. Es así, como SALEH (et al., 2012) presenta un Framework completo para facilitar el proceso de migración y ejecutar el modelo SaaS.

Sin embargo, MICHETI (2014), específicamente no orienta su propuesta hacia la construcción de software inicial u orientado al código fuente del sistema, sino que toma como punto de partida la personalización de las interfaces web del tenant en un modelo de distribución SaaS, siendo la interfaz web independiente por cada tenant, así que, construye un mecanismo llamado MHT (Multitenant Hierarchical Themes), basado en el patrón de diseño MVC (Modelo-Vista-Controlador).

Así mismo, KOZIOLEK (2010) propone documentar los conceptos y limitaciones de un software multi-tenant y describir un estilo arquitectónico llamado SPOSAD (Shared, Polymorphic, Scalable Application and Data), tales conceptos descritos deberán ser reutilizables y servirán para guiar el desarrollo de nuevas aplicaciones de software multi-tenant. KOZIOLEK (2010) identifica un nuevo estilo arquitectónico que ayuda a los desarrolladores en la creación de futuras aplicaciones multiusuario, debido a que todavía hay muchas decisiones de diseño en la aplicación y por ende el nivel de base de datos que tiene que hacerse por cada aplicación. Por lo tanto, el presente autor muestra un estilo arquitectónico que puede ayudar a los desarrolladores en sus decisiones. Es así también que KOZIOLEK (2010) cita a autores que proponen diferentes tecnologías, estilos o marcos para la implementación de aplicaciones multi-tenant, por ende, comenta que, ellos describen las pautas de seguridad, rendimiento e identifican los cuellos de botella y los enfoques de optimización para dichas aplicaciones. Sin embargo, se olvidan del nivel de aplicación en su investigación y por lo tanto de la construcción de software a través de los Frameworks.

También se cuentan con Frameworks comerciales como Symfony, cakePHP, kumbiaPHP, Zend Framework, entre otros; los cuales son Frameworks establecidos en el mercado, pero cuentan con muchos recursos, los cuales no definen exactamente un dominio específico para su aplicación, por lo tanto, se pueden volver engorrosos y por consiguiente aumentar el tiempo de construcción en el software. Cabe resaltar que estos Frameworks son buenos y recomendados por el autor pero, están orientados a la construcción de aplicaciones web sin contemplar una instancia inicial o un esqueleto ya diseñado sobre un sistema de información, por otro lado, solamente presentan una arquitectura que apunta a la creación de aplicaciones web que dan servicio a un tenant

o cliente, sin embargo, para el presente proyecto se necesita un Framework para la construcción de software en sistemas de información orientados al modelo SaaS.

La mayoría de Frameworks o arquitecturas estudiadas, proponen técnicas, métodos y metodologías orientadas al usuario final, rendimiento, escalabilidad, migraciones, sin enfocarse en que se debe contar con una arquitectura o estructura de trabajo definida para agilizar a construir software en un sistema de información orientado a SaaS, por ende, es importante considerar un marco de trabajo capaz de brindarle al desarrollador un conjunto de componentes estructurados, alcanzando así a la comunidad de desarrolladores; un Framework apuntando a un mismo dominio que es, la construcción de software en sistemas de información orientados a la prestación de servicios utilizando el modelo SaaS.

# **Capítulo 3**

## **Desarrollo del Framework**

---

### **3.1. INTRODUCCIÓN**

Para cualquier aplicación, sistema de información o software a desarrollar, es importante conocer y plantear objetivos; para ello, es necesario tener presente un escenario en donde se debe contemplar todo aquel requerimiento previo que nos dará el punto de partida a la implementación del software, en este caso, del Framework.

En primer lugar, los requerimientos van a ser colocados en un dominio del Framework, donde se expondrá todo lo necesario a construir.

En segundo lugar, teniendo previamente el análisis del dominio del Framework, se continuará con el desarrollo del Framework, donde tendremos que seguir una metodología de desarrollo de software, para este caso, se optará por la metodología RUP y la utilización del lenguaje de modelado UML.

## 3.2. ÁNALISIS DEL DOMINIO

### 3.2.1. Requerimientos del Framework

- **Instanciar software web** a través del Framework, es decir que, el Framework debe crear automáticamente la estructura de carpetas y la relación funcional entre sus componentes. Teniendo como componentes principales a: carpetas de trabajo, librerías, clases y funciones.
- **Establecer un estándar propio**, obteniendo un lenguaje común de codificación o gestión de recursos, entre el o los desarrolladores encargados de construir software a través del Framework.
- **Representar tablas de Base de Datos en el Sistema:** Extraer y representar las entidades principales de la base de datos en forma de código en la aplicación; es decir, extraer las tablas y representarlas en el Framework como entidades a través de clases.
- **Generar plantillas y controladores para procesos comunes;** debido a que la mayoría de Sistemas de Información tienen operaciones comunes como la inserción, actualización y/o listado de entidades, es que, el Framework deberá generar la lógica del negocio e interfaces a partir de la Base de Datos previamente creada, esto para acelerar la programación y dejar a lado la “forma” del Sistema y adentrarse más en el “fondo”; es decir en el negocio del Sistema.

- **Instanciar el sistema de información**, que soporte el manejo y registro de datos pertenecientes a diferentes empresas o clientes. Así mismo, cada empresa registrada podrá llevar el control de sus usuarios Administradores u Operadores como perfiles primordiales. Se deberá concebir una sola instancia para "N" clientes y usuarios.
- **Controlar el Acceso al sistema**, para obtener mayor seguridad en los Sistemas de Información a construir por medio del Framework, es que este tendrá una validación de usuarios (Login del Sistema).
- **Considerar un orden en la programación**, capaz de separar la parte visual, lógica y el acceso a la base de datos.
- **Desarrollar un Manual Técnico**, permitiendo al desarrollador un fácil aprendizaje del uso y utilización de los recursos y/o estructura de los componentes internos del Framework.
- **Desarrollar un Manual de Usuario**, permitiendo al desarrollador la facilidad de instanciar y manejar los recursos del Framework, pero, de manera externa y general.

### 3.3. DESARROLLO DEL FRAMEWORK

#### 3.3.1. Análisis, Diseño y Construcción del Framework

##### a. Diagrama de Casos de Uso

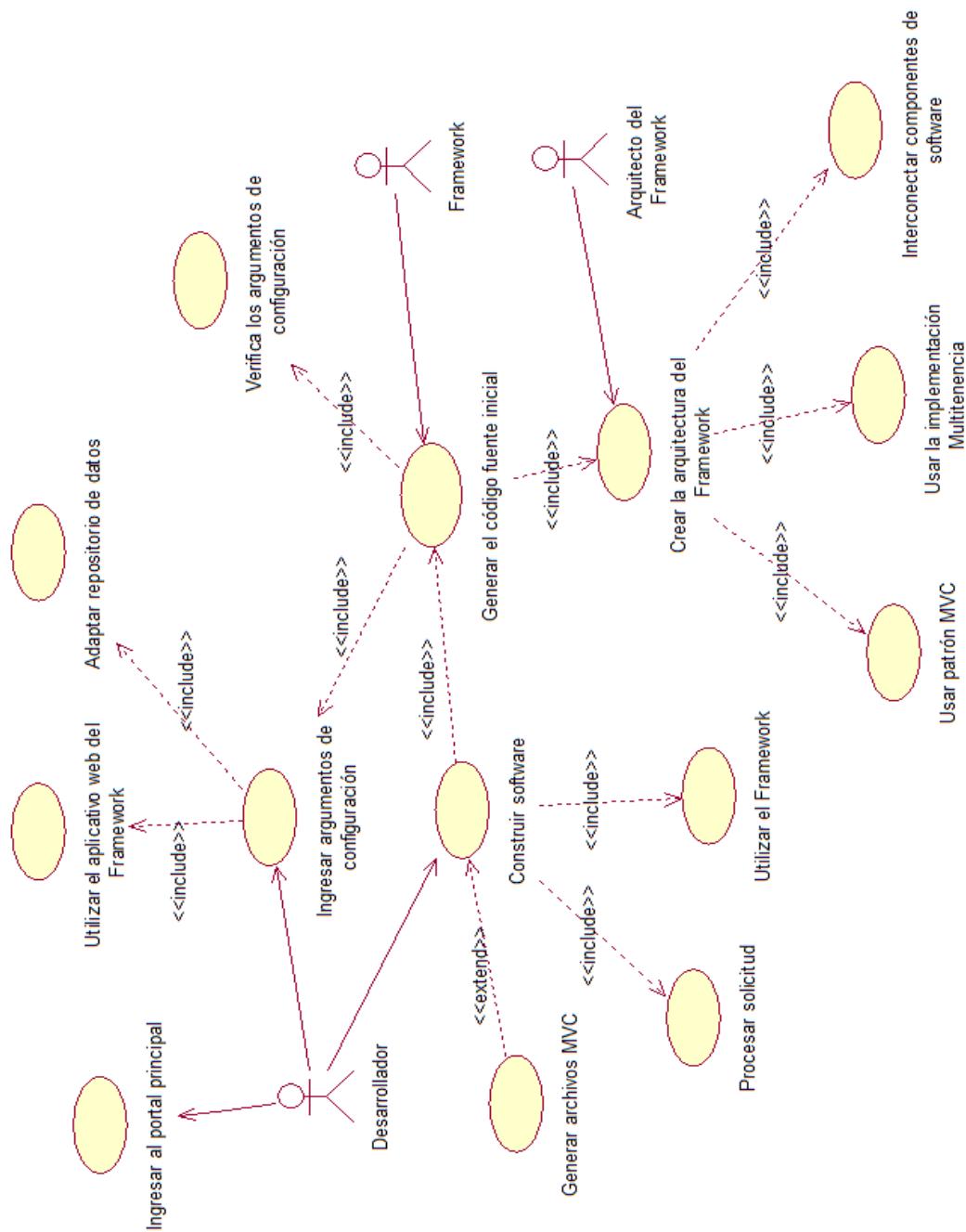


Figura 13: Diagrama de Casos de Uso del Framework.

Fuente: El Autor.

## b. Especificación de Casos de Uso

<b>Nombre</b>	Utilizar el aplicativo web del Framework
<b>Descripción</b>	El desarrollador debe iniciar el aplicativo web que trae consigo el Framework.
<b>Actor (es)</b>	Desarrollador
<b>Precondiciones</b>	<p>1.- Ingresar a SOURCEFORGE (<a href="https://sourceforge.net/projects/lizarframe/">https://sourceforge.net/projects/lizarframe/</a>) y realizar la descarga del Framework.</p> <p>2.- Tener configurado un servidor web de visualización de páginas web, utilización de PHP y módulos configurados para trabajar con gestores de base datos.</p> <p>3.- Colocar la carpeta descargada, de nombre LizarFrame, dentro del servidor web.</p>
<b>Eventos</b>	Iniciar el aplicativo web, colocando en el navegador la dirección http://localhost/LizarFrame, teniendo en cuenta que se utiliza el puerto 80 para brindar el servicio web.
<b>Postcondiciones</b>	La aplicación web se visualizará en el navegador. Si se utiliza otro puerto en el servicio web, se deberá hacer el cambio. Por ejemplo: http://localhost:8081/LizarFrame.

Tabla 2: Caso de Uso - Utilizar el aplicativo web del Framework.

Fuente: El Autor.

<b>Nombre</b>	Adaptar el repositorio de datos.
<b>Descripción</b>	En este caso de uso se creará la base de datos y las tablas ya pensadas en un análisis previo del Sistema de Información, así como también la ejecución del script de base de datos que trae consigo el

	Framework (se pueden descargar del aplicativo como texto plano del menú Descargas).
<b>Actor (es)</b>	Desarrollador
<b>Precondiciones</b>	<p>1.- Instalar el gestor de base de datos.</p> <p>2.- Crear la base de datos.</p> <p>3.- Descargar el script del Framework en la zona de descargas.</p>
<b>Eventos</b>	<p>1.- Ejecutar el script descargado del Framework.</p> <p>La ejecución deberá ser en la instancia de la base de datos creada anteriormente.</p>
<b>Postcondiciones</b>	Se podrá definir y crear los recursos necesarios del Sistema de Información a desarrollar.

*Tabla 3: Caso de Uso - Adaptar el repositorio de datos.*

*Fuente: El Autor.*

<b>Nombre</b>	Ingresar argumentos de configuración
<b>Descripción</b>	El desarrollador debe brindar los recursos o argumentos necesarios para que el Framework pueda instanciar el software de manera correcta.
<b>Actor (es)</b>	Desarrollador.
<b>Precondiciones</b>	<p>1.- Adaptar el repositorio de datos.</p> <p>2.- Desplegar el aplicativo web del Framework.</p>
<b>Eventos</b>	<p>El desarrollador deberá ir a la opción de menú llamada "Argumentos de configuración", dentro del aplicativo web. El desarrollador debe llenar los siguientes datos:</p> <p>1.- Carpeta contenedora: Nombre de carpeta contenedora del Sistema de Información a Instanciar.</p> <p>2.- Servidor: Nombre del servidor donde reside la base de datos.</p>

	<p>3.- Usuario: Nombre de usuario con el que se puede acceder a la base de datos.</p> <p>4.- Clave: Contraseña con la cual se accede a la base de datos.</p> <p>5.- Base de datos: Nombre de la base de datos.</p> <p>6.- Gestor de base de datos: Nombre del gestor de base de datos (Ejemplo: mssql, mysql).</p> <p>7.- Nombre de Aplicación.</p> <p>8.- Nombre del desarrollador.</p>
<b>Postcondiciones</b>	Se podrá utilizar el software instanciado o construido por el Framework.

*Tabla 4: Caso de Uso - Ingresar argumentos de configuración.*

*Fuente: El Autor.*

<b>Nombre</b>	Usar Patrón MVC
<b>Descripción</b>	El Arquitecto del Framework debe utilizar el patrón de diseño Modelo - Vista - Controlador para interconectar los componentes de software a ser utilizados por el desarrollador.
<b>Actor (es)</b>	Arquitecto del Framework.
<b>Precondiciones</b>	1.- Adquirir conocimiento sobre el patrón MVC.
<b>Eventos</b>	El Arquitecto del Framework utilizará e incorporará el patrón MVC en la Arquitectura a construir.
<b>Postcondiciones</b>	Se podrá utilizar el Framework, dividiendo las capas: vista, lógica del negocio y acceso a datos.

*Tabla 5: Caso de Uso - Usar Patrón MVC.*

*Fuente: El Autor.*

<b>Nombre</b>	Usar la implementación Multitenencia
<b>Descripción</b>	El Arquitecto del Framework debe utilizar la implementación Multitenencia para soportar la gestión de varios clientes a través de una sola instancia del software.
<b>Actor (es)</b>	Arquitecto del Framework
<b>Precondiciones</b>	1.- Adquirir conocimiento sobre la implementación Multitenencia.
<b>Eventos</b>	El Arquitecto del Framework utilizará e incorporará la implementación Multitenencia en la Arquitectura a construir.
<b>Postcondiciones</b>	Se podrá utilizar el Framework, gestionando una sola instancia de software para muchos clientes y usuarios.

*Tabla 6: Caso de Uso - Usar la implementación Multitenencia.*

*Fuente: El Autor.*

<b>Nombre</b>	Interconectar componentes de software.
<b>Descripción</b>	El Arquitecto del Framework debe interconectar los componentes tales como: carpetas, librerías, imágenes, estilos (CSS), archivos de configuración, archivos de validaciones, archivos que gestionan el patrón Modelo - Vista - Controlador, código SQL, implementación Multitenencia, estructura de carpetas.
<b>Actor (es)</b>	Arquitecto del Framework
<b>Precondiciones</b>	1.- Crear los componentes de manera independiente.
<b>Eventos</b>	El Arquitecto del Framework debe interconectar los componentes creados.
<b>Postcondiciones</b>	Se debe utilizar el Framework según la arquitectura

	creada, así mismo se deberá utilizar un manual de usuario provisto por el Framework.
--	--

*Tabla 7: Caso de Uso - Interconectar componentes.*

*Fuente: El Autor.*

<b>Nombre</b>	Crear la arquitectura del Framework
<b>Descripción</b>	El Arquitecto del Framework debe crear una arquitectura capaz de soportar los requerimientos funcionales analizados. La Arquitectura debe ser diseñada en base al patrón MVC y la implementación Multitenencia.
<b>Actor (es)</b>	Arquitecto del Framework.
<b>Precondiciones</b>	1.- Conocer el patrón MVC. 2.- Conocer la implementación Multitenencia. 3.- Interconectar los componentes del Framework.
<b>Eventos</b>	El Arquitecto del Framework analiza y crea una interconexión de componentes basados en el patrón MVC y la implementación Multitenencia.
<b>Postcondiciones</b>	Se podrá utilizar la arquitectura del Framework con la finalidad de construir software orientado a la prestación de servicios.

*Tabla 8: Caso de Uso - Crear la arquitectura del Framework.*

*Fuente: El Autor.*

<b>Nombre</b>	Verifica los argumentos de configuración.
<b>Descripción</b>	El Framework debe gestionar los argumentos de configuración colocados por el Desarrollador y verificar que dichos argumentos cumplan con el objetivo de instanciar el software.
<b>Actor (es)</b>	Framework

<b>Precondiciones</b>	1.- Ingresar los argumentos de configuración.
<b>Eventos</b>	El Framework verifica los argumentos de configuración en cuanto a: base de datos (nombre de servidor, puerto, usuario, contraseña, nombre de la base de datos), gestor de base de datos, nombre carpeta contenedora, nombre de aplicación.
<b>Postcondiciones</b>	El Framework resolverá y mostrará la debida notificación en cuanto a la aprobación o no de tales argumentos.

*Tabla 9: Caso de Uso - Verifica los argumentos de configuración.*

*Fuente: El Autor.*

<b>Nombre</b>	Generar el código fuente inicial.
<b>Descripción</b>	El Framework debe crear el código fuente necesario para poder brindar al desarrollador los estándares, técnicas y elementos necesarios capaces de llevar a cabo la construcción de software. Esta generación de código fuente debe realizarse sobre una Arquitectura previamente definida.
<b>Actor (es)</b>	Framework
<b>Precondiciones</b>	1.- Utilizar el aplicativo web del Framework. 2.- Ingresar los argumentos de configuración. 3.- Verifica si los argumentos de configuración son correctos. 4.- Crear una arquitectura para el Framework.
<b>Eventos</b>	El Framework genera código fuente: librerías, clases, objetos, estilos de páginas, formularios, imágenes, validaciones en formularios, despliegue de las vistas, llamadas a controladores, llamadas a modelos, conexión a la base de datos.
<b>Postcondiciones</b>	Una vez que se haya generado el código fuente en

	base a una arquitectura, se podrá construir software orientado a la prestación de servicios (modelo SaaS).
--	--

*Tabla 10: Caso de Uso - Generar el código fuente inicial.*

*Fuente: El Autor.*

<b>Nombre</b>	Utilizar el Framework
<b>Descripción</b>	El Desarrollador debe utilizar el Framework para construir el software necesario que llevará al desarrollo de un Sistema de Información como producto final.
<b>Actor (es)</b>	Desarrollador
<b>Precondiciones</b>	1.- Utilizar el aplicativo web del Framework. 2.- Ingresar los argumentos de configuración solicitados en la sección de "Definir Recursos" o "Argumentos de configuración". 3.- Esperar a que el Framework genere el código fuente y su arquitectura.
<b>Eventos</b>	El Desarrollador comienza a utilizar el Framework.
<b>Postcondiciones</b>	Revisión y prueba del código generado por el Desarrollador a través de la utilización del Framework.

*Tabla 11: Caso de Uso - Utilizar el Framework.*

*Fuente: El Autor.*

<b>Nombre</b>	Procesar solicitud.
<b>Descripción</b>	El Desarrollador debe procesar solicitudes en la construcción del software obtener la gestión de información provista o consultada.
<b>Actor (es)</b>	Desarrollador

<b>Precondiciones</b>	1.- Framework instanciado.
<b>Eventos</b>	1.- El Desarrollador envía una solicitud al Controlador y este gestiona el acceso a la base de datos o no, según se dé el caso. El controlador gestiona la visualización de la vista al usuario final. 3.- El Controlador debe gestionar toda conexión con la base de datos a través de un Modelo.
<b>Postcondiciones</b>	El Desarrollador visualiza la solicitud realizada.

Tabla 12: Caso de Uso - Procesar solicitud.

Fuente: El Autor.

<b>Nombre</b>	Generar archivos MVC
<b>Descripción</b>	El Desarrollador podrá generar los modelos que van a ser archivos que representarían a las tablas de la base de datos, creándose su propia clase en cada archivo modelo y definiendo en cada uno de ellos los atributos que vienen a ser los nombres de los campos de las tablas. Así también se generarán los controladores y las vistas para las tablas creadas en el repositorio de datos correspondiente.
<b>Actores</b>	Desarrollador.
<b>Precondiciones</b>	1.- Adaptar el repositorio de datos. 2.- El Framework debe instanciarse en el servidor web. 3.- Recursos creados (carpetas, archivos de configuración).
<b>Eventos</b>	1.- Ir al menú de "Generar Modelo - Vista - Controlador" en el aplicativo web del Framework. 2.- Colocar el nombre de la carpeta contenedora. 3.- Hacer clic en Crear Archivos, para que el Framework genere todos los archivos necesarios en

	la carpeta del Sistema de Información instanciado. 4- El Framework leerá las tablas creadas en la Base de Datos y los convertirá en archivos que corresponden a los modelos del Sistema Instanciado, así mismo, también se crearán sus correspondientes vistas y controladores.
<b>Postcondiciones</b>	Modelos, vistas y controladores creados.

*Tabla 13: Caso de Uso - Generar archivos MVC.*

*Fuente: El Autor.*

<b>Nombre</b>	Construir software.
<b>Descripción</b>	El Desarrollador debe comenzar a construir software orientado a la prestación de servicios.
<b>Actor (es)</b>	Desarrollador
<b>Precondiciones</b>	1.- Generar código fuente inicial. 2.- Crear una arquitectura de software definida. 3.- Utilizar la arquitectura del Framework.
<b>Eventos</b>	El Desarrollador construye software gracias a la utilización del Framework. El Desarrollador procesa solicitudes para gestionar los componentes involucrados en el Framework.
<b>Postcondiciones</b>	Realizar las pruebas necesarias en la construcción del software. El Desarrollador puede o no generar rápidamente los archivos MVC provenientes de las entidades construidas en la base de datos.

*Tabla 14: Caso de Uso - Construir software.*

*Fuente: El Autor.*

<b>Nombre</b>	Ingresar al portal principal.
<b>Descripción</b>	El Desarrollador debe iniciar una sesión de usuario a través de un portal principal. Deberá existir en el

	portal principal un formulario de inicio de sesión para ingresar al sistema instanciado por el Framework.
<b>Actor (es)</b>	Desarrollador
<b>Precondiciones</b>	1.- El Framework debe instanciar el software. 2.- El Framework debe crear el código fuente inicial y su arquitectura. 3.- El Framework provee un usuario y contraseña creados por defecto. Usuario: admin@miempresa.com y contraseña: admin.
<b>Eventos</b>	El Desarrollador debe ingresar al servidor web y colocar el nombre de su aplicación a través de la URL. Por ejemplo: http://localhost/miAplicacion. El desarrollador debe colocar sus credenciales y dar inicio a la sesión.
<b>Postcondiciones</b>	Revisar las funcionalidades del software creado.

*Tabla 15: Caso de Uso - Ingresar al portal principal.*

*Fuente: El Autor.*

### c. Diagramas de Secuencia

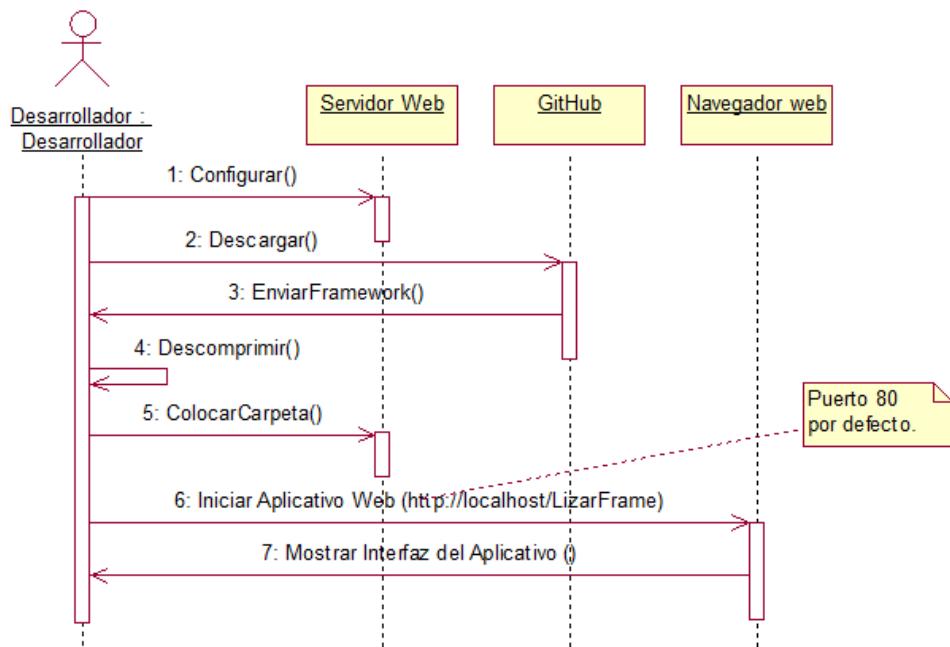


Figura 14: Diagrama de Secuencia - Utilizar el aplicativo web del Framework.

Fuente: El Autor.

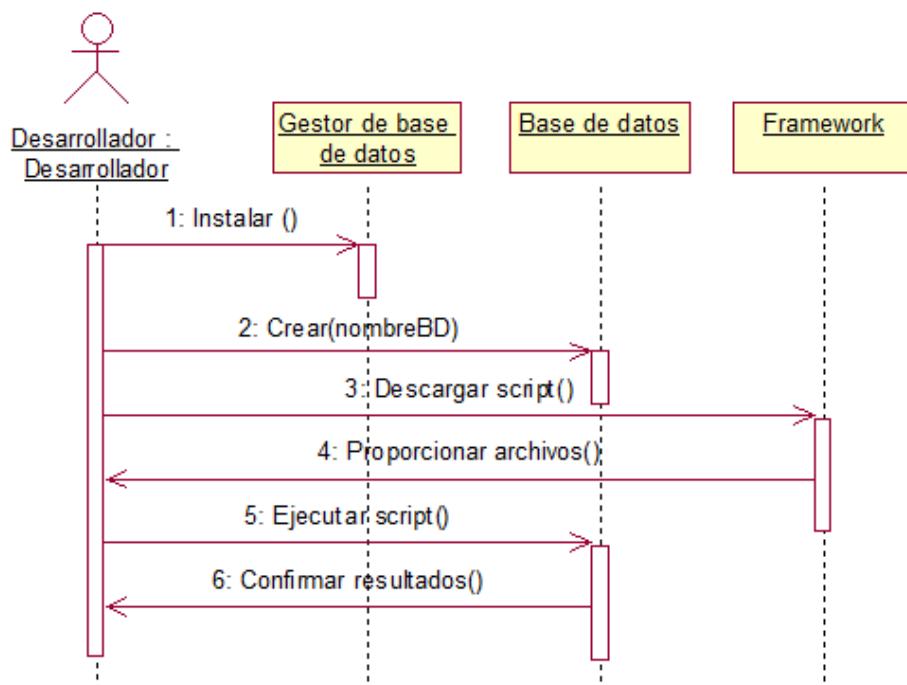


Figura 15: Diagrama de Secuencia - Adaptar el repositorio de datos.

Fuente: El Autor.

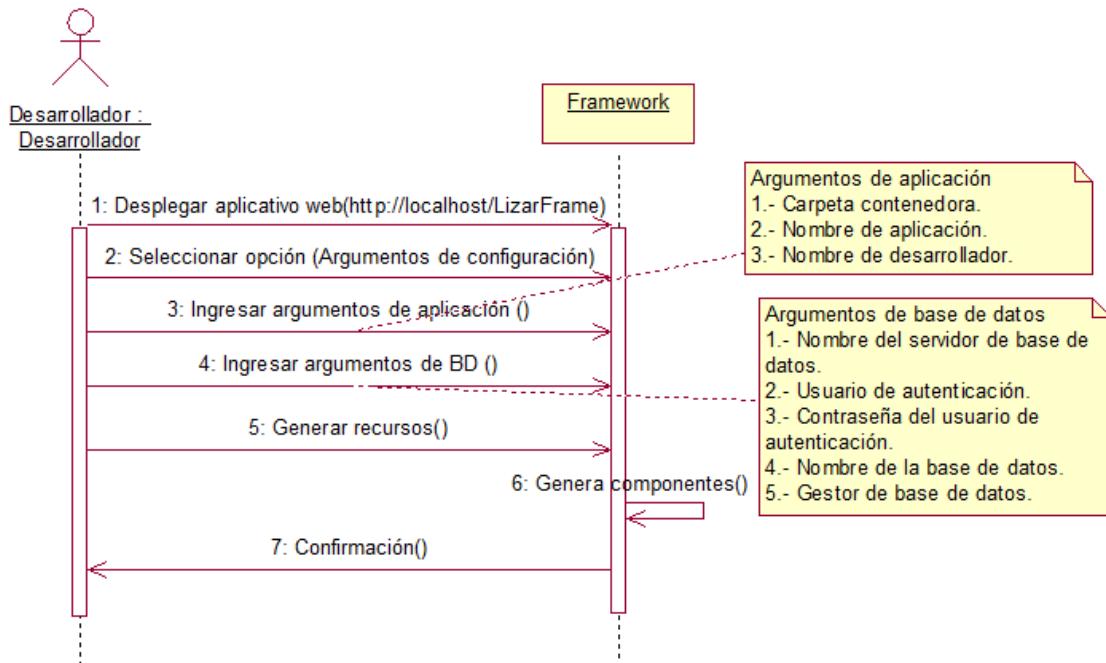


Figura 16: Diagrama de Secuencia - Ingresar argumentos de configuración.

Fuente: El Autor.

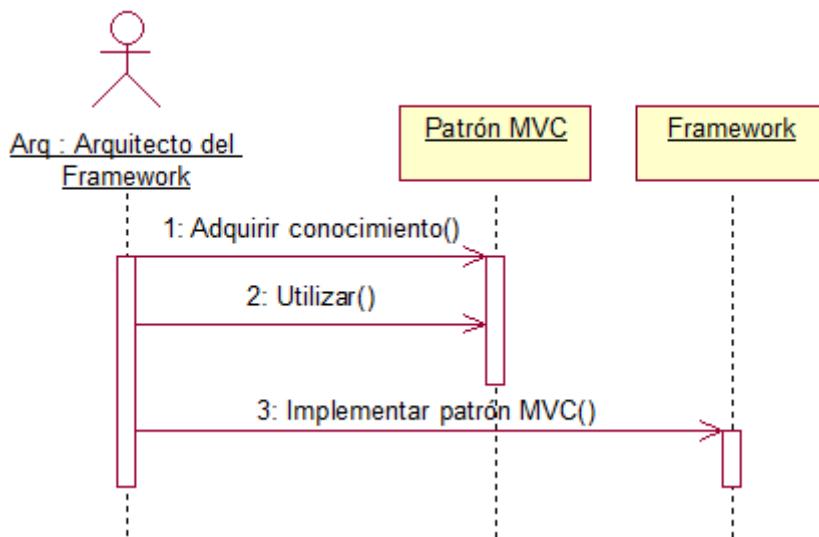


Figura 17: Diagrama de Secuencia - Usar Patrón MVC.

Fuente: El Autor.

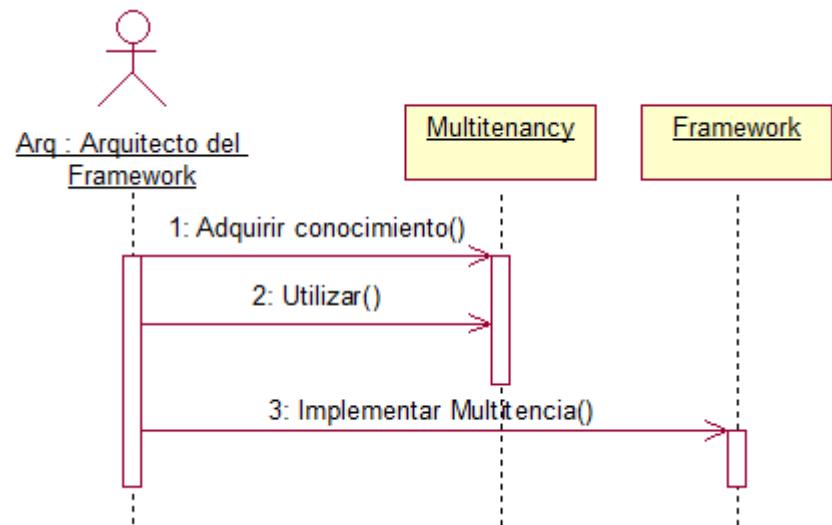


Figura 18: Diagrama de Secuencia - Usar la implementación Multitenencia.

Fuente: El Autor.

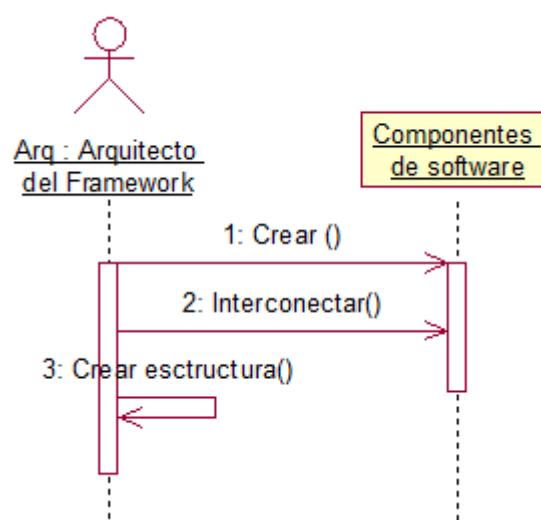


Figura 19: Diagrama de Secuencia - Interconectar componentes de software.

Fuente: El Autor.

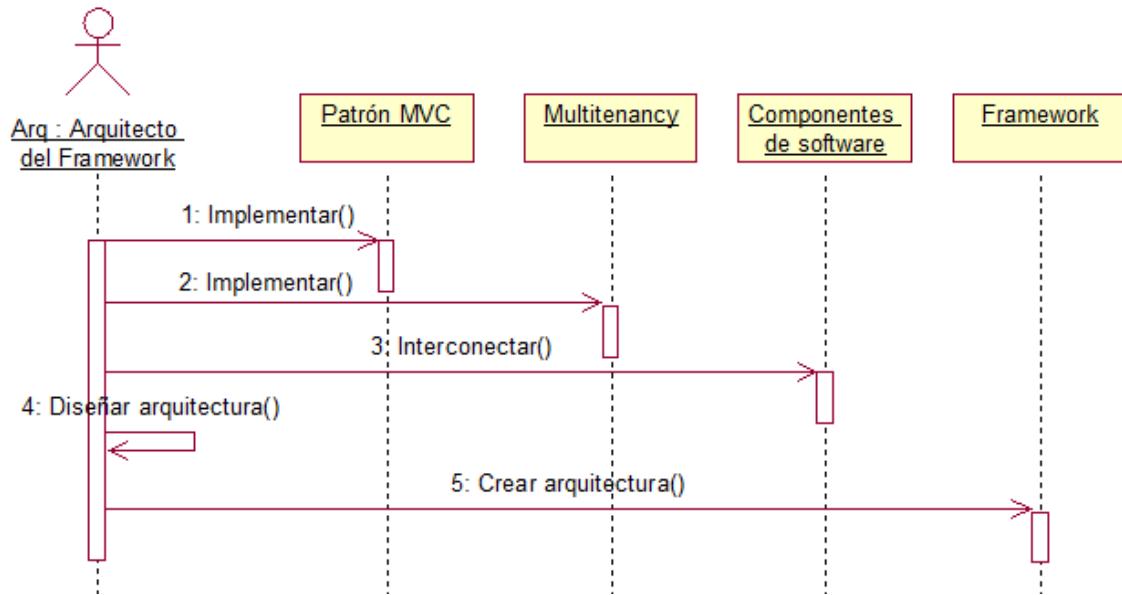


Figura 20: Diagrama de Secuencia - Crear la arquitectura del Framework.

Fuente: El Autor.

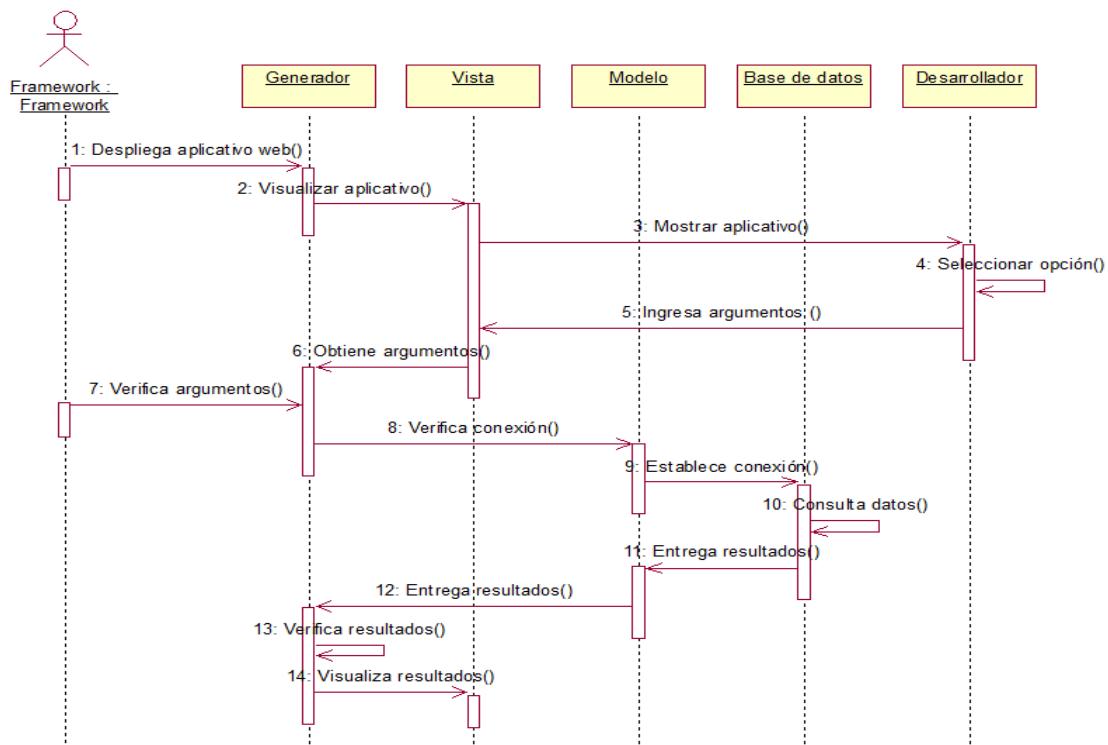


Figura 21: Diagrama de Secuencia - Verifica los argumentos de configuración.

Fuente: El Autor.

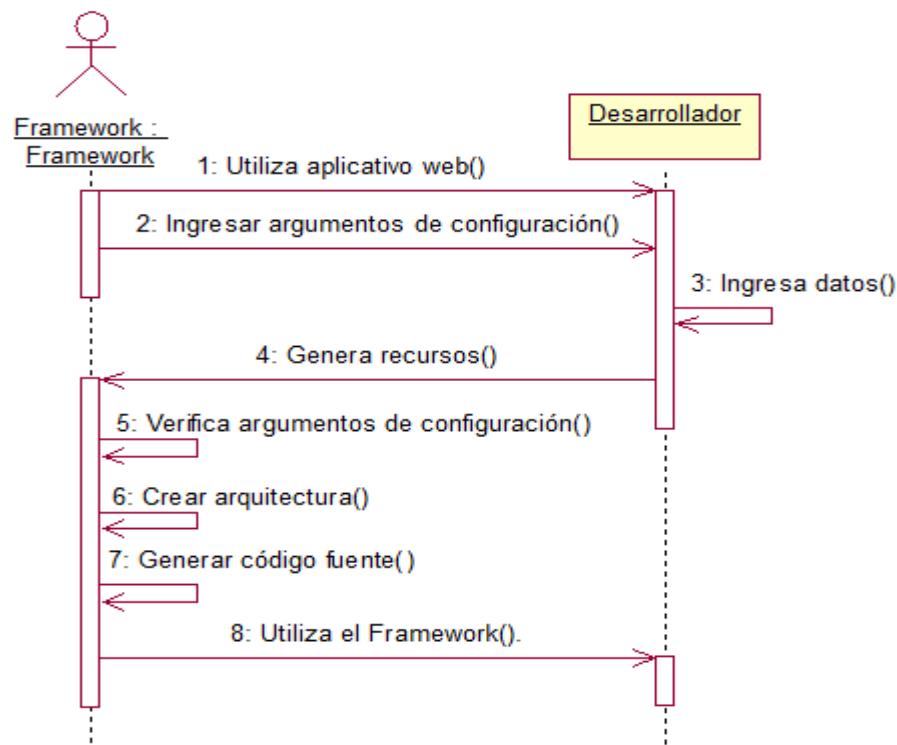


Figura 22: Diagrama de Secuencia - Generar el código fuente inicial.

Fuente: El Autor.

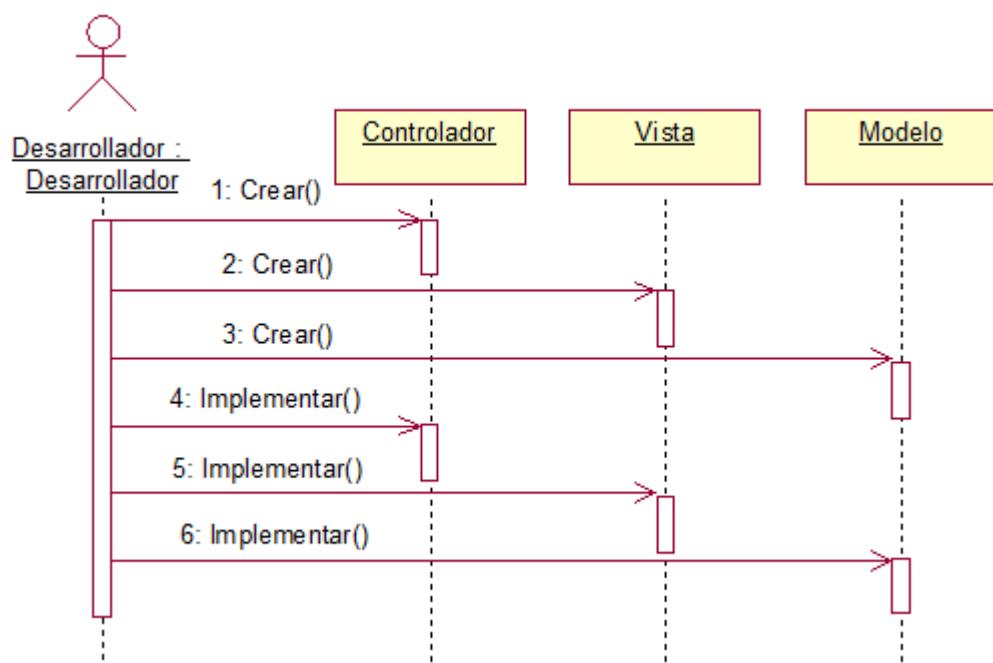


Figura 23: Diagrama de Secuencia - Utilizar el Framework.

Fuente: El Autor.

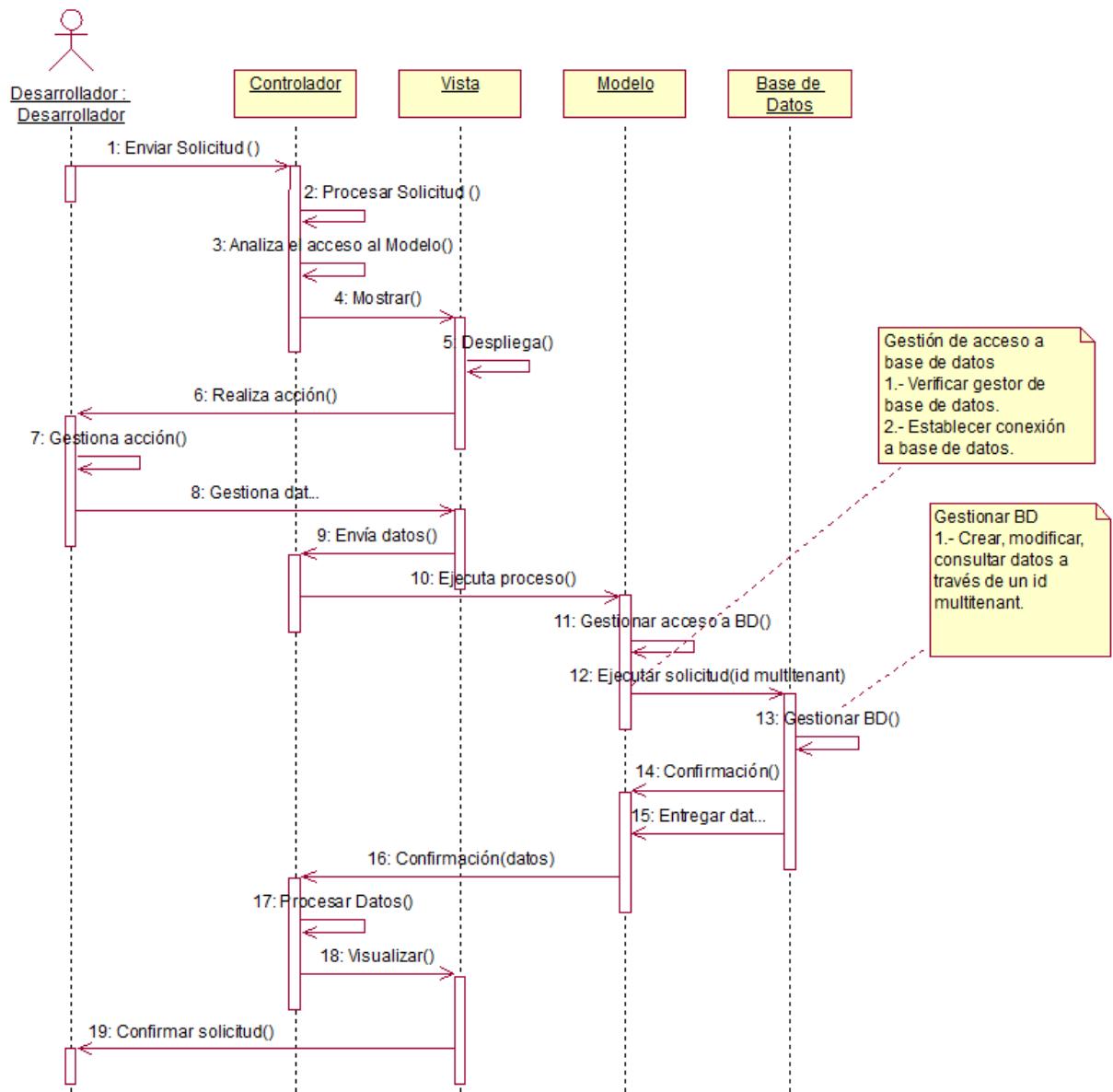


Figura 24: Diagrama de Secuencia - Procesar solicitud.

Fuente: El Autor.

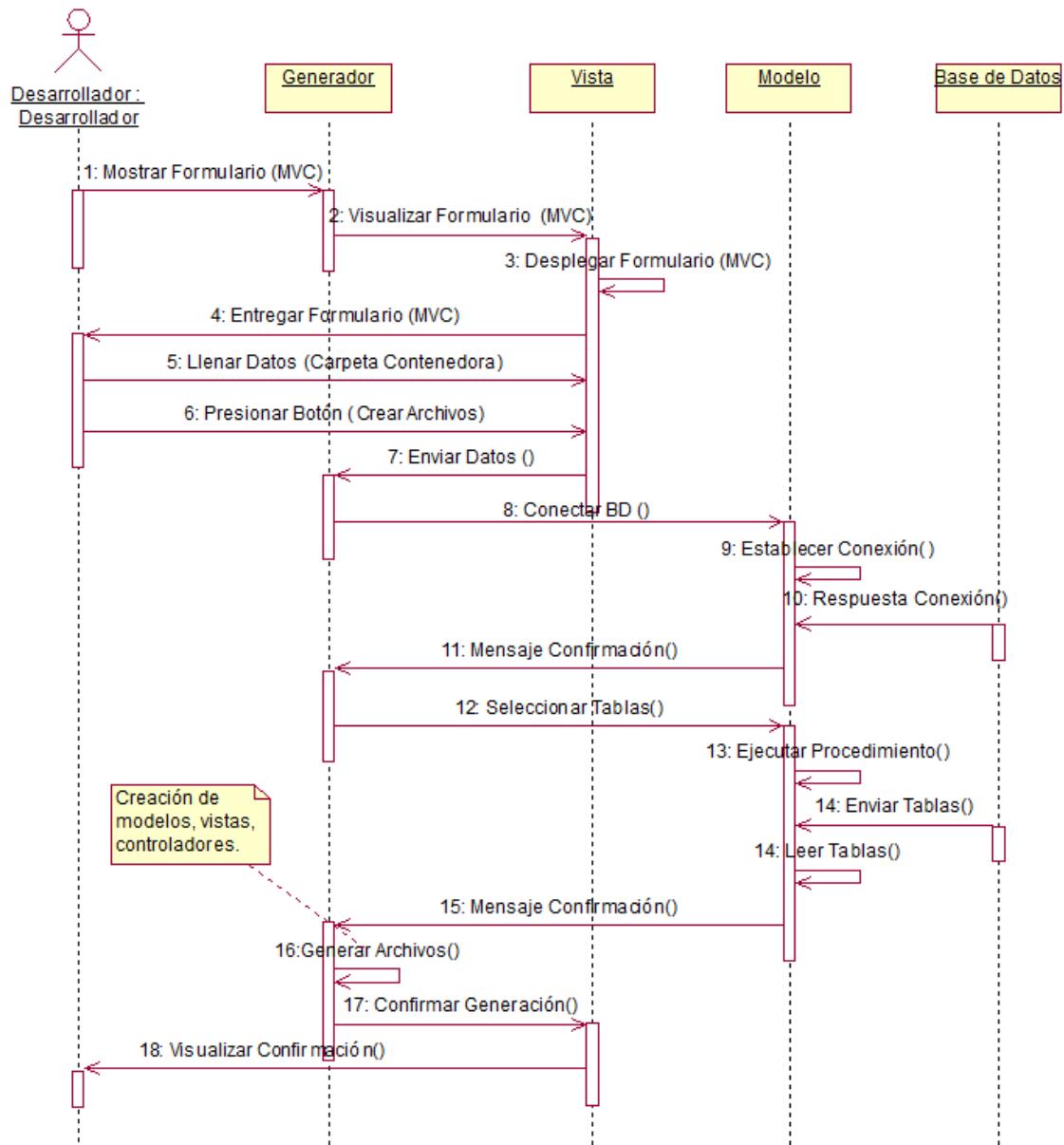


Figura 25: Diagrama de Secuencia - Generar archivos MVC.

