

**DISEÑO E IMPLEMENTACION DE UNA  
INFRAESTRUCTURA TECNOLOGICA CON FINES  
EDUCATIVOS SOBRE SISTEMAS DISTRIBUIDOS  
UTILIZANDO RASPBERRY PI.**

TRABAJO ESPECIAL DE GRADO

presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

como parte de los requisitos para optar al título de

INGENIERO EN INFORMÁTICA

REALIZADO POR

SAM COLINA, Héctor Félix.

TUTOR ACADEMICO

BARROETA, Carlos.

FECHA

Diciembre, 2013.

## **AGRADECIMIENTOS**

---

En primer lugar, quiero agradecerle a mi familia por haberme apoyado en estos meses, a mi padre, madre y hermanos, gracias por motivarme en los momentos difíciles.

A mi novia, Aileen Posadas, por haberme animado en todos estos meses y confiar en mí en todo momento.

A mi tutor, Carlos Barroeta, por su orientación y dedicación a la realización del presente trabajo de grado.

A los profesores de la Escuela de Ingeniería de Informática, en especial a: Lúcia Cardoso, Rafael Andara, Omar Hernández y Marcos Salazar. Muchas gracias por haberme formado como Ingeniero en Informática.

A los preparadores: David Hernández e Israel Fermín, muchas gracias por la motivación, dedicación y aprendizaje otorgado en cada una de sus clases.

A mis amigos y futuros colegas: Marcos Akerman, Juan Perozo, Luis Vicens, Alfredo Nava, Alejandro Martínez, Mimia Lo, Gary Bustillos, Juan Fuentes y Ángel Alberici por darme su amistad y confianza en todos los momentos que pasamos juntos en la universidad y fuera de ella.

¡A todos muchas gracias!

Héctor Sam.

## INDICE DE CONTENIDO

---

AGRADECIMIENTOS .....	ii
INDICE DE CONTENIDO .....	iii
INDICE DE TABLAS .....	xi
INDICE DE ILUSTRACIONES .....	xiii
SIPNOSIS .....	xvi
INTRODUCCIÓN .....	1
CAPÍTULO I. Problema.....	2
I.1. Planteamiento del Problema.....	2
I.2. Objetivos .....	4
I.2.1