Fuente: El Autor.

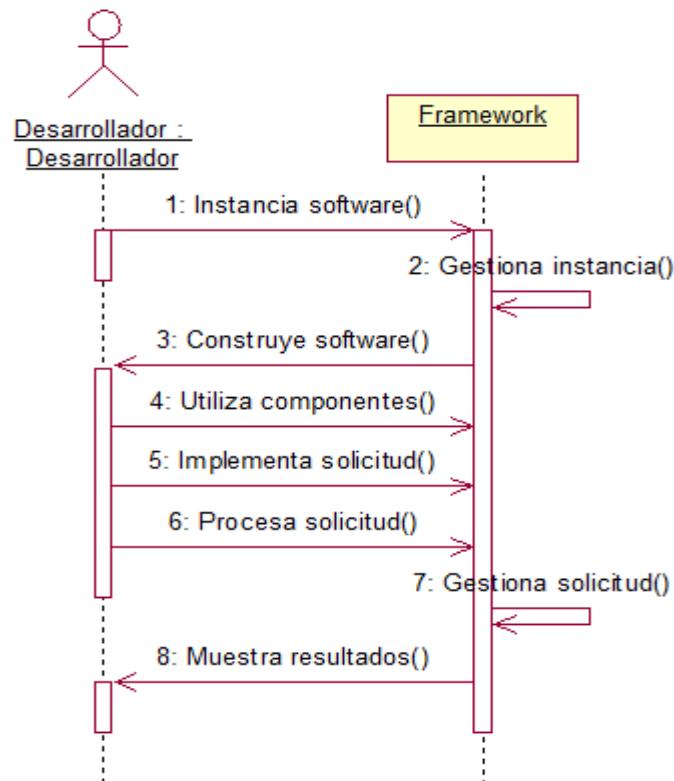


Figura 26: Diagrama de Secuencia - Construir software.

Fuente: El Autor.

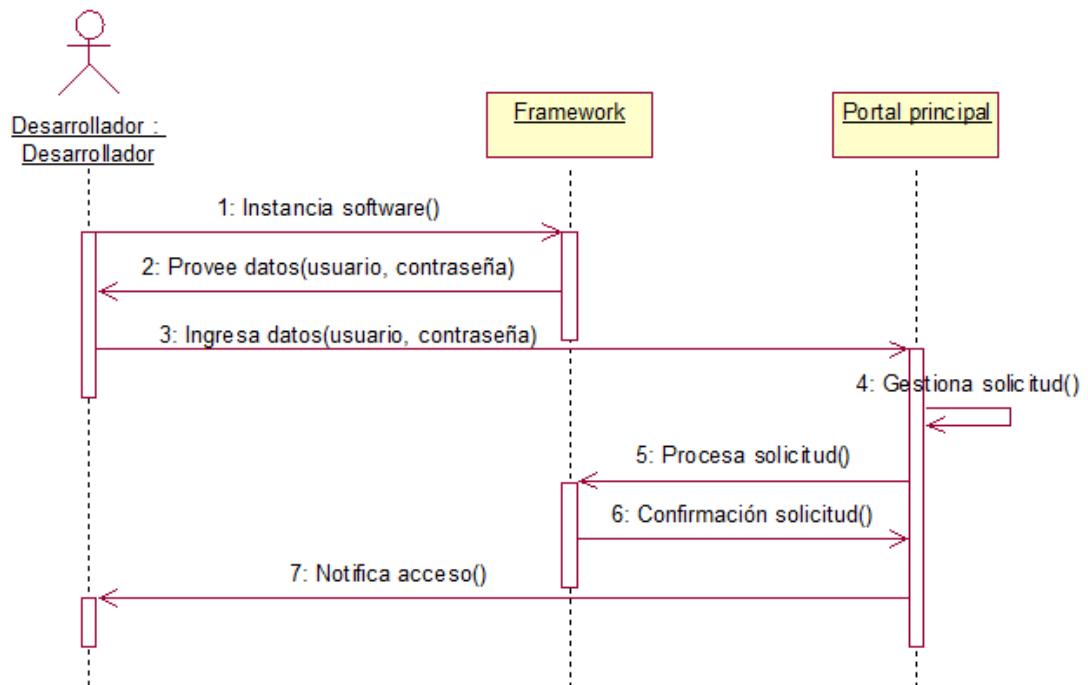


Figura 27: Diagrama de Secuencia - Ingresar al portal principal.

Fuente: El Autor.

## d. Diagrama de Clases

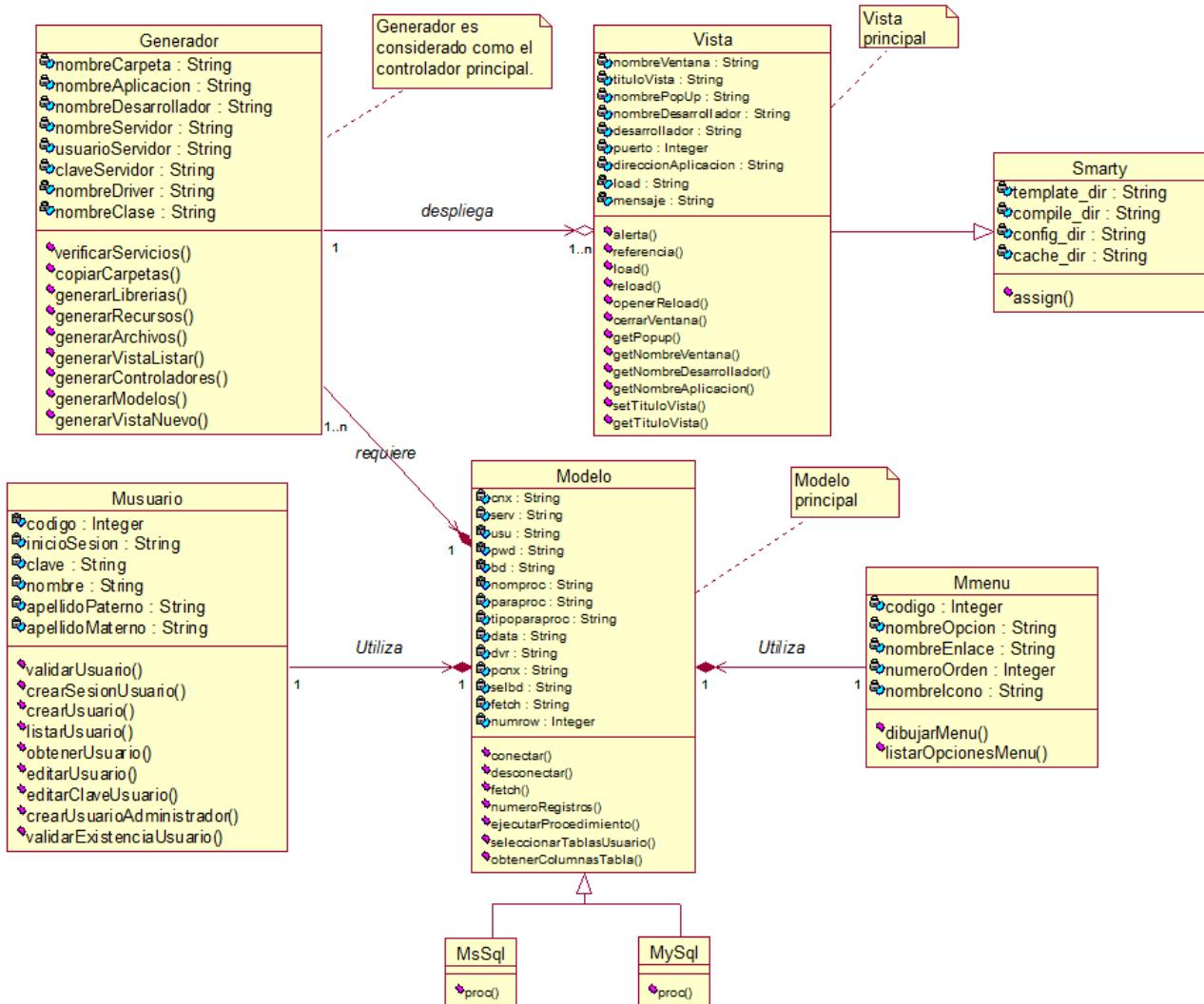


Figura 28: Diagrama de Clases del Framework.

Fuente: El Autor.

## e. Diagrama de Componentes

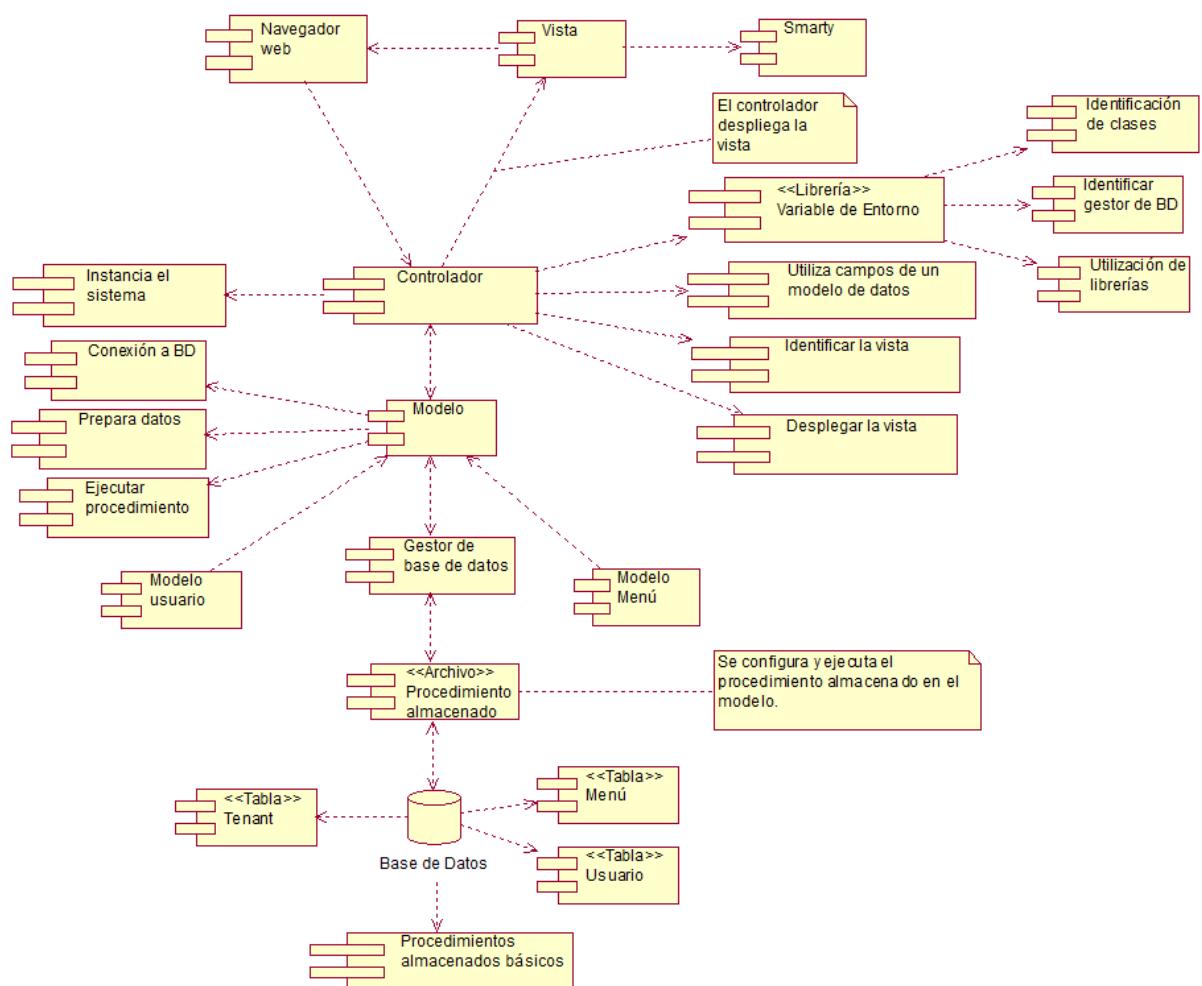
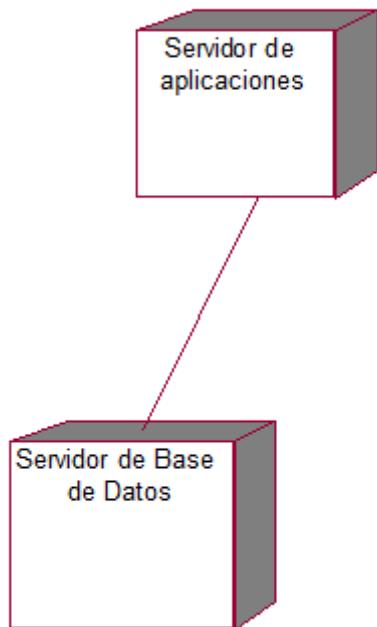


Figura 29: Diagrama de Componentes del Framework.

Fuente: El Autor.

#### f. Diagrama de Despliegue



*Figura 30: Diagrama de Despliegue del Framework.*

*Fuente: El Autor.*

### g. Diagrama Entidad Relación (E-R)

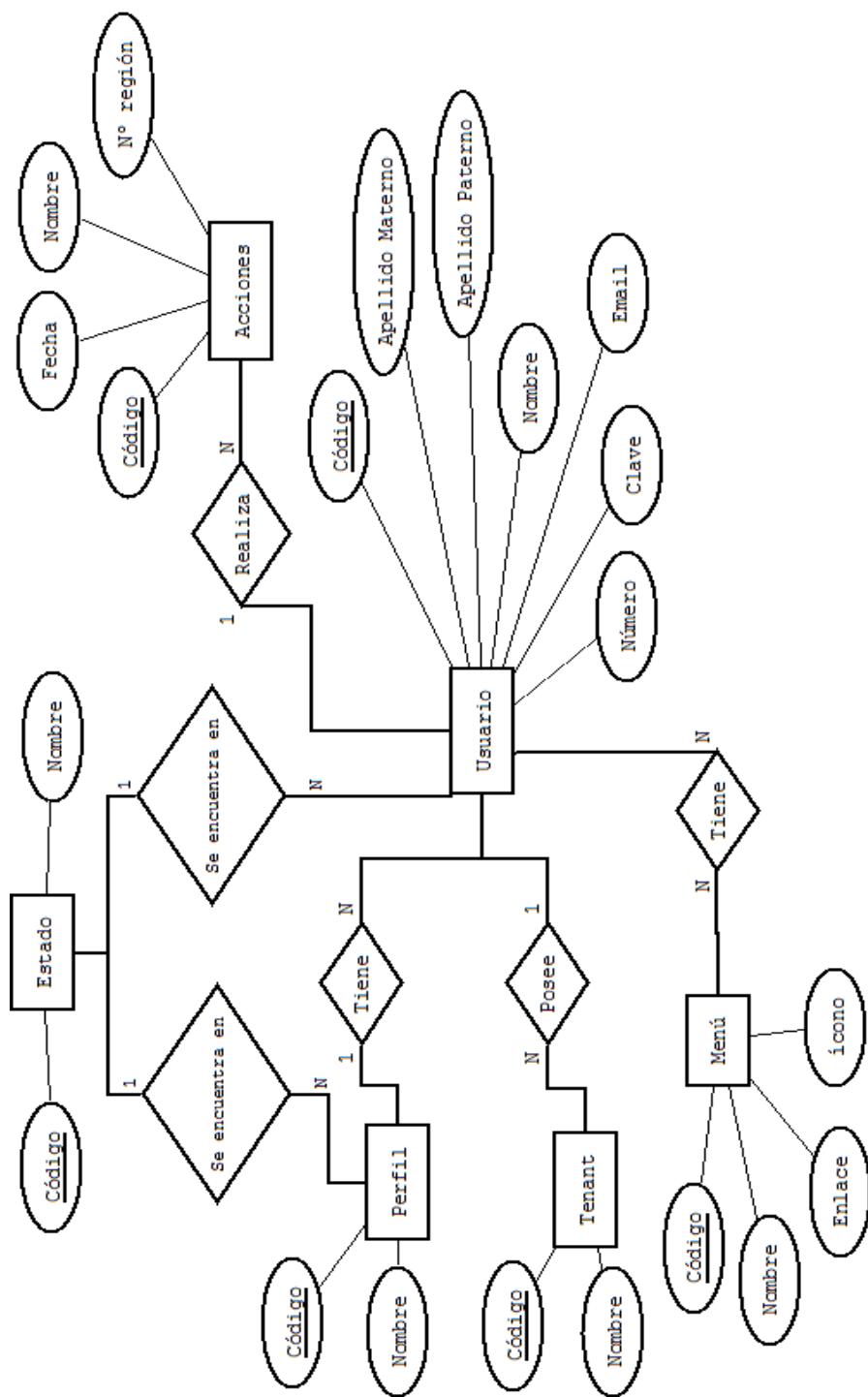


Figura 31: Diagrama Entidad Relación (E-R).

Fuente: El Autor.

## h. Diccionario de Base de Datos

Nombre de Campo	Clave Primaria	Clave Foránea	Tipo de Dato	Longitud	Valor nulo	Descripción
ipkcodaud	X		int	4	No	Código principal de auditoría.
dfecaud			datetime	8	No	Fecha de auditoría.
ifkcodusu		X	int	4	No	Código de usuario.
cnamaud			varchar	70	No	Nombre de auditoría.
inumreg			int	4	No	Número de región afectada.

*Tabla 16: Base de Datos - Tabla AuditoriaMaestros.*

*Fuente: El Autor.*

Nombre de Campo	Clave Primaria	Clave Foránea	Tipo de Dato	Longitud	Valor nulo	Descripción
ipkcodest	X		tinyint	1	No	Código principal de estado.
cdescest			varchar	15	No	Descripción de estado.

*Tabla 17: Base de Datos - Tabla Mestado.*

*Fuente: El Autor.*

Nombre de Campo	Clave Primaria	Clave Foránea	Tipo de Dato	Longitud	Valor nulo	Descripción
ipkcodmenu	X		smallint	2	No	Código principal del menú.
cnomopc			varchar	64	No	Nombre de la opción de menú.
cnomenla			varchar	500	No	Nombre de la página (referencia).
icodpadr			int	4	No	Código de padre en el menú.
iordmenu			int	4	Si	Número de orden de la opción de menú.
cicomenu			varchar	50	Si	Ícono de menú.

*Tabla 18: Base de Datos - Tabla Mmenu.*

*Fuente: El Autor.*

Nombre de Campo	Clave Primaria	Clave Foránea	Tipo de Dato	Longitud	Valor nulo	Descripción
ipkcodperf	X		tinyint	1	No	Código principal del perfil.
cnomperf			varchar	15	No	Nombre de perfil.

*Tabla 19: Base de Datos - Tabla Mperfil.*

*Fuente: El Autor.*

Nombre de Campo	Clave Primaria	Clave Foránea	Tipo de Dato	Longitud	Valor nulo	Descripción
ipkcodten	X		int	4	No	Código principal de tenant.
cnomten			varchar	30	No	Nombre de tenant.

Tabla 20: Base de Datos - Tabla Mtenant.

Fuente: El Autor.

Nombre de Campo	Clave Primaria	Clave Foránea	Tipo de Dato	Longitud	Valor nulo	Descripción
ipkcodusu	X		int	4	No	Código principal de usuario.
ifkcodperf		X	tinyint	1	No	Código de perfil.
inrousu			tinyint	1	No	Número de usuario.
cnomusu			varchar	50	No	Nombre(s) de usuario.
capepatusu			varchar	50	No	Apellido paterno de usuario.
capematusu			varchar	50	No	Apellido materno de usuario.
cmailusu			varchar	60	No	Correo electrónico de usuario.
cclavusu			varchar	50	No	Clave de usuario.
ifkcodest		X	tinyint	1	No	Código de estado.
ifkcodten		X	int	4	No	Código de tenant.

Tabla 21: Base de Datos - Tabla Musuario.

Fuente: El Autor.

Nombre de Campo	Clave Primaria	Clave Foránea	Tipo de Dato	Longitud	Valor nulo	Descripción
ifkcodusu		X	int	4	No	Código de usuario.
ifkcodmenu		X	smallint	2	No	Código de menú.

Tabla 22: Base de Datos - Tabla MusuarioMenu.

Fuente: El Autor.

Nombre de tabla	Columna	Nombre de restricción
AuditoriaMaestros	ipkcodaud	PK_AuditoriaMaestros
Mestado	ipkcodest	PK_Mestado
Mmenu	ipkcodmenu	PK_Mmenu
Mperfil	ipkcodperf	PK_Mperfil
Mtenant	ipkcodten	PK_Mtenant
Musuario	ipkcodusu	PK_Musuario

Tabla 23: Base de Datos - Claves Primarias.

Fuente: El Autor.

Clave Foránea	Tabla	Columna	Tabla referenciada	Columna referenciada
FK_Musuario_Mestado	Musuario	ifkcodest	Mestado	ipkcodest
FK_Musuario_Mperfil	Musuario	ifkcodperf	Mperfil	ipkcodperf
FK_Musuario_Mtenant	Musuario	ifkcodten	Mtenant	ipkcotden
FK_AuditoriaMaestros_Musuario	AuditoriaMaestros	ifkcodusu	Musuario	ipkcodusu
FK_MusuarioMenu_Mmenu	MusuarioMenu	ifkcodmenu	Mmenu	ipkcodmenu
FK_MusuarioMenu_Musuario	MusuarioMenu	ifkcodusu	Musuario	ipkcodusu

Tabla 24: Base de Datos - Claves Foráneas.

Fuente: El Autor.

Nº	Nombre
1	pa_activarUsuario
2	pa_crearUsuario
3	pa_crearUsuarioAdministrador
4	pa_editarClaveUsuario
5	pa_editarUsuario
6	pa_listaMenu
7	pa_listarEstado
8	pa_listarPerfiles
9	pa_listarUsuario
10	pa_listaUsuario
11	pa_obtenerUsuario
12	pa_seleccionarColumnasTabla
13	pa_seleccionarTablasUsuario
14	pa_validarExistenciaUsuario

Tabla 25: Base de Datos - Procedimientos almacenados.

Fuente: El Autor.

Para mayor detalle sobre el diseño de la base de datos, se debe ingresar a la plataforma web del Framework, en donde se encuentra el Script de Base de Datos.

## i. Selección de la Plataforma Tecnológica

**1. Lenguaje de programación:** Para construir el Framework se ha elegido a PHP como lenguaje de programación en su versión 5.2 para el lado del servidor. Esta elección obedece a las siguientes características analizadas:

- Permite a la mayoría de los desarrolladores, crear aplicaciones complejas con una curva de aprendizaje muy corta.
- Soporta XML, servicios web, librerías, integración con otros lenguajes de programación, independencia del sistema operativo, interacción con muchos motores de base de datos.
- Permite utilizar técnicas de programación orientada a objetos.
- Los desarrollos de aplicaciones se hacen rápidas debido a que es código interpretado y no necesita compilarse cada vez que haya cambios, lo cual beneficia al implementador en cuanto a rapidez en construcción de aplicaciones.
- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.

**2. Gestor de base de datos:** La elección del gestor de base de datos se realizó a través de la comparación en la Tabla 26. Se elige a SQL Server (versión Express) como gestor de base de datos nativo para la implementación del Framework, obedeciendo tal elección a las siguientes características:

- SQL SERVER como base de datos comercial cuenta con soporte y respaldo garantizado, es decir que, si existe algún tipo de daño en el producto y este comprometiese la información, existe una empresa

que va a responder por estos daños (no quiere decir que las otras bases de datos comerciales no cuenten con soporte y respaldo).

- Es una base de datos intuitiva y fácil de manejar.
- Utilización del paquete Business Intelligence. Para la presente investigación se toma en cuenta solamente Reporting Services como principal diseñador de informes.

	MySQL	SQL Server	Oracle
Fabricante	Oracle (desde 2010)	Microsoft	Oracle
Licencia (privada, gratuita)	Libre a nivel de usuario, pero, para las empresas que quieran incorporarlo en productos privados, deben comprar licencia.	Privada - Microsoft	Privada
Ventajas	Es rápido, fiable y fácil de usar, por la cooperación con los usuarios desarrolladores de Open Source a nivel mundial	Facilidad de instalación y configuración. Es SGBDR - Sistema de Gestión de Bases de Datos Relacionales	Puede ejecutarse en todas las plataformas, desde una PC hasta un supercomputador.
	Una gran cantidad de software de contribuciones está disponible para MySQL, y por esta razón se pueden usar muchas herramientas disponibles en modo consola y en entorno gráfico. Gran compatibilidad entre distintos sistemas o plataformas	Cuando SQL Server no tiene tareas de usuario para procesar, comienza a escribir automáticamente las memorias intermedias sucias del caché al disco. Como estas escrituras se realizan en los ciclos de inactividad del servidor, se denominan escrituras libres.	Permite el uso de particiones para la mejora de eficiencia, de replicación e incluso ciertas versiones admiten la administración de base de datos distribuidos. El software del servidor puede ejecutarse en multitud de sistemas operativos
	Soporta muchos modos asignados para comportarse como otros gestores de base de datos.	Permite administrar permisos a todo: nivel de servidor, seguridad en tablas, lectura, escritura, ejecución, seguridad en procedimientos almacenados.	Existe una versión personal, lo cual es bueno para los desarrolladores que se llevan trabajo a casa. Soporte de transacciones. Estabilidad.
Desventajas	No cuenta con soporte. No sincroniza los datos con otras (réplica). El soporte para	Si se utiliza para prácticas, no va a ser muy útil porque se prohíben muchas cosas, restricciones.	El mayor inconveniente de Oracle es quizás su precio, incluso las licencias de Personal Oracle son

disparadores es básico, por lo tanto, hay ciertas limitaciones.		excesivamente caras.
	La mayor desventaja sería la gran cantidad de memoria RAM que utiliza para la instalación y utilización del software.	Necesidad de ajustes: un Oracle mal configurado puede llegar a ser muy lento.
	La relación calidad-precio está muy debajo comparado con Oracle	También el elevado coste de formación.

*Tabla 26: Cuadro comparativo de Gestores de Base de Datos.*

*Fuente: El Autor.*

**3. Modelo SaaS:** En el modelo SaaS, se puede distinguir tres categorías que pueden ser utilizadas, las cuales son:

- Una base de datos por cada empresa o tenant.
- Una base de datos con N conjuntos de tablas.
- Una base de datos con ÚNICO conjunto de tablas

De las anteriores tres categorías, se utilizará la tercera, debido a las características ofrecidas y vistas en la siguiente figura:



*Figura 32: Beneficios del SaaS.*

*Fuente: <http://www.saasmania.com/blog/tag/multitenancy/>.*

Por lo tanto, al analizar la categoría del modelo SaaS, y por consiguiente su elección, se concluye que:

- La razón de elegir la tercera categoría, es por la facilidad de mantenimiento y flexibilidad en construir nuevos modelos de negocio en uno ya instanciado y que puedan ser propagados a todos los clientes que optan por tener el servicio prestado.
- En vez de recolectar los datos desde múltiples fuentes utilizando probablemente diferentes esquemas de bases de datos, toda la información de los clientes es almacenada en un esquema común y único. Por lo tanto, la ejecución de peticiones a la base de datos sobre los clientes para completar diferentes tareas, son mucho más simples de realizar.
- Solamente se tendrá una instancia de aplicación y base de datos para suministrar el servicio a varios clientes.

**4. Patrón de diseño:** Se utiliza el Patrón de Diseño Modelo Vista Controlador (MVC), debido a que separa cualquier aplicación en un mínimo de tres capas (Presentación, Lógica de Negocio y acceso a datos), siendo esto beneficioso para el desarrollador ya que le permite un fácil mantenimiento del código fuente. Debido a que se trata de un Framework, se debe facilitar también la reutilización de código y reducir el esfuerzo de programación, para lo cual este patrón de diseño es muy bueno para realizar dichas tareas.

## j. Arquitectura del Framework

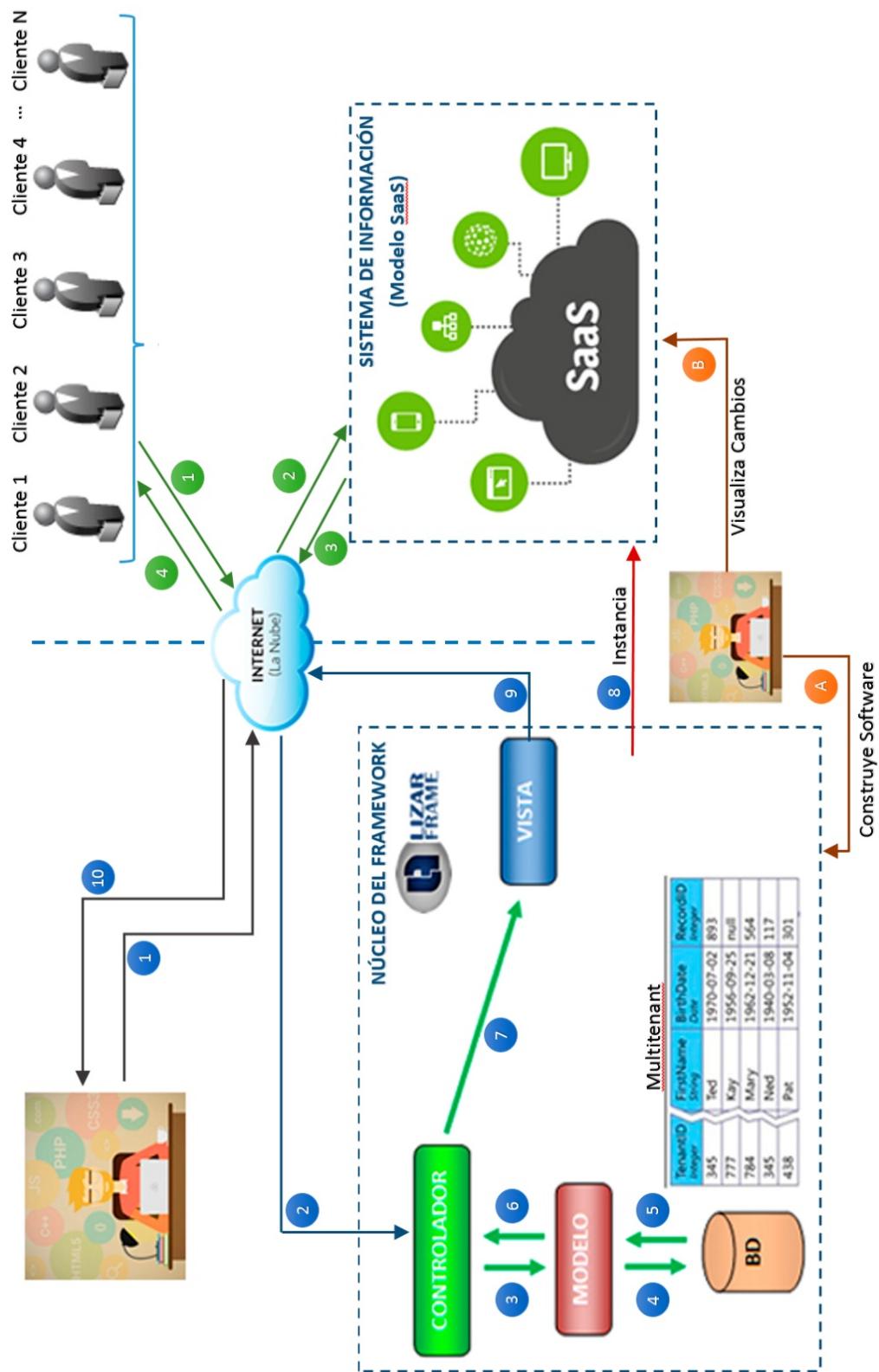


Figura 33: Arquitectura del Framework.

Fuente: El Autor.

A continuación, la interpretación de la secuencia en la Arquitectura del Framework:

**Para efectos de instanciar un Sistema de Información (modelo SaaS):**

- 1.** La primera vez que el desarrollador o entidad que desea instanciar un sistema de información orientado a la prestación de servicios, deberá colocar la dirección web del Framework.
- 2.** El núcleo del Framework recibe la acción del punto N° 1 y comienza a gestionar los recursos a través del Controlador.
- 3.** El controlador es el que decide si extrae o no datos del modelo, esto depende si la entidad externa requiere actualizar, ingresar, listar o realizar algún proceso con la base de datos. En el controlador se instancian objetos del modelo y/o vista, se llaman a funciones, se incluyen scripts, librerías, etc. así como también maneja los eventos de la entidad (ejemplo: GET - POST).
- 4.** El Modelo realiza la conexión y ejecución del procedimiento almacenado en la base de datos. Cabe resaltar que, la base de datos se encuentra bajo el modelo Multitenant, donde se tiene una sola base de datos, diferenciando a los tenant a través de un campo definido.
- 5.** La base de datos provee la información hacia el Modelo a través de la ejecución de su procedimiento almacenado.
- 6.** El Modelo se encarga de entregar los resultados al Controlador, quien se encargará de la lógica del negocio.
- 7.** El Controlador entrega la data procesada hacia la Vista.
- 8.** El núcleo del Framework es capaz de instanciar un sistema de información orientado a la prestación de servicios (modelo SaaS).

**9.** La Vista es colocada en la Internet para que sea visualizada por el desarrollador.

**10.** El desarrollador verifica los cambios a través de su monitor.

**Para efectos de utilizar el Sistema de Información (modelo SaaS):**

**1.** Un cliente o tenant deberá colocar la dirección web del sistema de información creado; pudiendo existir más de un cliente.

**2.** El sistema es requerido por el cliente a través de la nube o Internet.

**3.** El sistema de Información provee la información a través de la Vista del Framework y es colocada en la nube.

**4.** La nube proporciona al cliente, la vista del sistema de información.

**Para efectos de construir software en el Sistema de Información (modelo SaaS):**

**A.** El desarrollador debe construir software a través del núcleo del Framework.

**B.** El desarrollador verifica los cambios en el Sistema de Información previamente instanciado.

**C.** El desarrollador puede repetir los pasos del 1 al 10 sin necesidad de instanciar el Sistema de Información, si no, utilizando los componentes de la web y/o núcleo del Framework.

### **3.3.2. Pruebas del Framework**

#### **a. Especificación de casos de prueba**

Esta especificación provee una visión general para definir la estrategia de pruebas del Framework. Se utilizarán criterios de prueba bajo el concepto de caja negra, compatibilidad y concurrencia.

## **b. Objetivos de los casos de prueba**

Encontrar y solucionar los posibles errores para mejorar la integridad del Framework. A pesar de que no se puede realizar un trabajo de pruebas exhaustivo, se realizará la mayor cantidad de pruebas posible para lograr esta meta. Además, se desea lograr que todas las características presentadas en la especificación de requerimientos funcionen adecuadamente.

## **c. Plan de pruebas**

El flujo general del proceso de pruebas será presentado a continuación:

- Identificación de los requerimientos a ser probados.
- Identificación de los casos de prueba que serán utilizados para probar cada módulo.
- Realización de las pruebas.
- Se hará uso de una plantilla para reportar cada caso de pruebas del Framework.
- Cada caso de pruebas que no resulte exitoso, será verificado y corregido en el Framework; por consiguiente, se realizarán los casos de prueba nuevamente.

## **d. Tipos de pruebas**

Para la validación de las pruebas unitarias de la funcionalidad de cada módulo del sistema se elige el tipo de prueba llamado Caja Negra, el cual evalúa las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. Así mismo, se han realizado pruebas no funcionales del sistema (compatibilidad, concurrencia y carga de datos).

## **e. Alcance**

- Ejecución del script de base de datos que provee el Framework.
- Creación de recursos para la instancia del Framework.

- Creación de archivos Modelo - Vista - Controlador.
- Ingresar al sistema instanciado.
- Registrar un tenant o cliente.

#### **f. Ejecución de pruebas**

Para las pruebas funcionales de instancia del Framework, se han realizado las siguientes:

<b>Prueba Nº</b>	1
<b>Descripción</b>	Ejecutar el script de base de datos que provee el Framework.
<b>Objetivos y proceso</b>	Ejecutar correctamente el script de la base de datos que se encuentra colocado en la página web del Framework. Para ello es necesario: <ol style="list-style-type: none"> <li>1. Acceder a la página web del Framework.</li> <li>2. Seleccionar la opción "Descargas".</li> <li>3. Seleccionar la opción "Script de base de datos".</li> <li>4. Copiar y ejecutar el texto en una ventana dentro del gestor de base de datos.</li> </ol>
<b>Condiciones</b>	<ul style="list-style-type: none"> <li>▪ Configurar el Framework en un servidor.</li> <li>▪ Crear una base de datos.</li> </ul>
<b>Resultado esperado</b>	El script de base de datos no presente ningún error luego de su ejecución.
<b>Resultado obtenido</b>	El resultado fue satisfactorio.

*Tabla 27: Prueba unitaria - Ejecución de script de base de datos.*

*Fuente: El Autor.*

<b>Prueba Nº</b>	2
<b>Descripción</b>	Crear los recursos principales para que el Framework pueda instanciar el sistema de

	información orientado a la prestación de servicios.
<b>Objetivos y proceso</b>	<p>Realizar el llenado de los recursos principales para que el Framework pueda instanciar el sistema de información. Para ello es necesario:</p> <ol style="list-style-type: none"> <li>1. Acceder a la página web del Framework.</li> <li>2. Seleccionar la opción Argumentos de configuración.</li> <li>3. Ingresar los datos en el formulario, correspondientes a la aplicación.</li> <li>4. Ingresar los datos en el formulario, correspondientes a la base de datos.</li> <li>5. Presionar el botón de Generar Recursos.</li> <li>6. Ingresar al sistema creado con las credenciales: admin@miempresa.com y admin, siendo el usuario y clave respectivamente.</li> </ol>
<b>Condiciones</b>	<ul style="list-style-type: none"> <li>▪ Configurar el Framework en un servidor.</li> <li>▪ Ejecutar el script de base de datos.</li> </ul>
<b>Resultado esperado</b>	Creación de un sistema de información.
<b>Resultado obtenido</b>	El resultado fue satisfactorio.

Tabla 28: Prueba unitaria - Crear recursos para la instancia del Framework.

Fuente: El Autor.

<b>Prueba Nº</b>	3
<b>Descripción</b>	El Framework debe crear archivos orientados al patrón MVC. Tales archivos son generados desde las tablas creadas previamente en la base de datos.
<b>Objetivos y proceso</b>	Crear automáticamente archivos orientados al patrón MVC. Para ello es necesario:

	<ol style="list-style-type: none"> <li>1. Acceder a la página web del Framework.</li> <li>2. Seleccionar la opción Modelo - Vista - Controlador.</li> <li>3. Ingresar el nombre de la carpeta contenedora del sistema de información creado.</li> <li>4. Presionar el botón de Crear Archivos.</li> </ol>
<b>Condiciones</b>	<ul style="list-style-type: none"> <li>▪ Configurar el Framework en un servidor.</li> <li>▪ Instanciar el sistema de información.</li> </ul>
<b>Resultado esperado</b>	Creación de archivos en la estructura del sistema de información previamente creado.
<b>Resultado obtenido</b>	El resultado fue satisfactorio.

*Tabla 29: Prueba unitaria - Creación de archivos MVC.*

*Fuente: El Autor.*

<b>Prueba Nº</b>	4
<b>Descripción</b>	Ingresar al sistema instanciado por el Framework a través de las credenciales generadas.
<b>Objetivos y proceso</b>	<p>Probar la funcionalidad del login principal del sistema instanciado por el Framework. Para ello es necesario:</p> <ol style="list-style-type: none"> <li>1. Acceder a la URL del sistema de información instanciado y seleccionar la pestaña Login.</li> <li>2. Colocar el usuario y contraseña otorgado por el Framework, siendo admin@miempresa.com y admin respectivamente.</li> <li>3. Presionar el botón "Ingresar".</li> </ol>
<b>Condiciones</b>	<ul style="list-style-type: none"> <li>▪ Se debe instanciar un sistema a través</li> </ul>

	del Framework.
<b>Resultado esperado</b>	Ingresar a la pantalla principal del sistema, visualizando las opciones de menú y espacios de visualización del sistema otorgados por el Framework.
<b>Resultado obtenido</b>	El resultado fue satisfactorio.

*Tabla 30: Prueba unitaria - Ingresar al sistema instanciado.*

*Fuente: El Autor.*

<b>Prueba Nº</b>	5
<b>Descripción</b>	Se necesita crear un tenant o cliente que pueda tener todas las funcionalidades independientemente de otros clientes.
<b>Objetivos y proceso</b>	<p>Registrar un cliente. Para ello es necesario:</p> <ol style="list-style-type: none"> <li>1. Acceder a la URL del sistema de información instanciado y seleccionar la pestaña "Registrarse".</li> <li>2. Ingresar los campos solicitados.</li> <li>3. Presionar el botón "Registrar".</li> <li>4. Los datos de validación serán enviados al correo electrónico registrado.</li> <li>5. Abrir el correo enviado y hacer clic sobre el enlace de validación.</li> <li>6. Verificar que el sistema aprobó el registro.</li> </ol>
<b>Condiciones</b>	<ul style="list-style-type: none"> <li>▪ Se debe instanciar un sistema a través del Framework.</li> </ul>
<b>Resultado esperado</b>	Registrarse como cliente o tenant.
<b>Resultado obtenido</b>	El resultado fue satisfactorio.

*Tabla 31: Prueba unitaria - Registrar un tenant o cliente.*

*Fuente: El Autor.*

También se han considerado otro tipo de pruebas, las cuales serán capaces de determinar en qué escenario se puede utilizar el Framework; para ello, se han realizado las siguientes:

- Compatibilidad web del Framework.
- Compatibilidad del servidor donde estará alojado el Framework.

El Framework ha sido probado manualmente en el lado del cliente, es decir, en diferentes navegadores web, presentando una compatibilidad definida, como se puede visualizar en la siguiente tabla:

<b>Sistema operativo</b>	<b>Chrome v. 47</b>	<b>Firefox v. 43</b>	<b>Internet Explorer v. 11</b>	<b>Opera v. 34</b>	<b>Safari v. 9.1</b>
Ubuntu 15.04	Si	Si	-	Si	-
Mac OS X 10.8	Si	Si	-	Si	Si
Windows XP, 7, 8.	Si	Si	Si	Si	-

*Tabla 32: Compatibilidad web del Framework (forma manual).*

*Fuente: El Autor.*

Así mismo, se realizaron pruebas automatizadas con la aplicación crossbrowserTesting, como se puede visualizar en la siguiente figura:

Select Browsers (7)									
		Desktop (5)		Mobile (2)		Trial Browsers			
		Chrome		Firefox		IE	Opera	Safari	
Mac OSX 10.11	47x64	46x64	45x64	43	42	41	32	31	30
Mac OSX 10.10	47x64	46x64	45x64	43	42	41	32	31	30
Mac OSX 10.9	47x64	46x64	45x64	43	42	41	32	31	30
Mac OSX 10.8	47x64	46x64	45x64	42	41	40	32	31	30
Ubuntu 14.04				38	37	36			
Windows 10	47x64	47	46	43	42	41	Edge	11	32
Windows 8.1	47	47x64	46x64	43	42	41	11	32	31
Windows 8	47	47x64	46	43	42	41	10	32	31
Windows 7 64-Bit	47x64	47	46	43	42	41	11	10	9x64
Windows 7	47	46	45	43	42	41	9	8	32
Windows Vista	47	46	45	43	42	41	9	8	7
Windows XP SP3	47	46	45	43	42	41	8	32	31
Windows XP SP2	47	46	45	42	41	40	7	6	32

Figura 34: Compatibilidad web del Framework (forma automatizada).

Fuente: <https://www.crossbrowsertesting.com/>.

Teniendo como resultado la compatibilidad con diferentes sistemas operativos, navegadores web y resoluciones. A continuación, los resultados:

The screenshot shows a web browser window with the following details:

- Header:** Windows 8, 52x64, 1024x768.
- Title Bar:** Mi Framework - Sistemas
- Address Bar:** 192.168.0.101/lizarFrame/web/\_principal/\_index.php
- Content Area:**
  - LizarFrame Logo:** A blue and white circular logo with the text "LIZAR FRAME".
  - Welcome Message:** Bienvenido a LizarFrame
  - Navigation Menu:** Inicio, Argumentos de configuración, Modelo - Vista - Controlador, Descargas.
  - Text:** Usted visita nuestra web el: 2016-08-07
  - Text (Right Side):** Desarrollador para obtener mayor conocimiento del uso de "LizarFrame". [Leer +](#)
  - Text (Bottom Right):** Mi Framework v 1.0, es un framework libre escrito en PHP5, basado en las mejores prácticas de desarrollo web. LizarFrame favorece la velocidad y eficiencia en la creación y mantenimiento de Sistemas de Información Híbridos, reemplazando tareas de codificación repetitivas por poder, control y placer. [Descargar](#)
  - Footer:** Inicio | Argumentos de configuración | Modelo - Vista - Controlador | Descargas
  - Page Info:** LizarFrame v 2.0 | Todos los derechos reservados © 2015 | Resolución recomendada 1024 \* 768 píxeles.
  - Page Info:** Responsable en diseño: Edgar Frank Lizárraga Ugarte

Figura 35: Resultado Nro. 1 de Compatibilidad web del Framework (forma automatizada).

Fuente: <https://www.crossbrowsertesting.com/>.

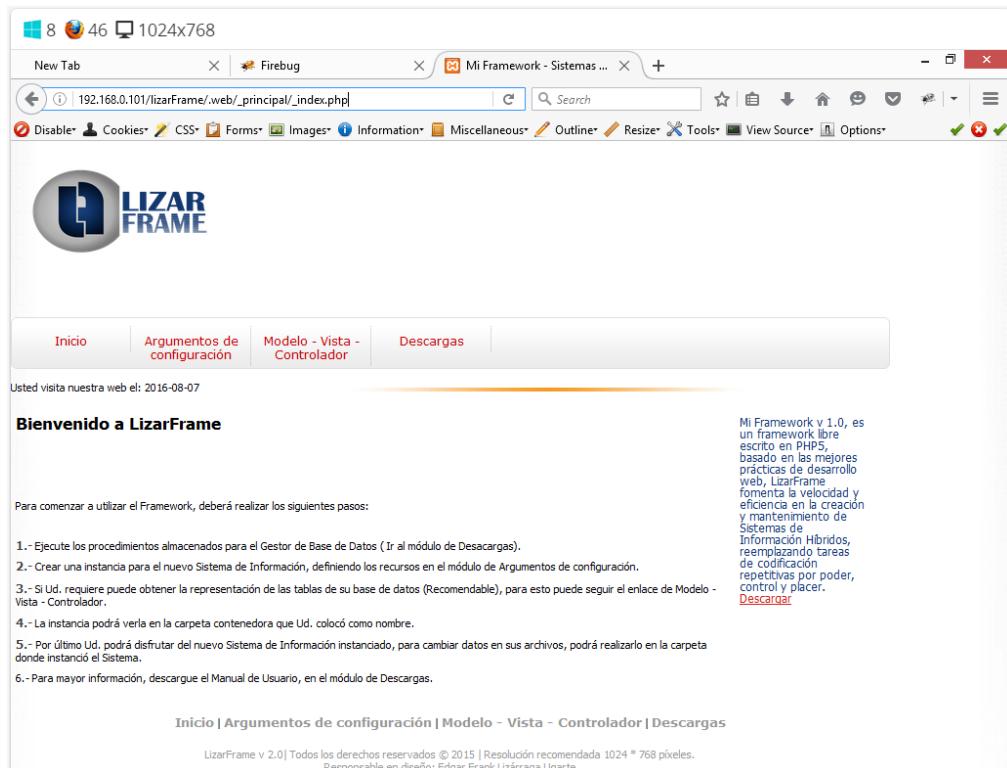


Figura 36: Resultado Nro. 2 de Compatibilidad web del Framework (forma automatizada).

Fuente: <https://www.crossbrowsertesting.com/>.

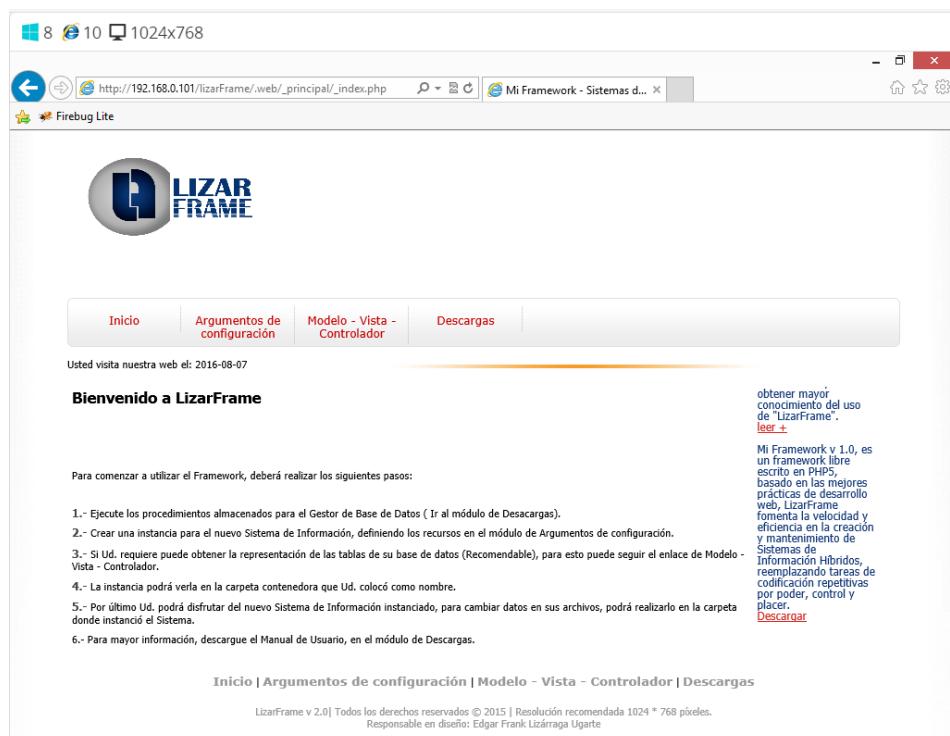


Figura 37: Resultado Nro. 3 de Compatibilidad web del Framework (forma automatizada).

Fuente: <https://www.crossbrowsertesting.com/>.

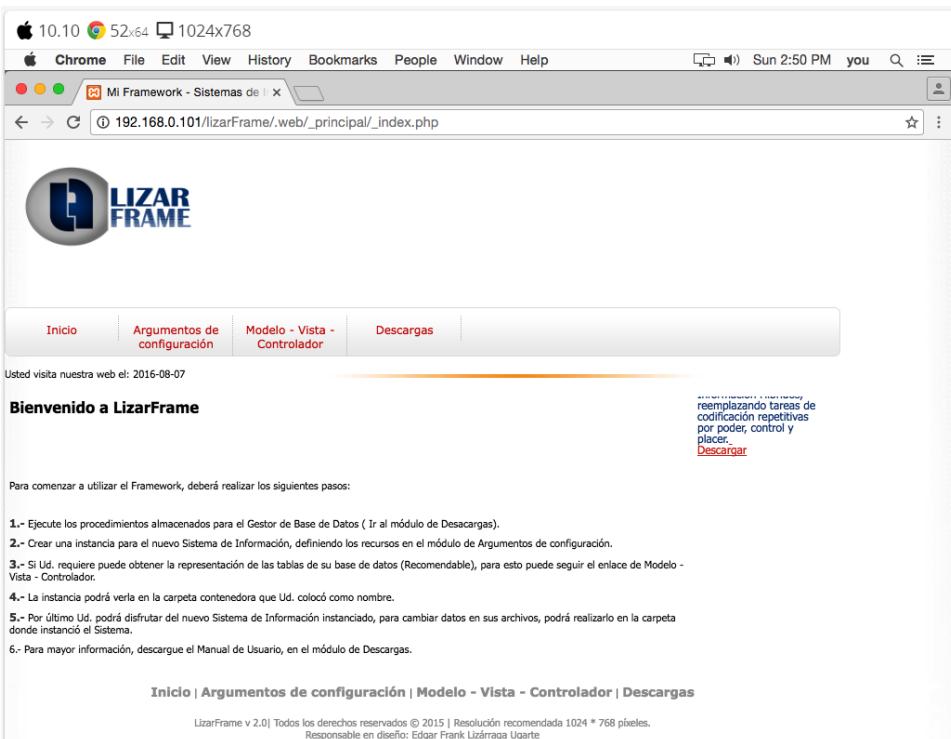


Figura 38: Resultado Nro. 4 de Compatibilidad web del Framework (forma automatizada).

Fuente: <https://www.crossbrowsertesting.com/>.

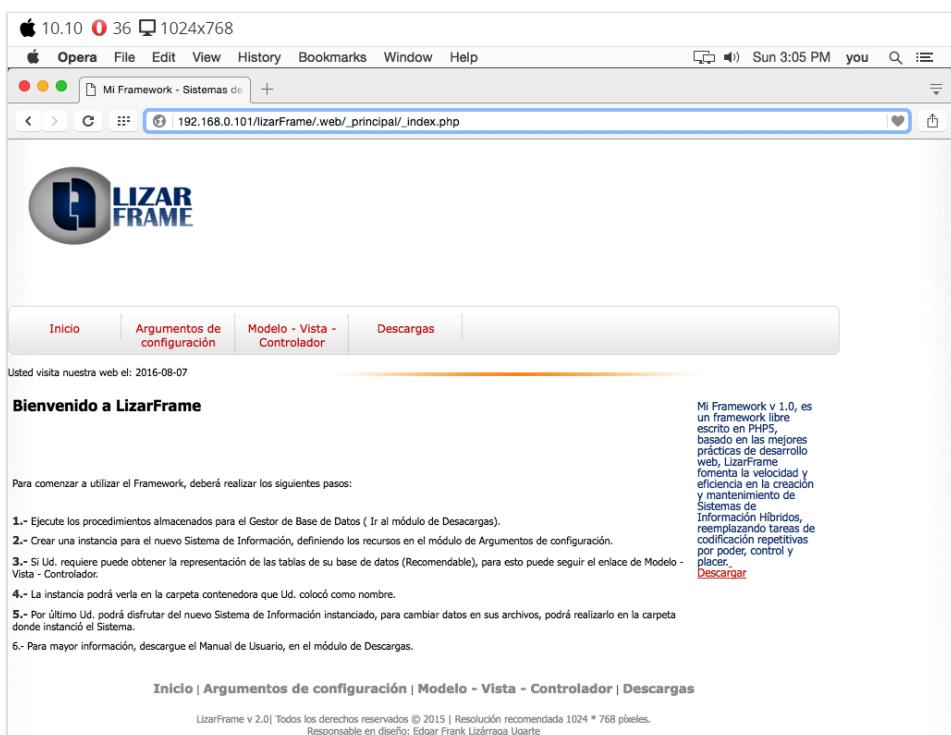


Figura 39: Resultado Nro. 5 de Compatibilidad web del Framework (forma automatizada).

Fuente: <https://www.crossbrowsertesting.com/>.

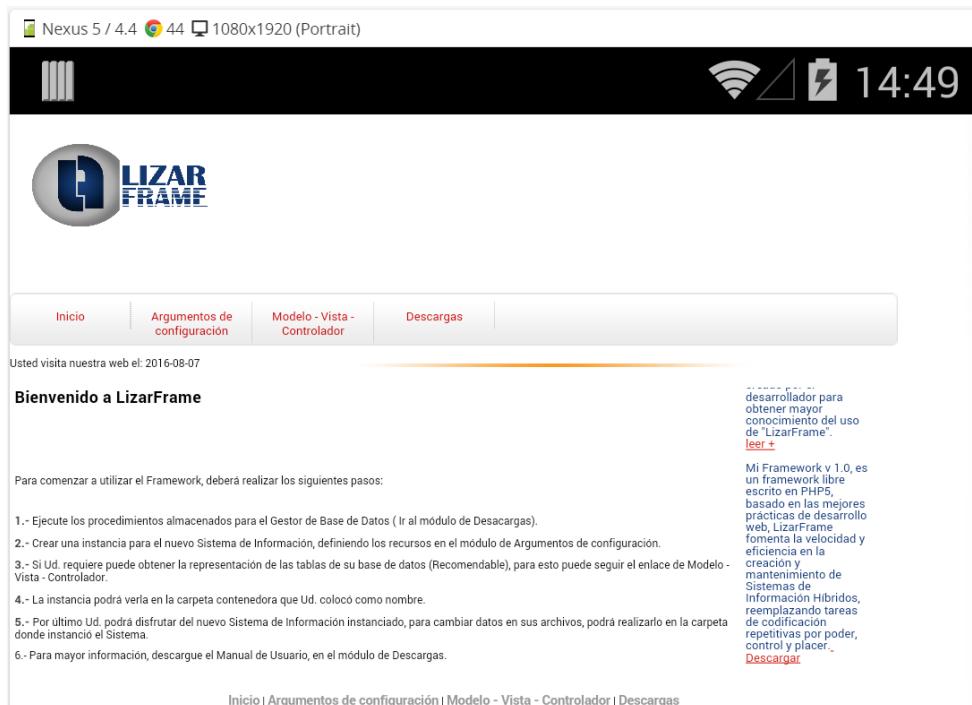


Figura 40: Resultado Nro. 6 de Compatibilidad web del Framework (forma automatizada).

Fuente: <https://www.crossbrowsertesting.com/>.

En referencia a las pruebas del lado del servidor, se ha hecho lo siguiente:

- Instalación y configuración del servidor Apache 2.2.4 y PHP 5.2 en un servidor con sistema operativo Windows 7 de 32 bits, obteniendo resultados satisfactorios.
- Instalación y configuración del servidor Apache 2.4 y PHP 5.5 en un servidor con sistema operativo Windows 7 de 32 bits, obteniendo resultados NO satisfactorios.

Por último, se ha realizado la prueba de concurrencia y carga en los módulos de Login y registro de tenant o clientes; dicha prueba se ha realizado de manera automática, a través de la herramienta Apache JMeter en su versión 2.13 (*Apache JMeter, 2015*). Antes de comenzar con las pruebas de rendimiento pasemos a ver algunos conceptos del Grupo de Hilos:

- Número de hilos: simulará el número de usuarios accediendo al escenario.

- Periodo de subida: tiempo en que tardan las peticiones de los hilos.
- Contador del bucle: será el número de iteraciones a realizar.
- Periodo de carga: indica cada cuánto tiempo un hilo realiza la petición.
- Media: Tiempo promedio en resolver una petición.
- Contador del bucle: Número de veces a ejecutar el plan de pruebas.

El periodo de subida se calcula con la siguiente fórmula:

$$\text{Periodo de subida} = \text{periodo de carga} * \text{número de hilos}$$

Un ejemplo de esto sería el siguiente: si queremos que un nuevo usuario virtual (hilo) ejecute el escenario cada 0,2 segundos, hasta llegar a los 500 usuarios:

$$\text{Periodo de subida} = 0,2 * 500$$

$$\text{Periodo de subida} = 100 \text{ segundos}$$

Ahora, para realizar la prueba, se debe configurar un servidor, por ende, se tienen las siguientes características del servidor de pruebas:

➤ **Plataforma:**

Apache/2.2.4 (Win 32) PHP/5.2

Base de datos: SQL Server 2008 R2

Procesador: Intel Core 2 Duo P8600 @2.4GHz

Memoria RAM: 6.00 GB

Sistema operativo: Windows 7 Ultimate de 64 bits.

A continuación, los módulos a probar:

**a. Módulo de Login**

- En primer lugar, se debe preparar el plan de pruebas, la Figura 41 muestra dicho plan.

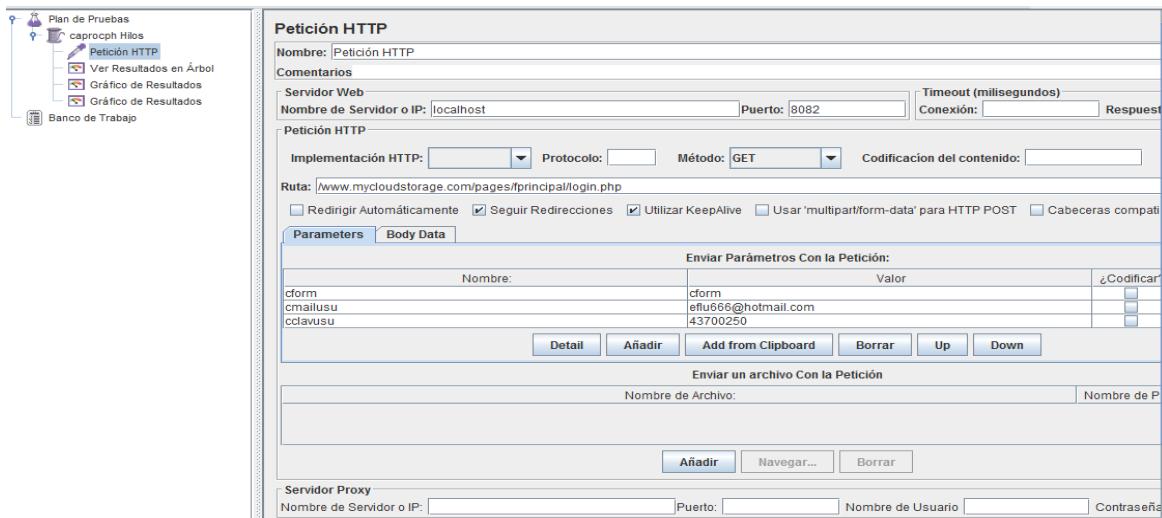


Figura 41: Plan de pruebas en JMeter para módulo de Login.

Fuente: El Autor.

- La Tabla 33 muestra las pruebas realizadas.

Nº iteración	Nº de Hilos	Periodo de subida (s)	Periodo de Carga (s)	Contador del bucle	Media (ms)	Resultado
1	10	10	1	1	101	Positivo
2	100	100	1	1	105	Positivo
3	200	200	1	1	108	Positivo
4	300	300	1	1	113	Positivo
5	400	400	1	1	113	Positivo
6	500	500	1	1	113	Positivo
7	600	600	1	1	124	Positivo
8	700	350	0.5	1	36	Positivo
9	1000	1000	1	1	35	Positivo
10	1000	200	0.2	1	41	Positivo
11	1000	100	0.1	1	34	Positivo
12	1000	20	0.02	1	3162	Positivo
13	1000	10	0.01	1	6166	Alerta
14	1000	5	0.005	1	6275	Alerta
15	1000	1	0.001	1	8722	Alerta

Tabla 33: Pruebas de concurrencia y carga a módulo de Login.

Fuente: El Autor.

- En las pruebas realizadas al módulo de Login se puede observar que los tiempos de respuesta del servidor son buenos. Se resuelven al 100% las solicitudes, por lo que no se producen errores. La carga al comienzo se realiza segundo a segundo en donde se visualizan resultados positivos, hasta que, el periodo de carga comienza a disminuir en tiempo, por consiguiente, los resultados comienzan a alertar por la gran carga de peticiones "http" en un periodo corto de tiempo. En términos generales, las peticiones "http" al módulo de Login del Framework fueron satisfactorias por la gran carga, alto periodo de subida y elevada cantidad de usuarios virtuales. Si se requiere tener mayor capacidad de usuarios realizando peticiones, se recomienda realizar las pruebas en una plataforma con mayores prestaciones.

### b. Módulo de registro de tenant o cliente

- En primer lugar, se debe preparar el plan de pruebas, la Figura 42 muestra dicho plan.

The screenshot shows the JMeter 'Plan de Pruebas' interface. On the left, there's a tree view with 'Plan de Pruebas' at the root, followed by 'Plan de pruebas - Registro tenant' which contains a single item named 'Petición HTTP'. Below the tree is a 'Banco de Trabajo' section. The main panel displays the 'Petición HTTP' configuration:

**Petición HTTP**

Nombre: Petición HTTP  
 Comentarios:  
 Servidor Web: Nombre de Servidor o IP: localhost Puerto: 8082 Timeout (milisegundos):  
 Conexión: Respuesta:  
**Petición HTTP**  
 Implementación HTTP: Protocolo: http Método: POST Codificación del contenido:  
 Ruta: /www.mycloudstorage.com/pages/fprincipal/registrar.usuario.php  
 Redirigir Automáticamente  Seguir Redirecciones  Utilizar KeepAlive  Usar 'multipart/form-data' para HTTP POST  Cabeceras compatibles con navegadores  
**Parameters** **Body Data**  
 Envíar Parámetros Con la Petición:  

Nombre:	Valor	Codificar?	Incluir Equals?
registro	registrar.usuario.php	<input type="checkbox"/>	<input checked="" type="checkbox"/>
capepatusu	Ltzarraga	<input type="checkbox"/>	<input checked="" type="checkbox"/>
capematusu	Ugarte	<input type="checkbox"/>	<input checked="" type="checkbox"/>
cromusu	Edgar Frank	<input type="checkbox"/>	<input checked="" type="checkbox"/>
cromemp	Empresa prueba	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ifcodrub	1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
cmailusu	eflizarraga@hotmail.com	<input type="checkbox"/>	<input checked="" type="checkbox"/>
cclavusu	1234567890	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Buttons at the bottom: Detail, Añadir, Add from Clipboard, Borrar, Up, Down.

Figura 42: Plan de pruebas en JMeter para módulo de registro de tenant o cliente.

Fuente: El Autor.

- Luego, se debe eliminar validaciones sobre valores repetidos, para poder insertar a "n" clientes con el mismo valor.

- La Tabla 34 muestra las pruebas realizadas.

Nº iteración	Nº de Hilos	Periodo de subida (s)	Periodo de Carga (s)	Contador del bucle	Media (ms)	Resultado
1	10	10	1	1	1265	Positivo
2	100	100	1	1	1314	Positivo
3	200	200	1	1	1305	Positivo
4	300	300	1	1	1336	Positivo
5	400	400	1	1	1320	Positivo
6	500	500	1	1	1337	Positivo
7	600	600	1	1	1330	Positivo
8	700	350	0.5	1	1436	Positivo
9	1000	1000	1	1	1330	Positivo
10	1000	200	0.2	1	1518	Positivo
11	1000	100	0.1	1	18266	Positivo
12	1000	20	0.02	1	37041	Alerta
13	1000	10	0.01	1	37665	Alerta
14	1000	5	0.005	1	39614	Alerta
15	1000	1	0.001	1	41314	Alerta

Tabla 34: Pruebas de concurrencia y carga a módulo de registro de tenant o cliente.

Fuente: El Autor.

- En las pruebas realizadas al módulo de registro de tenant o cliente se puede observar que los tiempos de respuesta del servidor son buenos. Se resuelven al 100% las solicitudes, por lo que no se producen errores. La carga al comienzo se realiza segundo a segundo en donde se visualizan resultados positivos, hasta que, el periodo de carga comienza a disminuir en tiempo, por consiguiente, los resultados comienzan a alertar por la gran carga de peticiones "http" en un periodo corto de tiempo. En términos generales, las peticiones "http" al módulo de Registro de clientes del Framework fueron satisfactorias por la gran carga, alto periodo de subida y elevada cantidad de usuarios virtuales. Si se requiere tener mayor capacidad de usuarios realizando peticiones, se recomienda realizar las pruebas en una plataforma con mayores prestaciones.

- La concurrencia también ha sido positiva, en cuanto a la capacidad de tener muchos clientes registrados en una sola instancia de base de datos; visualizando la diferencia en los datos de clientes a través del campo multitenant colocado en las diferentes tablas diseñadas de la base de datos.

ipkcodemp										Messages			
ipkcodemp	cnomemp	ifkcodrub	ipkcodusu	ifkcodperf	inrousu	cnomusu	capepatusu	capematusu	cclavusu	cmailusu	ifkcodalm	ifkcodest	ifkcodemp
997	997	Empresa prueba	1										
998	998	Empresa prueba	1										
999	999	Empresa prueba	1										
1000	1000	Empresa prueba	1										

ipkcodusu	ifkcodperf	inrousu	cnomusu	capepatusu	capematusu	cclavusu	cmailusu	ifkcodalm	ifkcodest	ifkcodemp	
996	996	1	1	Edgar Frank	Lizarraga	Ugarte	1234567890	eflizarraga@hotmail.com	996	2	996
997	997	1	1	Edgar Frank	Lizarraga	Ugarte	1234567890	eflizarraga@hotmail.com	997	2	997
998	998	1	1	Edgar Frank	Lizarraga	Ugarte	1234567890	eflizarraga@hotmail.com	998	2	998
999	999	1	1	Edgar Frank	Lizarraga	Ugarte	1234567890	eflizarraga@hotmail.com	999	2	999
1000	1000	1	1	Edgar Frank	Lizarraga	Ugarte	1234567890	eflizarraga@hotmail.com	1000	2	1000

ipkcodciu	inrociu	cnonciu	ifkcodemp
997	997	1	Demo
998	998	1	Demo
999	999	1	Demo
1000	1000	1	Demo

Figura 43: Tablas con campo de tenant o cliente.

Fuente: El Autor.

### g. Resultado de pruebas

- Se ha realizado satisfactoriamente las pruebas de caja negra, compatibilidad, concurrencia y carga de datos; comprobando el correcto funcionamiento de los requerimientos funcionales y no funcionales del Framework.
- De acuerdo a las diferentes pruebas realizadas, se tiene como resultado la Tabla 35, haciendo referencia a los requerimientos mínimos y recomendados para instalar, configurar e instanciar el Framework:

Componente	Descripción
Sistema operativo	Windows 7
Navegador Web	Internet Explorer, Google Chrome, Mozilla

	Firefox (recomendados).
Configuración del servidor web	Apache 2.2.4 como servicio web y PHP 5.2 como lenguaje de programación.
Procesador	Intel Core 2 Duo 2GHz (recomendado)
Memoria RAM	4 GB
Tarjeta de video	128 MB (mínimo)
Periféricos	Mouse y teclado
Comunicación	Tarjeta de red Ethernet, considerando un ancho de banda de 2Mbps.
Monitor	Monitor VGA a color con resolución mínima de 800 * 600 px. Calidad de color a 16 bits. Resolución de 1024 * 768 px (recomendado).

*Tabla 35: Características mínimas y recomendadas del equipo para utilizar el Framework.*

*Fuente: El Autor*

### **3.4. INSTANCIA DEL FRAMEWORK**

Para mayor detalle sobre la instancia del Framework, ver el anexo A, que corresponde al Manual Técnico del Framework.

# **Capítulo 4**

## **Desarrollo del Sistema de Información**

---

---

### **4.1. INTRODUCCIÓN**

El presente capítulo, desarrolla el subsistema de Control de Inventarios dentro del sistema de información logístico orientado a la prestación de servicios en MYPES que poseen la unidad de negocio sobre logística. Para tal desarrollo, se realiza el modelado del negocio, conociendo así su proceso actual, luego, se continúa con el desarrollo del subsistema, tomando en consideración las fases de la metodología RUP, en donde, el Framework será utilizado en la fase de la construcción del software.

Así mismo, es importante resaltar que, al usar el término “Sistema”, debe entenderse que se está haciendo referencia al subsistema de control de inventarios, siendo ese el criterio que se usará en el presente capítulo.

## 4.2. DESARROLLO DEL SISTEMA DE INFORMACIÓN

### 4.2.1. Modelado del Negocio

Debido a que el software está orientado a la prestación de servicios en MYPEs, es necesario considerar a micro y pequeñas empresas que cumplan con algún proceso de negocio, pues, para el interés del presente proyecto de tesis, se optó por elegir un proceso logístico que entra a tallar como un módulo en el flujo de trabajo principal de una MYPE que contempla el control de inventarios.

Por consiguiente, las MYPE a tratar son: UVICAR S.A, y Control Total (desde ahora EMV - Empresas de Monitoreo Vehicular), encargadas del control y monitoreo de unidades de transporte terrestre a nivel nacional. A continuación, el proceso logístico actual de las MYPE mencionadas:

- **Gestión de productos:** En el caso de una EMV, se da el manejo de equipos GPS, CHIP e insumos relacionados a tales equipos como: relay, porta relay, botones de pánico, entre otros. Solamente los productos como GPS y CHIP cuentan con una identificación única, por lo que se requiere tener un seguimiento y control a tales equipos.
- **Gestión de proveedores:** Manejo de los proveedores de equipos y/o productos.
- **Gestión de almacenes:** Las MYPE cuentan con almacenes a nivel nacional, teniendo la posibilidad de contar con dos o más almacenes en una misma ciudad.
- **Compra de productos:** Los productos pueden ser adquiridos a través de importaciones o compras locales.
- **Gestión de almacenes:** Una vez adquiridos los productos, estos rotan a través de los almacenes, pudiendo realizar ingresos, salidas y transferencias.

B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
<b>LIMA</b>					<b>AREQUIPA</b>					<b>TRUJILLO</b>					
STOCK DE ARTICULOS Y ACCESORIOS					STOCK DE ARTICULOS Y ACCESORIOS					STOCK DE ARTICULOS Y ACCESORIOS					
ITEM	ARTICULO	CANTIDAD	ESTADO	ITEM	ARTICULO	CANTIDAD	ESTADO	ITEM	ARTICULO	CANTIDAD	ESTADO	ITEM	ARTICULO	CANTIDAD	ESTADO
1	HANDS FREE	64	ALMACEN	1	HANDS FREE			1	HANDS FREE		ALMACEN	2	ARNES SIMPLE		3 ALMACEN
2	ARNES SIMPLE	156	ALMACEN	2	ARNES SIMPLE	34	ALMACEN	3	ARNES INTERMEDIO		ALMACEN	3	ARNES INTERMEDIO		ALMACEN
3	ARNES INTERMEDIO	85	ALMACEN	4	PORTARELAY	31	ALMACEN	4	PORTARELAY		3 ALMACEN	5	RELAY	37	ALMACEN
4	PORTARELAY	262	ALMACEN	5	RELAY	37	ALMACEN	5	RELAY		3 ALMACEN	6	FUSIBLE		3 ALMACEN
6	FUSIBLE	217	ALMACEN	6	PORTAFUSIBLE	23	ALMACEN	6	PORTAFUSIBLE		3 ALMACEN	7	PORTAFUSIBLE		3 ALMACEN
7	PORTAFUSIBLE	219	ALMACEN	8	BOTON DE PANICO	60	ALMACEN	8	BOTON DE PANICO		3 ALMACEN	9	SENSOR DE ACCIDENTES		ALMACEN
8	BOTON DE PANICO	156	ALMACEN	9	SENSOR DE ACCIDENTES		ALMACEN	10	DALLAS KIT		ALMACEN	10	DALLAS KIT		ALMACEN
9	SENSOR DE ACCIDENTES	3	ALMACEN	11	SENSOR DE PUERTA		ALMACEN	11	SENSOR DE PUERTA		ALMACEN	11	SENSOR DE PUERTA		ALMACEN
10	DALLAS KIT	1	ALMACEN	12	SENSOR DE COMBUSTIBLE		ALMACEN	12	SENSOR DE COMBUSTIBLE		ALMACEN	12	SENSOR DE COMBUSTIBLE		ALMACEN
11	SENSOR DE PUERTA	3	ALMACEN	13	ANTENA DE GPS	48	ALMACEN	13	ANTENA DE GPS		1 ALMACEN	13	ANTENA DE GPS		1 ALMACEN
12	SENSOR DE COMBUSTIBLE	1	QUEMADO	14	GPS	18	ALMACEN	14	GPS		4 ALMACEN	14	GPS		4 ALMACEN

Figura 44: EMV - Lista de artículos por almacén.

Fuente: El Autor.

C	D	E	F	G	H	I	J
Nº GPS	ALMACÉN	ESTADO		Nº CHIP	Nº CELULAR	ALMACÉN	ESTADO
373097	AREQUIPA	INSTALADO		8951061121011514185	978466388	Lima - San Borja	INSTALADO
319584	AREQUIPA	INSTALADO		8951061121011514474	978466544	Lima - San Borja	INSTALADO
319706	AREQUIPA	INSTALADO		8951061121010022701	957416629	Arequipa - Malogrados	MALOGRADO
319882	AREQUIPA	INSTALADO		8951061121011404684	957414652	Lima - San Borja	INSTALADO
319900	AREQUIPA	INSTALADO		8951061121010022784	957416634	Arequipa	INSTALADO
319886	AREQUIPA	MALOGRADO		8951061121010022776	957416631	Lima - San Borja	INSTALADO
319892	AREQUIPA	QUEMADO		8951061121010022693	957416635	Lima - San Borja	INSTALADO
354146	AREQUIPA	INSTALADO		8951061121010196901	956606193	Arequipa	INSTALADO
401559	AREQUIPA	INSTALADO		8951061121010196950	956602544	Lima - San Borja	INSTALADO
452204	AREQUIPA	INSTALADO		8951061121010196984	953978159	Lima - San Borja	INSTALADO
373028	AREQUIPA	INSTALADO		8951061121010196992	953977342	Lima - San Borja	INSTALADO
354216	AREQUIPA	INSTALADO		8951063121100564420	957891283	Arequipa - Malogrados	MALOGRADO
452202	AREQUIPA	INSTALADO		8951061121009479672	974997065	Lima - San Borja	INSTALADO
354185	CUZCO	INSTALADO		8951064121103344837	952806233	Lima - San Borja	INSTALADO
382609	CUZCO	INSTALADO		8951064121103344803	952809665	Trujillo	INSTALADO
452205	CUZCO	INSTALADO		8951063031112087338	979075698	Lima - San Borja	INSTALADO

Figura 45: EMV - Lista de GPS y CHIP por almacén.

Fuente: El Autor.

#### **4.2.2. Identificación del Problema**

Las EMV, no cuentan con un control adecuado ante sus productos, por tal razón, el jefe de logística no controla completamente la gestión de los productos, teniendo así, un gran retraso y descontrol en su unidad de negocio, así mismo el operador de almacén demora en gestionar las transacciones diarias. La unidad de logística no cuenta con un control exhaustivo sobre dichos productos, obteniendo un desorden y desabastecimiento no controlado, además que, muchas veces no se conoce a ciencia cierta la localidad y motivo de salida o ingreso de equipos y/o materiales en los almacenes. Tal falta de control se puede visualizar en las figuras 44 y 45, debido a que se encuentran tablas en Excel para el registro de los productos en diferentes almacenes.

#### **4.2.3. Requerimientos**

- Se debe controlar las acciones de los usuarios que utilizan la aplicación.
- Registrar, listar y editar a los productos que posee la empresa, tales como: equipos GPS, CHIP, relay, porta relay, fusibles. Cada GPS y CHIP es único, por lo que debe tratarse de manera diferente que los demás productos; por lo tanto, el GPS y CHIP tiene un Nº único de identificación.
- Registrar, listar y editar a los proveedores.
- Registrar, listar y editar a los almacenes por sede.
- El sistema debe permitir el registro y listado de compras por importación y por compras de forma local; teniendo la capacidad de comprar varios equipos GPS, CHIP y cualquier otro producto sobre el que se requiera tener control. Al momento de registrar una compra, se debe considerar lo siguiente:

- 1. Para compras locales y de importación:** Se debe tener el registro de la fecha de compra, fecha de factura del proveedor, Nº de factura del proveedor y localidad del almacén. Así mismo, se debe ingresar la cantidad de productos en general que se compran. En el caso de la EMV, para la compra de GPS, se debe colocar el Nº de serie de un equipo o el rango de números de serie de varios equipos GPS, ya que las

importaciones se realizan por cantidades y estos equipos GPS cuentan con números de serie secuenciales; como, por ejemplo:

Nombre (modelo)	Descripción	Nº GPS
GPS CELLO F	GPS CELLO F	475433
GPS CELLO F	GPS CELLO F	475434
GPS CELLO F	GPS CELLO F	475435
GPS CELLO F	GPS CELLO F	475436
GPS CELLO F	GPS CELLO F	475437
GPS CELLO F	GPS CELLO F	475438
GPS CELLO F	GPS CELLO F	475439
GPS CELLO F	GPS CELLO F	475440
GPS CELLO F	GPS CELLO F	475441
GPS CELLO F	GPS CELLO F	475442
GPS CELLO F	GPS CELLO F	475443
GPS COMPACT FLEET	GPS COMPACT FLEET	631621
GPS COMPACT FLEET	GPS COMPACT FLEET	631622
GPS COMPACT FLEET	GPS COMPACT FLEET	631623
GPS COMPACT FLEET	GPS COMPACT FLEET	631624
GPS COMPACT FLEET	GPS COMPACT FLEET	631625
GPS COMPACT FLEET	GPS COMPACT FLEET	631626
GPS COMPACT FLEET	GPS COMPACT FLEET	631627
GPS COMPACT FLEET	GPS COMPACT FLEET	631628
GPS COMPACT FLEET	GPS COMPACT FLEET	631629
GPS COMPACT FLEET	GPS COMPACT FLEET	631630
GPS COMPACT FLEET	GPS COMPACT FLEET	631631
GPS COMPACT FLEET	GPS COMPACT FLEET	631632
GPS COMPACT FLEET	GPS COMPACT FLEET	631633
GPS COMPACT FLEET	GPS COMPACT FLEET	631634
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046146
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046147
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046148
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046149
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046150
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046151
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046152
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046153
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046154
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046155
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046156
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046157
LMU623	LMU623, UBLOX GSM, INT ANT, GEN-HW	CAL1331046158

Figura 46: EMV - Lista de equipos GPS.

Fuente: El Autor.

Respecto a la compra de CHIP, estos no tienen un número secuencial, pero, se hace la compra por lotes, lo que significa que debe existir un proceso que pueda importar la gran cantidad de chip con que se cuenta, y no ingresarlos uno por uno al sistema, por ejemplo:

Nombre	Descripción	Nro. CHIP	Nro. Celular
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121103901563	957969352
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121103901571	957969357
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121105768556	958442587
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121201755184	957613534
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121202098733	957613439
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951061121202098741	971239838
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121202098758	944443497
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121202098766	974997065
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951061121202098774	958440675
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121202098782	957416635
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121202098790	957416629
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121202098808	957416631
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121202098824	957416634
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSPAGO DATOS	8951061121202098832	982987617
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951063121202444091	979075976
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951063121202444117	979076036
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951063121202444125	979076111
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951063121202444133	979076195
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951063121202444166	979076245
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951063121202444315	979076281
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951064031112526292	979076056
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951064031112526625	979076312
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951064031346053592	979075881
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951064031346068509	979076313
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951064031347347415	979076602
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951064031347347423	979076886
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951064031347347431	979076936
CHIP PREPAGO MOVISTAR - SMS	CHIP PREPAGO MOVISTAR DE MENSAJES DE TEXTO	8951064031349346027	979076817

Figura 47: EMV - Lista de CHIP.

Fuente: El Autor.

2. Además, para las compras que se realizaron con alguna importación se debe considerar: fecha de DUA y N° de DUA (Documento Único Administrativo correspondiente a la importación).
  - El sistema debe considerar las transacciones en la unidad de logística, tales como:
    1. Órdenes de salida de almacén.
    2. Aprobación de órdenes de salida de almacén.

- 3.** Ingreso de productos.
  - 4.** Salida de productos.
  - 5.** Transferencia entre almacenes (salida e ingreso por transferencia).
  - 6.** Reingreso de un producto.
- El sistema debe generar documentos a raíz de las anteriores transacciones, tales como:
  - 1.** Nota de salida de almacén.
  - 2.** Nota de ingreso de almacén.
- Los motivos de ingreso a almacén son: por reingreso, por transferencia y por compra.
- Los motivos de salida de un almacén son: por operación interna y por transferencia.
- Para el ingreso por reingreso y salida por operación interna debe considerarse los siguientes motivos:
  - 1.** Quemado, devolución, prueba de transmisión, sin transmisión, prueba de botón de pánico, posición errada, malogrado, fallado, bloqueado, préstamo, configuración, configurado, y separado.
- El sistema debe generar reportes principales de inventario, salidas e ingresos de almacén, movimiento de artículos, stock.

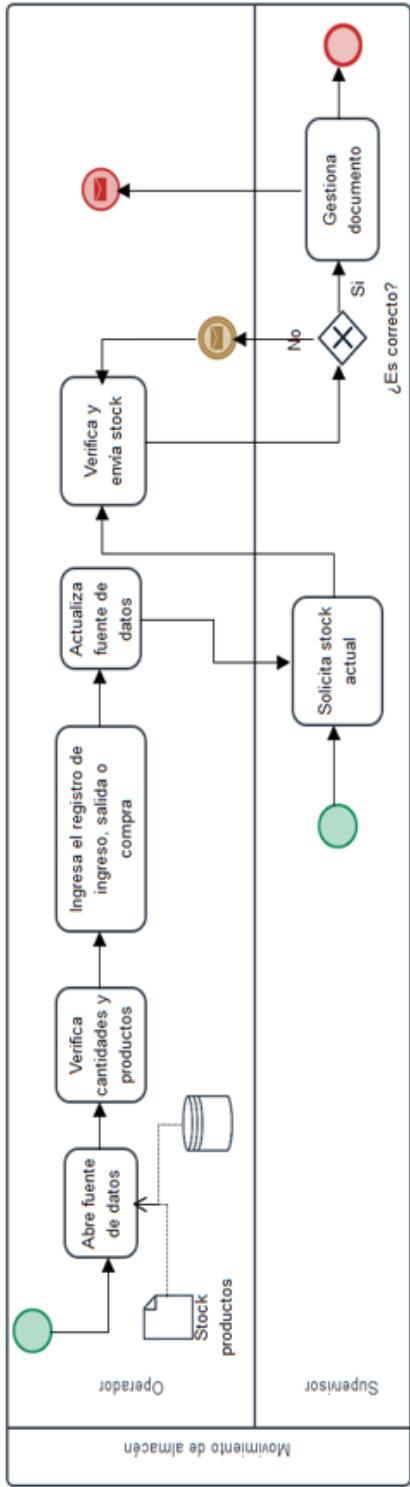


Figura 48: Diagrama BPM - Proceso actual del movimiento de almacén.

Fuente: El Autor.

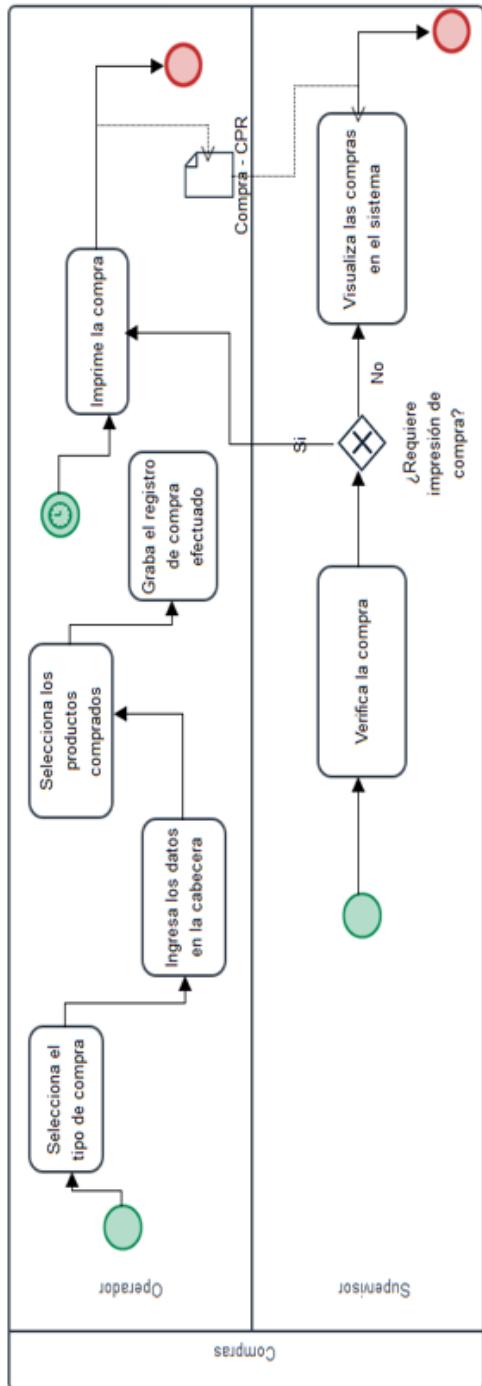


Figura 49: Diagrama BPM - Proceso propuesto para compras.

Fuente: El Autor.

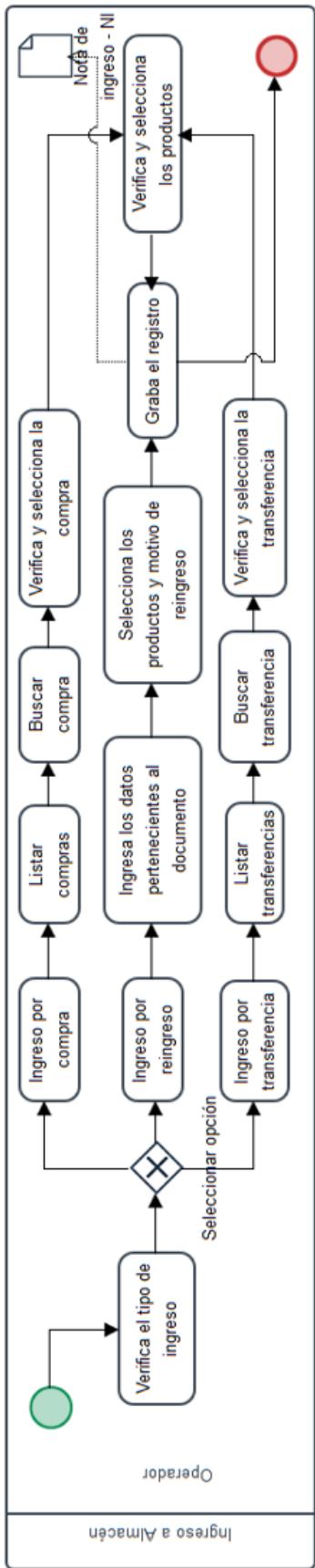


Figura 50: Diagrama BPM - Proceso propuesto para el ingreso a almacén.

Fuente: El Autor.

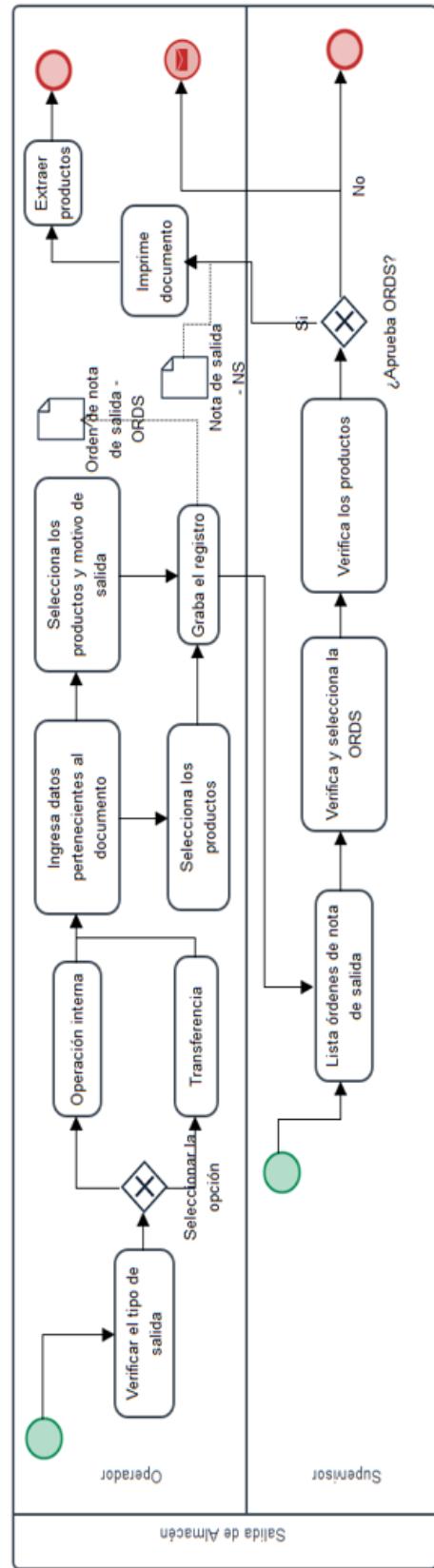


Figura 51: Diagrama BPM - Proceso propuesto para la salida de almacén.

Fuente: El Autor.

#### 4.2.4. Casos de Uso

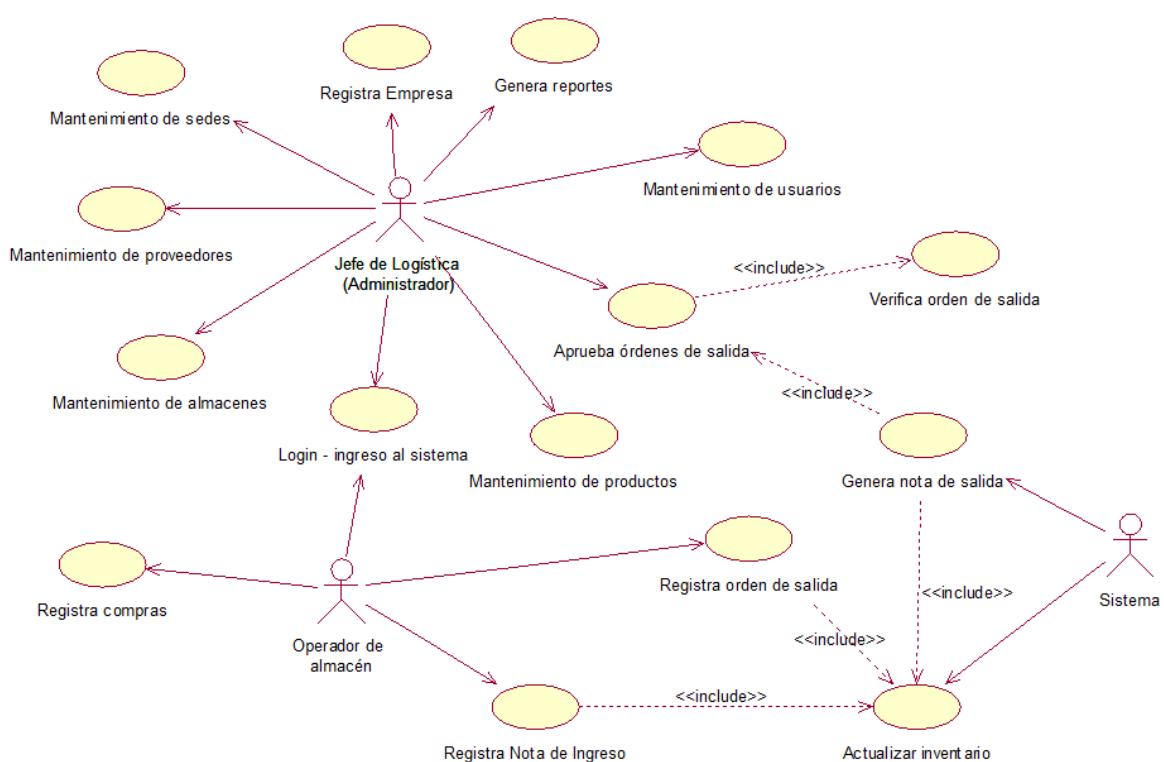


Figura 52: Diagrama de Casos de Uso del Sistema.

Fuente: El Autor.

#### 4.2.5. Especificación de Casos de Uso

<b>Nombre:</b>	Registra empresa.
<b>Descripción:</b>	El usuario a utilizar el sistema debe registrar su empresa para otorgarle un usuario y contraseña.
<b>Actores:</b>	Jefe de logística (Administrador).
<b>Precondiciones:</b>	<ol style="list-style-type: none"> <li>1. Ingresar a la dirección web del sistema de información.</li> <li>2. Ir a la opción de Registrarse.</li> </ol>
<b>Eventos:</b>	<ol style="list-style-type: none"> <li>1. Llenar la ficha de inscripción.</li> <li>2. Registrar sus datos.</li> <li>3. Verificar su correo electrónico.</li> </ol>
<b>Postcondiciones:</b>	Ingresar al sistema a través del Login.

*Tabla 36: Caso de Uso - Registrar empresa.*

*Fuente: El Autor.*

<b>Nombre:</b>	Login - ingreso al Sistema.
<b>Descripción:</b>	El usuario a utilizar el sistema tiene que introducir su usuario y contraseña en la pantalla de logueo.
<b>Actores:</b>	Jefe de logística y operador de almacén.
<b>Precondiciones:</b>	1. Ingresar al sistema mediante una dirección URL.
<b>Eventos:</b>	<ol style="list-style-type: none"> <li>1. Ingresar usuario y contraseña.</li> <li>2. El sistema determinará la autenticidad del usuario.</li> </ol>
<b>Postcondiciones:</b>	Si el usuario es correcto, el sistema se cargará.

*Tabla 37: Caso de Uso - Login - Ingreso al Sistema.*

*Fuente: El Autor.*

<b>Nombre:</b>	Mantenimiento de usuarios.
<b>Descripción:</b>	El administrador del sistema es el encargado de agregar, modificar y listar a los usuarios del sistema.
<b>Actores:</b>	Jefe de logística (Administrador)
<b>Precondiciones:</b>	<p>1. Ingresar al sistema por medio del login, ingresando su clave y contraseña.</p> <p>2. Elegir una opción de menú.</p>
<b>Eventos:</b>	<p>1. Para agregar un usuario, se tiene que desplegar la opción de menú "Mantenimiento", luego desplegar "Usuarios" y por último hacer clic en "Nuevo".</p> <p>2. Para listar a los usuarios, se tiene que desplegar la opción de menú "Mantenimientos", luego desplegar "Usuarios" y por último hacer clic en "Listado".</p> <p>3. Para editar un usuario, se realiza el paso N° 2 y luego se escoge un usuario para ver sus datos a modificar.</p>
<b>Postcondiciones:</b>	Luego de realizar uno de los eventos descritos, se tendrá un nuevo registro, se verán los registros o se cambiará un registro en la base de datos.

Tabla 38: Caso de Uso - Mantenimiento de usuarios.

Fuente: El Autor.

<b>Nombre:</b>	Mantenimiento de productos.
<b>Descripción:</b>	El administrador del sistema es el encargado de agregar, modificar y listar los productos.
<b>Actores:</b>	Jefe de logística (Administrador).
<b>Precondiciones:</b>	<p>1. Ingresar al sistema por medio del login, ingresando su clave y contraseña.</p> <p>2. Elegir una opción de menú.</p>

<b>Eventos:</b>	1. Para agregar un producto, se tiene que desplegar la opción de menú "Mantenimiento", luego desplegar "Productos" y por último hacer clic en "Nuevo". 2. Para listar productos, se tiene que desplegar la opción de menú "Mantenimientos", luego desplegar "Producto" y por último hacer clic en "Listado". 3. Para modificar a un producto, se realiza el paso número "2" y luego se escoge a un producto para ver sus datos a modificar.
<b>Postcondiciones:</b>	Luego de realizar uno de los eventos descritos, se tendrá un nuevo registro, se verán los registros o se cambiará un registro en la base de datos.

*Tabla 39: Caso de Uso - Mantenimiento de productos.*

*Fuente: El Autor.*

Tener en cuenta que las especificaciones de los casos de uso de: mantenimiento de sedes, proveedores y almacenes vienen siendo iguales a los descritos en los casos de uso de mantenimiento de usuarios y productos.

<b>Nombre:</b>	Registro de compras.
<b>Descripción:</b>	El operador de almacén se encarga de registrar compras de productos, ya sea equipos GPS, CHIP o cualquier otro producto sobre el que se requiera tener control.
<b>Actores:</b>	Operador de almacén
<b>Precondiciones:</b>	1. Ingresar al sistema por medio del login, ingresando su clave y contraseña. 2. Elegir la correspondiente opción de menú (compra por importación o localidad). 3. Visualizar el formulario a registrar.

<b>Eventos:</b>	<ol style="list-style-type: none"> <li>1. Llenar el formulario referente a información de la compra local o por importación.</li> <li>2. Agregar artículos en general, artículos GPS o CHIP según se dé el caso.</li> <li>3. Verificar el ingreso de los productos en el listado que deberá mostrar el sistema.</li> <li>4. El usuario debe guardar los cambios.</li> </ol>
<b>Postcondiciones:</b>	<p>El sistema generará el registro de compra correspondiente, identificándose de la siguiente manera: CPR - 000001-001, donde:</p> <ul style="list-style-type: none"> <li>▪ NI: Nota de ingreso.</li> <li>▪ 000001: N° de la nota de ingreso.</li> <li>▪ 001: Código de almacén a donde pertenece.</li> </ul> <p>El usuario visualiza el registro de compra creado.</p> <p>El usuario busca el registro de compra creado a través del almacén, el N° de compra o fechas de compra.</p>

*Tabla 40: Caso de Uso - Registro de compras.*

*Fuente: El Autor.*

<b>Nombre:</b>	Registro de nota de ingreso por reingreso.
<b>Descripción:</b>	El operador de almacén se encarga de registrar las notas de ingreso por reingreso, siempre y cuando el producto haya salido de almacén por una operación interna y un motivo.
<b>Actores:</b>	Operador de almacén, Sistema.
<b>Precondiciones:</b>	<ol style="list-style-type: none"> <li>1. Ingresar al sistema por medio del login, ingresando su clave y contraseña.</li> <li>2. Elegir la opción de ingreso a almacén por reingreso.</li> </ol>

<b>Eventos:</b>	<ol style="list-style-type: none"> <li>1. Para productos en general, se debe seleccionar el producto, la cantidad del producto y el motivo del reingreso.</li> <li>2. Para GPS, se debe buscar el N° del equipo y colocar el motivo del reingreso.</li> <li>3. Para CHIP, se debe buscar el N° del chip y colocar el motivo del reingreso.</li> <li>4. El usuario debe guardar los cambios.</li> <li>5. El sistema debe actualizar el inventario.</li> </ol>
<b>Postcondiciones:</b>	<p>El sistema generará la nota de ingreso correspondiente, identificándose de la siguiente manera: NI - 000001-001, donde:</p> <ul style="list-style-type: none"> <li>▪ NI: Nota de ingreso.</li> <li>▪ 000001: N° de la nota de ingreso.</li> <li>▪ 001: Código de almacén a donde pertenece.</li> </ul> <p>El usuario visualiza el registro de ingreso creado.</p> <p>El usuario busca el registro de ingreso creado a través del almacén, el N° de nota de ingreso o fecha de ingreso.</p>

*Tabla 41: Caso de Uso - Registro de nota de ingreso por reingreso.*

*Fuente: El Autor.*

<b>Nombre:</b>	Registro de nota de ingreso por transferencia.
<b>Descripción:</b>	El operador de almacén se encarga de registrar las notas de ingreso por transferencia, siempre y cuando exista la transferencia de un almacén origen al que se encuentra manejando el operador de almacén.
<b>Actores:</b>	Operador de almacén, Sistema.
<b>Precondiciones:</b>	<ol style="list-style-type: none"> <li>1. Ingresar al sistema por medio del login, ingresando su clave y contraseña.</li> <li>2. Elegir la opción de ingreso a almacén por</li> </ol>

	<p>transferencia.</p> <p>3. Listar el o los registros de transferencia, es importante saber que el sistema debe mostrar solamente aquellas transferencias al almacén indicado o de destino; en el presente caso, viene siendo el almacén donde se está trabajando.</p> <p>4. Seleccionar el registro de transferencia que se quiere aceptar.</p>
<b>Eventos:</b>	<p>1. Verificar los productos que están siendo transferidos.</p> <p>2. Colocar la fecha de ingreso y el almacén correspondiente.</p> <p>3. El usuario debe guardar los cambios.</p> <p>4. El sistema debe actualizar el inventario.</p>
<b>Postcondiciones:</b>	<p>El sistema generará la nota de ingreso, identificándose de la siguiente manera: NI - 000001-001, donde:</p> <ul style="list-style-type: none"> <li>▪ NI: Nota de ingreso.</li> <li>▪ 000001: N° de la nota de ingreso.</li> <li>▪ 001: Código de almacén a donde pertenece.</li> </ul> <p>El usuario visualiza el ingreso por transferencia.</p> <p>El usuario busca el registro de ingreso creado a través del almacén, el N° de nota de ingreso o fecha de ingreso.</p>

Tabla 42: Caso de Uso - Registro de nota de ingreso por transferencia.

Fuente: El Autor.

<b>Nombre:</b>	Registro de nota de ingreso por compra.
<b>Descripción:</b>	El operador de almacén se encarga de registrar las notas de ingreso por compra, siempre y cuando exista un registro de compra previo.
<b>Actores:</b>	Operador de almacén, Sistema.
<b>Precondiciones:</b>	1. Ingresar al sistema por medio del login, ingresando su clave y contraseña.

	<p>2. Elegir la opción de ingreso a almacén por compra.</p> <p>3. Listar el o los registros de compra, es importante saber que el sistema debe mostrar solamente aquellas compras en el almacén donde actualmente trabaja el operador de almacén.</p> <p>4. Seleccionar el registro de compra.</p>
<b>Eventos:</b>	<p>1. Verificar los productos que serán ingresados por la compra.</p> <p>2. Colocar la fecha de ingreso y el almacén correspondiente.</p> <p>3. El usuario debe guardar los cambios.</p> <p>4. El sistema debe actualizar el inventario.</p>
<b>Postcondiciones:</b>	<p>El sistema generará la nota de ingreso, identificándose de la siguiente manera: NI - 000001-001, donde:</p> <ul style="list-style-type: none"> <li>▪ NI: Nota de ingreso.</li> <li>▪ 000001: N° de la nota de ingreso.</li> <li>▪ 001: Código de almacén a donde pertenece.</li> </ul> <p>El usuario visualiza el ingreso por compra.</p> <p>El usuario busca el registro de ingreso creado a través del almacén, el N° de nota de ingreso o fecha de ingreso.</p> <p>El sistema actualiza el inventario.</p>

Tabla 43: Caso de Uso - Registro de nota de ingreso por compra.

Fuente: El Autor.

<b>Nombre:</b>	Registro de orden de salida por operación interna.
<b>Descripción:</b>	El operador de almacén se encarga de registrar las órdenes de salida por operación interna, siempre y cuando el producto se encuentre en almacén.
<b>Actores:</b>	Operador de almacén, Sistema.

<b>Precondiciones:</b>	<ol style="list-style-type: none"> <li>1. Ingresar al sistema por medio del login, ingresando su clave y contraseña.</li> <li>2. Elegir la opción de salida de almacén por operación interna.</li> </ol>
<b>Eventos:</b>	<ol style="list-style-type: none"> <li>1. Para productos en general, se debe seleccionar el producto, la cantidad del producto y el motivo de la salida.</li> <li>2. Para GPS, se debe buscar el N° del equipo, el motivo y observación de la salida.</li> <li>3. Para CHIP, se debe buscar el N° del chip, el motivo y observación de la salida.</li> <li>4. El usuario debe guardar los cambios.</li> <li>5. El sistema debe actualizar el inventario.</li> </ol>
<b>Postcondiciones:</b>	<p>El sistema generará la orden de salida correspondiente, identificándose de la siguiente manera: ORDS - 000001-001, donde:</p> <ul style="list-style-type: none"> <li>▪ ORDS: Orden de salida.</li> <li>▪ 000001: N° de la orden de salida.</li> <li>▪ 001: Código de almacén a donde pertenece.</li> </ul> <p>El usuario visualiza el registro de orden de salida creado.</p> <p>El usuario busca el registro de orden de salida creado a través del almacén, el N° o fecha de orden de salida.</p>

Tabla 44: Caso de Uso - Registro de orden de salida por operación interna.

Fuente: El Autor.

<b>Nombre:</b>	Registro de orden de salida por transferencia.
<b>Descripción:</b>	El operador de almacén se encarga de registrar las órdenes de salida por transferencia, siempre y cuando el producto se encuentre en almacén.
<b>Actores:</b>	Operador de almacén, Sistema.

<b>Precondiciones:</b>	<ol style="list-style-type: none"> <li>1. Ingresar al sistema por medio del login, ingresando su clave y contraseña.</li> <li>2. Elegir la opción de salida de almacén por transferencia.</li> </ol>
<b>Eventos:</b>	<ol style="list-style-type: none"> <li>1. Para productos en general, se debe seleccionar el producto, la cantidad del producto y el motivo de la salida.</li> <li>2. Para GPS, se debe buscar el Nº del equipo.</li> <li>3. Para CHIP, se debe buscar el Nº del chip.</li> <li>4. El usuario debe guardar los cambios.</li> <li>5. El sistema debe actualizar el inventario.</li> </ol>
<b>Postcondiciones:</b>	<p>El sistema generará la orden de salida correspondiente, identificándose de la siguiente manera: ORDS - 000001-001, donde:</p> <ul style="list-style-type: none"> <li>▪ ORDS: Orden de salida.</li> <li>▪ 000001: Nº de la orden de salida.</li> <li>▪ 001: Código de almacén a donde pertenece.</li> </ul> <p>El usuario visualiza el registro de orden de salida creado.</p> <p>El usuario busca el registro de orden de salida creado a través del almacén, el Nº o fecha de orden de salida.</p>

Tabla 45: Caso de Uso - Registro de orden de salida por transferencia.

Fuente: El Autor.

<b>Nombre:</b>	Aprueba orden de salida de almacén.
<b>Descripción:</b>	El jefe de logística aprueba o no las órdenes de salida creadas de manera total o parcial (ciertos productos).
<b>Actores:</b>	Jefe de logística (Administrador).
<b>Precondiciones:</b>	<ol style="list-style-type: none"> <li>1. Ingresar al sistema por medio del login, ingresando su clave y contraseña.</li> <li>2. Elegir la opción de listar órdenes de salida de</li> </ol>

	<p>almacén.</p> <p>3. Buscar y seleccionar la orden de salida.</p>
<b>Eventos:</b>	<p>1. Verificar el listado de productos.</p> <p>2. Seleccionar el o los productos para aprobar su salida.</p> <p>3. Aprobar la orden de salida de almacén.</p>
<b>Postcondiciones:</b>	<p>El sistema generará la nota de salida correspondiente, identificándose de la siguiente manera: NS - 000001-001, donde:</p> <ul style="list-style-type: none"> <li>▪ NS: Nota de salida.</li> <li>▪ 000001: N° de la nota de salida.</li> <li>▪ 001: Código de almacén a donde pertenece.</li> </ul> <p>El usuario visualiza el registro de nota de salida creado.</p> <p>El usuario busca el registro de nota de salida creado a través del almacén, el N° o fecha de nota de salida.</p>

*Tabla 46: Caso de Uso - Aprobación de orden de salida de almacén.*

*Fuente: El Autor.*

<b>Nombre:</b>	Genera nota de salida
<b>Descripción:</b>	El sistema debe generar automáticamente una nota de salida a través de orden de nota de salida previamente aprobada.
<b>Actores:</b>	Sistema.
<b>Precondiciones:</b>	1. Aprobar una orden de nota de salida.
<b>Eventos:</b>	<p>1. El sistema debe verificar los productos seleccionados y aprobados por el jefe de logística.</p> <p>2. El sistema debe generar la nota de salida.</p> <p>3. El sistema debe actualizar el inventario.</p>
<b>Postcondiciones:</b>	<p>Se listará las notas de salida creadas, identificándolas de la siguiente manera: NS - 000001-001, donde:</p> <ul style="list-style-type: none"> <li>▪ NS: Nota de salida.</li> </ul>

	<ul style="list-style-type: none"> <li>▪ 000001: N° de la nota de salida.</li> <li>▪ 001: Código de almacén a donde pertenece.</li> </ul> <p>El usuario visualiza el registro de nota de salida creado.</p> <p>El usuario busca el registro de nota de salida creado a través del almacén, el N° o fecha de nota de salida.</p>
--	---

*Tabla 47: Caso de Uso - Genera nota de salida.*

*Fuente: El Autor.*

<b>Nombre:</b>	Actualizar inventario.
<b>Descripción:</b>	El operador de almacén realiza un movimiento de ingreso u orden de salida de productos y posterior a ello el operador debe guardar los cambios para que se genere una nota de ingreso u orden de salida a ser aprobada o no.
<b>Actores:</b>	Operador de almacén, Sistema.
<b>Precondiciones:</b>	1. Crear un movimiento de almacén, utilizando los productos que se quieran ingresar o sacar del almacén por algún motivo.
<b>Eventos:</b>	1. El operador guarda el movimiento de almacén. 2. El sistema procesa el movimiento de almacén. 3. El sistema actualiza el inventario. 4. El sistema procesa el inventario.
<b>Postcondiciones:</b>	El stock de productos se ha debido de actualizar según el movimiento realizado por el usuario operador del almacén.

*Tabla 48: Caso de Uso - Actualizar inventario.*

*Fuente: El Autor.*

<b>Nombre:</b>	Generar reportes
<b>Descripción:</b>	El jefe de logística se encarga de generar los reportes diarios, mensuales y aquellos que sean necesarios en el transcurso del día.
<b>Actores:</b>	Jefe de logística (Administrador).
<b>Precondiciones:</b>	<ol style="list-style-type: none"> <li>1. Ingresar al sistema por medio del login, ingresando su clave y contraseña.</li> <li>2. Elegir el reporte a visualizar.</li> </ol>
<b>Eventos:</b>	<ol style="list-style-type: none"> <li>1. Ingresar los parámetros de entrada del reporte.</li> <li>2. Hacer clic sobre el botón de Generar.</li> </ol>
<b>Postcondiciones:</b>	El reporte es generado y a continuación se podrá exportar a un tipo de archivo, como, por ejemplo: Archivo CSV, PDF, MHTML, EXCEL, TIFF, WORD, XML.

*Tabla 49: Caso de Uso - Generar reportes.*

*Fuente: El Autor.*

#### 4.2.6. Diagramas de secuencia

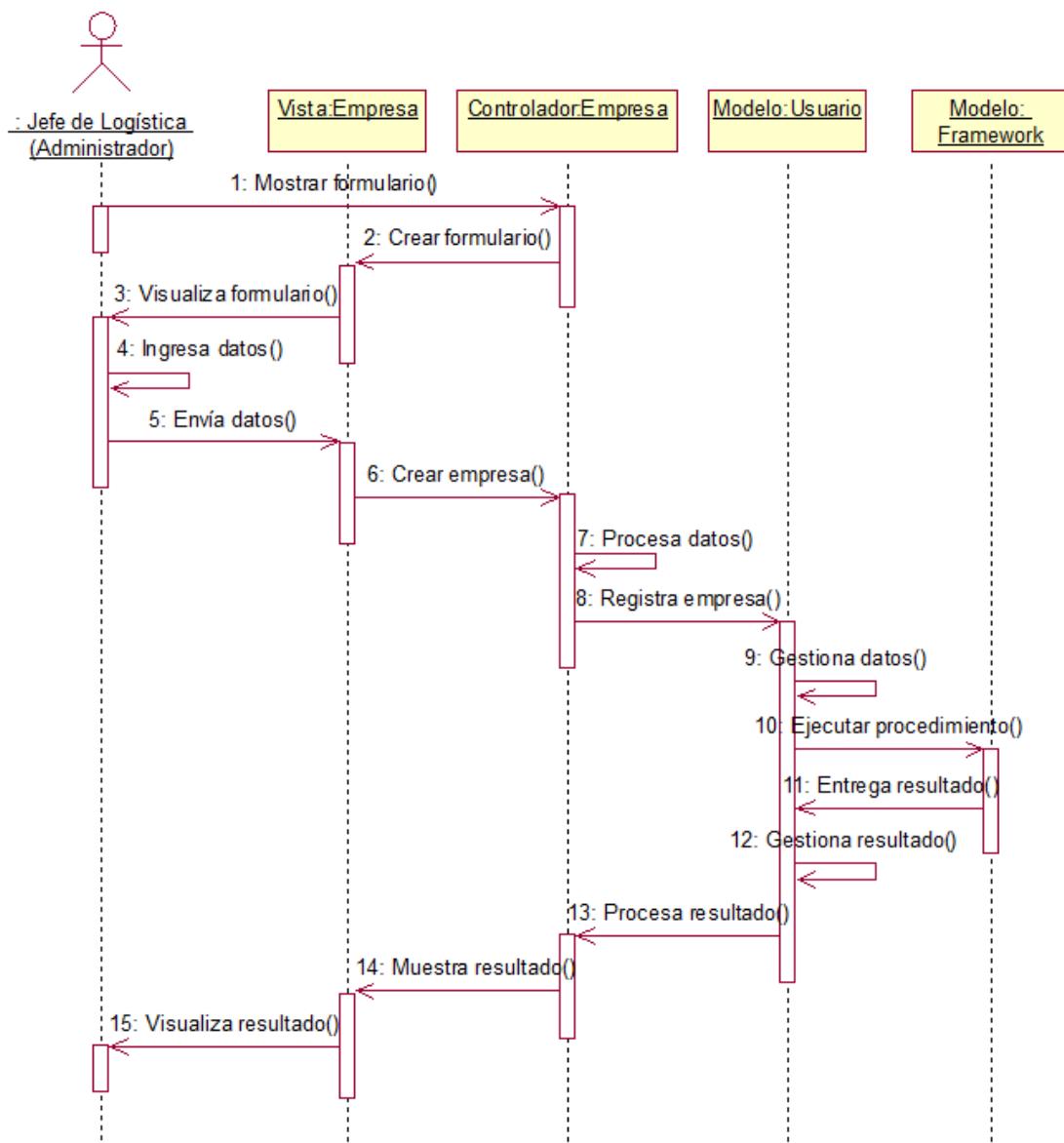


Figura 53: Diagrama de Secuencia - Registra empresa.

Fuente: El Autor.

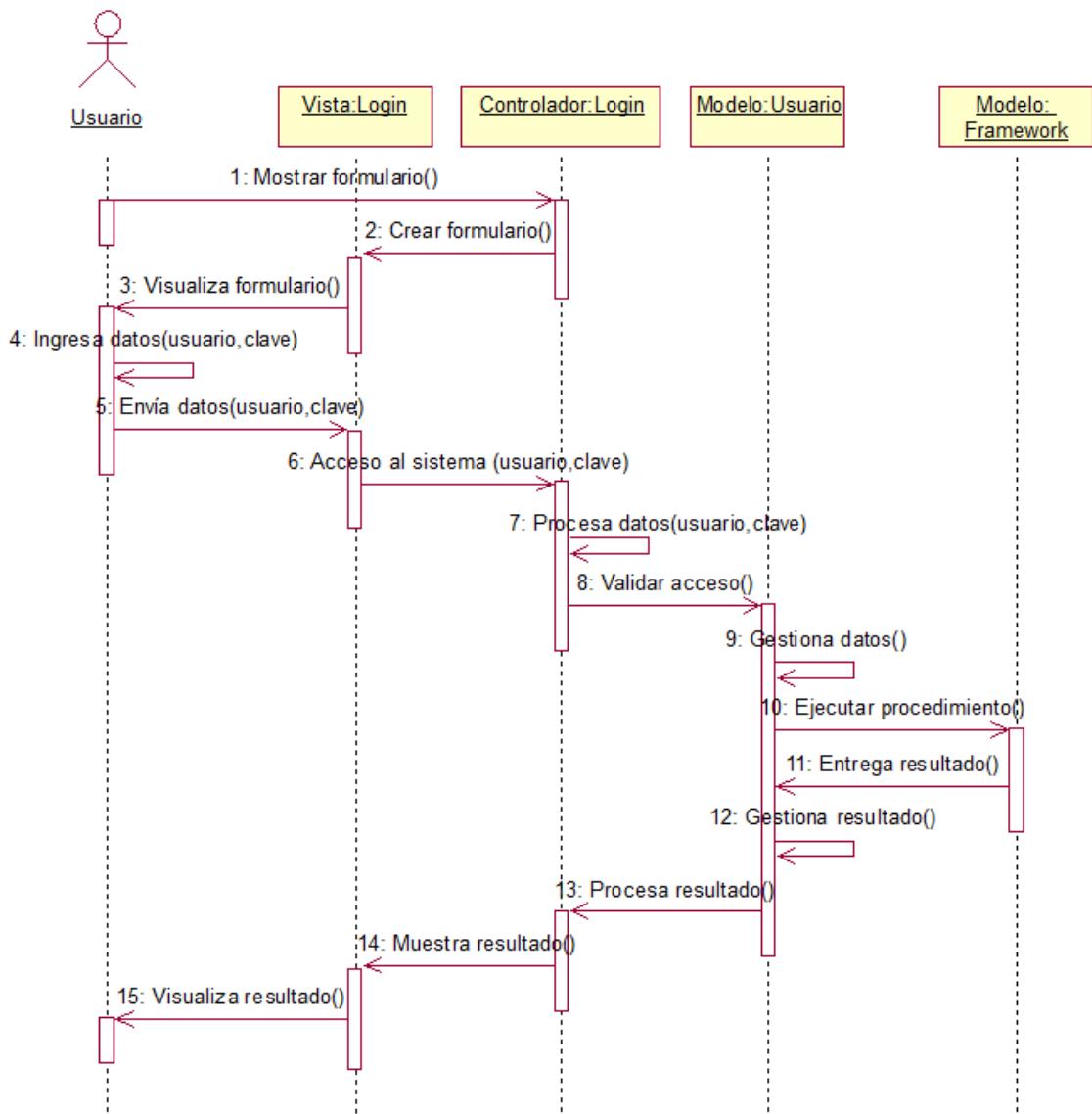


Figura 54: Diagrama de Secuencia - Login e ingreso al sistema.

Fuente: El Autor.

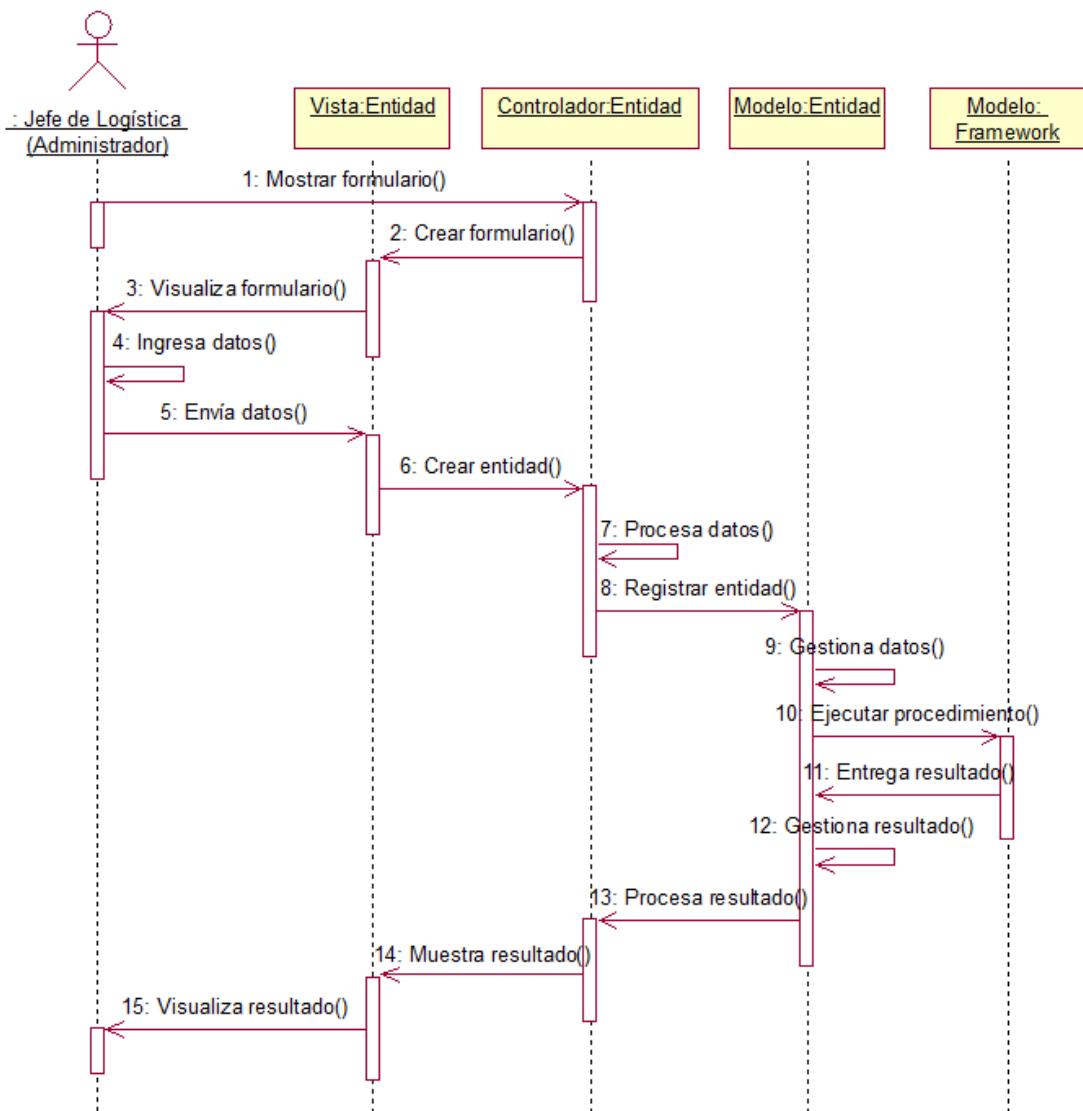


Figura 55: Diagrama de Secuencia - Mantenimiento de entidades principales.

Fuente: El Autor.

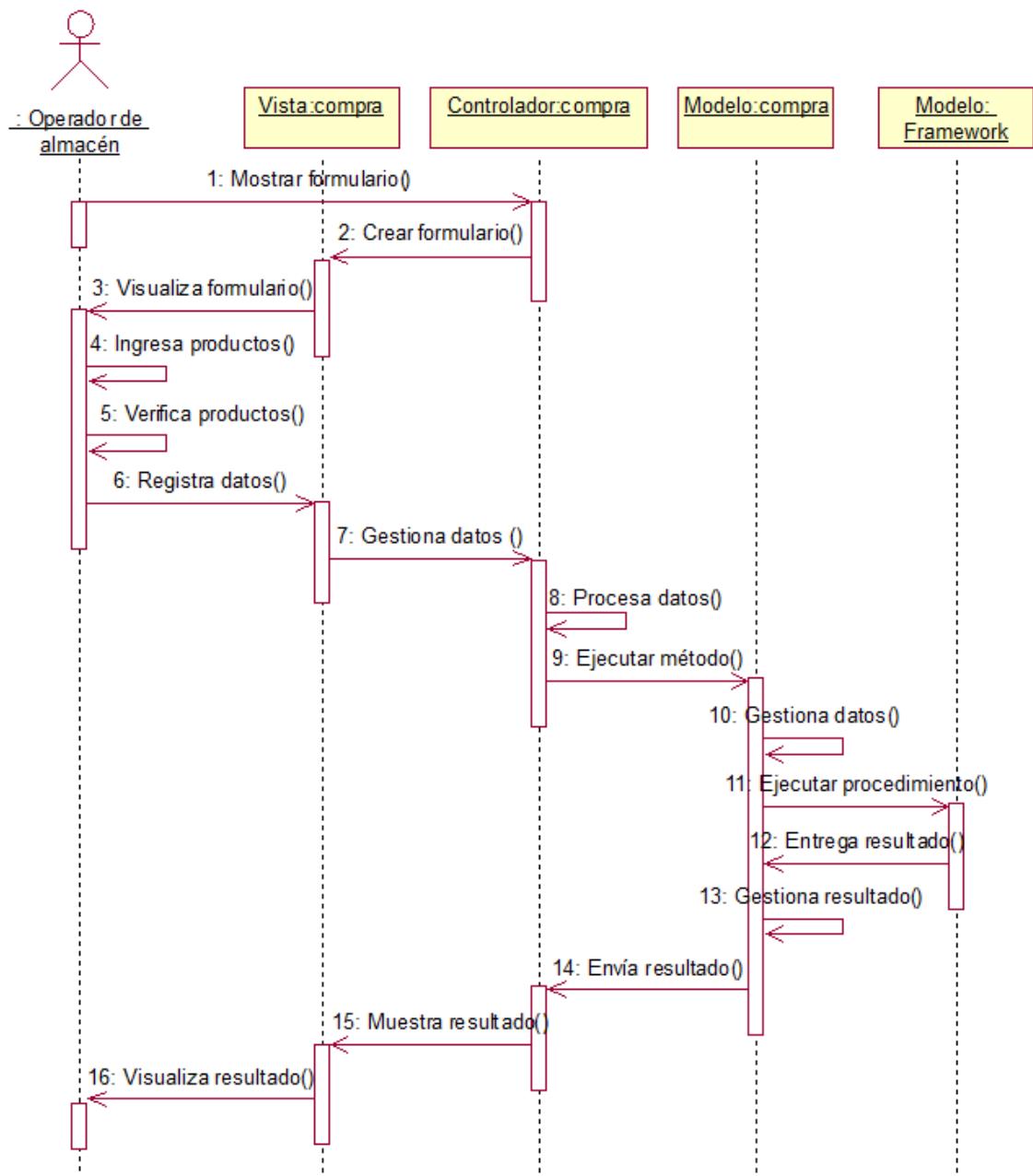


Figura 56: Diagrama de Secuencia - Registro de compras.

Fuente: El Autor.

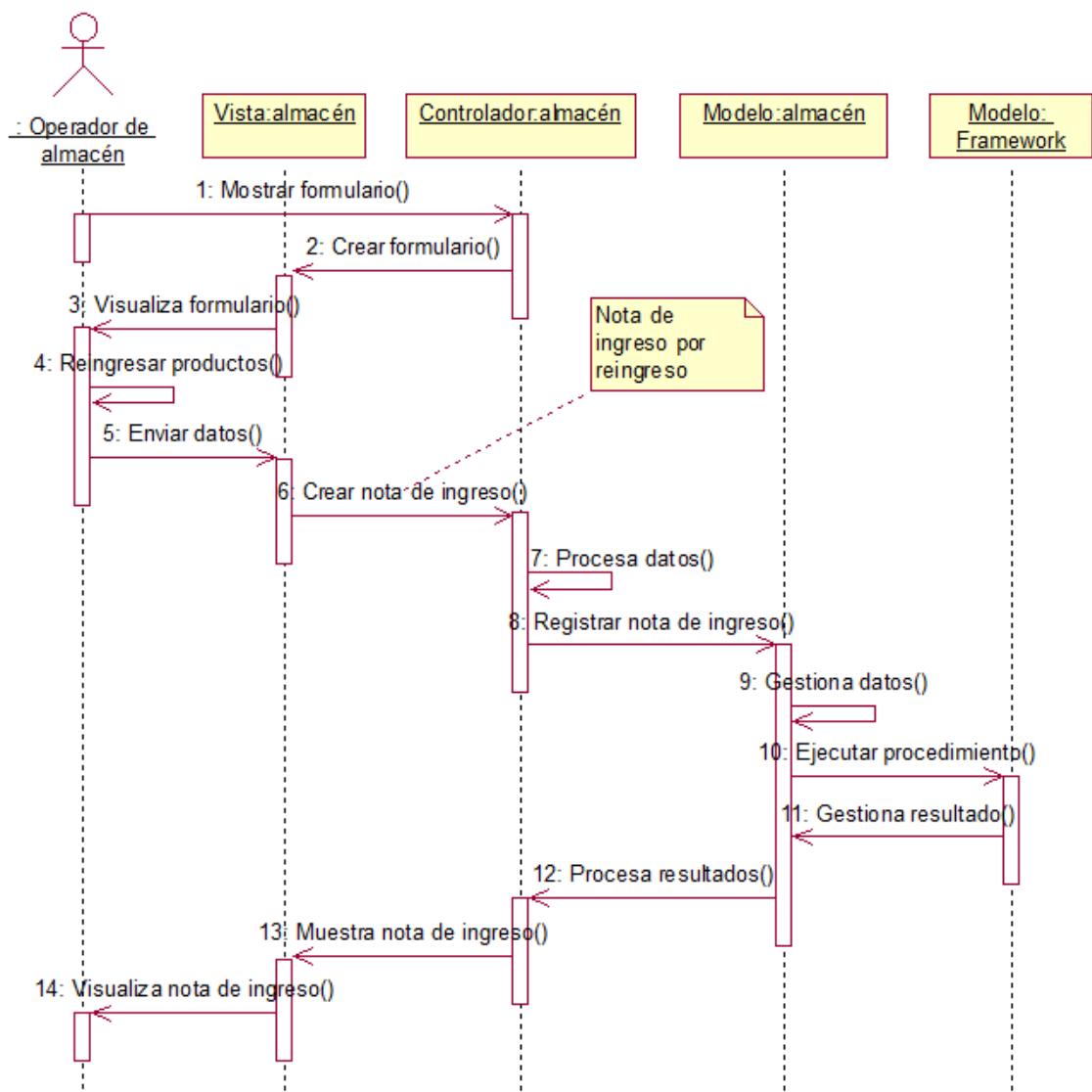


Figura 57: Diagrama de Secuencia - Registro de nota de ingreso por reingreso.

Fuente: El Autor.

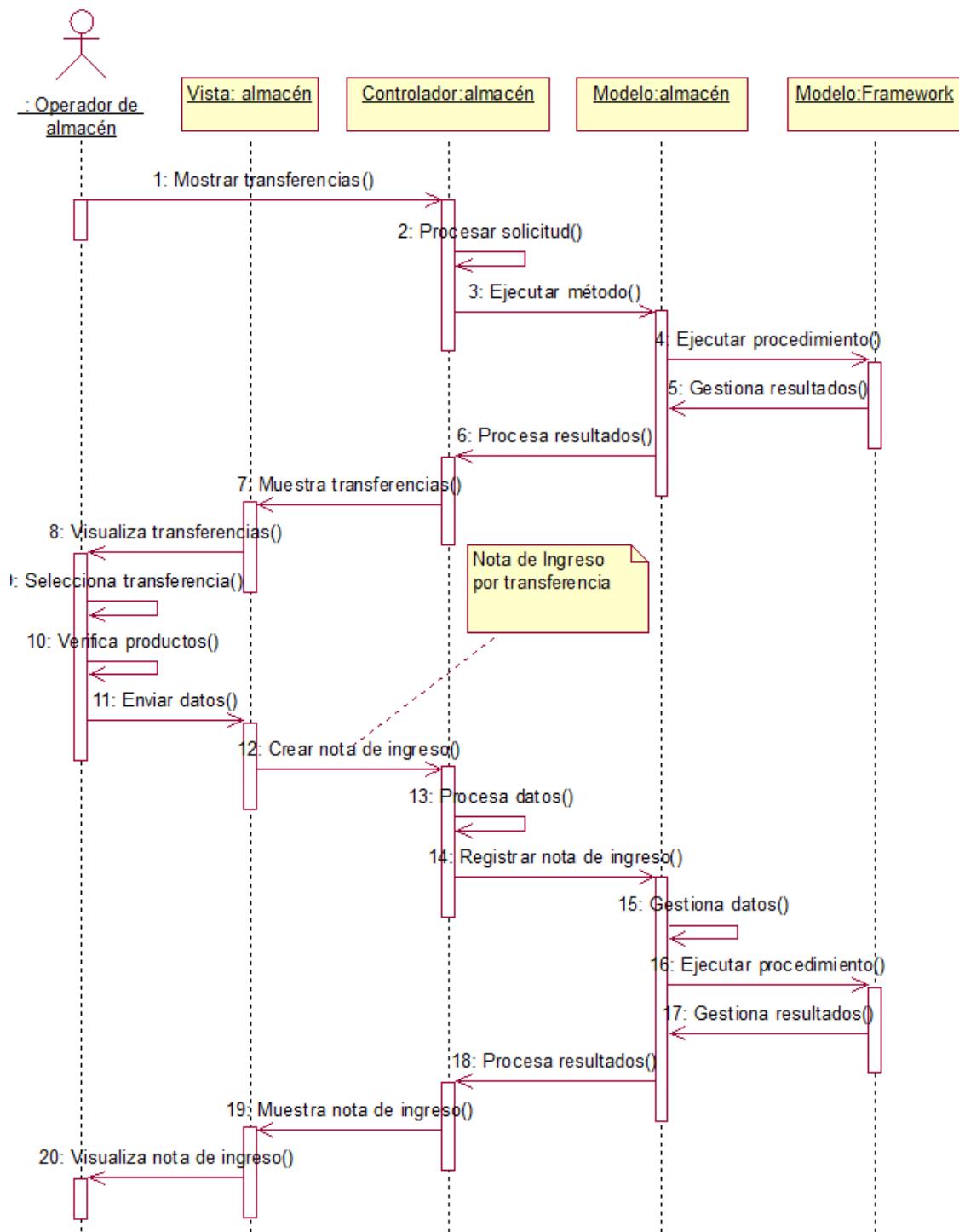


Figura 58: Diagrama de Secuencia - Registro de nota de ingreso por transferencia.

Fuente: El Autor.

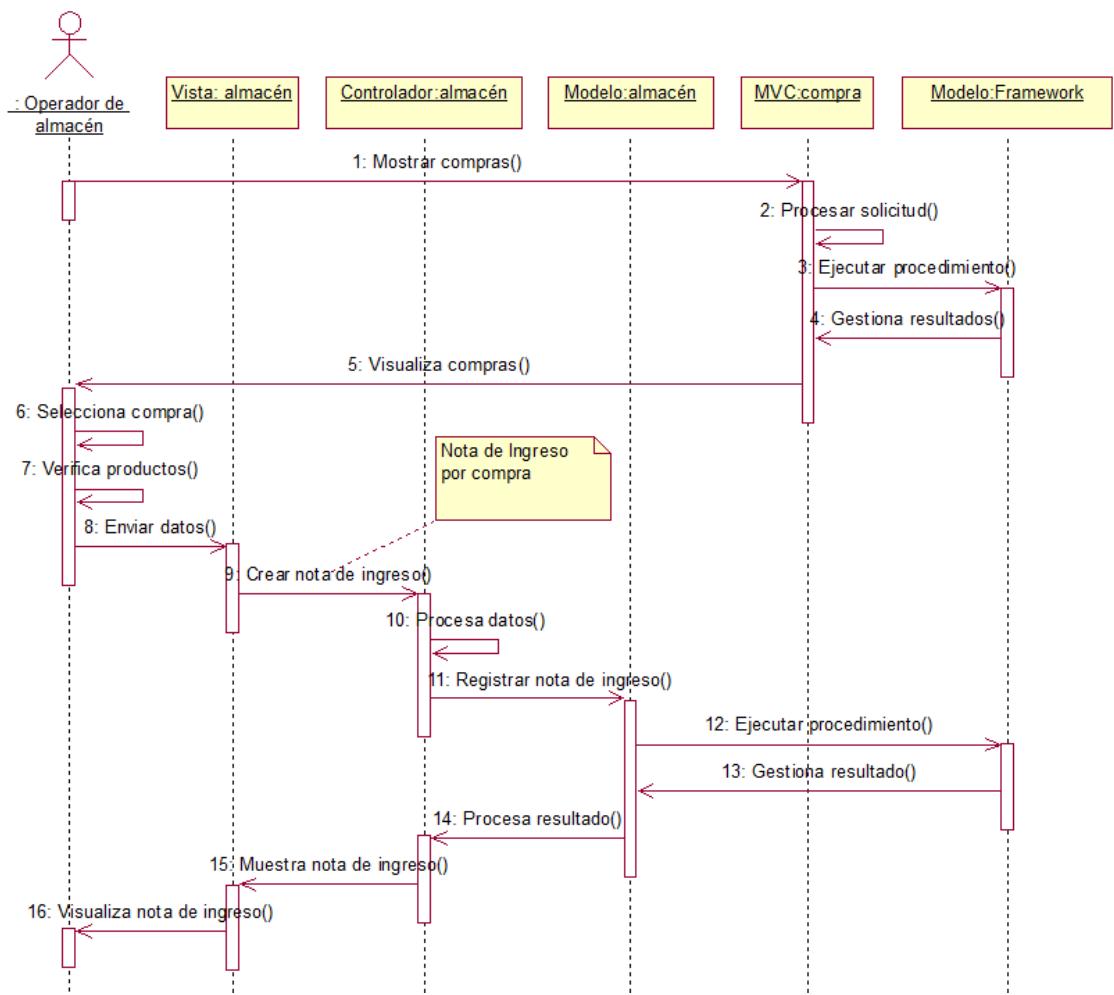


Figura 59: Diagrama de Secuencia - Registro de nota de ingreso por compra.

Fuente: El Autor.

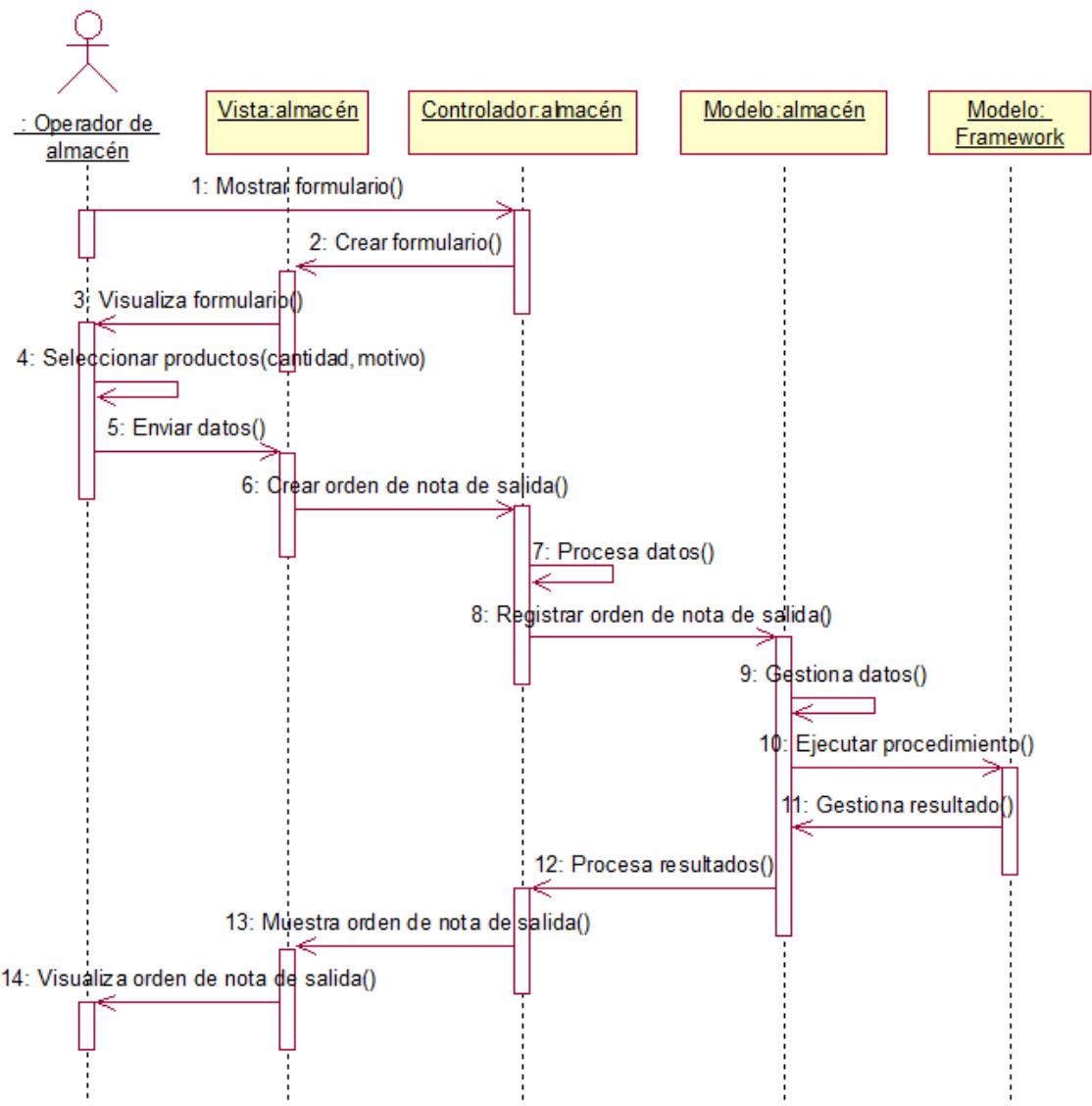


Figura 60: Diagrama de Secuencia - Registro de orden de salida.

Fuente: El Autor.

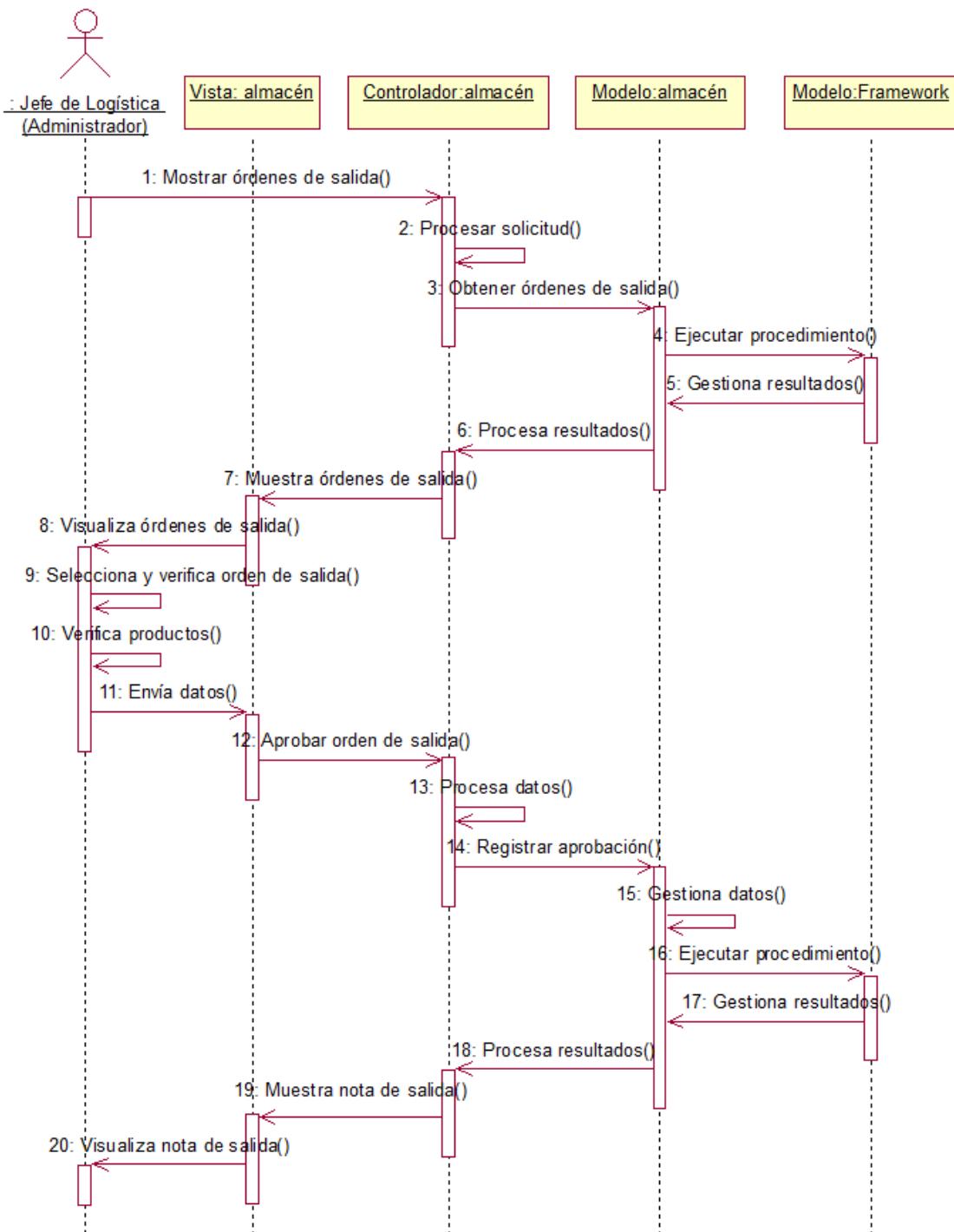
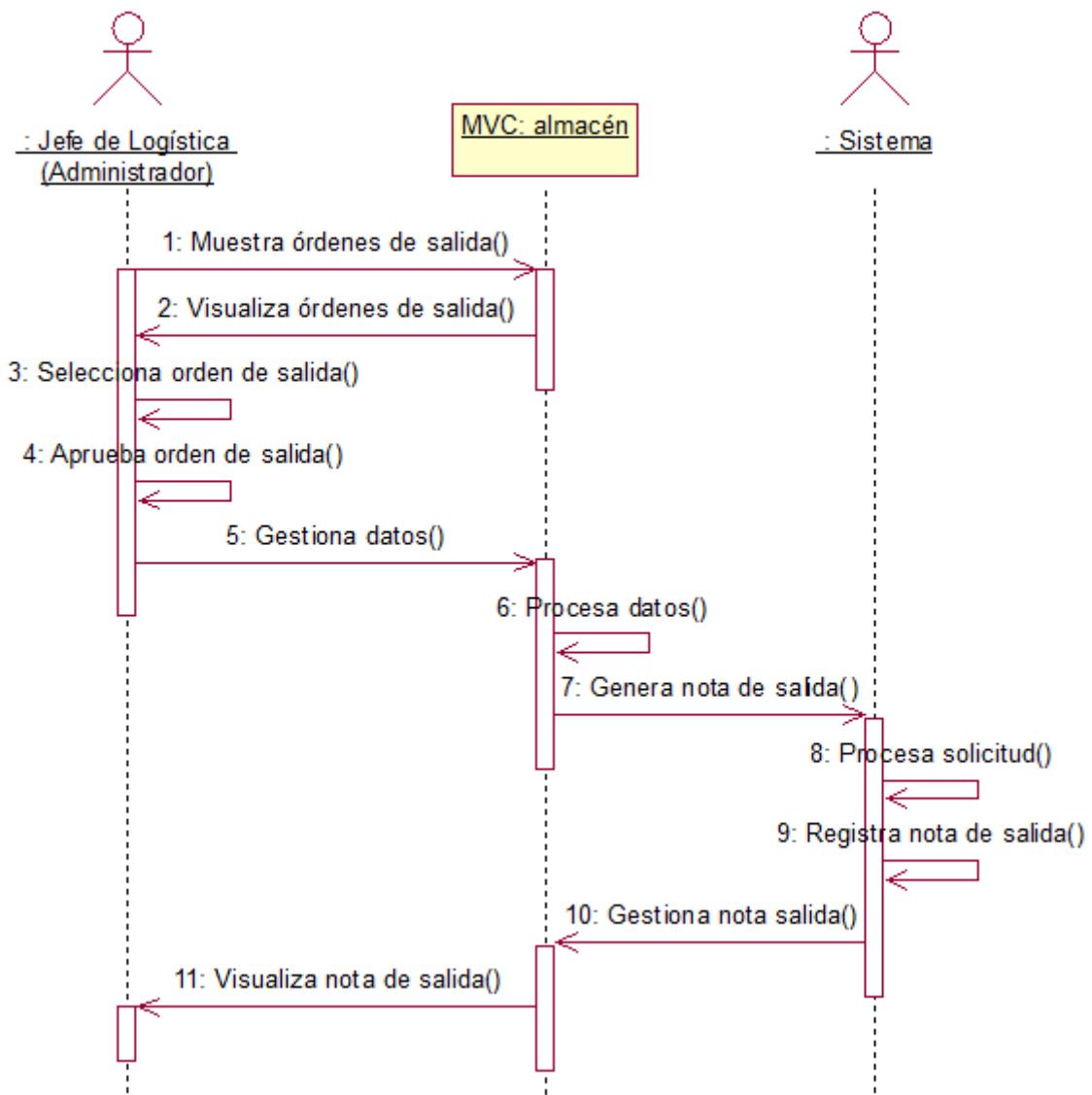


Figura 61: Diagrama de Secuencia - Aprueba orden de salida de almacén.

Fuente: El Autor.



*Figura 62: Diagrama de Secuencia - Genera nota de salida.*

*Fuente: El Autor.*

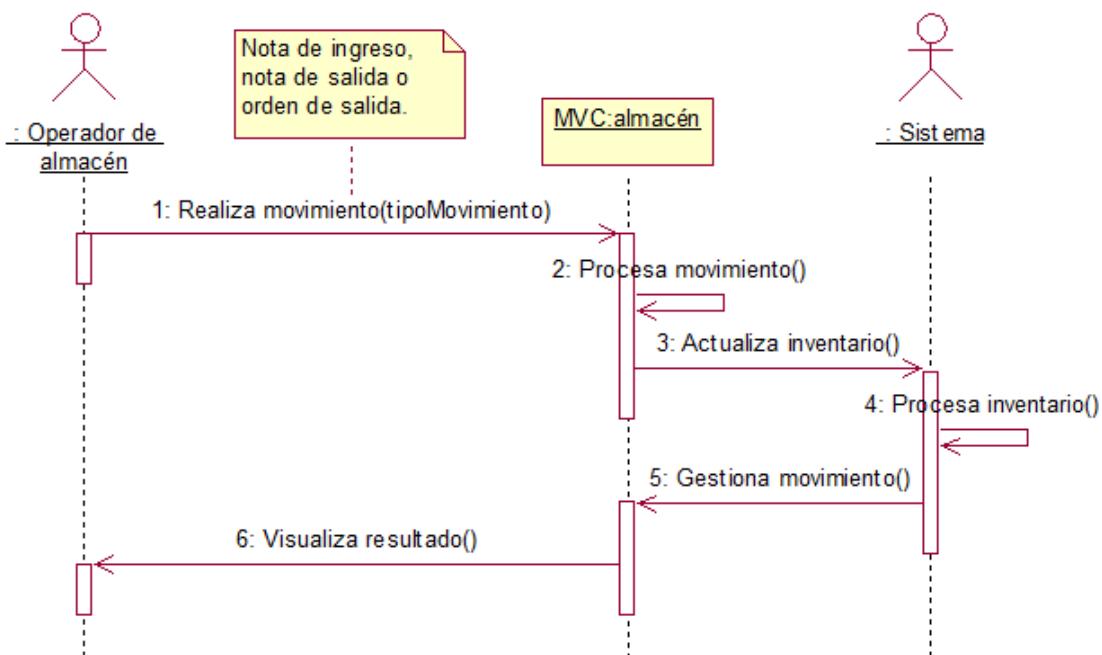


Figura 63: Diagrama de Secuencia - Actualizar inventario.

Fuente: El Autor.

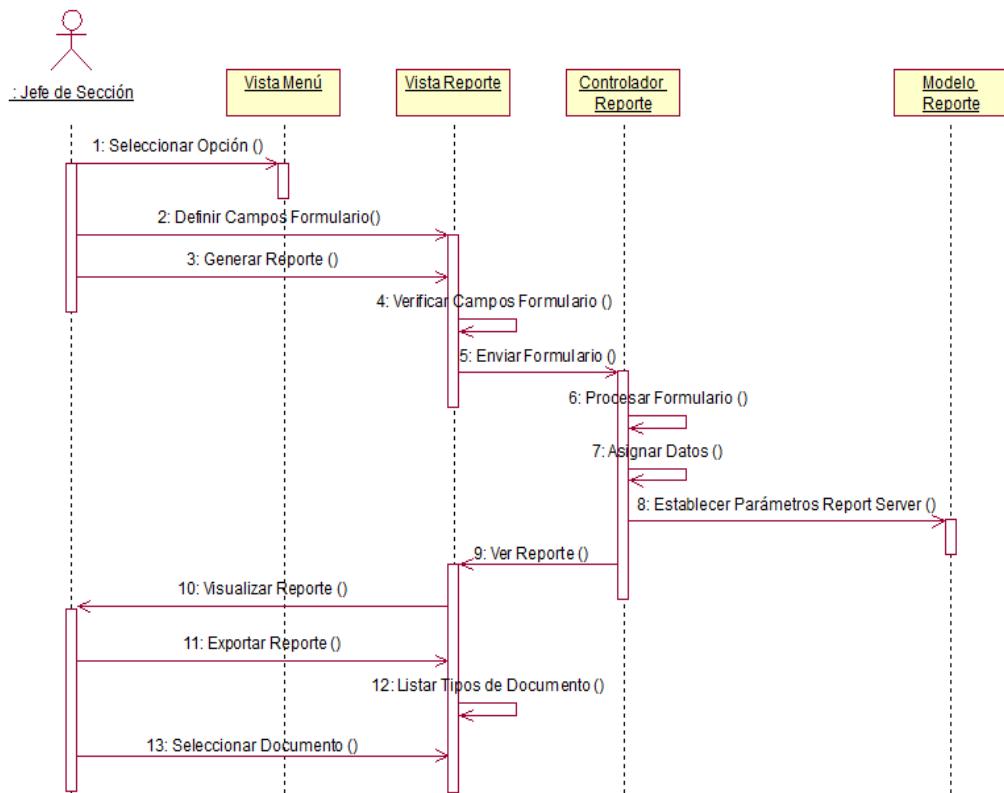


Figura 64: Diagrama de Secuencia - Generar reportes.

Fuente: El Autor.

#### 4.2.7. Diagrama de clases

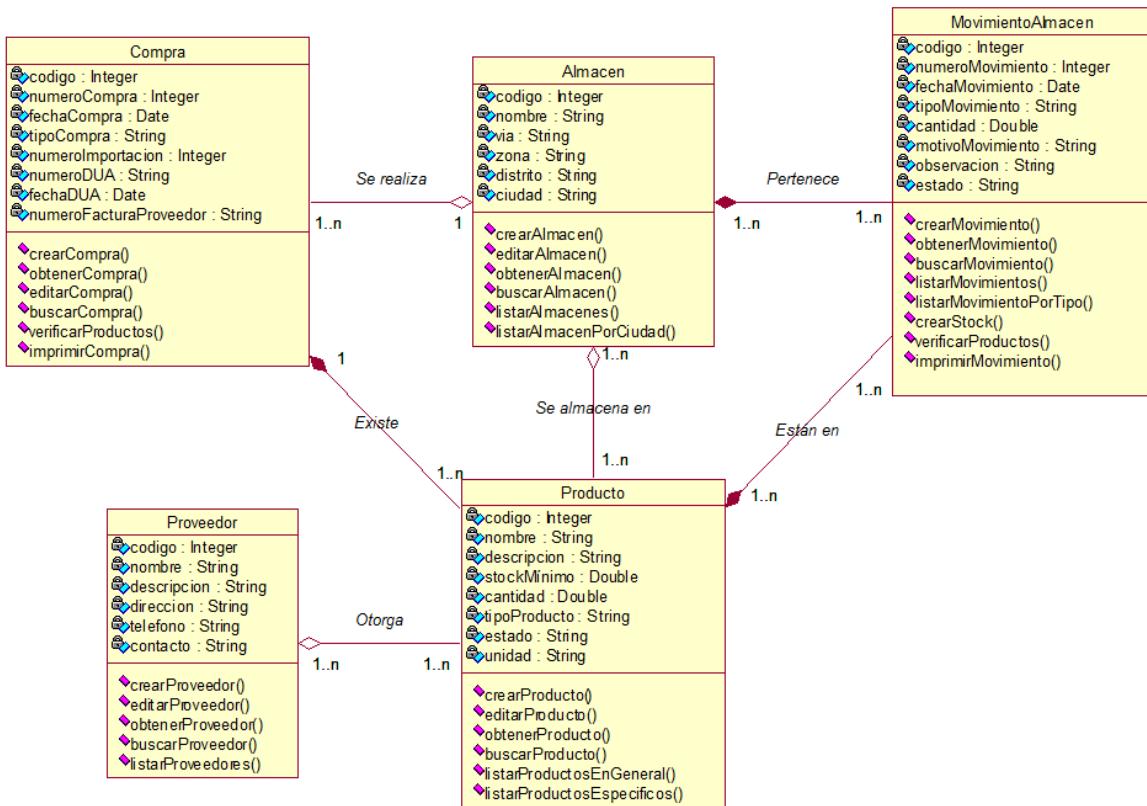


Figura 65: Diagrama de clases.

Fuente: El Autor.

#### 4.2.8. Diagrama Entidad Relación (E-R)

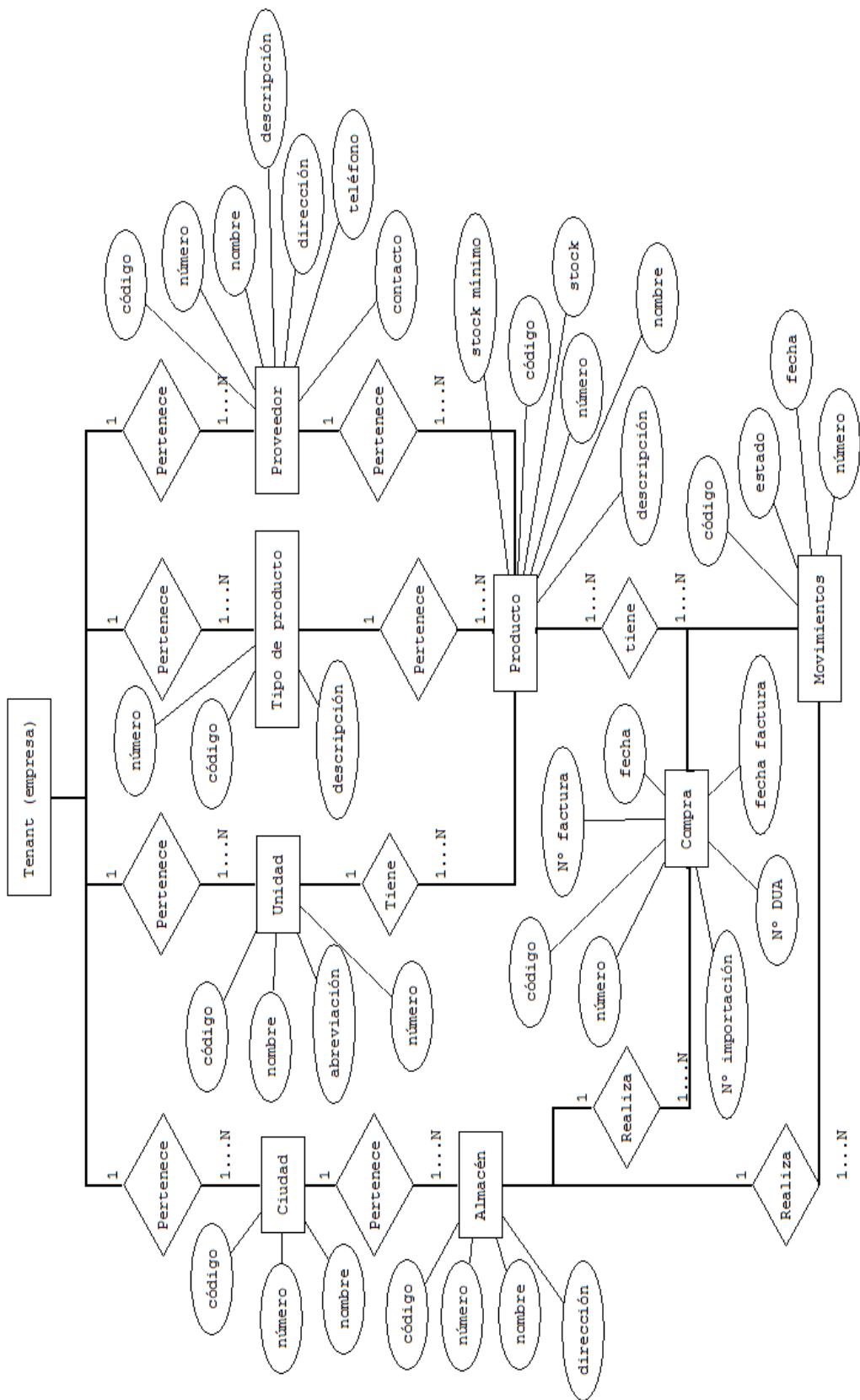


Figura 66: Diagrama Entidad - Relación (E-R).

Fuente: El Autor.

#### **4.2.9. Construcción del Sistema**

El sistema ha sido instanciado por el Framework y es ahora en dónde el desarrollador debe construir el software alineado a la lógica del negocio y/o requerimientos planteados anteriormente. Para construir software a través del Framework, es necesario ver el punto N° 6.4 del Anexo “A”, relacionado a la construcción de Software. Para mayor detalle sobre la implementación, ver el anexo C.

#### **4.2.10. Pruebas del Sistema**

El sistema ha sido instanciado por el Framework y ya se han realizado diferentes pruebas en el anterior capítulo, por lo que, solamente queda realizar pruebas de caja negra para verificar que los datos registrados por un usuario perteneciente a un cliente no sean vistos por otro usuario que pertenezca a un cliente diferente, con esta prueba podremos obtener la confianza de que los datos son únicos en cada empresa, cliente o tenant. Para el presente, se ha tomado en cuenta a usuarios finales tanto de UVICAR S.A y Control Total.

<b>Prueba N°</b>	1
<b>Descripción</b>	Visualizar que los datos sean independientes y consistentes por cada cliente que utilice el sistema de información.
<b>Objetivos y proceso</b>	Verificar la concurrencia de los usuarios finales, independencia y consistencia de los datos. Para ello es necesario: <ol style="list-style-type: none"><li>1. Registrarse como cliente.</li><li>2. Verificar el correo electrónico enviado en la bandeja de entrada y seguir el link para activar la cuenta.</li><li>3. Ingresar con el usuario y contraseña creado.</li><li>4. Realizar el mantenimiento a entidades como almacén, usuarios, ciudades,</li></ol>

	<p>productos, proveedores.</p> <ol style="list-style-type: none"> <li>5. Realizar compra de productos.</li> <li>6. Realizar movimientos de almacén.</li> <li>7. Generar reportes.</li> <li>8. Constatar los datos que sean del cliente.</li> </ol>
<b>Condiciones</b>	<ul style="list-style-type: none"> <li>▪ Registrarse como cliente.</li> <li>▪ Activar la cuenta.</li> <li>▪ Ingresar con un usuario y contraseña.</li> </ul>
<b>Resultado esperado</b>	Los datos deben pertenecer únicamente al cliente, tenant o empresa que utiliza el sistema de información como servicio.
<b>Resultado obtenido</b>	El resultado fue satisfactorio.

*Tabla 50: Prueba unitaria - verificación de datos.*

*Fuente: El Autor.*

# **Capítulo 5**

## **Pruebas y resultados**

---

### **5.1. INTRODUCCIÓN**

Para las presentes pruebas se utiliza el Criterio de Expertos. Este método permitirá consultar a expertos y validar el Framework desarrollado, sustentado en sus conocimientos, investigaciones, experiencia, estudios bibliográficos, etc. Los estudios de Jakob Nielsen, muestran que un número de entre 3 y 5 evaluadores es suficiente para la evaluación por criterios de un sitio web. Este número puede ser incrementado en caso de que la usabilidad sea crucial para el éxito del sitio web.

## **5.2. EVALUACIÓN DE EXPERTOS**

Para evaluar el Framework, ha sido necesario contar con la presencia y elección de expertos que pertenecen a empresas que desarrollan software y que tienen conocimientos técnicos en cuanto a la construcción de software en sistemas de información con marcos de trabajo (Framework); para esto, se realizó la presente evaluación a dos (02) equipos de 2 personas cada uno, pertenecientes a diferentes empresas.

El escenario para la evaluación de expertos, es el siguiente:

- 1.** Explicar el concepto de SaaS (Software as Service).
- 2.** Presentación del subsistema de información logístico construido en el presente proyecto.
- 3.** Demostración de la instancia del Framework a través de la interface web que presenta.
- 4.** Presentación del Manual Técnico del Framework, mostrando los módulos principales para comenzar a instanciar y construir software en sistemas de información orientados a la prestación de servicios.
- 5.** Realizar una encuesta a cada experto perteneciente a los equipos de desarrollo de software.

## **5.3. PERFIL DE EXPERTOS**

- Estudios superiores en análisis, diseño y construcción de aplicaciones web, sistemas de información operativos, sistemas de información gerenciales, entre otros sistemas informáticos.
- Experiencia superior a cuatro años en desarrollo de sistemas informáticos utilizando Frameworks.

- Conocimientos acerca de tecnología aplicada en entorno web.

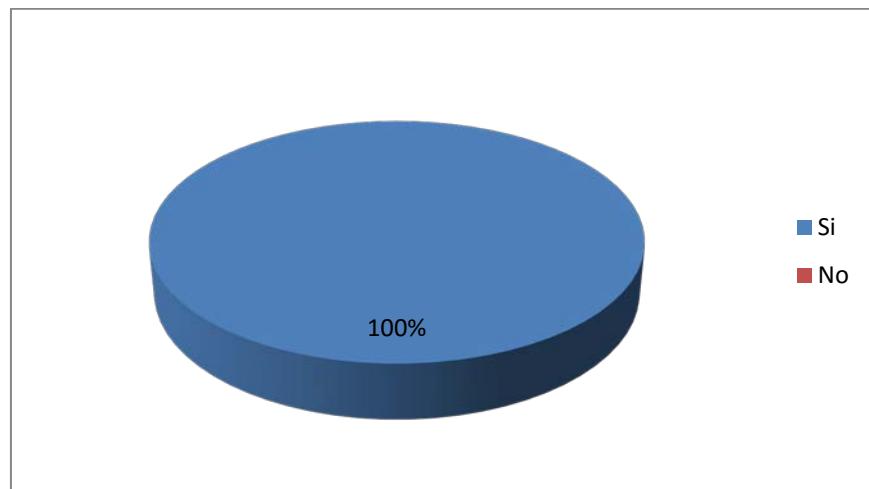
## 5.4. INTERPRETACIÓN DE EVALUACIÓN DE EXPERTOS

### 1. ¿Puede Ud. trabajar con el Framework en navegadores que utiliza?

RESPUESTA	FRECUENCIA	PORCENTAJE
Si	4	100%
No	0	0%
<b>Total</b>	<b>4</b>	<b>100%</b>

*Tabla 51: Framework en navegadores web.*

*Fuente: El Autor.*



*Figura 67: Framework en navegadores web.*

*Fuente: El Autor.*

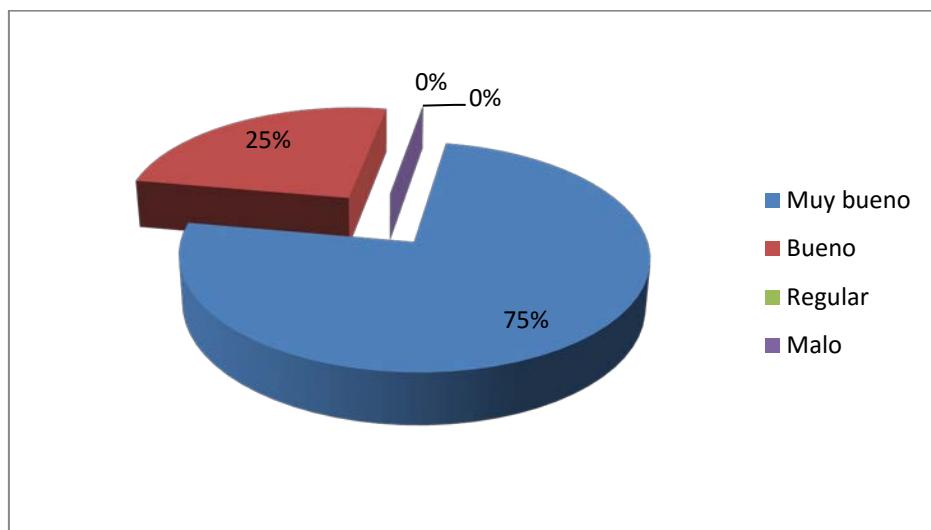
Los 4 expertos encuestados que representan el 100% visualizaron el Framework en sus diferentes navegadores web, tales como: Google Chrome, Internet Explorer y Mozilla Firefox, tomando como criterio su experiencia de desarrollo con cada navegador. El resultado final fue favorable debido a que realizaron además la instancia del sistema de información sin ningún problema, sin embargo, 1 experto recomendó realizar una reingeniería en cuanto al diseño o presentación del módulo instanciado y utilizar HTML 5.

**2. ¿Qué opina sobre el desarrollo del Framework basado en el patrón de diseño MVC (Modelo - Vista - Controlador)?**

ESPUESTA	FRECUENCIA	PORCENTAJE
Muy Bueno	3	75%
Bueno	1	25%
Regular	0	0%
Malo	0	0%
<b>Total</b>	<b>4</b>	<b>100%</b>

*Tabla 52: Patrón de diseño MVC.*

*Fuente: El Autor.*



*Figura 68: Patrón de diseño MVC.*

*Fuente: El Autor.*

De los 4 expertos encuestados, el 100% tiene conocimiento sobre el patrón MVC, pareciéndole al 75% de ellos (3 expertos) muy bueno el empleo del patrón MVC, debido a que, estructurar los archivos en diferentes capas conlleva a tener una facilidad en la utilización y mantenimiento del código fuente y el manejo de archivos. El 25% que corresponde a 1 experto, le pareció buena la utilización del patrón MVC debido a que aún piensa en que la separación de tantas capas no es la más adecuada y debería de tal vez considerarse la unión por lo menos del modelo y el controlador.

### 3. ¿Qué opina sobre el desarrollo del Framework basado en el modelo SaaS elegido?

ESPUESTA	FRECUENCIA	PORCENTAJE
Muy Bueno	4	100%
Bueno	0	0%
Regular	0	0%
Malo	0	0%
<b>Total</b>	<b>4</b>	<b>100%</b>

Tabla 53: Utilización del modelo SaaS.

Fuente: El Autor.

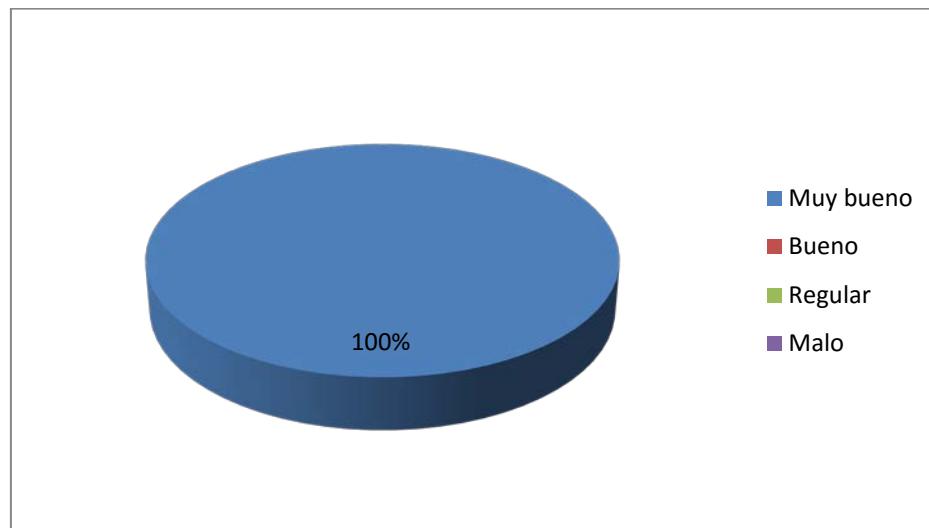


Figura 69: Utilización del modelo SaaS.

Fuente: El Autor.

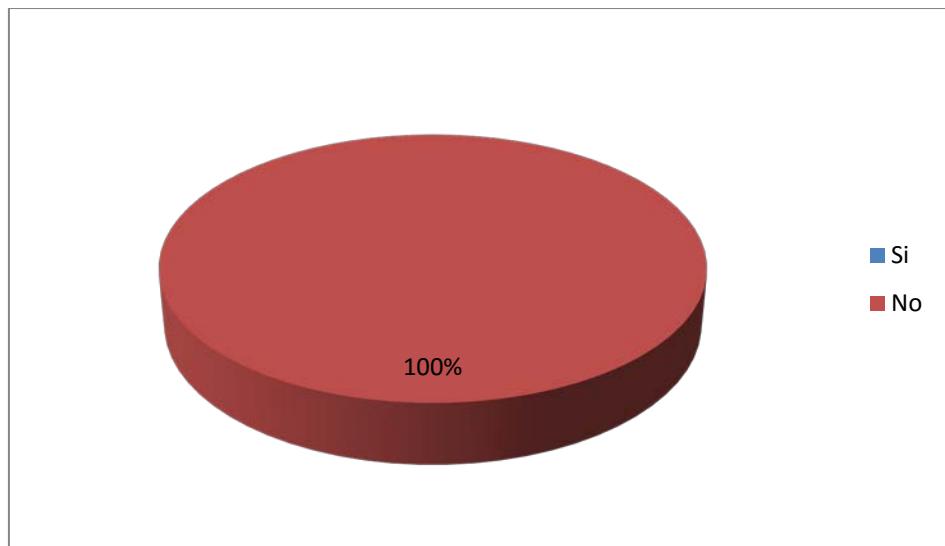
De los 4 expertos encuestados, el 75% no tenía conocimiento sobre el modelo SaaS utilizado y la arquitectura multi - tenencia que existe en tal modelo. Gracias a una previa explicación antes de comenzar el cuestionario es que, a los 4 expertos que representan el 100% les pareció muy buena opción la utilización de un modelo que contempla ofrecer un servicio de software a muchos clientes a través de una sola instancia en la aplicación y aún mejor que todo ya se encuentra definido en el Framework desarrollado en base a una arquitectura como la multi - tenencia.

**4. Para crear o instanciar una aplicación a través del Framework, ¿realiza muchos pasos?**

RESPUESTA	FRECUENCIA	PORCENTAJE
Si	0	0%
No	4	100%
<b>Total</b>	<b>4</b>	<b>100%</b>

*Tabla 54: Creación o instancia de una aplicación.*

*Fuente: El Autor.*



*Figura 70: Creación o instancia de una aplicación.*

*Fuente: El Autor.*

A los 4 expertos encuestados que representan el 100%, respondieron que NO realizan muchos pasos en crear una aplicación, debido a la capacidad del Framework en crear el proyecto estructurando todos sus componentes y aún mejor, otorgando la utilidad de que el subsistema de información no ha sido instanciado desde cero sino que ya cuenta con módulos que siempre se reutilizan en las aplicaciones como: acceso a través de un formulario (login del sistema), administración de usuarios, cambio en contraseñas, entre otros. Algunos recomiendan la implementación de más módulos con frecuencia en su usabilidad.

## 5. ¿Cómo considera la curva de aprendizaje del Framework?

ESPUESTA	FRECUENCIA	PORCENTAJE
Alta	0	0%
Regular	1	25%
Baja	3	75%
<b>Total</b>	<b>4</b>	<b>100%</b>

Tabla 55: Curva de aprendizaje del Framework.

Fuente: El Autor.

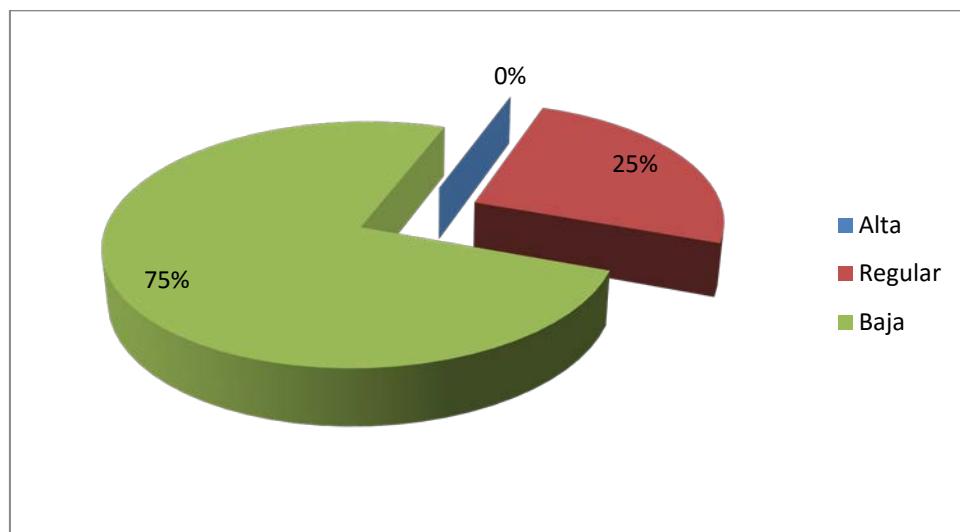


Figura 71: Curva de aprendizaje del Framework.

Fuente: El Autor.

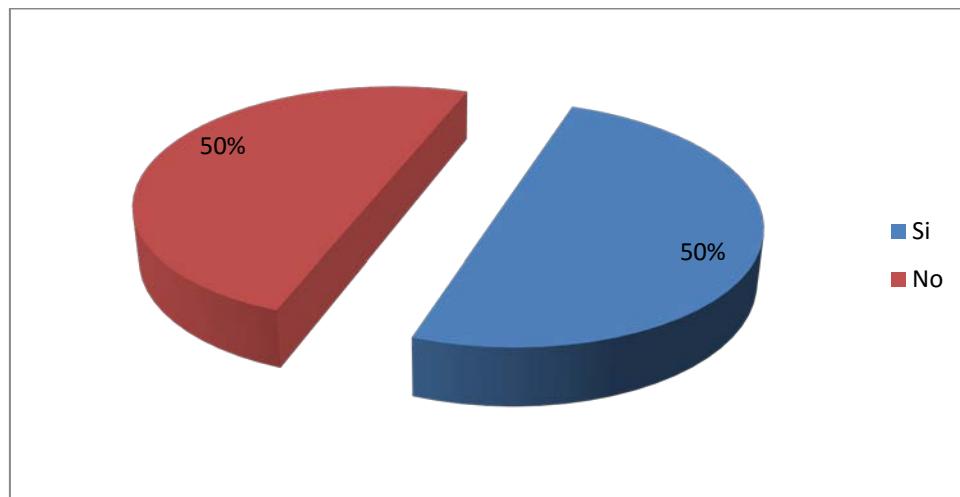
De los 4 expertos encuestados, 3 personas que representan el 75%, respondieron que la curva de aprendizaje del Framework es baja, ya que se cuenta con un Manual Técnico capaz de resolver interrogantes o dificultades en la utilización e instancia del Framework, 1 persona que representa el 25% piensa que es regular debido a que debería existir la opción de conocer la utilización del Framework a través de videos.

**6. ¿Le costó mucho tiempo comenzar a crear software en un sistema de información a través del Framework?**

RESPUESTA	FRECUENCIA	PORCENTAJE
Si	2	50%
No	2	50%
<b>Total</b>	<b>4</b>	<b>100%</b>

*Tabla 56: Construcción de software en un sistema de información.*

*Fuente: El Autor.*



*Figura 72: Construcción de software en un sistema de información.*

*Fuente: El Autor.*

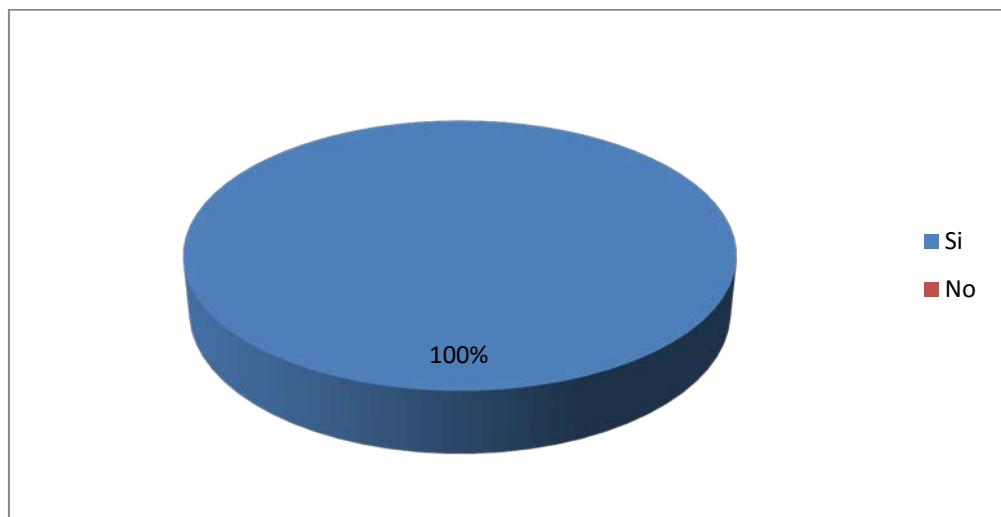
De los 4 expertos encuestados, 2 personas que representan el 50%, respondieron que no les costó mucho comenzar a crear software en la aplicación instanciada, ya que, se cuenta con el manual técnico correspondiente y la organización e integración de carpetas, archivos, librerías y clases en la estructura del Framework, 2 personas que representan el otro 50% comentaron que si les costó crear software debido a que no se cuenta con el control de las opciones de menú asignados a los usuarios.

**7. ¿Cree Ud. que el Framework le proveerá mayor agilidad en la construcción de software en Sistemas de Información orientados a la prestación de servicios (modelo SaaS)?**

RESPUESTA	FRECUENCIA	PORCENTAJE
Si	4	100%
No	0	0%
<b>Total</b>	<b>4</b>	<b>100%</b>

*Tabla 57: Agilidad en construir software.*

*Fuente: El Autor.*



*Figura 73: Agilidad en construir software.*

*Fuente: El Autor.*

De los 4 expertos encuestados, 4 personas que representan el 100%, respondieron que el Framework si permitirá agilizar la construcción de software en Sistemas de Información, debido a la estructura y propiedad de instanciar un software con características definidas, por ende, no comenzar desde cero el proceso de construcción de software.

## 8. ¿Cuántas veces se puede crear software a través del Framework?

RESPUESTA	FRECUENCIA	PORCENTAJE
Muchas veces	4	100%
Pocas veces	0	0%
Ninguna	0	0%
<b>Total</b>	<b>4</b>	<b>100%</b>

Tabla 58: Número de instancias del Framework.

Fuente: El Autor.

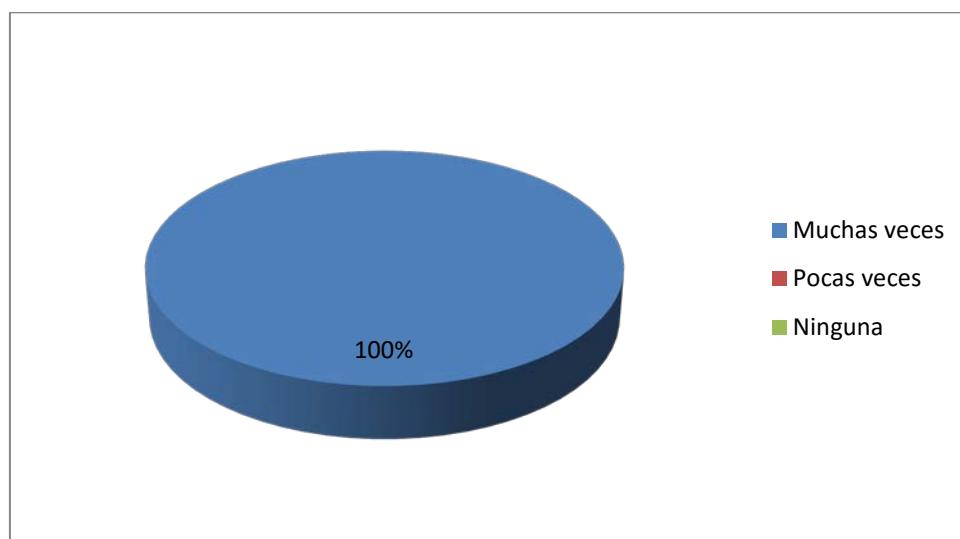


Figura 74: Número de instancias del Framework.

Fuente: El Autor.

De los 4 expertos encuestados, 4 personas que representan el 100%, respondieron que la potencialidad del Framework es su utilización de un portal web y que a través de tal portal se puede crear o instanciar muchas veces a diferentes sistemas de información teniendo una instancia no en vacío, sino que, al contrario, el Framework crea un sistema con módulos ya asignados y controlados para que los usuarios de cada tenant pueda utilizarlos, por lo tanto, si el Framework instancia muchas veces a sistemas de información, por ende, muchas veces se puede crear software a través del Framework, considerando a tal como reutilizable.

## 9. ¿Considera que la estructura del Framework es?

RESPUESTA	FRECUENCIA	PORCENTAJE
Muy Buena	2	50%
Buena	1	25%
Regular	1	25%
Mala	0	0%
<b>Total</b>	<b>4</b>	<b>100%</b>

Tabla 59: Estructura del Framework.

Fuente: El Autor.

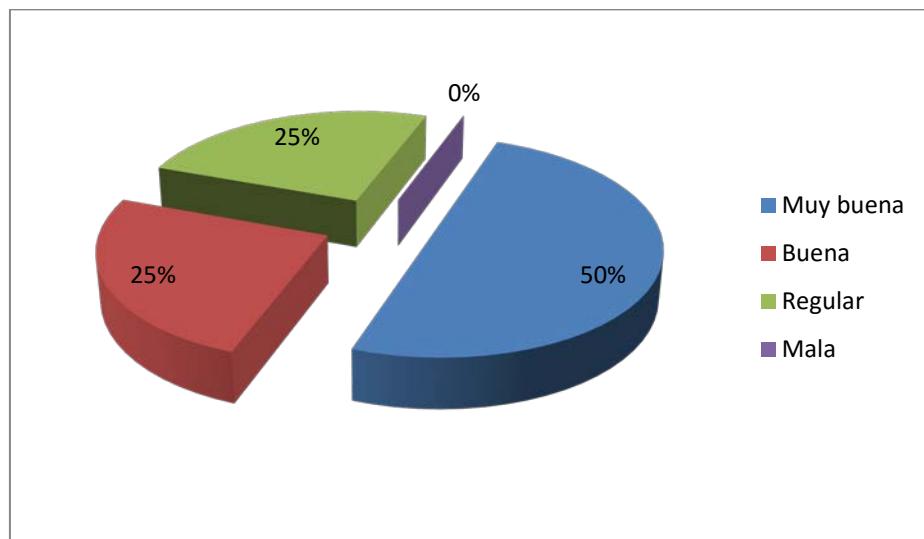


Figura 75: Estructura del Framework.

Fuente: El Autor.

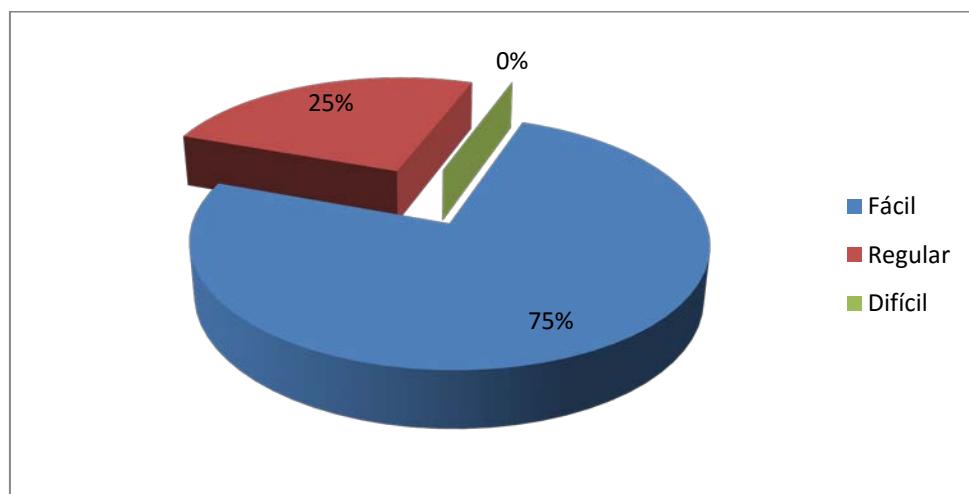
De los 4 expertos encuestados, 2 personas que representan el 50% les pareció que es muy buena la estructura del Framework siendo fácil el acceso a sus componentes, carpetas y archivos, 1 persona que representa el 25% le pareció buena la estructura debido a que podría mejorar la organización de las librerías y la última persona que representa el 25% comentó que le parecía regular la estructura del Framework ya que debería de separarse en carpetas a los controladores de las vistas para así tener mayor reutilización de tales controladores.

**10. El dar mantenimiento al software instanciado por el Framework, considera que es:**

RESPUESTA	FRECUENCIA	PORCENTAJE
Fácil	3	75%
Regular	1	25%
Difícil	0	0%
<b>Total</b>	<b>4</b>	<b>100%</b>

*Tabla 60: Mantenimiento del Framework.*

*Fuente: El Autor.*



*Figura 76: Mantenimiento del Framework.*

*Fuente: El Autor.*

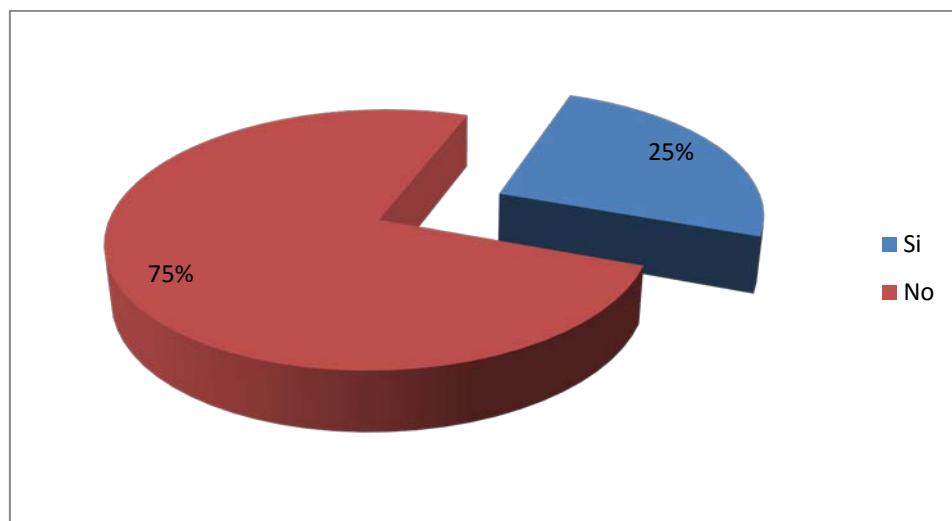
De los 4 expertos encuestados, 3 personas que representan el 75% les pareció que dar mantenimiento al software es fácil debido a los recursos con que cuenta el Framework en cuanto a su estructura, fácil acceso a los archivos, facilidad en la programación, entre otros; 1 persona que representa el 25% le pareció regular ya que, se debería contar con mayor cantidad de clases teniendo la posibilidad de controlar y aumentar la gestión de reutilización del software.

**11. ¿Alguna vez escuchó sobre el término SaaS (Software as a service)?**

RESPUESTA	FRECUENCIA	PORCENTAJE
Si	1	25%
No	3	75%
<b>Total</b>	<b>4</b>	<b>100%</b>

*Tabla 61: Modelo SaaS (Software as a service).*

*Fuente: El Autor.*



*Figura 77: Modelo SaaS (Software as a Service).*

*Fuente: El Autor.*

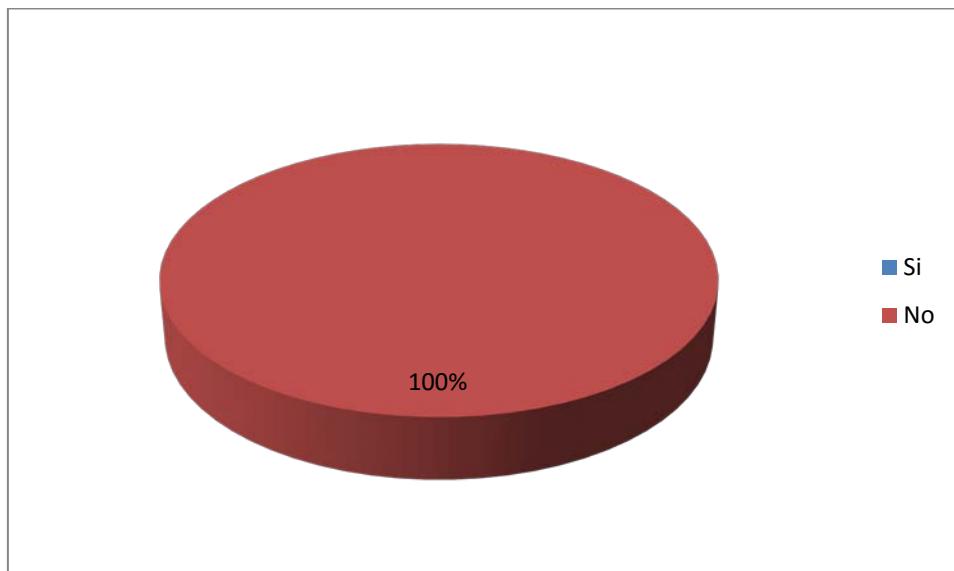
De los 4 expertos encuestados, 3 personas que representan el 75% no escucharon sobre el término SaaS hasta la previa explicación de la presentación y utilización del Framework, pareciéndoles muy buena alternativa en la construcción de software en la prestación de servicios con una sola instancia, disminuyendo el costo de mantenimiento de la aplicación; 1 persona que representa el 25% comentó que si escuchó sobre tal término ya que realizó la utilización de salesforce.com en una demo para su cliente.

**12. ¿Conoce Ud. algún Framework que ayude a construir aplicaciones web orientadas a la prestación de servicios?**

RESPUESTA	FRECUENCIA	PORCENTAJE
Si	0	0%
No	4	100%
<b>Total</b>	<b>4</b>	<b>100%</b>

*Tabla 62: Opinión sobre Frameworks.*

*Fuente: El Autor.*



*Figura 78: Opinión sobre Frameworks.*

*Fuente: El Autor.*

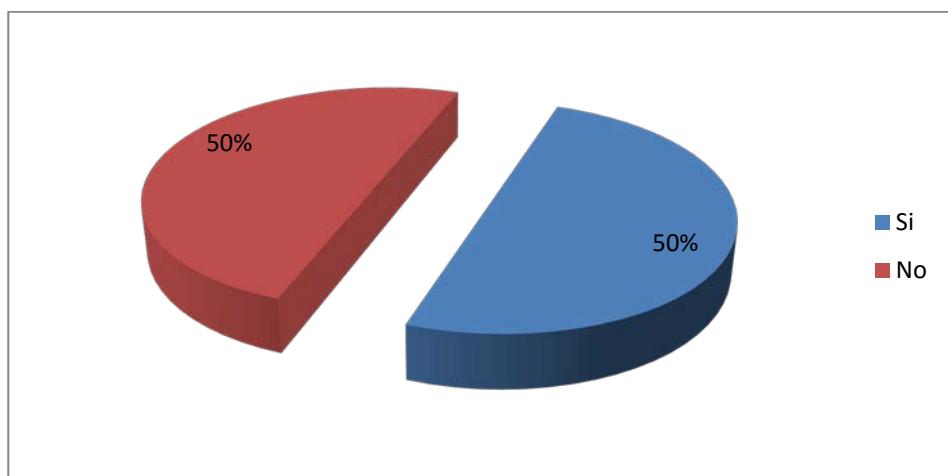
De los 4 expertos encuestados, 4 personas que representan el 100% no conocen Framework alguno que esté orientado a construir software en el modelo SaaS y que la mayoría de marcos de trabajo, solamente se orientan a construir aplicaciones web para un solo cliente, por ende, si se quiere construir software de tipo multi propietario, se debe adecuar los Framework que se encuentran disponibles en el mercado, lo que conlleva a tener un retraso en la creación de software orientado a tal modelo.

**13. ¿Cree Ud. que el software como servicio incremente?**

RESPUESTA	FRECUENCIA	PORCENTAJE
Si	2	50%
No	2	50%
<b>Total</b>	<b>4</b>	<b>100%</b>

*Tabla 63: Incremento del Software como Servicio.*

*Fuente: El Autor.*



*Figura 79: Incremento del Software como Servicio.*

*Fuente: El Autor.*

De los 4 expertos encuestados, 2 personas que representan el 50% piensan que si se incrementará el SaaS, lo cual conlleva a requerir este tipo de Framework en la construcción de software orientados al modelo SaaS, sin embargo, 2 personas que representan el otro 50% comentan que la resistencia al cambio y la poca confiabilidad en otorgar los datos a una empresa en prestar el servicio, serán los criterios que retrasarán la creación de sistemas de información orientados al modelo SaaS, por ende, se seguirán utilizando marcos de trabajo actuales.

## CONCLUSIONES

**PRIMERA:** Se logró desarrollar un Framework para construir software en Sistemas de Información orientados a la prestación de servicios, utilizando características del modelo de distribución SaaS, patrones de diseño y tecnologías web, con la finalidad de agilizar la construcción y aumentar la modularidad en el software.

**SEGUNDA:** Se llegó a implementar el patrón de diseño MVC (Modelo – Vista - Controlador) y la característica de tenencia múltiple (multitenancy en inglés) como principales núcleos del Framework; lo cual, ayudará a construir software de una manera modular, ejecutándose dicho software en una sola instancia con la capacidad de servir a múltiples clientes u organizaciones.

**TERCERA:** Se diseñó la estructura de carpetas, librerías, clases y estándares de programación orientados a obtener una buena disposición y orden de los recursos del Framework, lo cual nos lleva a tener mayor facilidad y/o flexibilidad en el mantenimiento del software.

**CUARTA:** Se llevó a cabo el diseño de una Arquitectura que brinda y/o muestra las características relevantes del Framework, lo que conlleva a que se pueda interactuar con el código fuente del Framework.

**QUINTA:** De acuerdo a las pruebas realizadas, resultados obtenidos y opinión de expertos, el Framework es fácil de usar, posee baja curva de aprendizaje y reutilización respecto a instanciar y construir software en sistemas de información orientados a la prestación de servicios, por consiguiente, puede ser utilizado por otros desarrolladores, logrando obtener una nueva concepción y opción en el desarrollo de Sistemas de Información orientados a la prestación de servicios.

**SEXTA:** Se logró demostrar la construcción de software en un subsistema de control de inventarios dentro de un sistema de información logístico orientado a la prestación de servicios, a través del Framework; siendo importante tal demostración, debido a que puede aplicarse la instancia del software en otros sistemas de información.

## **RECOMENDACIONES**

**PRIMERA:** Optimizar la concurrencia y rendimiento en el acceso a la base de datos bajo la Arquitectura de Multitenencia (en inglés, Multitenancy), debido a que solamente se ha realizado pruebas con una cantidad de 1000 usuarios virtuales y teniendo en cuenta que, el sistema podría albergar una mayor cantidad de accesos, por ende, se debe verificar su funcionamiento y/o comportamiento con una mayor carga.

**SEGUNDA:** Mejorar el proceso de instancia de Sistemas de Información a través del Framework, utilizando la gestión de diferentes gestores de base de datos, así mismo, mejorar la parte visual, dando a los clientes la funcionalidad de cambiar dinámicamente la forma de visualización del Sistema de Información.

**TERCERA:** Analizar, diseñar e implementar una estructura de seguridad en el acceso a la información en las aplicaciones SaaS.

**CUARTA:** Proponer que, se implemente en las micro, pequeñas y medianas empresas, aplicaciones bajo el modelo de distribución SaaS, incrementando y mejorando el servicio de software bajo demanda.

## BIBLIOGRAFÍA

- ÁLVAREZ CONTRERAS, V. A. Desarrollo de Software Orientado a la Prestación de Servicios (modelo SaaS). 2009. 130 p. Tesis (Título de Ingeniero en Ciencias y Sistemas) - Facultad de Ingeniería, Escuela de Ingeniería en Ciencias y Sistemas, Universidad de San Carlos de Guatemala, Guatemala, 2009.
- APACHE JMeter. The Apache Software Foundation. 2015. Disponible en: <http://jmeter.apache.org/index.html>. Consulta: 10 de diciembre. 2015.
- BERNAL, A. Descomposición de Procesos de Negocio en Servicios SOA (Paradigma de orientación a servicios). Disponible en: <http://www.baware.com.mx/es/index.php/blog/54-descomposicion-de-procesos-de-negocio-en-servicios-soa-paradigma-de-orientacion-a-servicios>. Consulta: 12 de mayo. 2016
- CakePHP CookBook - Introducción a CakePHP. 2015. Disponible en: <http://book.cakephp.org/2.0/es/cakephp-overview/understanding-model-view-controller.html>. Consulta: 10 de junio. 2015.
- CA'ZORZI, A.; Las TIC en el desarrollo de la PyME. 2011. Algunas experiencias de América Latina. Disponible en: [http://www.oitcinterfor.org/sites/default/files/file\\_publicacion/tic\\_pyme.pdf](http://www.oitcinterfor.org/sites/default/files/file_publicacion/tic_pyme.pdf). Consulta: 26 de abril. 2015.
- FAEDPYME, Fundación para el Análisis Estratégico y Desarrollo de la Pequeña y Mediana Empresa. 2014. Disponible en: <http://www.gaedpyme.upct.es/index.php>. Consulta: 04 de junio. 2015.
- GARIMELLA K.; LEES M.; WILLIAMS B. Introducción a BPM para DUMMIES; Edición Especial de Software AG. Publicado por Wiley Publishing. 2, Inc. 111 River Street Hokoben, NJ 07030-5774. 2008. Disponible en: [http://www.managementensalud.com.ar/ebooks/Introduccion\\_a\\_BPM\\_para\\_Dummies.pdf](http://www.managementensalud.com.ar/ebooks/Introduccion_a_BPM_para_Dummies.pdf). Consulta: 12 de mayo. 2016.

- GUTIÉRREZ, J. ¿Qué es un Framework web? 2009. Disponible en: [http://www.lsi.us.es/~javierj/investigacion\\_ficheros/Framework.pdf](http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf). Consulta: 10 de abril. 2015.
- GRAVITAR. Información sin Límites. 2010. Disponible en: <http://gravitar.biz/tecnologia-negocios/arquitecturas-multi-tenant/>. Consulta: 12 de junio. 2015.
- KOZIOLEK H. Towards an Architectural Style for Multi-Tenant Software Applications. ABB Corporate Research, Industrial Software Systems, Wallstadter Str. 59, 68526 Ladenburg, Alemania. 2010.
- KREBS R.; MOMM C.; KOUNEV S. Architectural Concerns in Multi - Tenant SaaS Applications. SAP AG, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany, Alemania, 2013.
- LIZÁRRAGA UGARTE, E. F. Propuesta de un Framework para la implementación de sistemas de información híbridos utilizando tecnologías web. 2011. 250 p. Tesis (Título de Ingeniero de Sistemas) - Facultad de Ciencias e Ingenierías Físicas y Formales, Programa Profesional de Ingeniería de Sistemas, Universidad Católica de Santa María, Arequipa - Perú, 2011.
- MARROQUÍN VILLATORO, J. A. SAAS PARA ASEGURADORAS. 2010. 119 p. Tesis (Título de Ingeniero en Ciencias y Sistemas) - Facultad de Ingeniería, Escuela de Ingeniería en Ciencias y Sistemas, Universidad de San Carlos de Guatemala, Guatemala, 2010.
- MICHETTI MANDUCA, A. Customização de interfaces web para clientes de software como serviço multitenant. 2014. 67 p. Tesis (Título de Magister en Ciencias) - Instituto de Ciências Matemáticas e de Computação, Ciências de Computação e Matemática Computacional, USP, São Carlos, 2014.
- MORENO, J. C.; Saasmania. 2008. Disponible en: <http://www.saasmania.com>. Consulta: 06 de abril. 2014.

- MYPES podrían reducir sus costos si usaran más tecnologías de información y comunicación. Disponible en: <http://gestion.pe/mercados/mypes-podrian-reducir-sus-costos-si-usaran-mas-tecnologias-informacion-y-comunicacion-2093444>. Consulta: 04 de junio. 2015.
- NIEDRĪTIS, A. Architecture for Multi-Tenant Adaptive Information Systems. 2013. Tesis Doctoral (Grado Doctoral en Ciencias de la Computación) - Facultad de Computación, Universidad de Latvia, Riga - Letonia, 2013.
- PERÚ. Decreto Legislativo Nº 1086, 28 de junio de 2008. Aprueba la Ley de Promoción de la Competitividad, Formalización, y Desarrollo de la Micro y Pequeña Empresa y del Acceso al Empleo Decente. Perú, Título I, Artículo Nº 2, 2008.
- PHP. The php group. Disponible en: <http://docs.php.net/manual/es/intro-whatis.php>. Consulta: 02 de Julio. 2015.
- RIEHLE, D. Framework Design, A Role Modeling Approach. Disertación para optar el grado de Doctor en Ciencias Técnicas. Instituto de Tecnología en Zúrich. Universidad de Hamburgo, Alemania. 2000.
- RODRÍGUEZ RODRÍGUEZ, P. A.; Rediseño del Modelo de Negocios del Datacenter de Telefónica Empresas en función de prácticas ITIL. Proyecto de grado para optar al grado de Magíster en Ingeniería de Negocios con Tecnologías de Información. Santiago de Chile. Chile. 2007. Disponible en: [http://www.thesis.uchile.cl/thesis/uchile/2007/rodriguez\\_pr/sources/rodriguez\\_pr.pdf](http://www.thesis.uchile.cl/thesis/uchile/2007/rodriguez_pr/sources/rodriguez_pr.pdf) Consulta: 08 de agosto. 2015.
- SALEH E.; SHAABANI N.; MEINEL C. A Framework for Migrating Traditional Web Applications into Multi-Tenant SaaS. The second International Conference on Advanced Communications and Computation. University of Potsdam, Alemania, 2012. Disponible en: [http://www.thinkmind.org/download.php?articleid=infocomp\\_2012\\_6\\_10\\_10161](http://www.thinkmind.org/download.php?articleid=infocomp_2012_6_10_10161). Consulta: 20 de abril del 2015.

- SUNAT Superintendencia Nacional de Administración Tributaria. Disponible en: [http://www.guiatributaria.sunat.gob.pe/index.php?option=com\\_content&view=article&id=519:01-condiciones-para-ser-micro-empresa-y-para-ser-pequena-empresa&catid=83:mypes&Itemid=134](http://www.guiatributaria.sunat.gob.pe/index.php?option=com_content&view=article&id=519:01-condiciones-para-ser-micro-empresa-y-para-ser-pequena-empresa&catid=83:mypes&Itemid=134). Consulta: 04 de junio. 2015.
- PEREA PATRICIA. Tres modelos de servicio cloud: SaaS, PaaS, IaaS. Disponible en: <http://www.aunclicdelastic.com/test/tres-modelos-de-servicio-cloud-saas-paas-iaas/>. Consulta: 25 de Setiembre del 2015.
- URUEÑA A.; FERRARI A.; BLANCO D.; VALDECASA E. Cloud Computing Retos y Oportunidades. Observatorio Nacional de las Telecomunicaciones y de la SI, Ministerio de Industria, Energía y Turismo, Gobierno de España. 2012. Disponible en: [http://www.ontsi.red.es/ontsi/sites/default/files/1-\\_estudio\\_cloud\\_computing\\_retos\\_y\\_oportunidades\\_vdef.pdf](http://www.ontsi.red.es/ontsi/sites/default/files/1-_estudio_cloud_computing_retos_y_oportunidades_vdef.pdf). Consulta: 18 de mayo. 2014.
- WIKI Framework. 2015. Disponible en: <https://es.wikipedia.org/wiki/Framework>. Consulta: 05 de abril. 2015.
- WIKIPEDIA Computación en la nube. 2015. Disponible en: [https://es.wikipedia.org/wiki/Computaci%C3%B3n\\_en\\_la\\_nube#Historia](https://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube#Historia). Consulta: 23 de Julio. 2015.
- YAMAKAWA, P.; DEL CASTILLO, C.; BALDEÓN, J.; ESPINOZA, L.M.; GRANDA, J.C.; VEGA, L. Modelo Tecnológico de integración de servicios para la mype peruana. Lima: Universidad ESAN, 2010. 165 p. Impreso por Editorial Cordillera S.A.C.

# **ANEXOS**

## **ANEXO A**

### **MANUAL TÉCNICO DEL FRAMEWORK**

#### **1. INTRODUCCIÓN**

El Framework diseñado e implementado por el autor es un conjunto de herramientas y estándares que son necesarios para los desarrolladores que van a iniciar a construir un Sistema de Información. El Manual Técnico supone que el desarrollador posee conocimientos de PHP, programación orientada a objetos (POO) y tecnologías web. Para la utilización del Framework es necesario tener la carpeta "LizarFrame" en un servidor web y luego comenzar a utilizar el aplicativo web que trae consigo. El Framework puede ser descargado de la siguiente ruta:

#### **2. ESTRUCTURA DE COMPONENTES**

Las carpetas propias del Framework llevarán al inicio un punto (Ejemplo: .classes), para indicar que son propias del Framework y que no son totalmente usadas por los desarrolladores, sino que son prácticamente re- utilizables por ellos.



*Figura 80: Estructura de carpetas.*

*Fuente: El Autor.*

Nombre corto	Nombre completo	Descripción
.classes	Clases	Aquí se encuentran las clases del Framework, conteniendo los siguientes archivos: _generador.php, _menu.php, _modelo.php, _mssql.php, _mysql.php, _reporte.php, _usuario.php, _vista.php.
.css	Css - Hoja de Estilos	Carpeta que contiene las hojas de estilos.
.js	JavaScript	Carpeta que contiene archivos de tipo JavaScript.
.lib	Librerías	Carpeta en donde se encuentran los archivos de configuración, variables de entorno, conexión de archivos "php" y "tpl" y librerías como smarty, calendarios, etc.
.part	Parte	Carpeta que contiene a archivos que forman parte de otros archivos tpl.
.res	Recursos	Carpeta que contiene los recursos, como imágenes correspondientes a: la estructura del software a instanciar, aplicables a iconografía en general, aplicables a iconografía del menú principal, aplicables a la aplicación web.
.validaciones	Validaciones	Carpeta que contiene las validaciones en los componentes utilizados dentro de un formulario. Por ejemplo: cajas de texto, botones, entre otros.
pages	Páginas	Carpeta en donde el desarrollador comenzará a construir el software.
web	Web	Carpeta donde se localiza la aplicación web del Framework.

Tabla 64: Descripción de Carpetas del Framework.

Fuente: El Autor.

Dentro de cada carpeta, se tiene un conjunto de archivos y/o carpetas en donde se tiene un estándar propio para el nombramiento de estos. A continuación, se puntualiza la forma en que se debe nombrar:

- Las carpetas principales del Framework, deberán anteponerse un punto al comienzo.

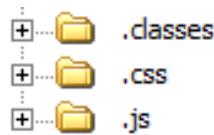


Figura 81: Estándar de Carpeta Principal.

Fuente: El Autor.

- La carpeta pages es la única que no contiene el punto al comienzo, debido a que será la carpeta donde el desarrollador comenzará a construir el software. Esta carpeta tiene su propia estructura, allí, se encontrarán otras como fnew, flist, etc., estas últimas servirán para colocar los archivos .php y .tpl tanto para realizar inserciones, listados, ediciones y otras operaciones que tiene un Sistema de Información.

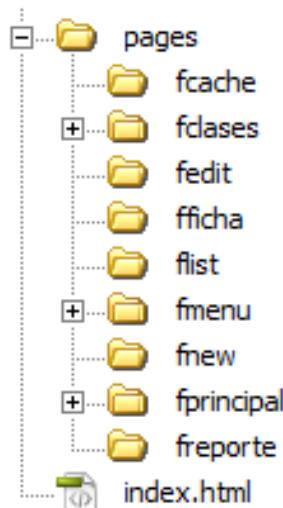


Figura 82: Estructura de carpeta pages.

Fuente: El Autor.

- Debido a que se sigue el patrón de diseño Modelo Vista Controlador, es que cada archivo debe ser independiente de los demás, teniendo de esta manera lo siguiente:

- a. **Modelo:** Los archivos que representan la información del software a construir son colocados en la carpeta fclases. Tales archivos en extensión php.
- b. **Vista:** Los archivos que representan la vista del software a construir son colocados en las carpetas que hacen alusión al tipo de acción que realizará el usuario final. Tales archivos en extensión tpl.
- c. **Controlador:** Los archivos que representan el controlador del software a construir son colocados en las carpetas que hacen alusión al tipo de acción que realizará el usuario final. tales archivos en extensión php.

Los controladores y vistas deben crearse con el mismo nombre y colocarlos en la misma ruta.

A continuación, se muestra una tabla en referencia a la Figura 82.

Nombre corto	Nombre completo	Descripción
fclases	Carpeta de Clases.	Carpeta que contiene los modelos del software a construir.
fedit	Carpeta de Ediciones.	Carpeta que contiene a archivos relacionados con la edición.
fficha	Carpeta de Fichas.	Carpeta que contiene a archivos que son llamados fichas o pequeñas ventanas que se asemejan a fichas (ventanas emergentes).
flist	Carpeta de Listado.	Carpeta que contiene a archivos relacionados con el listado de información.
fmenu	Carpeta del Menú.	Carpeta que contiene a archivos que se encargan de construir el menú principal del sistema a construir.
fnew	Carpeta de Nuevo.	Carpeta que contiene a archivos

		relacionados con el registro de datos.
fprincipal	Carpeta para la estructura principal.	Carpeta que contiene a archivos de la estructura principal del Sistema de Información como: posición del menú principal, posición de herramientas, etc.
freporte	Carpeta para Reportes.	Carpeta que contiene los reportes del Sistema.

Tabla 65: Definición de carpetas en pages.

Fuente: El Autor.

- La carpeta que se encuentra en el interior de una carpeta principal, deberá anteponerse un guión bajo al comienzo.
- El archivo que se encuentra en el interior de una carpeta principal deberá anteponerse un guión bajo al comienzo (se exceptúa a nombre de imágenes como por ejemplo).

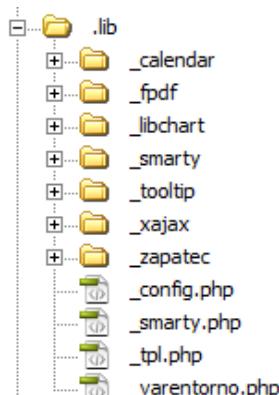


Figura 83: Estándar de Carpeta - Archivo.

Fuente: El Autor.

- Al dar un nombre a una carpeta o archivo, deberá comenzar la primera palabra de su nombre con minúscula en caso que el archivo o carpeta sólo tenga un nombre. En el caso que el nombre sea compuesto de dos o más palabras, la primera letra de la segunda palabra deberá ir con mayúscula y si tiene mayor cantidad de palabras, pues se seguirá con la misma operación. Para este punto tenemos una excepción para

nombrar a archivos que son los modelos para el software a construir, ya que, tendrán el mismo nombre de las tablas de la base de datos adaptada.

### 3. DESCRIPCIÓN DE COMPONENTES

Entre archivos que se ejecutan tanto en el lado del servidor como en el lado del cliente tenemos:

- **CLASES**

- **Generador:** Clase encargada de generar modelos, vistas, controladores, generar recursos, entre otras funciones.

- **Archivo contenedor:** \_generador.php

Función	Parámetro Entrada	Descripción
verificarServicios	-	Verifica los parámetros acerca del acceso a base de datos.
copiarCarpetas	origen, destino	Copia las carpetas del Framework, instanciando así el software a construir.
generarLibrerias	-	Crea el archivo _varentorno.php y _config.php, los cuales rigen el funcionamiento del Framework.
generarRecursos	-	Función principal que invoca a copiarCarpetas y generarLibrerias.
generarArchivos	-	Función principal que invoca a: generarVistaListar generarControladores generarModelos y generarVistaNuevo.
generarVistaListar	tablasUsuario, objeto modelo	Genera las vistas desde las tablas de la base de datos en cuanto a listado de registros se refiere. Extrae el esquema de las tablas creadas.

generarVistaNuevo	tablasUsuario, objeto modelo	Genera las vistas desde las tablas de la base de datos en cuanto a creación de registros se refiere. Extrae el esquema de las tablas creadas.
generarModelos	tablasUsuario, objeto modelo	Genera las clases a partir de las tablas de la base de datos a emplear.
generarControladores	tablasUsuario, objeto modelo	Genera los controladores para consultar data y mostrar las vistas creadas.

Tabla 66: Descripción de métodos de la clase Generador.

Fuente: El Autor.

- **Modelo:** Clase responsable de la recuperación de datos convirtiéndolos en conceptos significativos para la aplicación, así como su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos.

- **Archivo contenedor:** \_modelo.php

Función	Parámetro Entrada	Descripción
_construct	-	Constructor de la clase. Coloca los valores iniciales que serán utilizados para la posterior conexión a base de datos.
conectar	-	Conexión a la base de datos.
fetch	-	Función que retorna los registros de la base de datos en una matriz.
numeroRegistros	Resultado de la ejecución del procedimiento almacenado.	Función que retorna la cantidad de registros.
ejecutarProcedimiento	-	Función que ejecuta el procedimiento almacenado según el gestor de base de

		datos utilizado.
seleccionarTablasUsuario	-	Función que selecciona las tablas creadas por el usuario en su base de datos.
obtenerColumnasTabla	Nombre de tabla.	Función que obtiene las columnas de una tabla específica.

*Tabla 67: Descripción de métodos de la clase Modelo.*

*Fuente: El Autor.*

- **Vista:** Clase que se encarga de instanciar objetos de la interfaz del usuario, también provee la interfaz de login inicial.
- **Archivo contenedor:** \_vista.php

Función	Parámetro Entrada	Descripción
_construct	Nombre de la ventana, nombre del desarrollador, nombre de la aplicación.	Constructor de la clase. Coloca valores a la pantalla inicial de acceso al software a construir y crea el cache a utilizar.
alerta	Texto del mensaje.	Muestra mensaje de alerta.
referencia	Ruta relativa de la página.	Ir hacia una página específica.
confirmar	Texto del mensaje, referencia de la página.	Muestra mensaje de confirmación para el usuario.
load	Nombre del campo.	Coloca el foco del formulario en el campo que se quiera.
reload	-	Actualiza la página local.

openerReload	-	Actualiza la página padre que abrió a la página en uso.
cerrarVentana	-	Cerrar ventana actual o en uso.
cerrarVentanaPrincipal	-	Cierra la sesión del sistema.
cargarCalendario	-	Carga el calendario en la vista.
cargarValidacionText	-	Carga la validación para cajas de texto.
cargarValidacionArea		Carga la validación para áreas de texto.
cargarCss	Nombre de hoja de estilos.	Carga la hoja de estilos solicitada.
cargarJs	Nombre del archivo Javascript.	Carga el Javascript solicitado.
getPopup	-	Obtener el nombre del popup a mostrar.
getNombreVentana	-	Obtener el nombre de la ventana actual.
getNombreAplicacion	-	Obtener nombre de la aplicación.
getNombreDesarrollador	-	Obtener nombre del desarrollador.
setTituloVista	-	Colocar el título a una vista.
getTituloVista	-	Obtener el título de una vista.

Tabla 68: Descripción de métodos de la clase Vista.

Fuente: El Autor.

- **mssql:** Clase que se encarga de ejecutar los procedimientos almacenados correspondientes al Gestor de Base de Datos SQL SERVER. Cabe resaltar que, si el

usuario decide utilizar otro gestor de base de datos, tendrá que crear un archivo para implementar una nueva clase y su función para que ejecute los procedimientos almacenados según el gestor de base de datos.

- **Archivo contenedor:** \_mssql.php

Función	Parámetro Entrada	Descripción
proc	Objeto del modelo.	Método que ejecuta los procedimientos almacenados.

*Tabla 69: Descripción de métodos de la clase mssql.*

*Fuente: El Autor.*

- **Mmenu:** Clase encargada de manejar el menú principal del sistema a instanciar.

- **Archivo contenedor:** \_menu.php

Función	Parámetro Entrada	Descripción
dibujarMenu	-	Método que despliega las opciones de menú de la tabla Mmenu otorgada por el framework (ver más adelante).

*Tabla 70: Descripción de métodos de la clase Mmenu.*

*Fuente: El Autor.*

- **Musuario:** Clase encargada de manejar a los usuarios del sistema a instanciar. La presente clase es un modelo.

- **Archivo contenedor:** \_usuario.php

Función	Parámetro Entrada	Descripción
validarUsuario	-	Método que verificar si existe o no el usuario creado en la base de datos.
crearSesionUsuario	-	Método que crea una sesión de

		usuario.
crearUsuario	-	Método que registra a un usuario en la base de datos.
listarUsuario	-	Método que lista a los usuarios existentes en la base de datos.
obtenerUsuario	-	Método que obtiene a un usuario en específico de la base de datos.

Tabla 71: Descripción de métodos de la clase Musuario.

Fuente: El Autor.

- **Mreporte:** Clase encargada de manejar a los reportes creados por el usuario gracias a la herramienta Report Server (paquete de Business Intelligence de SQL SERVER).

- Archivo contenedor: \_reporte.php

Función	Parámetro Entrada	Descripción
verReporte	-	Función que visualiza al reporte creado en Reporting Service.

Tabla 72: Descripción de Función de la clase Mreporte.

Fuente: El Autor.

#### • HOJAS DE ESTILO

- **\_estilo.0.css:** Una de las hojas de estilo que ofrece el Framework. Recomendado para los Sistemas de Información a instanciar.
- **\_estilo.arbol.css:** Hoja de estilo destinado al menú principal del Sistema de Información.
- **\_estilo.web.css:** Hoja de Estilo para el aplicativo web del Framework.
- **\_estilo.web2.css:** Hoja de Estilo para las noticias que se desplazan, recomendado para el aplicativo web del Framework.

- **JAVASCRIPT**

- **\_arbol.js:** Archivo para dibujar y/o desplegar el menú principal que provee la base de datos con la que se trabaja.

Función dTree(objName)

- **\_clock.js:** Archivo que se encarga de mostrar la hora en formato HH:MM:SS, simulando un reloj.

Función mueveReloj()

- **\_date.js:** Archivo que se encarga de mostrar la fecha en formato dd-mm-yyyy.

Función datetime():

- **\_ventana.js:** Archivo en donde se trabajan con funciones para abrir ventanas, popup, etc.
- **\_cabecera.js:** Archivo en donde se encuentra la funcionalidad de botones que simulan a las herramientas de un Sistema tales como: Atrás - Adelante - Actualizar - Imprimir - Ayuda - Información - Salir.

- **Archivo contenedor:** \_cabecera.js

Función	Parámetro Entrada	Descripción
cambiar	Nombre de imagen sin efecto, nombre de imagen con efecto.	Realiza el efecto de sombra en una imagen.
refrescar	-	Función actualiza una página.
imprimir	-	Función que imprime una página.
ayudas	-	Abre una ventana en donde se muestra la ayuda del sistema de información. Tal ayuda es creada por el desarrollador

infos	-	Abre una ventana en donde se muestra la información del sistema de información. Tal información es creada por el desarrollador.
-------	---	---

Tabla 73: Descripción de Funciones de archivo \_cabecera.js.

Fuente: El Autor.



Figura 84: Herramientas del Sistema de Información.

Fuente: El Autor.

- **\_noticias.js:** Archivo que se encarga de mover el contenido de otros archivos html o imágenes como el que se encuentra en la página inicial de la aplicación web del Framework.
- **\_tabs.js:** Archivo que se encarga de crear pestañas para separar las interfaces del software a construir.
- **\_tooltip.js:** Archivo necesario para obtener ventanas emergentes. Funciona al situar o pulsar con el ratón sobre algún elemento gráfico, mostrando una ayuda adicional para informar al usuario de la finalidad del elemento sobre el que se encuentra.

## • LIBRERIAS

- **\_almanaque:** Librería para utilizar un calendario. Si se requiere visualizar un calendario se tiene que llamar a tres archivos .js y su correspondiente .css:  
 \_almanaque/\_calendar.js,  
 \_almanaque/lang/\_espanol.js  
 \_almanaque/\_setup.js  
 \_almanaque/\_calendar.css

Otra forma sería crear un objeto "Vista" y de allí llamar a la función cargarCalendario();

```
$vist = new Vista();
$vist -> cargarCalendario;
```

Para mayor detalle sobre la utilización del código fuente, ver el anexo B.

- **\_smarty:** Librería que actúa como motor de plantillas para PHP. Smarty separa el código PHP como lógica de negocios del código HTML como lógica de presentación. Es una de las principales librerías para el funcionamiento del Framework. La clase Smarty se encuentra en el archivo \_smarty/libs/Smarty.class.php.
- **\_tpl.php:** Librería que se encarga de realizar la conexión de páginas php con páginas tpl.
- **\_config.php:** Archivo de configuración, en donde se colocan los datos para la conexión con la base de datos. Aquí se encuentran las constantes utilizadas en el modelo del Framework y son las siguientes:

Constante	Descripción
C_SERVIDOR	Constante que representa el nombre de instancia de la base de datos a conectar.
C_USUARIO	Constante que representa el nombre del usuario con el cual se conecta a la base de datos
C_PASSWORD	Constante que representa la contraseña del usuario con el cual se conecta a la base de datos
C_BD	Constante que representa el nombre de la base de datos.
C_DVR	Constante que representa el gestor de base de datos, por defecto se encuentra en mssql.
C_CLASS	Constante que representa el nombre de la clase que contiene al método que ejecuta el procedimiento almacenado.

Tabla 74: Descripción de constantes para acceder a Base de Datos.

Fuente: El Autor.

- **\_xajax:** Librería que utiliza tecnología AJAX. Xajax designa qué funciones de código PHP se convierten en funciones AJAX. A continuación, se muestra cómo se debe utilizarla:
  - Hacer referencia al archivo \_xajax/xajax.inc.php, donde se encuentra la clase xajax.
  - Crear una instancia de la clase xajax.  
`$xajax = new xajax();`
  - Registrar la función que va a realizar la acción AJAX.  
`registerFunction( "miFuncion" );`
  - Le indicamos a Xajax que procese las peticiones:  
`$xajax -> processRequests();`
- **\_varentorno.php:** Librería que se encarga de administrar las constantes del Framework, las cuales como propias de este, son:

Constante	Descripción
CF_GENERADOR	Constante para ubicar el Generador del Framework.
C_MODELO	Constante para ubicar el Modelo del Framework.
CF_VISTA	Constante para ubicar la Vista del Framework.
CF_SMARTY	Constante del Framework donde se encuentra la clase Smarty para generar los archivos TPL y enviar todas las variables, valores, etc., desde el controlador hacia la vista.
CF_CONFIG	Constante del Framework donde se encuentra el archivo de configuración para conectarse con la base de datos.
CF_CLASES	Constante del Framework donde se encuentran los modelos creados por el Framework (Representación de tablas de la base de datos).
CF_USUARIO	Constante del Framework donde se encuentra la clase para manejar a los usuarios.
CF_MENU	Constante del Framework donde se encuentra la clase para manejar al menú.
CF_REPORTES	Constante del Framework donde se encuentra la clase para manejar a los reportes.

CF_SQLGEST	Constante del Framework donde se encuentra el gestor de base de datos a utilizar.
CF_TPL	Constante del Framework que se encarga de desplegar la vista para cierto controlador.
CF_LIBSESION	Constante del Framework que se encarga de controlar el tiempo de sesión definido por el desarrollador.
CF_LIB1	Constante del Framework que maneja la colación respecto a los caracteres entre la aplicación y la base de datos. Gestiona y controla los caracteres especiales.

*Tabla 75: Descripción de constantes principales del Framework.*

*Fuente: El Autor.*

Constante	Descripción
CD_NOMBRECARPETA	Nombre de la carpeta contenedora del Framework o Sistema de Información instanciado.
CD_NOMBREAPLICACION	Nombre de la aplicación.
CD_NOMBREDESARROLLADOR	Nombre del desarrollador.

*Tabla 76: Descripción de constantes creadas por el desarrollador.*

*Fuente: El Autor.*

Nota: Se debe tener un especial cuidado con las variables propias del Framework, ya que estas determinan el orden de las clases y archivos de configuración del Framework.

## ▪ VALIDACIONES

- **\_textField.js:** Valida los campos de texto.

Para utilizar el archivo de validación, se debe de incluir en la vista:

.validaciones/\_textField.js

.validaciones/\_textField.css

Otra forma sería, crear un objeto "Vista" y de allí llamar a la función cargarValidacionText() desde el controlador:

```
$vist = new Vista();
$vist -> cargarValidacionText();
```

Para mayor detalle sobre la utilización del código fuente, ver el anexo B.

- **\_textArea.js:** Valida las áreas de texto.

Para utilizar el archivo de validación, se debe de incluir:

```
.validaciones/_textArea.js
.validaciones/_textArea.css
```

Otra forma sería, crear un objeto "Vista" y de allí llamar a la función cargarValidacionArea() desde el controlador:

```
$vist = new Vista();
$vist -> cargarValidacionText();
```

Para mayor detalle sobre la utilización del código fuente, ver el anexo B.

- **WEB:** Existe una carpeta web, donde se encuentran los archivos diseñados para instanciar el software a construir. Estos archivos vienen siendo en su conjunto la página web para utilizar el Framework.

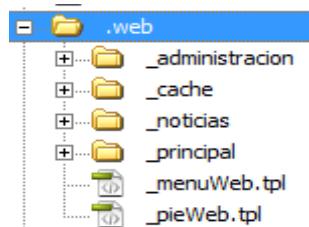


Figura 85: Carpetas - archivos que contienen a la página web.

Fuente: El Autor.

## 4. ESTÁNDAR EN CÓDIGO FUENTE

- Para declarar variables propias se debe utilizar las 3 primeras letras de la palabra a nombrar o puede ser también máximo 4 letras, por ejemplo:

Código de Usuario: codusu

Nombre de Producto: nomprod

- Los parámetros que se envían a los procedimientos almacenados de la base de datos en uso se deberá anteponerse la letra "p" de parámetro, por ejemplo:

p\_nomusu: parámetro de un nombre de usuario.

p\_nomprod: parámetro de un nombre de producto.

- Los nombres de variables del Framework deben ir en mayúsculas y con su nombre completo, por ejemplo:

CF\_GENERADOR: Constante del Framework para el Generador.

C\_SERVIDOR: Constante que tiene el nombre del servidor.

- Para declarar a funciones se podrá colocar el nombre completo. El nombre debe comenzar con minúscula y luego, a partir de la segunda palabra, el primer carácter deber colocarse con mayúscula, por ejemplo:

```
function generarArchivos{  
    //código a implementar.  
}
```

- Las clases creadas por el Framework tienen el mismo nombre que el de las tablas de la base de datos a utilizar, las cuales podrán ser cambiadas por el desarrollador, por ejemplo:

Mproducto: Clase de productos (proviene de un maestro en base de datos).

TcabeceraIngresoAlmacen: Clase de ingreso a almacén (proviene de una transacción en base de datos).

## 5. ESTÁNDAR EN BASE DE DATOS

- Tipo de Tabla:

- Maestro	: M
- Transacciones	: T
- Puente	: P

M	Nombre
---	--------

*En el caso de tratarse de Tablas Puente, la nomenclatura variaría ligeramente poniéndole dos nombres de tablas que serán las que están relacionadas:*

P	Nombre1	Nombre2
---	---------	---------

- Campos:

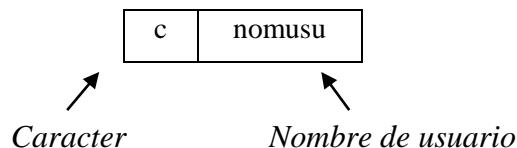
- Primero: El tipo de dato.

Tipo de Dato	Nombramiento
Entero	Se antepone una "i".
Decimal, Float	Se antepone una "f".
Date	Se antepone una "d".
Caracter	Se antepone una "c".
Booleano	Se antepone una "b".

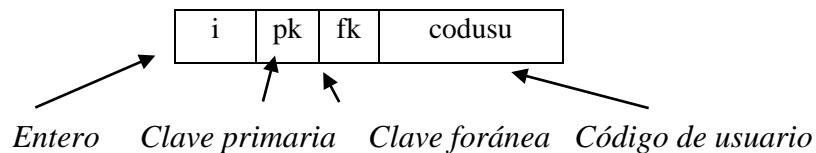
*Tabla 77: Nombrar a tipos de datos.*

*Fuente: El Autor.*

- Segundo: El nombre del campo, si es compuesto solo 3 letras por cada palabra (máximo 4).



- Tercero: Existen campos que vienen a ser claves primarias o foráneas.



- Vistas: v = Para especificar que es una vista.

v	Nombre
---	--------

- Procedimientos Almacenados: pa\_ = Para especificar que es un procedimiento.

pa_	Nombre
-----	--------

- Triggers: d = Para especificar que es un trigger o disparador.

- Momento:

Before: b

After: a

- Evento

Insert: i

Update: u

Delete: d:

d	Tabla	Momento	Evento
---	-------	---------	--------

## 6. UTILIZACIÓN DEL FRAMEWORK

Para comenzar a utilizar el Framework, su carpeta contenedora debe estar bien instalada en el servidor donde se va a comenzar a trabajar con él.

### 6.1. ACCESO AL FRAMEWORK

Se accede como a cualquier página web. Se debe colocar la siguiente dirección:

*http://localhost:8082/LizarFrame (para este caso el puerto del servidor Apache es el 8082).*

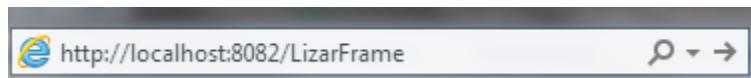


Figura 86: Acceso al Framework.

Fuente: El Autor.

La dirección nombrada, ingresará a la página inicial del Framework.

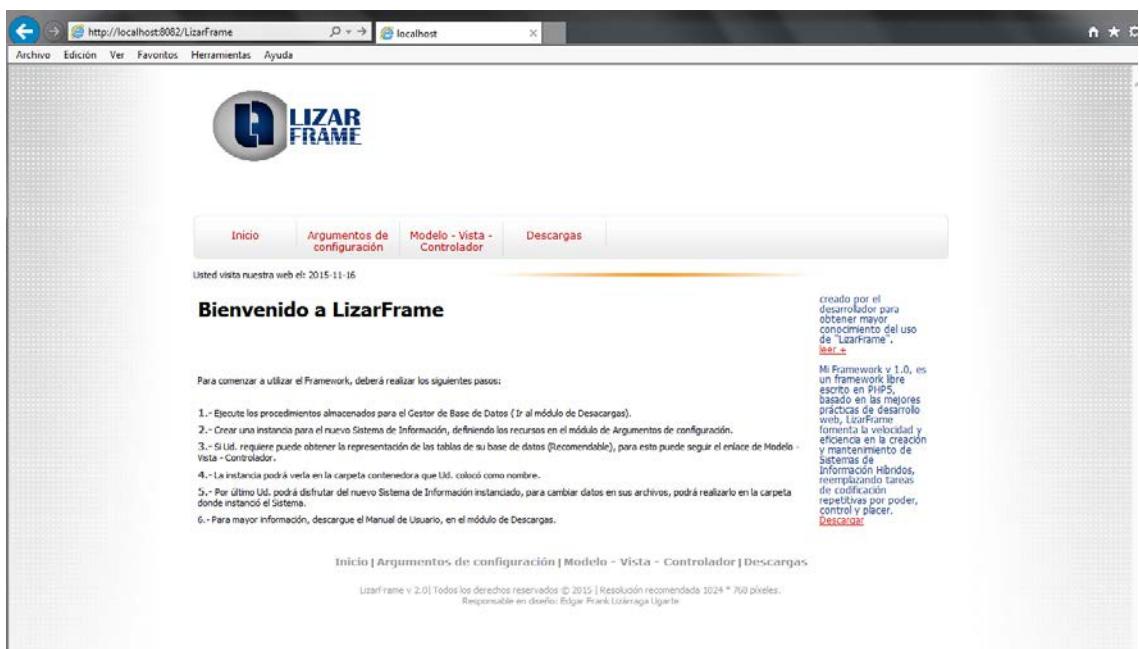


Figura 87: Aplicativo Web - Pantalla Inicial.

Fuente: El Autor.

## 6.2. INSTANCIAR EL SOFTWARE

- El aplicativo web del Framework es capaz de instanciar el software y así iniciar su construcción. En primer lugar, se debe llenar los Argumentos de configuración, donde existe un formulario que contiene los siguientes campos:
  - Carpeta contenedora: Nombre de la carpeta en donde el Framework instanciará al Sistema de Información.
  - Nombre de Aplicación
  - Nombre del Desarrollador
  - Servidor: Nombre del servidor de la base de datos.

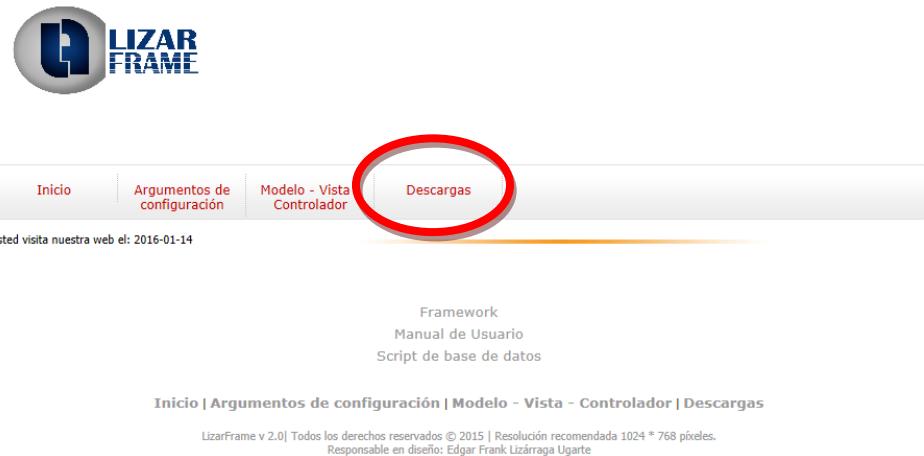
- Usuario: Nombre del usuario con el cual se conecta a la base de datos.
- Clave: Contraseña del usuario con la cual se conecta a la base de datos.
- Base de BD: Nombre de la base de datos con la que se va a trabajar.

The screenshot shows the LizarFrame web application interface. At the top, there is a navigation bar with tabs: 'Inicio', 'Argumentos de configuración' (which is highlighted with a red oval), 'Modelo - Vista - Controlador', and 'Descargas'. Below the navigation bar, there is a message: 'Usted visita nuestra web el: 2015-01-01'. The main content area is divided into two sections: 'Aplicación:' and 'Base de Datos:'. Under 'Aplicación:', there are fields for 'Carpeta contenedora:' (with a long input field), 'Nombre de Aplicación:' (empty input field), and 'Nombre del Desarrollador:' (empty input field). Under 'Base de Datos:', there are fields for 'Servidor:' (empty input field), 'Usuario:' (empty input field), 'Clave:' (empty input field), and 'Nombre de BD:' (empty input field). To the right of these fields is a 'Gestor de BD:' section with radio buttons for 'mssql' (selected) and 'mysql'. At the bottom of the form is a 'Generar Recursos' button. Below the form, there is a footer with links: 'Inicio | Argumentos de configuración | Modelo - Vista - Controlador | Descargas' and copyright information: 'LizarFrame v 2.0 | Todos los derechos reservados © 2015 | Resolución recomendada 1024 \* 768 píxeles. Responsable en diseño: Edgar Frank Lizárraga Ugarte'.

*Figura 88: Aplicativo Web - Argumentos de configuración.*

*Fuente: El Autor.*

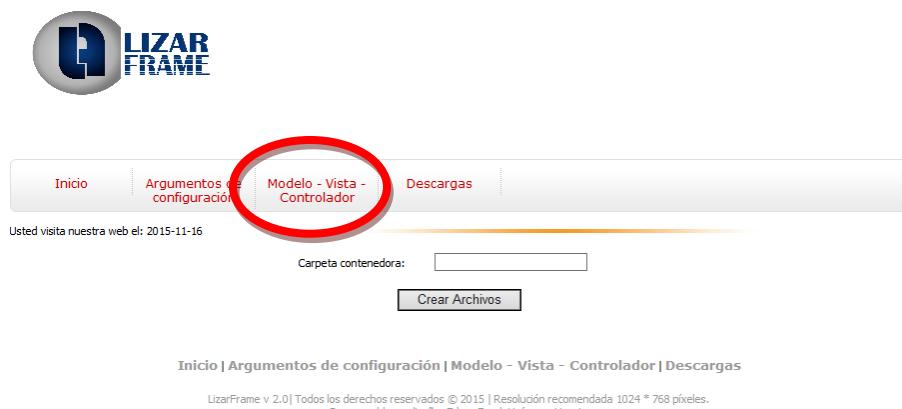
- Para llenar el anterior formulario es necesario adaptar la base de datos con la cual va a trabajar el software a instanciar y también ejecutar el script de base de datos que trae consigo el Framework, para descargarlos Ud. puede ir a la pestaña de "Descargas".



*Figura 89: Aplicativo Web - Descargas.*

*Fuente: El Autor.*

- Luego de definir los argumentos de configuración, se puede extraer los modelos desde la base de datos, para esto, deberá dirigirse al módulo de "Modelo - Vista - Controlador" y luego llenar el campo con el nombre de la Carpeta Contenedora, es decir el nombre de la carpeta que se ingresó en el formulario de la Figura 88.



*Figura 90: Aplicativo Web - Modelo Vista Controlador.*

*Fuente: El Autor.*

- Si se siguieron los pasos anteriores, entonces el software ya se encuentra instanciado, para visualizar el acceso, coloque la siguiente dirección en el navegador:

**http://localhost:8082/nombreCarpetaContedora**

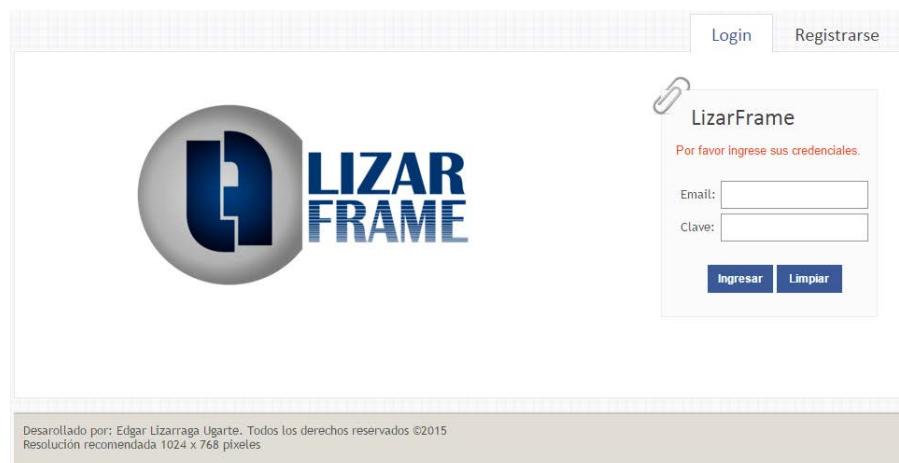


Figura 91: Login del software instanciado.

Fuente: El Autor.

- Para acceder al software instanciado, el usuario y contraseña para el usuario de perfil administrador son: admin@miempresa.com y admin.

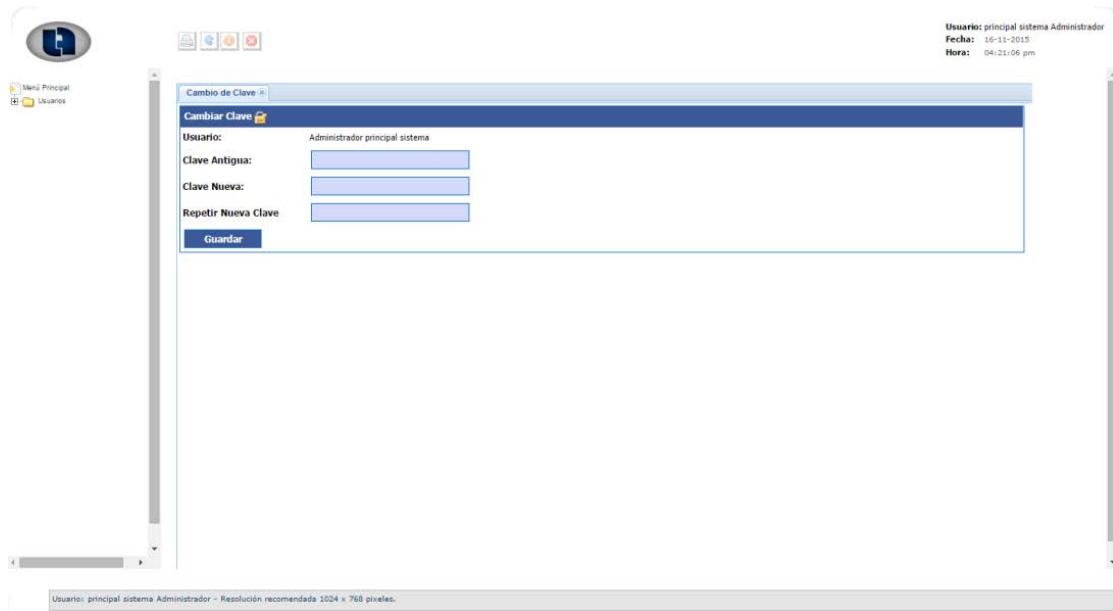


Figura 92: Estructura Principal del software instanciado.

Fuente: El Autor.

### 6.3. REGISTRO DE UN TENANT

El Framework está orientado a construir software orientado a la prestación de servicios, por lo que, es necesario la habilitación de un módulo para el registro de los usuarios "tenant" o propietarios que administrarán su propio servicio:

The screenshot shows a registration form titled "Ficha de Inscripción". At the top right are "Login" and "Registrarse" buttons. Below the title, a note says "Llene el siguiente formulario para tener acceso a LizarFrame.". The form contains six input fields with labels: "Apellido Paterno (\*):", "Apellido Materno (\*):", "Nombres (\*):", "Empresa (\*):", "Email (\*):", and "Clave (\* - En minúsculas)". A note at the bottom left says "(\*) Datos obligatorios a llenar." and a "Registrar" button is at the bottom right.

Figura 93: Registro de un tenant.

Fuente: El Autor.

### 6.4. CONSTRUCCIÓN DE SOFTWARE

Una vez instanciado el software, se deberá continuar con su construcción, teniendo como ayuda la estructura del Framework, para esto, se ha tomado como ejemplo la creación de Productos de un Sistema de Información logístico orientado a la prestación de servicios, construido por el autor de la presente tesis.

#### 6.3.1. Archivos principales del Framework

- Archivo: `_config.php`
- Dirección del Archivo: `nombreProyecto/.lib/_config.php`
- Descripción: Archivo que realiza la conexión con la base de datos. El significado de cada variable se puede revisar en la Tabla 74.

```

1 <?php
2 /* Archivo de configuración
3      @version 1.0
4      @license GNU Lesser General Public License, http://www.gnu.org/copyleft/lesser.html
5      @author Edgar Lizárraga Ugarte
6      @created 2012-01-01
7      @updated 2015-09-19
8      @link ./lib/_config.php
9      */?>
10 <?php
11 $serv='EFLU-PC';
12 $usu='sa';
13 $pwd='123456';
14 $bd='SISLOG';
15 $dvr='mssql';
16 $clas='MsSql';
17 define('C_SERVIDOR',$serv);
18 define('C_USUARIO', $usu);
19 define('C_PASSWORD', $pwd);
20 define('C_BD',$bd);
21 define('C_DVR',$dvr);
22 define('C_CLASS', $clas);
23 ?>
```

Figura 94: Código Fuente del archivo de Configuración.

Fuente: El Autor.

- Archivo: \_varentorno.php
- Dirección del Archivo: *nombreProyecto/.lib/\_varentorno.php*
- Descripción: Archivo que maneja las variables principales del Framework.

A continuación de `$_SERVER['DOCUMENT_ROOT']` siempre debe colocarse el nombre de la carpeta contenedora del software a construir. El significado de cada variable se puede revisar en la Tabla 75.

```

1 <?php
2 //Constante del framework donde se encuentra el controlador del framework.
3 define ('CF_GENERADOR,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.classes/_generador.php');
4 //Constante del framework donde se encuentra el modelo del framework.
5 define ('CF_MODELO,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.classes/_modelo.php');
6 //Constante del framework donde se encuentra la vista del framework.
7 define ('CF_VISTA,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.classes/_vista.php');
8 //Constante del framework donde se encuentra la clase para manejar a los usuarios.
9 define ('CF_USUARIO,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.classes/_usuario.php');
10 //Constante del framework donde se encuentra la clase para manejar al menú.
11 define ('CF_MENU,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.classes/_menu.php');
12 //Constante del framework donde se encuentra la clase para manejar a los reportes.
13 define ('CF_Reporte,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.classes/_reporte.php');
14 //Constante del framework donde se encuentra el gesto de base de datos a utilizar.
15 define ('CF_SQLGEST,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.classes/_mssql.php');
16 //Constante del framework donde se encuentra la clase Smarty para generar los archivos TPL y enviar todas la
17 define ('CF_SMARTY,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.lib/_smarty/libs/Smarty.class.php');
18 //Constante del framework donde se encuentra el archivo para conectarse con la base de datos.
19 define ('CF_CONFIG,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.lib/_config.php');
20 //Constante del framework que se encarga de desplegar la vista para cierto controlador.
21 define ('CF_TPL,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.lib/_tpl.php');
22 //Constante del framework donde se encuentran los modelos creados por el framework (Representación de tablas
23 define ('CF_CLASES,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/pages/fclases/');
24 /*Variables de entorno definidos por el desarrollador.*/
25 //Nombre de la carpeta contenedora.
26 define ('CD_NOMBRECARPETA,'www.mycloudstorage.com');
27 //Nombre de la aplicación.
28 define ('CD_NOMBREAPLICACION,'MyCloudStorage');
29 //Nombre del desarrollador, empresa, etc.
30 define ('CD_NOMBREDESARROLLADOR,'Edgar Lizarraga Ugarte');
31 define ('CD_TIEMPOSESION,'4800');
32 define ('CF_LIB1,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.lib/_lib.1.php');
33 define ('CF_DELDATATEMP,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.lib/_eliminarDataTemp.php');
34 define ('CF_AGREGARCEROS,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.lib/_agregarCeros.php');
35 define ('CF_REDONDEAR,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.lib/_redondear.php');
36 define ('CF_NUMALET,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.lib/_numeroALetras.php');
37 define ('CF_READER,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.lib/_excelReader/reader.php');
38 define ('CF_LIBSESION,$_SERVER['DOCUMENT_ROOT'].'./www.mycloudstorage.com/.lib/_sesion.php');
39 ?|

```

*Figura 95: Código Fuente del archivo de Variables de Entorno.*

*Fuente: El Autor.*

### 6.3.2. Creación de Productos

El Framework se basa en el patrón de diseño Modelo Vista Controlador; por lo tanto, se deben crear 3 archivos principales:

- **Vista de Productos:**

- Archivo: fnew.Mproducto.tpl

- Dirección del Archivo:

- nombreProyecto/pages/fnew/fnew.Mproducto.tpl

- Descripción: Diseño realizado para la captura del Nombre, Descripción, Proveedor, Stock Mínimo, Tipo y Estado del Producto.

Nombre:	<input type="text"/>	Campo requerido.
Descripción:	<input type="text"/>	Campo requerido.
Proveedor:	{include file=../../fpart/fpart.proveedor.tpl}	
Stock Mínimo:	1	Campo requerido. • Formato incorrecto.
Tipo de Producto:	{include file=../../fpart/fpart.tipo.producto.tpl}	
Estado:	{include file=../../fpart/fpart.estado.tpl}	
<input type="button" value="Guardar"/>		

Figura 96: Diseño de la Vista - Mproducto.

Fuente: El Autor.

La Figura 97 representa el código fuente del diseño de la Figura 96, donde se puede observar que no contiene código del lenguaje de programación PHP, lo que conlleva a que la vista pueda ser manejada por un diseñador de páginas web sin que tenga conocimientos en programación. En la parte inferior de la imagen se puede notar la utilización de "JavaScript" para realizar validaciones de los campos, teniendo tanto a Nombre, Descripción y stock mínimo como campos requeridos.

```

1 <html><head><meta http-equiv='Content-Type' content='text/html; charset=utf-8' />
2 <title>${titcab}</title>
3 </head>
4 <body ${fload} >
5 <form id='cform' name='cform' method='GET' action=''>
6 <div id='panel'>
7 <table width='100%' border='0' cellpadding='3' cellspacing='3'>
8 <tr>
9   <td colspan='2' id='titulito'>${titcab} <img src='../../../../res/_img/_opt/brick_add.png' alt=' ' width='16' height='16' align='absmiddle'>
10  <input type="hidden" name="tipo" id="tipo" value="${tipo}">
11 </td>
12 </tr>
13  <tr>
14    <td class="fontForm" width="13%">Nombre:</td>
15    <td width="87%">
16      <span id='camp1'>
17        <input title='cnomusu' type='text' name='cnomprod' id='cnomprod' size='50' maxlength='50' />
18        <span class='textfieldRequiredMsg'>Campo requerido.</span><span>
19      </td>
20    </tr>
21    <tr>
22      <td class="fontForm"> Descripción:</td>
23      <td>
24        <span id='camp2'>
25          <input title='capeusu' type='text' name='cdescprod' id='cdescprod' size='100' maxlength='100' />
26          <span class='textfieldRequiredMsg'>Campo requerido.</span><span>
27        </td>
28      </tr>
29      <tr>
30        <td class="fontForm">Proveedor:</td>
31        <td>{include file=../../fpart/fpart.proveedor.tpl}</td>
32      </tr>
33      <tr>
34        <td class="fontForm">Stock M&iacute;nimo:</td>
35        <td>
36          <span id='camp3'>
37            <input title='fstkminprod' value="1" type='text' name='fstkminprod' id='fstkminprod' size='5' maxlength='5' />
38            <span class='textfieldRequiredMsg'>Campo requerido. </span><span class='textfieldInvalidFormatMsg'>#8226; Formato incorrecto.</span><span>
39          </td>
40        </tr>
41        <tr>
42          <td class="fontForm">Tipo de Producto:</td>
43          <td>{include file=../../fpart/fpart.tipo.producto.tpl}</td>
44        </tr>
45        <tr>
46          <td class="fontForm">Estado:</td>
47          <td>{include file=../../fpart/fpart.estado.tpl}</td>
48        </tr>
49      </table>
50    </div>
51  </form>
52 </body>
53  <script type='text/javascript'>
54  var camp1 = new Spry.Widget.ValidationTextField('camp1');
55  var camp2 = new Spry.Widget.ValidationTextField('camp2');
56  var camp3 = new Spry.Widget.ValidationTextField('camp3','real');
57  </script>
58 </html>
```

Figura 97: Código fuente de la Vista - Mproducto.

Fuente: El Autor.

■ **Modelo de Productos:**

- Archivo: Mproducto.php
- Dirección del Archivo: nombreProyecto/pages/fclases/Mproducto.php
- Descripción: Archivo que se encarga del manejo de los datos en la aplicación o software. En todo modelo es necesario colocar al comienzo del código fuente, la siguiente cabecera:

```
<?php include (CF_MODELO); ?>
```

El Framework solamente soporta la utilización de procedimientos almacenados de una base de datos. Por ejemplo, si existe un procedimiento almacenado, *pa\_createProducto*, con dos parámetros de entrada: *@p\_nomprod* y *@p\_descprod* se tiene que llamar a funciones del modelo que provee el Framework como muestra la Figura 98.

Donde: \$i es la numeración de la cantidad de parámetros del procedimiento almacenado.

*data[\$i]* es el valor del parámetro que requiere el procedimiento almacenado.

*paraproc[\$i]* es el nombre del parámetro del procedimiento almacenado.  
*tipoparaproc[\$i]* es el tipo de dato del parámetro del procedimiento almacenado.

*nomproc* es el nombre del procedimiento almacenado.

```

1 <?php include_once ('CF_MODELO');?>
2 <?php
3 class Mproducto{
4
5 /*Función que registra a un producto en la base de datos.*/
6     function crearProducto(){
7         $mod = new Modelo ();
8         $mod -> data [0] = $this -> cnomprod;
9         $mod -> paraproc [0] = "p_nomprod";
10        $mod -> tipoparaproc [0] = "varchar";
11        $mod -> data [1] = $this -> cdescprod;
12        $mod -> paraproc [1] = "p_descprod";
13        $mod -> tipoparaproc [1] = "varchar";
14        $mod -> data [2] = $this -> fstkminprod;
15        $mod -> paraproc [2] = "p_stkminprod";
16        $mod -> tipoparaproc [2] = "float";
17        $mod -> data [3] = $this -> ifkcodusu;
18        $mod -> paraproc [3] = "p_codusu";
19        $mod -> tipoparaproc [3] = "int";
20        $mod -> data [4] = $this -> ifkcodprov;
21        $mod -> paraproc [4] = "p_codprov";
22        $mod -> tipoparaproc [4] = "int";
23        $mod -> data [5] = $this -> ifkcodtipoprod;
24        $mod -> paraproc [5] = "p_codtipoprod";
25        $mod -> tipoparaproc [5] = "int";
26        $mod -> data [6] = $this -> ifkcodenst;
27        $mod -> paraproc [6] = "p_codenst";
28        $mod -> tipoparaproc [6] = "int";
29        $mod -> data [7] = $this -> ifkcodempp;
30        $mod -> paraproc [7] = "p_codempp";
31        $mod -> tipoparaproc [7] = "int";
32        $mod -> nomproc = "pa_crearProducto";
33        $recset = $mod -> ejecutarProcedimiento ();
34        $rowset = $mod -> fetch ($recset);
35        unset($mod);
36        if ($rowset['codprod'])
37            return true;
38        else
39            return false;
40    }
}

```

*Figura 98: Código fuente del Modelo Mproducto.*

*Fuente: El Autor.*

#### ▪ Controlador de Productos:

- Archivo: fnew.Mproducto.php

- Dirección del Archivo:

nombreProyecto/pages/fnew/fnew.Mproducto.php

- Descripción: Archivo que se encarga de la comunicación entre la vista y el modelo, así como también de realizar la lógica de negocio para la creación de los Productos. En todo archivo controlador es necesario colocar al comienzo del código fuente, la siguiente cabecera:

```
<?php include_once ('../../lib/_varentorno.php'); ?>
```

En todo archivo controlador es necesario colocar al final del código fuente, lo siguiente:

```
<?php include_once(CF_TPL); ?>
```

Si se requiere utilizar funciones de la vista que provee el Framework, se debe colocar: *include (CF\_VISTA);* y luego utilizar funciones que trae la vista como, por ejemplo:

- Instanciar un objeto Vista: *\$vist = new Vista();*
- Para cargar la validación de campos: *\$vist->cargarValidacionText();*
- Para cargar la hoja de estilos: *\$vist -> cargarCss('\_estilo.0');*

Si se requiere utilizar un modelo creado por el programador se debe colocar: *include (CF\_CLASES."Mproducto.php");*, al comienzo del código fuente, luego utilizar funciones del modelo:

- Instanciar un objeto del modelo Mproducto:

```
$prod = new Mproducto();
```

- Invocar al método del modelo:

```
$tra ->crearProductos
```

```

1  <?php session_start(); include_once ('../../../../lib/_parentorno.php'); include (CF_VISTA);  include (CF_CLASES.'Mproducto.php');?>
2  <?php
3  $vist = new Vista();
4  $vist -> cargarValidacionText();
5  $vist -> cargarCss('_estilo.0');
6
7  if(isset($_GET['Guardar'])){
8      $prod = new Mproducto();
9      $prod -> cnomprod=$_GET['cnomprod'];
10     $prod -> cdescprod=$_GET['cdescprod'];
11     $prod -> ifkcodprov=$_GET['ifkcodprov'];
12     $prod -> fstkmnprod=$_GET['fstkmnprod'];
13     $prod -> fkcodtipoprod = $_GET['fkcodtipoprod'];
14     $prod -> ifkcodusu = $_SESSION['USUARIO'];
15     $prod -> ifkcodemp = $_SESSION['EMPRESA'];
16     $prod -> ifkcodest = $_GET['ifkcodest'];
17     $codprod = $prod ->crearProducto();
18     if($codprod){
19         $vist -> alerta ("Se creó el Producto satisfactoriamente.");
20         $vist -> referencia ("../../fnew/Mproducto.php");
21     }
22 }
23 $vist -> assign('titcab', "Nuevo Producto");
24 $vist -> assign('LISTPROV', $listprov);
25 $vist -> assign('LISTTIPOPROD', $listtipoprod);
26 $vist -> assign('load', $vist -> load("cnomprod"));
27 $vist -> assign('LISTEST', $listest);
28 ?>
29 <?php include_once(CF_TPL);?>

```

Figura 99: Código fuente del controlador - Mproducto.

Fuente: El Autor.

▪ **Crear opción de menú:**

Para visualizar la opción de menú que invocará al controlador "fnew.Mproducto.php", es obligatorio ingresar un registro dentro de la tabla Mmenu:

- ipkcodmenu: Código del menú (autogenerado).
- cnomopc: Nombre de la opción a mostrar.
- cnomenla: Enlace del controlador.
- icodpadr: Código del padre; corresponde a la opción raíz donde pertenecerá la opción de menú a crear.
- iordmenu: Orden de la opción de menú.
- cicomenu: Nombre de imagen a mostrar. Por defecto, las imágenes se encuentran en: .res/\_img/\_opt.

ipkcodmenu	cnomopc	cnomenla	icodpadr	iordmenu	cicomenu
1	Menú Principal	NULL	-1	1	NULL
2	Usuarios	NULL	12	3	NULL
3	Nuevo Usuario	../fnew/fnew.Musuario.php	2	4	user_add.png
4	Listar Usuarios	../flist/flist.Musuario.php	2	5	application_view_list.png
5	Cambiar Clave	../fedit/fedit.clave.usuario.php	2	6	lock_edit.png
6	Productos	NULL	12	7	NULL
7	Nuevo Producto	../fnew/fnew.Mproducto.php?tipo=a	6	8	brick_add.png
8	Listar Productos	../flist/flist.Mproducto.php	6	9	bricks.png
9	Movimientos	NULL	1	19	NULL
10	Almacén	NULL	9	20	NULL

*Figura 100: Registro de opción de menú.*

*Fuente: El Autor.*

## ANEXO B

### CÓDIGO FUENTE DEL FRAMEWORK

A continuación, se presenta el principal código fuente del Framework:

#### • CLASES

- **Generador:** Clase encargada de generar modelos, vistas, controladores, generar recursos, entre otras funciones.

#### - Archivo contenedor: \_generador.php

```
include ('../../lib/_varentorno.php'); include (CF_MODELO);

class Generador {
    // Código (Atributos, funciones, etc.).
}
```

#### Atributos de la clase Generador:

```
var $nomcarp; // Nombre de la carpeta a instanciar el framework.
var $nomapli; // Nombre de la aplicación a instanciar.
var $nomdesa; // Nombre del desarrollador.
var $nomserv; // Nombre del servidor.
var $ususerv; // Nombre del usuario.
var $clavserv; // Nombre de clave del servidor.
var $nombd; // Nombre de base de datos.
var $dvr; // Nombre de driver.
var $clas; // Nombre de clase que representa al modelo de base de datos.
```

**Función verificarServicios:** Verifica si los argumentos de configuración son correctos.

```
function verificarServicios(){
    $mod = new Modelo ();
    if($mod -> verificarServicios($this -> nomserv,$this -> ususerv,$this ->
        clavserv,$this -> nombd,$this -> dvr))
```

```

        return true;
    } else
        return false;
}

```

**Función copiarCarpetas:** Copia las carpetas del framework hacia el sistema a instanciar.

```

function copiarCarpetas( $source, $target ) {
if ( is_dir( $source ) ) {
    $d = dir( $source );
    mkdir($target);
    if(!$d)
        return false;
    while ( FALSE !== ( $entry = $d->read() ) ) {
        if ( $entry == '.' || $entry == '..' || $entry == ".web" || $entry == "_generador.php" ) {
            continue;
        }
        $Entry = $source . '/' . $entry;
        if ( is_dir( $Entry ) ) {
            $this -> copiarCarpetas( $Entry, $target . '/' . $entry );
            continue;
        }
        @copy( $Entry, $target . '/' . $entry );
    }
    $d->close();
} else {
    @copy( $source, $target );
}
return true;
}
/*Crea las librerías en el archivo _varentorno.php y el archivo de configuración en _config.php, las cuales rigen el funcionamiento del framework.*/
function generarLibrerias(){
    $br = "\r";
    $libEnt = "<?php".$br;
    $libEnt .= /* Archivo Variables de Entorno
    @version 1.0
    @license GNU Lesser General Public License, http://www.gnu.org/copyleft/lesser.html
    @author Edgar Lizárraga Ugarte
    @created 2012-01-01
    @updated 2015-09-19
    @link ./lib/_varentorno.php
    */;
    $libEnt .= "?>".$br;
    //Librería variable de entorno.
    $libEnt .="<?php". $br;
    $libEnt .= "//Constante del framework donde se encuentra el controlador del
framework.". $br;
    $libEnt .="//Constante del framework donde se encuentra el modelo del framework.".
$br;
}

```

```

$libEnt .= " define (CF_MODELO,"."._SERVER['DOCUMENT_ROOT'].'/". $this ->
nomcarp."/classes/_modelo.php') ; ". $br;
$libEnt .="//Constante del framework donde se encuentra la vista del framework.". $br;
$libEnt .= " define (CF_VISTA,"."._SERVER['DOCUMENT_ROOT'].'/". $this ->
nomcarp."/classes/_vista.php') ; ". $br;
$libEnt .="//Constante del framework donde se encuentra la clase para manejar a los
usuarios.". $br;
$libEnt .= " define (CF_USUARIO,"."._SERVER['DOCUMENT_ROOT'].'/". $this ->
nomcarp."/classes/_usuario.php') ; ". $br;
$libEnt .="//Constante del framework donde se encuentra la clase para manejar al
menú.". $br;
$libEnt .= " define (CF_MENU,"."._SERVER['DOCUMENT_ROOT'].'/". $this ->
nomcarp."/classes/_menu.php') ; ". $br;
$libEnt .="//Constante del framework donde se encuentra la clase para manejar a los
reportes.". $br;
$libEnt .= " define (CF_Reporte,"."._SERVER['DOCUMENT_ROOT'].'/". $this ->
nomcarp."/classes/_reporte.php') ; ". $br;
$libEnt .="//Constante del framework donde se encuentra el gesto de base de datos a
utilizar.". $br;
$libEnt .= " define (CF_SQLGEST,"."._SERVER['DOCUMENT_ROOT'].'/". $this ->
nomcarp."/classes/_".$this -> dvr.".php'); ". $br;
$libEnt .="//Constante del framework donde se encuentra la clase Smarty para
generar los archivos TPL y enviar todas las variables, valores,etc., desde el controlador
hacia la vista.". $br;
$libEnt .= " define (CF_SMARTY,"."._SERVER['DOCUMENT_ROOT'].'/". $this ->
nomcarp."/lib/_smarty/libs/Smarty.class.php"); ". $br;
$libEnt .="//Constante del framework donde se encuentra el archivo para conectarse
con la base de datos.". $br;
$libEnt .= " define (CF_CONFIG,"."._SERVER['DOCUMENT_ROOT'].'/". $this ->
nomcarp."/lib/_config.php') ; ". $br;
$libEnt .="//Constante del framework que se encarga de desplegar la vista para cierto
controlador.". $br;
$libEnt .= " define (CF_TPL,"."._SERVER['DOCUMENT_ROOT'].'/". $this ->
nomcarp."/lib/_tpl.php') ; ". $br;
$libEnt .="//Constante del framework donde se encuentran los modelos creados por el
framework (Representación de tablas de la base de datos).". $br;
$libEnt .= " define (CF_CLASES,"."._SERVER['DOCUMENT_ROOT'].'/". $this ->
nomcarp."/pages/fclases/") ; ". $br;
$libEnt .="/*Variables de entorno definidos por el desarrollador.*". $br;
$libEnt .="//Nombre de la carpeta contenedora.". $br;
$libEnt .= " define (CD_NOMBRECARPETA,".$this -> nomcarp."); ". $br;
$libEnt .="//Nombre de la aplicación.". $br;
$libEnt .= " define (CD_NOMBREAPLICACION,".$this -> nomapli."); ". $br;
$libEnt .="//Nombre del desarrollador, empresa, etc.". $br;
$libEnt .= " define (CD_NOMBREDESARROLLADOR,".$this -> nomdesa."); ". $br;
$libEnt .="?>";
/* ----- Crear librería _lib.varentorno.php -----*/
$nomArc = "../../".$this -> nomcarp."/lib/"._varentorno.php";
$arc = fopen($nomArc, 'w') or die("no se puede abrir");
$write = fputs($arc, $libEnt);
if($nomArc== false || $arc == false || $write == false)
    return false;
fclose($arc);

$libConf = "<?php".$br;
$libConf .= /* Archivo de configuración
@version 1.0
@license GNU Lesser General Public License, http://www.gnu.org/copyleft/lesser.html

```

```

@author Edgar Lizárraga Ugarte
@created 2012-01-01
@updated 2015-09-19
@link ./lib/_config.php
*/";
$libConf .= "?>".$br;
$libConf .="<?php". $br;
$libConf .= "$"."serv=\"$". $this -> nomserv.";" . $br;
$libConf .= "$"."usu=\"$". $this -> ususerv.";" . $br;
$libConf .= "$"."pwd=\"$". $this -> clavserv.";" . $br;
$libConf .= "$"."bd=\"$". $this -> nombd.";" . $br;
$libConf .= "$"."dvr=\"$". $this -> dvr.";" . $br;
$libConf .= "$"."clas=\"$". $this -> clas.";" . $br;
$libConf .= "define('C_SERVIDOR', ". "$"."serv);". $br;
$libConf .= "define('C_USUARIO', ". "$"."usu);". $br;
$libConf .= "define('C_PASSWORD', ". "$"."pwd);". $br;
$libConf .= "define('C_BD', ". "$"."bd);". $br;
$libConf .= "define('C_DVR', ". "$"."dvr);". $br;
$libConf .= "define('C_CLASS', ". "$"."clas);". $br;
$libConf .="?>";
/* ----- Crear librería _lib.config.php. PARA LA INSTANCIA -----*/
$nomArc = "../../".$this -> nomcarp."/lib"/"_config.php";
$arc = fopen($nomArc, 'w') or die("no se puede abrir");
$write = fputs($arc, $libConf);

if($nomArc== false || $arc == false || $write == false)
    return false;

fclose($arc);
/* ----- Crear librería _lib.config.php. PARA EL MISMO FRAMEWORK -----*/
*/
$nomArc = "../../lib/_config.php";
$arc = fopen($nomArc, 'w') or die("no se puede abrir");
$write = fputs($arc, $libConf);

if($nomArc== false || $arc == false || $write == false)
    return false;

fclose($arc);
$index = "<script language='javascript' type='text/javascript'>". $br;
$index .= "function redireccionar()". $br;
$index .= "{location.href='./pages/fprincipal/login.php';}". $br;
$index .= "setTimeout ('redireccionar()', 5);". $br;
$index .= "</script>". $br;
/* ----- Crear archivo index para la nueva instancia. -----*/
$nomArc = "../../".$this -> nomcarp."/index.html";
$arc = fopen($nomArc, 'w') or die("no se puede abrir");
$write = fputs($arc, $index);

if($nomArc== false || $arc == false || $write == false)
    return false;
fclose($arc);

return true;
}

```

**Función generarRecursos:** Función principal que invoca a copiarCarpetas y generarLibrerias.

```
function generarRecursos(){
    //Crear carpeta contenedora.
    if(!mkdir("../../".$this -> nomcarp))
        return 1;
    /*Copiar archivos de LizarFrame hacia una carpeta, la cual es ingresada por el
desarrollador.*/
    $orig = "../../LizarFrame";
    $dest = "../../".$this -> nomcarp;

    if(!$this -> copiarCarpetas ($orig,$dest))
        return 2;

    if(!$this -> generarLibrerias ())
        return 3;
    return 4;
}
```

**Función generarArchivos:** Función principal que invoca a: generarVistaListar generarControladores generarModelos y generarVistaNuevo.

```
function generarArchivos (){
    $mod = new Modelo ();
    //Seleccionar las tablas creadas por el usuario.
    //Generar las vistas para archivos en modo NUEVO REGISTRO.
    $recset = $mod -> seleccionarTablasUsuario();
    if(!$this -> generarVistaNuevo($recset,$mod)){
        return "Alguno de los procedimientos almacenados no fueron ejecutados.";
        exit();
    }
    unset ($mod);
    $mod = new Modelo ();
    //Generar las vistas para archivos en modo LISTADO DE REGISTROS.
    $recset = $mod -> seleccionarTablasUsuario();
    if(!$this -> generarVistaListar($recset,$mod)){
        return "Alguno de los procedimientos almacenados no fueron ejecutados.";
        exit();
    }
}
```

```

}

unset ($mod);
$mod = new Modelo ();
//Generar todos los controladores.
$recset = $mod -> seleccionarTablasUsuario();
if(!$this -> generarControladores ($recset,$mod)){
    return "Alguno de los procedimientos almacenados no fueron ejecutados.";
    exit();
}

unset ($mod);
$mod = new Modelo ();
//Generar todos los modelos en la carpeta fclases.
$recset = $mod -> seleccionarTablasUsuario();
if(!$this -> generarModelos ($recset,$mod)){
    return "Alguno de los procedimientos almacenados no fueron ejecutados.";
    exit();
}

unset ($mod);
return "Los archivos se generaron correctamente.";
}

```

**Función generarVistaListar:** Genera las vistas desde las tablas de la base de datos en cuanto a listado de registros se refiere. Extrae el esquema de las tablas creadas.

```

function generarVistaListar ($recsettab,$mod){
    $rowset = $mod -> fetch ($recsettab);
    if($rowset){
        do {
            //Colocar el nombre de las tablas en un array.
            $tablas [$conttab] = $rowset ['name'];

            /* ----- Columnnas de una tabla. ----- */
            $recset = $mod -> obtenerColumnasTabla($tablas [$conttab]);
            $rowset = $mod -> fetch ($recset);
            //contador de columnas de la tabla en uso.
            $contcol = 0;

            unset($coltab);
    }
}

```

```

do
{
    $coltab[$contcol][NOMB] = $rowset['COLUMN_NAME'];
    $coltab[$contcol][LONG] = $rowset['LENGTH'];
    $rowset = $mod -> fetch ($recset);
    $contcol++;
}while ($rowset);

/* ----- Crear las vistas de las tablas. MODO LISTAR REGISTROS -----
--- */
$br = "\r";
$arcvist = "<html>" . $br . "<head>" . $br .
"<meta http-equiv='Content-Type' content='text/html; charset=utf-8'/>
<title></title>" . $br .
"</head>" . $br ;
$arcvist .= "<body>" . $br;
$arcvist .= "<div id='panel'>" . $br;
$arcvist .= "<table width='100%' border='1' align='center' cellpadding='2' cellspacing='0' bordercolor='#CCCCCC'>" . $br ;
$arcvist = "<tr>" . $br . " <td colspan='14' id='titulo'{$ptitle} <img src='../../res/_img/_opt/application_view_list.png' alt=' ' width='16' height='16' align='absmiddle'> </td>" . $br . "</tr>" . $br;
$arcvist .= "<tr bgcolor='#D8DDFC'>" . $br;
for($i = 0; $i < count($coltab) ; $i++){
    $arcvist .= " <th width='20%'>" . $coltab[$i][NOMB]. "</th>" .
    $br;
}
$arcvist .= "</tr>" . $br;
$arcvist .= " {section name = rshow loop = ".$"."."ARRAY"} . $br;
$arcvist .= " <tr onMouseOver='this.style.background='#E1E1E1';'>" . $br;
"onMouseOut='this.style.background=''" . $br;
for($i = 0; $i < count($coltab) ; $i++){
    $arcvist .= " <td align='center'>" . $br;
    "{$"."."ARRAY[rshow].".$coltab[$i][NOMB]."}" . "</td>" . $br;
}
$arcvist .= " </tr>" . $br;
$arcvist .= " {/section}" . $br;

```

```

$arcvist .= "</table>" . $br;
$arcvist .= "</div>" . $br;
$arcvist .= "</body>" . $br;
$arcvist .= "</html>" . $br;

/* ----- Crear los archivos para las vistas. -----*/
$nomarcvist = "../../".$this -> nomcarp."/pages/flist/flist.".$tablas
[$conttab];
$nomarcvist .= ".tpl";
$arc = fopen($nomarcvist, 'w') or die("no se puede abrir");

fputs($arc, $arcvist);
fclose($arc);

//Contador para el array de tablas.

$conttab++;
$rowset = $mod -> fetch ($recsettab);
}while($rowset);

return true;
unset ($arcvist);
}
else
return false;
}

```

**Función generarControladores:** Genera los controladores para consultar data y mostrar las vistas creadas..

```

function generarControladores ($recsettab,$mod){
$rowset = $mod -> fetch ($recsettab);
if($rowset){
do{
//Colocar el nombre de las tablas en un array.
$tablas [$conttab] = $rowset ['name'];
/* ----- Columnnas de una tabla. ----- */
$recset = $mod -> obtenerColumnasTabla($tablas [$conttab]);
$rowset = $mod -> fetch ($recset);
$contcol = 0;

```

```

unset($coltab);

do
{
$coltab[$contcol][NOMB] = $rowset['COLUMN_NAME'];
$coltab[$contcol][LONG] = $rowset['LENGTH'];
$rowset = $mod -> fetch ($recset);
$contcol++;
}while ($rowset);

//Crear los controladores.

$br = "\r";
$cont = "<?php include_once ('../../lib/_varentorno.php'); include
(CF_VISTA);?> ".$br;
$cont .= "<?php //Sección para implementar. ".$br;
$cont .= "$"."vist = new Vista ();".$br;
$cont .= "$"."vist -> cargarValidacionText();".$br;
$cont .= "$"."vist -> cargarCss('_estilo.0');".$br;
$cont .= "?>".$br;
$cont .= "<?php include(CF_TPL); ?>".$br;

$nomCont = "../../".$this -> nomcarp."/pages/fnew/fnew.". $tablas
[$conttab];
$nomCont .= ".php";
$arc = fopen($nomCont, 'w');
fputs($arc, $cont);
fclose($arc);

$nomCont = "../../".$this -> nomcarp."/pages/flist/flist.". $tablas
[$conttab];
$nomCont .= ".php";
$arc = fopen($nomCont, 'w');
$write = fputs($arc, $cont);
fclose($arc);

unset ($cont);

//Contador para el array de tablas.
$conttab++;
$rowset = $mod -> fetch ($recsettab);
}while($rowset);

return true;
unset ($nomModelo);
}
else
return false;
}

}

```

**Función generarModelos:** Genera las clases a partir de las tablas de la base de datos a emplear.

```
function generarModelos ($recsettab,$mod){
```

```

$rowset = $mod -> fetch ($recsettab);
if($rowset){
    do {
        //Colocar el nombre de las tablas en un array.
        $tab [$conttab] = $rowset ['name'];
        /* ----- Columnnas de una tabla. ----- */
        $recset = $mod -> obtenerColumnasTabla($tab [$conttab]);
        $rowset = $mod -> fetch ($recset);
        $contcol = 0;

        unset($coltab);
        do
        {
            $coltab [$contcol][NOMB] = $rowset['COLUMN_NAME'];
            $coltab [$contcol][LONG] = $rowset['LENGTH'];
            $rowset = $mod -> fetch ($recset);
            $contcol++;
        }while ($rowset);

        //Crear los modelos (clases en base a tablas).
        if ($tab [$conttab] != "Musuario" && $tab [$conttab] != "Mmenu"){
            $br = "\r";
            $clas = "<?php include (CF_MODELO); ?> ".$br;
            $clas .= "<?php".$br;
            $clas .= "class " . $tab [$conttab] . "{". $br . $br;
            for($i = 0; $i < count($coltab) ; $i++)
            {
                $clas .= "var ".$coltab[$i][NOMB] . ";" . $br;
            }
            $clas .= " }". $br;
            $clas .= "?>";

            $nommod = " ../../../../$this -> nomcarp."/pages/fclases/".$tab
[$conttab];
            $nommod .= ".php";
            $arc = fopen($nommod, 'w');
            fputs($arc, $clas);
            fclose($arc);
            unset ($clas);
        }
    }
}

```

```

    }
    //Contador para el array de tablas.
    $conttab++;
    $rowset = $mod -> fetch ($recsettab);
    }while($rowset);

    return true;
    unset ($nommod);
}
else
    return false;
}

```

**Función generarVistaNuevo:** Genera las vistas desde las tablas de la base de datos en cuanto a creación de registros se refiere. Extrae el esquema de las tablas creadas.

```

function generarVistaNuevo ($recsettab,$mod){
    $rowset = $mod -> fetch ($recsettab);
    if($rowset){
        do {
            //Colocar el nombre de las tablas en un array.
            $tab [$conttab] = $rowset ['name'];

            /* ----- Columnnas de una tabla. ----- */
            $recset = $mod -> obtenerColumnasTabla($tab[$conttab]);
            $rowset = $mod -> fetch ($recset);
            $contcol = 0;

            unset($coltab);

            do
            {
                $coltab [$contcol][NOMB] = $rowset['COLUMN_NAME'];
                $coltab [$contcol][LONG] = $rowset['LENGTH'];
                $rowset = $mod -> fetch ($recset);
                $contcol++;
            }while ($rowset);

            /* ----- Crear las vistas de las tablas. MODO NUEVO REGISTRO
----- */
    }
}

```

```

$br = "\r";
$arcvist = "<html>" . $br . "<head>" . $br .
"<meta http-equiv='Content-Type' content='text/html; charset=utf-8'/>
<title></title>" . $br .
"</head>" . $br ;

$arcvist .= "<body onLoad = " . ".\"." . "load".") >" . $br ;
$arcvist .= "<form id='cform' name='cform' method='GET' action=>" .
$br ;
$arcvist .= "<div id='panel'" . $br;
$arcvist .= "<table width='100%' border='0' cellpadding='3'
cellspacing='3'>" . $br .
"<tr>
<td colspan='2' id='titulo'>{$ptitle}<img
src='../../res/_img/_opt/application_add.png' alt=' ' width='16' height='16' align='absmiddle'>
</td>";

```

for(\$i = 0; \$i < count(\$coltab) ; \$i++){
\$arcvist .= "<tr>
<td> ".\$coltab[\$i][NOMB]."</td>
<td>
<span id= 'camp".\$i.">
<input title= '".\$.coltab[\$i][NOMB]." type='text' name=
'".\$coltab[\$i][NOMB]."' id= '".\$coltab[\$i][NOMB] ."' size= '".\$coltab[\$i][LONG]."' maxlength=
'".\$coltab[\$i][LONG]."' /s>
<span class='textfieldRequiredMsg'>\*</span></span>
</td>
</tr>" . \$br;
}

\$arcvist .= "<tr>
<td colspan='2'><input name='Guardar' type='submit' class='inputbot'
id='Submit' value='Guardar' /></td>
</tr> " . \$br . "</table>" . \$br;

\$arcvist .= "</div>" . \$br;

\$arcvist .= "</form>" . \$br;

```

$arcvist .= "</body>" . $br;

/* ----- Validaciones para la vista. ----- */
$arcvist .= "<script type='text/javascript'>". $br;

for($i = 0; $i < count($coltab) ; $i++)
{
    $arcvist .= "var camp".$i." = new
Spry.Widget.ValidationTextField('camp".$i.");". $br;
}

$arcvist .= "</script> ". $br . "</html>";

/* ----- Crear los archivos para las vistas. -----*/
$nomarcvist = "../../".$this -> nomcarp."/pages/fnew/fnew.". $tab
[$conttab];
$nomarcvist .= ".tpl";
$arc = fopen($nomarcvist, 'w') or die("no se puede abrir");

fputs($arc, $arcvist);
fclose($arc);

//Contador para el array de tablas.
$conttab ++;

$rowset = $mod -> fetch ($recsettab);
}while($rowset);

return true;
unset ($arcvist);
}
else
    return false;
}

```

- **Modelo:** Clase que se comunica con la base de datos a trabajar.

- **Archivo contenedor:** \_modelo.php

```
session_start();
```

```

class Modelo{
    // Código (Atributos, funciones, etc.).
}

```

### Atributos de la clase Modelo:

```

var $cnx; // Conexión a la base de datos.
var $serv; //Nombre del servidor de base de datos.
var $usu; //Nombre del usuario de la base de datos.
var $pwd; //Password de acceso a la base de datos.
var $bd; //Nombre de la base de datos a conectar.
var $nomproc; //Nombre del Procedimiento Almacenado.
var $paraproc = array (); //Parámetro de entrada al Procedimiento Almacenado.
var $tipoparaproc = array (); //Tipo ce parámetro de entrada al Procedimiento Almacenado.
var $data = array(); //Data entregada al Procedimiento Almacenado.
var $dvr ; // Nombre de driver para gestor de base de datos.
var $pcnx ; // Nombre de conexión a gestor de base de datos.
var $selbd ; // Nombre de selección de base de datos.
var $fetch ; // Extraer registros.
var $numrow ; // Número de filas o registros.

```

**Función Constructora:** Encargada de asignar los valores de variables de entorno a valores de atributos de la clase modelo.

```

function __construct(){
    include_once (CF_CONFIG); include_once (CF_SQLGEST);
    if(C_SERVIDOR){
        $this -> serv = C_SERVIDOR;
        $this -> usu = C_USUARIO;
        $this -> pwd = C_PASSWORD;
        $this -> bd = C_BD;
        $this -> dvr = C_DVR;
        $this -> clas = C_CLASS;
        $this -> pcnx = $this->dvr._connect';
        $this -> selbd = $this->dvr._select_db';
        $this -> fetch = $this->dvr._fetch_assoc';
        $this -> numrow = $this->dvr._num_rows';
    }
}

```

**Función Conectar:** Establece la conexión con la base de datos.

```
function conectar (){
    $pcnx = $this -> pcnx;
    $selbd = $this -> selbd;
    $this -> cnx = $pcnx ($this -> serv,$this -> usu,$this -> pwd, false);
    if ( $selbd ($this -> bd, $this -> cnx)){
        session_register ('USUARIO');
        session_register ('NOMBRE');
        return true;
    }
    else
        return false;
}
```

**Función Desconectar:** Elimina la conexión con la base de datos.

```
function desconectar (){
    mssql_close($this -> cnx);
}
```

**Función fetch:** Función que retorna los registros de la base de datos en una matriz.

```
function fetch ($recset){
    $fetch = $this -> fetch;
    return $fetch ($recset);
}
```

**Función numeroRegistros:** Función que retorna la cantidad de registros.

```
function numeroRegistros ($recset){
    $numrow = $this -> numrow;
    return $numrow($recset);
}
```

**Función ejecutarProcedimiento:** Función que ejecuta el procedimiento almacenado según el gestor de base de datos empleado.

```

function ejecutarProcedimiento (){
    /* Aquí utilizamos Polimorfismo. Depende del tipo de driver para ejecutar el
    procedimiento almacenado para diferente
        gestor de base de datos */
    $obj = new $this -> dvr();
    $this -> conectar ();
    $recset = $obj -> proc ($this); // En $this envío el modelo.
    $this -> desconectar();
    return $recset;
}

```

**Función seleccionarTablasUsuario:** Función que selecciona las tablas creadas por el usuario en su base de datos.

```

function seleccionarTablasUsuario (){
    try {
        $this -> nomproc = "sp_seleccionarTablasUsuario";
        $recset = $this -> ejecutarProcedimiento ();
        $totreg= $this -> numeroRegistros($recset);
        if(!$totreg)
            throw new Exception(
                'Problemas con la información sumistrada. Es posible que el
                Procedimiento Almacenado no se encuentre disponible.');
    }
    catch (Exception $e) {
        $msje = $e -> getMessage();
    }
    if ($msje)
        return $msje;
    else
        return $recset;
}

```

**Función obtenerColumnasTabla:** Función que obtiene las columnas de una tabla específica.

```

function obtenerColumnasTabla($nomtab){
    $this -> nomproc = "sp_seleccionarColumnasTabla";
    $this -> paraproc [0] = "p_nomtab";

```

```

        $this -> tipoparaproc [0] = "varchar";
        echo $nomtab;
        $this -> data [0] = $nomtab;
        $recset = $this -> ejecutarProcedimiento ();
        $totreg= $this -> numeroRegistros($recset);
        return $recset;
    }

```

**Función obtenerColumnasTabla:** Verifica los servicios de la base de datos.

```

function obtenerColumnasTabla($nomtab){
    $this -> nomproc = "sp_seleccionarColumnasTabla";
    $this -> paraproc [0] = "p_nomtab";
    $this -> tipoparaproc [0] = "varchar";
    echo $nomtab;
    $this -> data [0] = $nomtab;
    $recset = $this -> ejecutarProcedimiento ();
    $totreg= $this -> numeroRegistros($recset);
    return $recset;
}

```

- **Vista:** Clase que maneja eventos para mostrar en la parte de la interfaz.

- **Archivo contenedor:** \_vista.php

```

include_once ('../../lib/_varentorno.php'); include_once (CF_SMARTY);

class Vista extends Smarty{
    // Código (Atributos, funciones, etc.).
}

```

### Atributos de la clase Vista:

```

var $nomvent; // Nombre de la aplicación.
var $titvist; // Título de la vista.
var $popup; //Nombre del popup a mostrar.
var $nomserv; //Servidor en donde se encuentra la aplicación.
var $nomdesa; //Nombre del desarrollador o empresa.
var $port; //Puerto en el que se encuentra la aplicación, por defecto en el 8080.
var $dirapli = ""; //Directorio de la carpeta donde coloco mi aplicación.

```

**Función Constructora:** Encargada de hacer el llamado a la clase smarty para mostrar la interfaz de cada controlador. Declara funciones para trabajar con la interfaz.

```
function __construct($nombre = null,$nomdesa = null,$directorío = null){  
    parent::__construct();  
    $this -> template_dir = "";  
    $this -> compile_dir = './fcache';  
    $this -> cache_dir = './fcache';  
    $this -> config_dir = './fcache';  
    $this -> nomvent = $nombre;  
    $this -> nomdesa = $nomdesa;  
    $this -> nomserv = $_SERVER['SERVER_NAME'];  
    $this -> port = $_SERVER['SERVER_PORT'];  
    if($directorío){  
        $this -> dirapli = $directorío;  
        $this -> popup = "http://".$this -> nomserv.":".$this -> port."/". $this -> dirapli."/pages/fprincipal/login.php";  
    }  
    else  
        $this -> popup = "http://".$this -> nomserv.":".$this -> port."/pages/fprincipal/login.php";  
}
```

**Función alerta:** Muestra mensaje de alerta.

```
function alerta ($msje){  
    echo "<script> alert(\"".$msje."\");</script>";  
}
```

**Función referencia:** Ir hacia una página específica.

```
function referencia ($ref){  
    echo "<script>location.href=\"".$ref."\"</script>";  
}
```

**Función load:** Coloca el foco del formulario en el campo que se quiera.

```
function load ($text){  
    return "document.getElementById('$text').focus()";  
}
```

**Función reload:** Actualiza la página local.

```
function reload (){
    echo "<script>location.reload()</script>";
}
```

**Función openerReload:** Actualiza la página padre que abrió a la página en uso.

```
function openerReload (){
    echo "<script>window.opener.location.reload()</script>";
}
```

**Función cerrarVentana:** Cerrar ventana actual o en uso.

```
function cerrarVentana (){
    echo "<script>window.close()</script>";
}
```

**Función cerrarVentanaPrincipal:** Cerrar ventana actual o en uso.

```
function cerrarVentanaPrincipal (){
    echo "<script>top.window.close()</script>";
}
```

**Función cargarCalendario:** Carga el calendario en la vista.

```
function cargarCalendario(){
    echo      "<script          language='javaScript'        type='text/javascript'";
    echo      "src='../../lib/_almanaque/_calendar.js'></script>";
    echo      "<script          language='javaScript'        type='text/javascript'";
    echo      "src='../../lib/_almanaque/lang/_espanol.js'></script>";
    echo      "<script          language='javaScript'        type='text/javascript'";
    echo      "src='../../lib/_almanaque/_setup.js'></script>";
    echo      "<link           rel='stylesheet'           type='text/css'";
    echo      "href='../../lib/_almanaque/_calendar.css'>";
```

**Función cargarValidacionText:** Carga la validación para cajas de texto.

```
function cargarValidacionText(){
    echo      "<script          type='text/javascript'        src='../../validaciones/_textField.js'";
    echo      "type='text/javascript'></script>";
```

```

echo      "<link      href='../../validaciones/_textField.css'      rel='stylesheet'
type='text/css'/'>";
}


```

**Función cargarValidacionArea:** Carga la validación para áreas de texto.

```

function cargarValidacionArea(){
    echo      "<script      src='../../validaciones/_textArea.js'
type='text/javascript'></script>";
    echo      "<link      href='../../validaciones/_textArea.css'      rel='stylesheet'
type='text/css'/'>";
}


```

**Función cargarCss:** Carga la hoja de estilos solicitada.

```

function cargarCss($text){
    echo "<link href='../../css/".$text.".css' rel='stylesheet' type='text/css'/'>";
}


```

**Función cargarVentanaPrincipal:** Carga la ventana principal de acceso al sistema.

```

function cargarVentanaPrincipal(){
    echo "<script src='../../js/_ventana.js' type='text/javascript'></script>";
}


```

**Función getNombreVentana:** Obtener el nombre de la ventana actual.

```

function getNombreVentana(){
    return $this -> nomvent;
}


```

**Función getNombreDesarrollador:** Obtener nombre del desarrollador.

```

function getNombreDesarrollador(){
    return $this -> nomdesa;
}


```

**Función getNombreAplicacion:** Obtener nombre de la aplicación.

```

function getNombreAplicacion(){
    return $this -> nomvent;
}

```

**Función setTituloVista:** Colocar el título a una vista.

```

function setTituloVista($tit){
    $this -> titvist = $tit;
}

```

**Función getTituloVista:** Obtener el título de una vista.

```

function getTituloVista(){
    return $this -> titvist;
}

```

- **mssql:** Clase que se encarga de ejecutar los procedimientos almacenados correspondiente al Gestor de Base de Datos SQL SERVER.  
 - **Archivo contenedor:** \_mssql.php

```
include ('../../../../lib/_varentorno.php'); include_once (CF_MODELO)
```

```

class mssql extends Modelo{
    // Código (Atributos, funciones, etc.).
}

```

**Función proc:** Función que ejecuta los procedimientos almacenados.

```

function proc ($obj){ //obj es el objeto de la clase modelo, creado desde un modelo en
'fclases'
    $strproc = mssql_init($obj -> nomproc);
    if (count($obj -> paraproc)){
        for ($i = 0 ; $i < count ($obj -> data); $i++){
            if ($obj -> tipoparaproc[$i] =="varchar")
                mssql_bind($strproc, "@".$obj-> paraproc[$i], $obj -> data[$i],
SQLVARCHAR, FALSE, FALSE);
            if ($obj -> tipoparaproc[$i] =="int")
                mssql_bind($strproc, "@".$obj-> paraproc[$i], $obj -> data[$i],
SQLINT4, FALSE, FALSE);
            if ($obj -> tipoparaproc[$i] =="float")

```

```

        mssql_bind($strproc, "@".$obj->paraproc[$i], $obj -> data[$i],
SQLFLT8, FALSE, FALSE);
        if ($obj -> tipoparaproc[$i] == "text")
            mssql_bind($strproc, "@".$obj->paraproc[$i], $obj -> data[$i],
SQLTEXT, FALSE, FALSE);
    }
}
if($recset = mssql_execute($strproc))
    return $recset;
else
    return false;
}

```

- **Mmenu:** Se encarga del menú principal del sistema a instanciar.

- **Archivo contenedor:** \_menu.php

```

include(CF_MODELO);
class Mmenu extends Modelo{
    // Código (Atributos, funciones, etc.).
}

```

**Atributo de la clase Mmenu:**

var \$ipkcodusu;

**Función dibujarMenu:** Despliega los nodos del menú según los registros de la tabla mmenu, la cual es parte del framework.

```

function dibujarMenu(){
    $mod = new Modelo ();
    $mod -> data [0] = $this -> ipkcodusu ;
    $mod -> paraproc [0] = "p_codusu";
    $mod -> tipoparaproc [0] = "int";
    $mod -> nomproc = "pa_listamenu";
    $recset = $mod -> ejecutarProcedimiento ();
    $rowset = $mod -> fetch ($recset);
    $cont = 0;
    do {
        if ($rowset['cicomenu'])

```

```

    $icomenu = "../../res/_img/_opt/".$rowset['cicomenu'];
else
    $icomenu = $rowset['cicomenu'];
$menu[$cont] = array (
'codmenu'      => $rowset['ipkcodmenu'],
'codpadr'      => $rowset['icodpadr'],
'nomopc'        => $rowset['cnomopc'],
'nomenla'       => $rowset['cnomenla'],
'icomenu'       => $icomenu
);
$cont++;
$rowset = $mod -> fetch ($recset);
} while($rowset);
unset($mod);
return $menu;
}

```

**Función listarOpcionesMenu:** Obtiene y lista las opciones de menú asignadas al usuario.

```

function listarOpcionesMenu (){
    $mod = new Modelo ();
    $mod -> nomproc = "pa_listarOpcionesMenu";
    $recset = $mod -> ejecutarProcedimiento ();
    $rowset = $mod -> fetch ($recset);
    $cont = 0;
    do {
        $mod2 = new Modelo ();
        $mod2 -> data [0] = $this -> ipkcodusu ;
        $mod2 -> paraproc [0] = "p_codusu";
        $mod2 -> tipoparaproc [0] = "int";
        $mod2 -> data [1] = $rowset['ipkcodmenu'] ;
        $mod2 -> paraproc [1] = "p_codmenu";
        $mod2 -> tipoparaproc [1] = "int";
        $mod2 -> nomproc = "pa_listarOpcionesMenuPorUsuario";
        $recset2 = $mod2 -> ejecutarProcedimiento ();
        $numreg = $mod2 -> numeroRegistros ($recset2);
        if ($numreg)
            $flagchec = 1;
    }
}

```

```

        else
            $flagchec = 0;
        unset ($mod2);
        $opcmenu[$cont] = array(
            'codmenu' => $rowset['ipkcodmenu'],
            'codpadr' => $rowset['icodpadr'],
            'nomopc' => htmlentities($rowset['cnomopc']),
            'nomenla' => $rowset['cnomenla'],
            'flagchec' => $flagchec
        );
        $cont++;
        $rowset = $mod -> fetch ($recset);
    }while($rowset);
    unset($mod);
    return $opcmenu;
}

```

- **Musuario:** Clase encargada de manejar a los usuarios del sistema a instanciar.

- **Archivo contenedor:** \_usuario.php

```

include_once (CF_MODELO);

class Musuario extends Modelo{
    // Código (Atributos, funciones, etc.).
}

```

#### **Atributos de la clase Musuario:**

```

var $cclavusu; // Clave de usuario.
var $clogusu; // Login de usuario.
var $cnomusu; // Nombre de usuario.
var $capeusu; // Apellido de usuario.

```

**Función validarUsuario:** Función que verificar si existe o no el usuario creado en la base de datos.

```

function validarUsuario(){
    $mod = new Modelo ();
    $mod -> data [0] = $this -> cclavusu;
}

```

```

$mod -> paraproc [0] = "p_clavusu";
$mod -> tipoparaproc [0] = "varchar";
$mod -> data [1] = $this -> cmailusu;
$mod -> paraproc [1] = "p_mailusu";
$mod -> tipoparaproc [1] = "varchar";
$mod -> nomproc = "pa_listaUsuario";
$recset = $mod -> ejecutarProcedimiento ();
$rowset = $mod -> fetch ($recset);
unset($mod);
if ($rowset['cmailusu'] == $this -> cmailusu && $rowset['cclavusu'] == $this ->
cclavusu ){
    session_start();
    $_SESSION['USUARIO'] = $rowset['ipkcodusu'];
    $_SESSION['PERFIL'] = $rowset['ifkcodperf'];
    $_SESSION['TENANT'] = $rowset['ifkcodten'];
    $_SESSION['TIEMPOSESION'] = date('Y-n-j H:i:s');
    $_SESSION['NOMBRE']      = htmlentities($rowset['capepatusu'] . '
'.$rowset['capematusu']. ' ' . $rowset['cnomusu']);
    return true;
}
else
    return false;
}

```

**Función crearUsuario:** Función que registra un usuario en la base de datos.

```

function crearUsuario (){
    $mod = new Modelo ();
    $mod -> data [0] = $this -> cnomusu ;
    $mod -> paraproc [0] = "p_nomusu";
    $mod -> tipoparaproc [0] = "varchar";
    $mod -> data [1] = $this -> capepatusu;
    $mod -> paraproc [1] = "p_apepatusu";
    $mod -> tipoparaproc [1] = "varchar";
    $mod -> data [2] = $this -> cclavusu;
    $mod -> paraproc [2] = "p_clavusu";
    $mod -> tipoparaproc [2] = "varchar";
    $mod -> data [3] = $this -> cmailusu;
    $mod -> paraproc [3] = "p_mailusu";
}

```

```

$mod -> tipoparaproc [3] = "varchar";
$mod -> data [4] = $this -> capematusu;
$mod -> paraproc [4] = "p_apematusu";
$mod -> tipoparaproc [4] = "varchar";
$mod -> data [5] = $this -> ifkcodperf;
$mod -> paraproc [5] = "p_codperf";
$mod -> tipoparaproc [5] = "int";
$mod -> data [6] = $this -> ifkcodest;
$mod -> paraproc [6] = "p_codest";
$mod -> tipoparaproc [6] = "int";
$mod -> data [7] = $this -> ifkcodten;
$mod -> paraproc [7] = "p_codten";
$mod -> tipoparaproc [7] = "int";
$mod -> data [8] = $this -> ifkcodusu;
$mod -> paraproc [8] = "p_codusu";
$mod -> tipoparaproc [8] = "int";
$mod -> nomproc = "pa_crearUsuario";
$mod -> ejecutarProcedimiento ();
unset($mod);
return true;
}

```

**Función listarUsuario:** Función que lista a los usuarios existentes en la base de datos.

```

function listarUsuario (){
    $mod = new Modelo ();
    $mod -> data [0] = $this -> ifkcodten ;
    $mod -> paraproc [0] = "p_codten";
    $mod -> tipoparaproc [0] = "int";
    $mod -> nomproc = "pa_listarUsuario";
    $recset = $mod -> ejecutarProcedimiento ();
    $rowset = $mod -> fetch ($recset);
    $cont = 0;
    do {
        $usu[$cont] = array (
            'codusu'           => $rowset['ipkcodusu'],
            'nomusu'          => $rowset['cnomusu'],
            'apepatusu'       => $rowset['capepatusu'],

```

```

'apematusu'    => $rowset['capematusu'],
'mailusu'       => $rowset['cmailusu'],
'clavusu'       => $rowset['cclavusu'],
'nrousu'         => $rowset['inrousu'],
'nomperf'        => $rowset['cnomperf'],
'descest'        => $rowset['cdescest'],
'nomalm'         => $rowset['cnomalm']
);
$cont++;
$rowset = $mod -> fetch ($recset);
} while($rowset);
unset($mod);
return $usu;
}

```

**Función obtenerUsuario:** Función que obtiene a los usuarios existentes en la base de datos.

```

function obtenerUsuario (){
    $mod = new Modelo ();
    $mod -> data [0] = $this -> ipkcodusu ;
    $mod -> parapro [0] = "p_codusu";
    $mod -> tipoparapro [0] = "int";
    if ($this -> cmailusu){//Obtener usuario por email.
        $mod -> data [1] = $this -> cmailusu ;
        $mod -> parapro [1] = "p_mailusu";
        $mod -> tipoparapro [1] = "varchar";
    }
    $mod -> nomproc = "pa_obtenerUsuario";
    $recset = $mod -> ejecutarProcedimiento ();
    $rowset = $mod -> fetch ($recset);
    $cont = 0;
    if($rowset){
        do {
            $usu[$cont] = array (
                'codusu'           => $rowset['ipkcodusu'],
                'nomusu'          => $rowset['cnomusu'],
                'apepatusu'        => $rowset['capepatusu'],
                'apematusu'        => $rowset['capematusu'],
                'clavusu'          => $rowset['cclavusu'],

```

```

'mailusu'           => $rowset['cmailusu']
);
$cont++;
$rowset = $mod -> fetch ($recset);
} while($rowset);
}
unset($mod);
return $usu;
}

```

**Función editarUsuario:** Función que edita a un usuario existente en la base de datos.

```

function editarUsuario (){
    $mod = new Modelo ();
    $mod -> data [0] = $this -> cnomusu ;
    $mod -> paraproc [0] = "p_nomusu";
    $mod -> tipoparaproc [0] = "varchar";
    $mod -> data [1] = $this -> capepatusu;
    $mod -> paraproc [1] = "p_apepatusu";
    $mod -> tipoparaproc [1] = "varchar";
    $mod -> data [2] = $this -> cmailusu;
    $mod -> paraproc [2] = "p_mailusu";
    $mod -> tipoparaproc [2] = "varchar";
    $mod -> data [3] = $this -> ipkcodusu;
    $mod -> paraproc [3] = "p_codusu";
    $mod -> tipoparaproc [3] = "int";
    $mod -> data [4] = $this -> capematusu;
    $mod -> paraproc [4] = "p_apematusu";
    $mod -> tipoparaproc [4] = "varchar";
    $mod -> data [5] = $this -> codusuEditor;
    $mod -> paraproc [5] = "p_codusuEditor";
    $mod -> tipoparaproc [5] = "int";
    $mod -> nomproc = "pa_editarUsuario";
    $mod -> ejecutarProcedimiento ();
    unset($mod);
    return true;
}

```

**Función editarClaveUsuario:** Función que edita la clave de un usuario.

```
function editarClaveUsuario (){
    $mod = new Modelo ();
    $mod -> data [0] = $this -> ipkcodusu ;
    $mod -> paraproc [0] = "p_codusu";
    $mod -> tipoparaproc [0] = "int";
    $mod -> data [1] = $this -> cclavusu;
    $mod -> paraproc [1] = "p_clavusu";
    $mod -> tipoparaproc [1] = "varchar";
    $mod -> nomproc = "pa_editarClaveUsuario";
    $mod -> ejecutarProcedimiento ();
    unset($mod);
    return true;
}
```

**Función crearUsuarioAdministrador:** Función que registra a un usuario de perfil administrador en la base de datos.

```
function crearUsuarioAdministrador (){
    $mod = new Modelo ();
    $mod -> data [0] = $this -> capepatusu ;
    $mod -> paraproc [0] = "p_apepatusu";
    $mod -> tipoparaproc [0] = "varchar";
    $mod -> data [1] = $this -> capematusu;
    $mod -> paraproc [1] = "p_apematusu";
    $mod -> tipoparaproc [1] = "varchar";
    $mod -> data [2] = $this -> cnomusu;
    $mod -> paraproc [2] = "p_nomusu";
    $mod -> tipoparaproc [2] = "varchar";
    $mod -> data [3] = $this -> cnomemp;
    $mod -> paraproc [3] = "p_nomemp";
    $mod -> tipoparaproc [3] = "varchar";
    $mod -> data [4] = $this -> cmailusu;
    $mod -> paraproc [4] = "p_mailusu";
    $mod -> tipoparaproc [4] = "varchar";
    $mod -> data [5] = $this -> cclavusu;
    $mod -> paraproc [5] = "p_clavusu";
```

```

$mod -> tipoparaproc [5] = "varchar";
$mod -> nomproc = "pa_crearUsuarioAdministrador";
$mod -> ejecutarProcedimiento ();
unset($mod);
return true;
}

```

**Función validarExistenciaUsuario:** Función que valida la existencia de un usuario.

```

function validarExistenciaUsuario (){
    $mod = new Modelo ();
    $mod -> data [0] = $this -> cmailusu ;
    $mod -> paraproc [0] = "p_mailusu";
    $mod -> tipoparaproc [0] = "varchar";
    $mod -> nomproc = "pa_validarExistenciaUsuario";
    $recset = $mod -> ejecutarProcedimiento ();
    $rowset = $mod -> fetch ($recset);
    if($rowset['cantUsu'] > 0)
        return true;
    else
        return false;
}

```

- **Mreporte:** Clase que maneja los reportes.

- **Archivo contenedor:** \_reporte.php

```

include_once ('../../../../lib/_varentorno.php');?>

class Mreporte{
    // Código (Atributos, funciones, etc.).
}

```

#### Atributos de la clase Mreporte:

```

var $ip = "localhost"; //Dirección ip del servidor donde se almacenan los Reportes.
var $nomservrpt = "ReportServer_EFLU2008/"; //Nombre del Servidor (Report Server).
var $nomrep; //Nombre del Reporte.
var $url; // Url del Reporte a mostrar.

```

**Función verReporte:** Función que visualiza al reporte ya creado.

```
function verReporte (){
    return "http://".$this -> ip.$this->nomservrpt.$this -> url;
}
```

- **LIBRERIAS**

- **\_almanaque**

Para cargar un calendario:

```
$vist = new Vista();
$vist -> cargarCalendario;
```

A continuación se muestra el código para poder utilizarlo:

```
<input name="fecha" readonly id="fecha" type="text" class="dates" size="10"
maxlength="10" tabindex="1">

this.style.background="#3271AF;" onMouseOut= "this.style.background="">

<script language="javaScript" type="text/javascript">Calendar.setup({l delim}inputField: "
fecha ", ifFormat: "%Y-%m-%d", button: "f_trigger_c1", align: "A2", singleClick: true{r delim})
</script>
```

Nota: Cuando se utiliza las llaves para abrir y cerrar una porción de código javascript en un archivo con extensión .tpl, se debe de utilizar {l delim} y {r delim}.

- **\_tpl.php:**

```
$vist -> display(str_replace("php", "tpl", basename($_SERVER['PHP_SELF'])));
```

- **\_config.php:**

```
$serv='nombreServidor';
$usu= 'usuarioServidor';
$pwd='claveServidor';
```

```

$bd='nombreBasedeDatos';
$dvr='nombreGestorBasedeDatos'; // Ejemplo: mssql
$clas=Nombre de clase modelo principal';//Ejemplo: Mssql
define('C_SERVIDOR',$serv);
define('C_USUARIO', $usu);
define('C_PASSWORD',$pwd);
define('C_BD',$bd);
define('C_DVR',$dvr);
define('C_CLASS',$clas);

```

- **\_varentorno.php:**

```

//Constante del framework donde se encuentra el controlador del framework.
define
(CF_GENERADOR,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.classes/_generador.php');

//Constante del framework donde se encuentra el modelo del framework.
define
(CF_MODELO,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.classes/_modelo.php');

//Constante del framework donde se encuentra la vista del framework.
define (CF_VISTA,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.classes/_vista.php');

//Constante del framework donde se encuentra la clase para manejar a los usuarios.
define
(CF_USUARIO,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.classes/_usuario.php');

//Constante del framework donde se encuentra la clase para manejar al menú.
define (CF_MENU,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.classes/_menu.php');

//Constante del framework donde se encuentra la clase para manejar a los reportes.
define
(CF_Reporte,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.classes/_reporte.php');

//Constante del framework donde se encuentra el gesto de base de datos a utilizar.
define
(CF_SQLGEST,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.classes/_mssql.php');

//Constante del framework donde se encuentra la clase Smarty para generar los archivos
TPL y enviar todas las variables, valores,etc., desde el controlador hacia la vista.
define
(CF_SMARTY,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.lib/_smarty/libs/Smarty.cl
ass.php');

//Constante del framework donde se encuentra el archivo para conectarse con la base de
datos.

```

```

define (CF_CONFIG,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.lib/_config.php');
//Constante del framework que se encarga de desplegar la vista para cierto controlador.
define (CF_TPL,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.lib/_tpl.php');
//Constante del framework donde se encuentran los modelos creados por el framework
(Representación de tablas de la base de datos).
define (CF_CLASES,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/pages/fclases/');
//Constante del framework donde se coloca la configuración del cache.
define (CF_LIB1,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.lib/_lib.1.php');
//Constante del framework donde se coloca el formato para agregar ceros a la derecha.
define
(CF_AGREGARCEROS,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.lib/_agregarCero
s.php');
//Constante del framework para redondear números.
define
(CF_REDONDEAR,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.lib/_redondear.php');
//Constante del framework para referenciar un número en letras.
define
(CF_NUMALET,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.lib/_numeroALetras.php')
;
//Constante del framework para crear una sesión de usuario de acuerdo al tiempo que
coloque el desarrollador.
define (CF_LIBSESION,$_SERVER['DOCUMENT_ROOT'].'/LizarFrame/.lib/_sesion.php');
/*Variables de entorno definidos por el desarrollador.*/
//Nombre de la carpeta contenedora.
define (CD_NOMBRECARPETA,'LizarFrame');
//Nombre de la aplicación.
define (CD_NOMBREAPLICACION,'LizarFrame');
//Nombre del desarrollador, empresa, etc.
define (CD_NOMBREDESARROLLADOR,'Edgar Lizarraga Ugarte');
//Tiempo que un usuario permanece en una sesión activa.
define (CD_TIEMPOSESION,'4800');

```

- **SCRIPT DE BASE DE DATOS DEL FRAMEWORK:** Se encuentra en la plataforma Web del Framework y puede ser descargado a través de tal medio.

## ANEXO C

### IMPLEMENTACIÓN DEL SUBSISTEMA DE INFORMACIÓN ORIENTADO A LA PRESTACIÓN DE SERVICIOS (MODELO SAAS)

1. **Código fuente:** La construcción de software para el sistema de información logístico necesita de código fuente implementado en dicho sistema. Para ver a detalle el código fuente, se puede abrir el proyecto en un entorno de desarrollo integrado capaz de leer archivos php, por ejemplo, Dreamweaver, PHPStorm, entre otros. El código fuente se puede consultar en el disco óptico adjunto al presente proyecto.
  
2. **Pantallas del Software:**



Figura 101: Login del sistema de información instanciado.

Fuente: El Autor.

**Sede:** Arequipa  
**Usuario:** Lizárraga Ugarte Edgar Frank  
**Fecha:** 22-01-2016  
**Hora:** 01:26:52 pm

Código	Proveedor	Nombre	Descripción	Tipo	Unidad	Stock Mínimo	Estado	...
001	Pointer	GPS CELLO F	GPS CELLO F CT7700200-000	GPS	Unidades	10.00	Activo	
002	Pointer	GPS COMPACT FLEET	GPS COMPACT FLEET GE6600193-ER3N	GPS	Unidades	10.00	Activo	
003	Pointer	GPS TRACKER 105	GPS TRACKER MODELO 105	GPS	Unidades	10.00	Activo	
004	Pointer	GPS CR300B	-GAIN FOR 370-50	GPS	Unidades	10.00	Activo	
005	Movistar	CHIP MOVISTAR - POST PAGO DATOS	CHIP MOVISTAR POSPAGO DATOS	CHIP	Unidades	10.00	Activo	
006	Pointer	CHIP MOVISTAR - VOZ Y DATOS	CHIP MOVISTAR VOZ Y DATOS PARA HANDS FREE	CHIP	Unidades	10.00	Activo	
007	Claro	CHIP PREPAGO CLARO - SMS	CHIP PREPAGO CLARO DE MENSAJES DE TEXTO	CHIP	Unidades	10.00	Activo	
008	Pointer	RELAY 12V	12V 711-20000	Insumos	Unidades	1.00	Activo	
009	Pointer	RELAY 24V	24V 711-20006	Insumos	Unidades	1.00	Activo	
010	Pointer	BOTÓN DE PÁNICOS	711-20001	Insumos	Unidades	10.00	Activo	
011	Pointer	FUSIBLE	3A Fusible	Insumos	Unidades	10.00	Activo	
012	Pointer	ARNES 156	9 WIRES INSTALLATION HARNESS	Insumos	Unidades	10.00	Activo	
013	Pointer	ARNES 199	16 WIRES INSTALLATION HARNESS	Insumos	Unidades	10.00	Activo	
014	Pointer	ANTENA GPS	STANDARD GPS RECEIVER – 3 METER LENGTH	Insumos	Unidades	1.00	Activo	
015	Pointer	KIT DE HANDSFREE	Comunicación entre conductor y representante del CCMF. Incluye HandsFree, micrófono, y parlante.	Insumos	Unidades	5.00	Activo	
016	Pointer	ARNES 248	FULL HARNESS INSTALLATION	Insumos	Unidades	1.00	Activo	
017	Pointer	BATERIA	BATERIA 7.4V 750MAH	Insumos	Unidades	1.00	Activo	
018	Pointer	BUZZER	AVISADOR DE VELOCIDAD SUPERIOR A KM/HORA	Insumos	Unidades	10.00	Activo	
019	Pointer	LLAVE DALLAS ROJA	LLAVE DALLAS COLOR ROJO PARA ACCESORIO DALLAS KIT	Insumos	Unidades	5.00	Activo	

Total de Registros: 19

Figura 102: Listado de productos de un primer cliente.

Fuente: El Autor.

**Sede:** Lima  
**Usuario:** Catañeda Rettis Claudia  
**Fecha:** 22-01-2016  
**Hora:** 03:19:43 pm

Código	Proveedor	Nombre	Descripción	Tipo	Unidad	Stock Mínimo	Estado	...
001	Pointer	GPS CELLO F	GPS CELLO F CT7700200-000	GPS	Unidades	10.00	Activo	
002	Pointer	GPS COMPACT FLEET	GPS COMPACT FLEET GE6600193-ER3N	GPS	Unidades	10.00	Activo	
003	Pointer	GPS TRACKER 105	GPS TRACKER MODELO 105	GPS	Unidades	10.00	Activo	
004	Pointer	GPS CR300B	-GAIN FOR 370-50	GPS	Unidades	10.00	Activo	
005	Movistar	CHIP MOVISTAR - POST PAGO DATOS	CHIP MOVISTAR POSPAGO DATOS	CHIP	Unidades	10.00	Activo	
006	Pointer	CHIP MOVISTAR - VOZ Y DATOS	CHIP MOVISTAR VOZ Y DATOS PARA HANDS FREE	CHIP	Unidades	10.00	Activo	
007	Claro	CHIP PREPAGO CLARO - SMS	CHIP PREPAGO CLARO DE MENSAJES DE TEXTO	CHIP	Unidades	10.00	Activo	
008	Pointer	RELAY 12V	12V 711-20000	Insumos	Unidades	1.00	Activo	
009	Pointer	RELAY 24V	24V 711-20006	Insumos	Unidades	1.00	Activo	
010	Pointer	BOTÓN DE PÁNICOS	711-20001	Insumos	Unidades	10.00	Activo	
011	Pointer	FUSIBLE	3A Fusible	Insumos	Unidades	10.00	Activo	
012	Pointer	ARNES 156	9 WIRES INSTALLATION HARNESS	Insumos	Unidades	10.00	Activo	
013	Pointer	ARNES 199	16 WIRES INSTALLATION HARNESS	Insumos	Unidades	10.00	Activo	
014	Pointer	ANTENA GPS	STANDARD GPS RECEIVER – 3 METER LENGTH	Insumos	Unidades	1.00	Activo	
015	Pointer	KIT DE HANDSFREE	Comunicación entre conductor y representante del CCMF. Incluye HandsFree, micrófono, y parlante.	Insumos	Unidades	5.00	Activo	
016	Pointer	ARNES 248	FULL HARNESS INSTALLATION	Insumos	Unidades	1.00	Activo	
017	Pointer	BATERIA	BATERIA 7.4V 750MAH	Insumos	Unidades	1.00	Activo	
018	Pointer	BUZZER	AVISADOR DE VELOCIDAD SUPERIOR A KM/HORA	Insumos	Unidades	10.00	Activo	
019	Pointer	LLAVE DALLAS ROJA	LLAVE DALLAS COLOR ROJO PARA ACCESORIO DALLAS KIT	Insumos	Unidades	5.00	Activo	

Total de Registros: 19

Figura 103: Listado de productos de un segundo cliente.

Fuente: El Autor.

Sede: Arequipa  
 Usuario: Lizárraga Ugarte Edgar Frank  
 Fecha: 22-01-2016  
 Hora: 01:28:13 pm

Figura 104: Registro de compras.

Fuente: El Autor.

Nombre	Descripción	Nro. Artículo	Cantidad
BOTÓN DE PÁNICO	711-20001		1.00
GPS CELLO F	GPS CELLO F CT7700200-000	202020	1.00
CHIP MOVISTAR - POST PAGO	CHIP MOVISTAR POSTPAGO DATOS	2546454653211	1.00
FUSIBLE	3A Fusible	-	2.00
ARMES 156	9 WIRES INSTALLATION HARNESS	-	1.00
LLAVE DALLAS ROSA	LLAVE DALLAS COLOR ROJO PARA ACCESORIO DALLAS KIT	-	2.00
GPS COMPACT FLEET	GPS COMPACT FLEET GE6600191-ERJN	303030	1.00
GPS COMPACT FLEET	GPS COMPACT FLEET GE66001912-ERJN	303031	1.00
GPS COMPACT FLEET	GPS COMPACT FLEET GE66001913-ERJN	303032	1.00
GPS COMPACT FLEET	GPS COMPACT FLEET GE66001915-ERJN	303033	1.00
GPS COMPACT FLEET	GPS COMPACT FLEET GE66001913-ERJN	303034	1.00

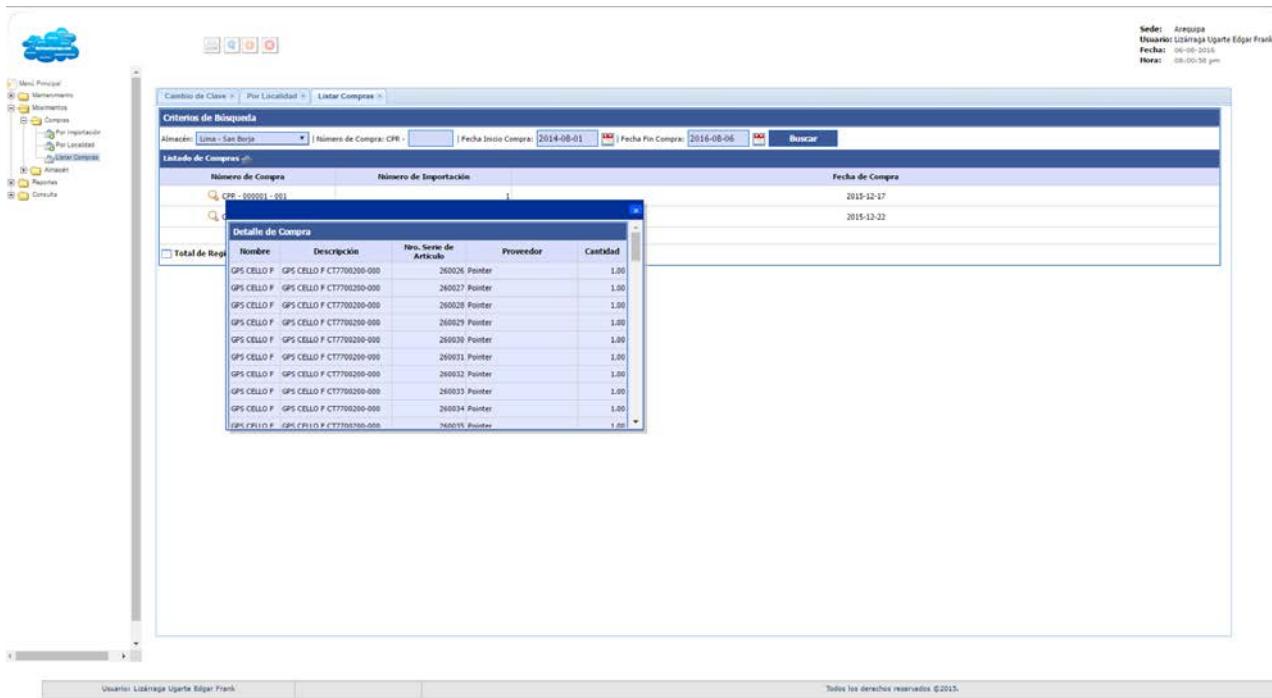
Total de Registros: 11

Agregar Artículo | Agregar GPS | Agregar CHIP | Agregar CHIP Excel

Sede: Arequipa  
 Usuario: Lizárraga Ugarte Edgar Frank  
 Fecha: 06-08-2016  
 Hora: 07:59:49 pm

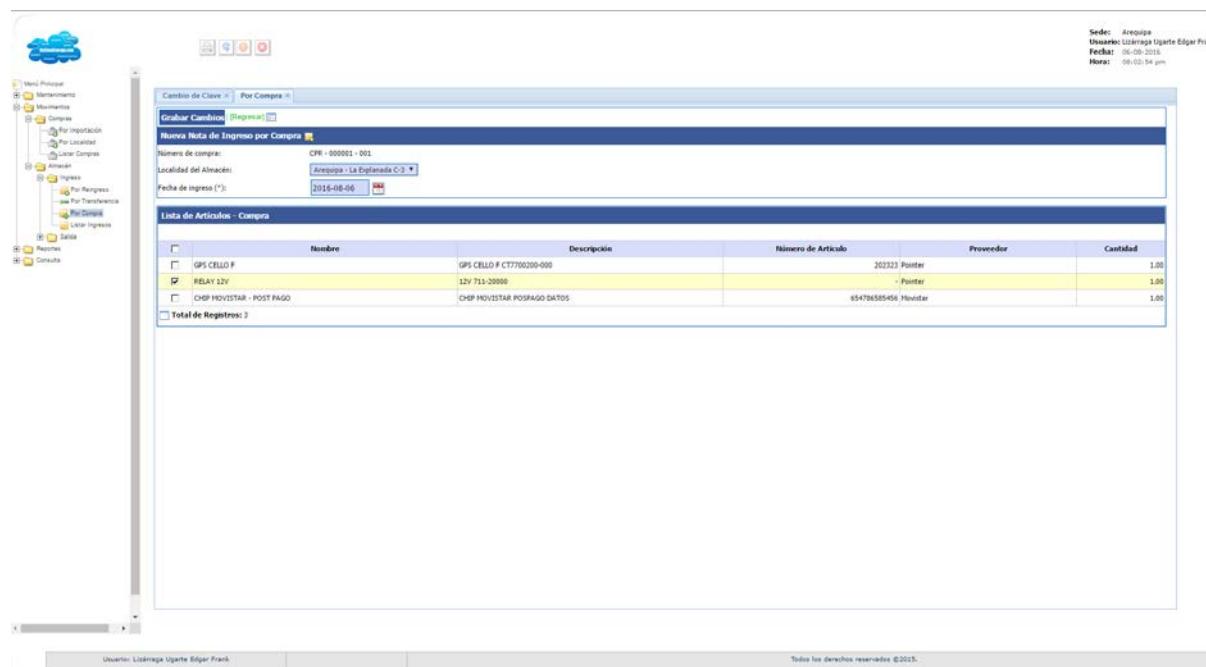
Figura 105: Selección de productos.

Fuente: El Autor.



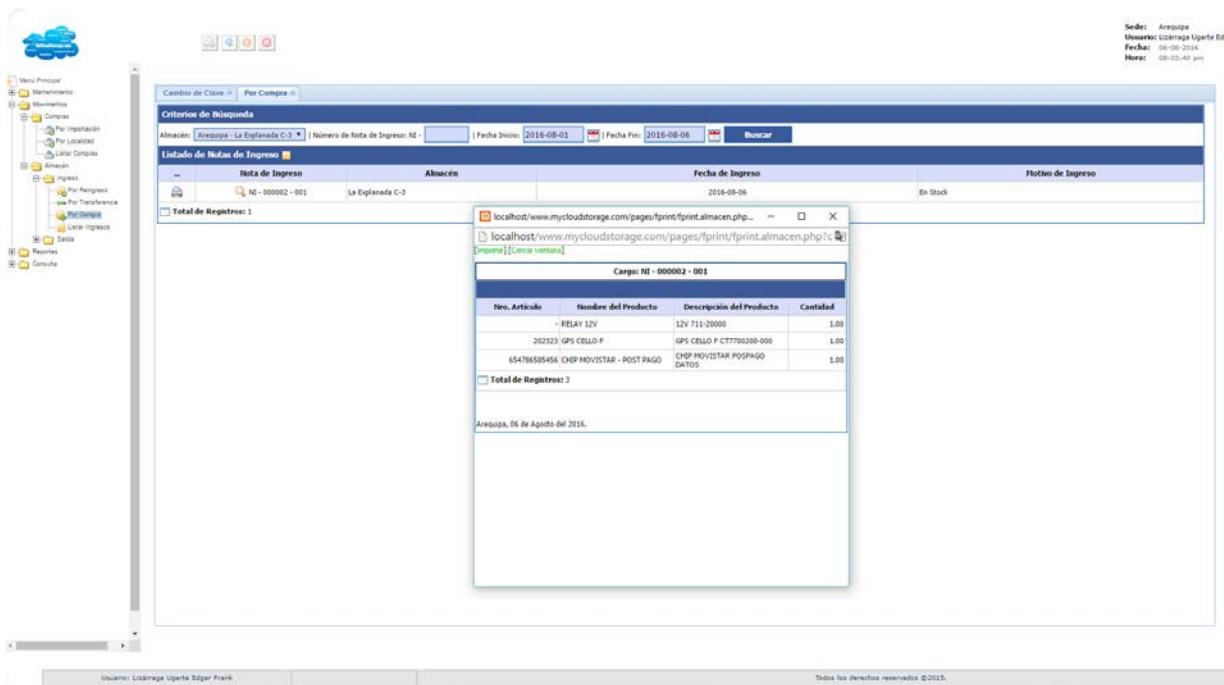
*Figura 106: Listado de compras.*

*Fuente: El Autor.*



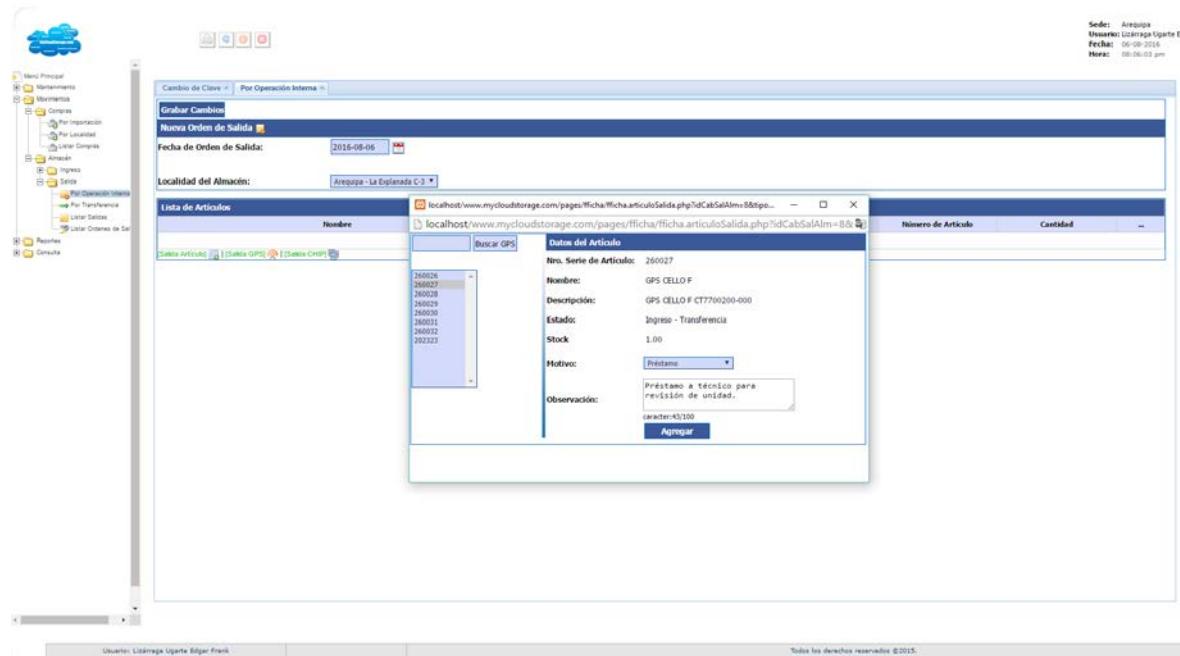
*Figura 107: Nueva nota de ingreso por compra.*

*Fuente: El Autor.*



*Figura 108: Impresión de nota de ingreso.*

Fuente: El Autor.



*Figura 109: Orden de salida por operación interna.*

Fuente: El Autor.

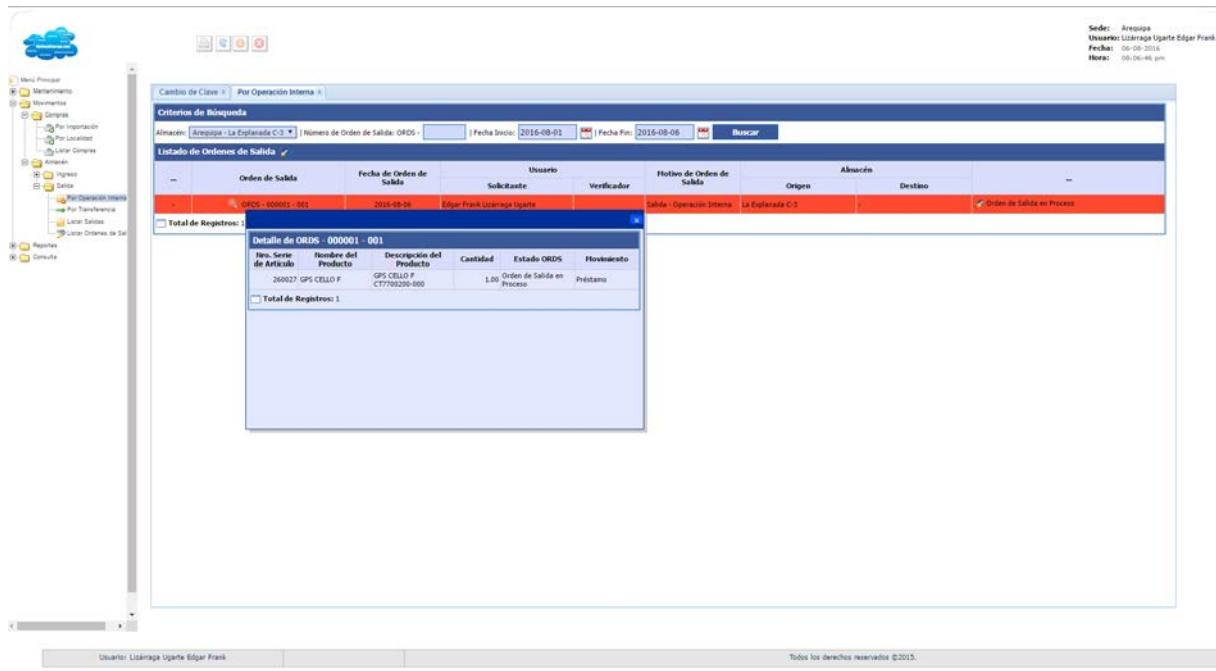


Figura 110: Listado de órdenes de salida sin aprobar.

Fuente: El Autor.

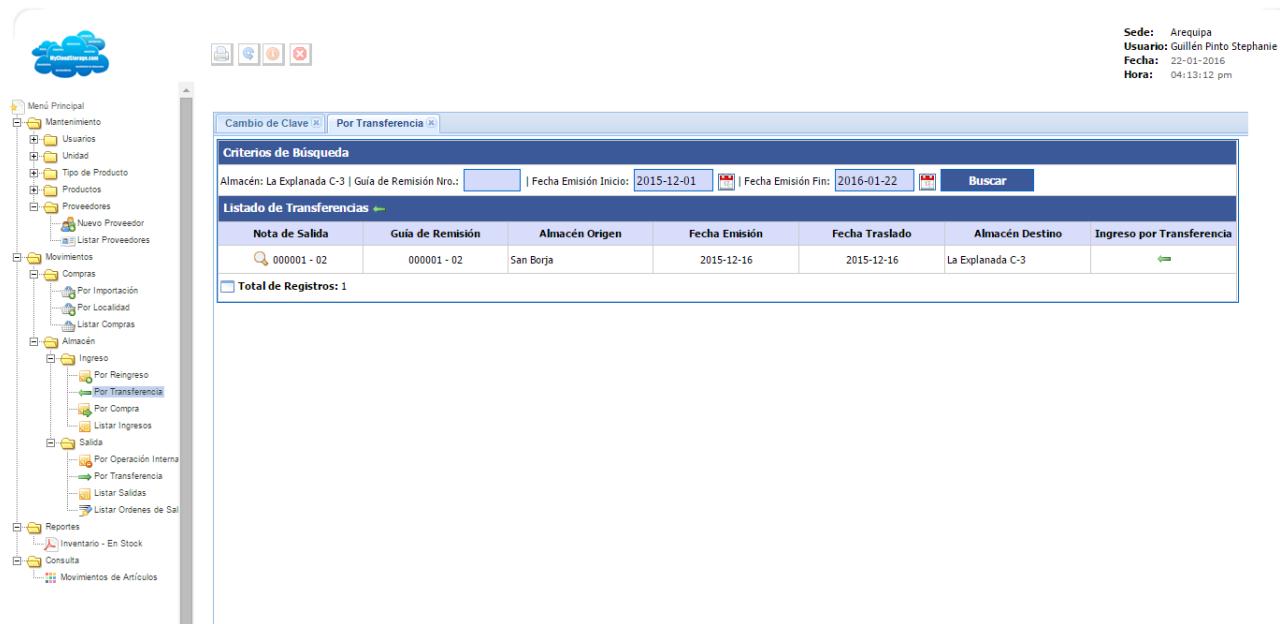
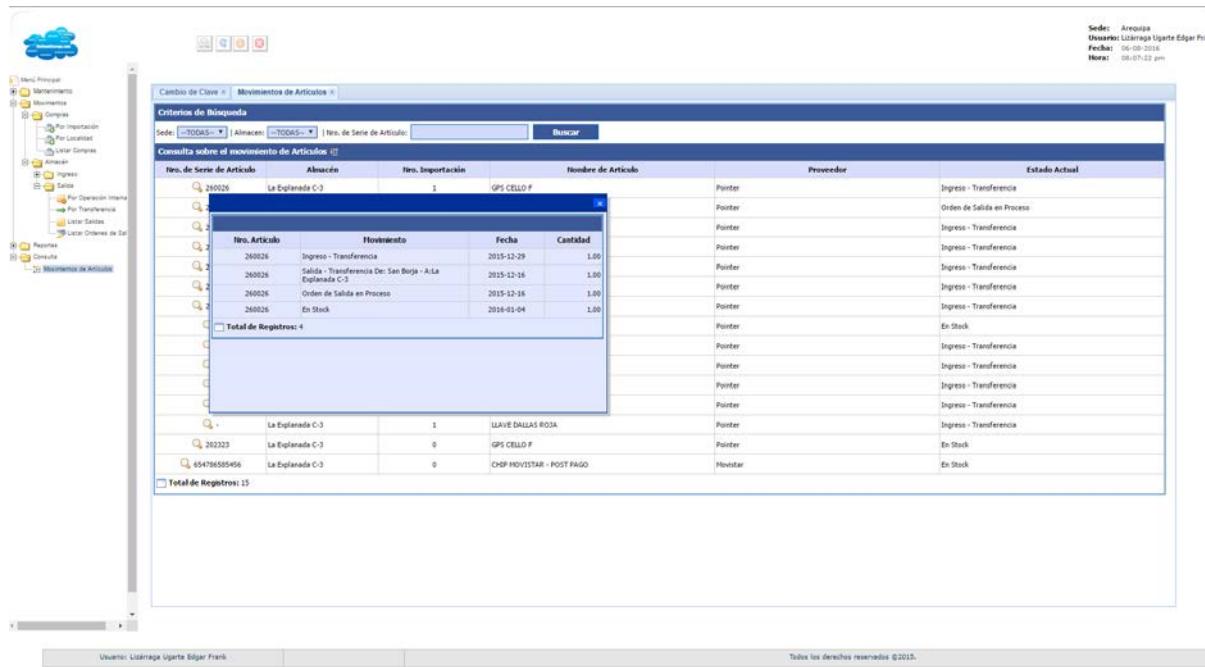


Figura 111: Ingreso de productos por transferencia de almacén.

Fuente: El Autor.



*Figura 112: Consulta del movimiento de los productos.*

*Fuente: El Autor.*

## **ANEXO D**

### **CUESTIONARIO**

**1. ¿Puede Ud. trabajar con el Framework en navegadores que utiliza?**

- a) Si
- b) No

**2. ¿Qué opina sobre el desarrollo del Framework basado en el patrón de diseño MVC (Modelo - Vista - Controlador)?**

- a) Muy Bueno
- b) Bueno
- c) Regular
- d) Malo

**3. ¿Qué opina sobre el desarrollo del Framework basado en el modelo SaaS elegido?**

- a) Muy Bueno
- b) Bueno
- c) Regular
- d) Malo

**4. Para crear o instanciar una aplicación a través del Framework, ¿realiza muchos pasos?**

- a) Si
- b) No

**5. ¿Cómo considera la curva de aprendizaje del Framework?**

- a) Alta
- b) Regular
- c) Baja

**6. ¿Le costó mucho tiempo comenzar a crear software en un sistema de información a través del Framework?**

- a) Si
- b) No

**7. ¿Cree Ud. que el Framework le proveerá mayor agilidad en la construcción de software en Sistemas de Información orientados a la prestación de servicios (modelo SaaS)?**

- a) Si
- b) No

**8. ¿Cuántas veces se puede crear software a través del Framework?**

- a) Muchas veces
- b) Pocas veces
- c) Ninguna

**9. ¿Considera que la estructura del Framework es?**

- a) Muy Buena
- b) Bueno
- c) Regular
- d) Mala

**10. El dar mantenimiento al software instanciado por el Framework, considera que es:**

- a) Fácil
- b) Regular
- c) Difícil

**11. ¿Alguna vez escuchó sobre el término SaaS (Software as a service)?**

- a) Si
- b) No

**12. ¿Conoce Ud. algún Framework que ayude a construir aplicaciones web orientadas a la prestación de servicios?**

- a) Si
- b) No

**13. ¿Cree Ud. que el software como servicio incremente?**

- a) Si
- b) No

## ANEXO E

### GLOSARIO DE TÉRMINOS

- **Amazon EC2:** Amazon Elastic Compute Cloud es un servicio web que proporciona capacidad de cómputo con tamaño modifiable en la cloud. Está diseñado para facilitar a los desarrolladores el cloud computing escalable basado en web.
- **Amazon S3:** Amazon Simple Storage Service ofrece a los desarrolladores y los profesionales de TI un almacenamiento en la cloud seguro, duradero y altamente escalable. Amazon S3 es un almacenamiento de objetos fácil de utilizar, con una sencilla interfaz de servicios web para almacenar y recuperar la cantidad de datos que desee desde cualquier ubicación de la web.
- **AMD: Advanced Micro Devices** es una compañía estadounidense de semiconductores establecida en Sunnyvale, California, que desarrolla procesadores de cómputo y productos tecnológicos relacionados para el mercado de consumo. Sus productos principales incluyen microprocesadores, chipsets para placas base, circuitos integrados auxiliares, procesadores embebidos y procesadores gráficos para servidores, estaciones de trabajo, computadores personales y aplicaciones para sistemas embebidos.
- **ASP:** Application Service Providers (en español, Proveedores de Servicios de Aplicaciones), empresas que proporcionan servicios de software a sus clientes a través de una red. Esencialmente, los **ASP** son una manera de adquirir externamente algunos o casi todos los aspectos de la tecnología de información que necesitan las compañías.
- **BBDD:** Base de datos que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto.
- **Brightcove:** Compañía que proporciona plataformas de alojamiento y de reproducción de videos en línea.

- **CEO:** Chief executive officer (en español, Ejecutivo en jefe) persona con más alta responsabilidad de una organización o corporación.
- **Cloud Computing:** La computación en la nube, conocida también como servicios en la nube, informática en la nube, nube de cómputo o nube de conceptos, es un paradigma que permite ofrecer servicios de computación a través de una red.
- **CRM:** Customer Relationship Management (en español, Administración de la relación con el cliente), es un modelo de gestión de toda la organización, basada en la satisfacción al cliente. El concepto más cercano es marketing relacional.
- **CSV:** Comma separated values (en español, Valores separados por comas), formato para almacenar datos tabulares y separados por comas.
- **EMV:** Empresa de Monitoreo Vehicular, dedicada a las soluciones de ubicación satelital y monitoreo permanente a través de dispositivos GPS.
- **ERP:** Enterprise Resource Planning (en español, Planificación de Recursos Empresariales), son sistemas de información gerenciales que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía en la producción de bienes o servicios.
- **Instanciar:** Se refiere a la acción de crear un objeto, sistema o aplicación.
- **Mainframes:** Computadora grande, potente y costosa, usada principalmente por grandes organizaciones para el procesamiento de una gran cantidad de datos; por ejemplo, para el procesamiento de transacciones bancarias.
- **MHTML:** Multipurpose Internet Mail Extension (en español, protocolo de transferencia de hipertexto Multiuso de la Extensión del Correo del Internet). Es un estándar para incluir recursos que en páginas HTTP usualmente están enlazados externamente, tal como los archivos de imágenes y sonido.
- **REST:** Representational State Transfer (en español, Transferencia de Estado Representacional), es un estilo de arquitectura de software para sistemas hipermedia distribuidos como la World Wide Web.

- **RUP:** Rational Unified Process (en español, Proceso Racional Unificado), es un proceso de desarrollo de software desarrollado por la empresa Rational Software, actualmente propiedad de IBM. Junto con el Lenguaje UML, constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.
- **SourceForge:** Es un sitio web de colaboración para proyectos de software.
- **Tenant:** En español, significa el arrendatario. En el presente trabajo se hace bastante referencia a este término debido a que, es el arrendatario o cliente quien tendrá que utilizar el servicio SaaS que se provea a través del software.
- **TIFF:** Es un formato de archivo informático para imágenes panorámicas y en tercera dimensión.
- **UML:** Unified Modeling Language (en español, Lenguaje Unificado de Modelado), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.
- **URL:** Uniform Resource Locator (en español, Localizador de Recursos Uniforme), es un identificador de recursos uniforme cuyos recursos referidos pueden cambiar, esto es, la dirección puede apuntar a recursos variables en el tiempo.
- **XML:** eXtensible Markup Language (en español, Lenguaje de Marcas Extensible), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

Los términos en el presente Glosario, han sido descritos según WIKIPEDIA (2015).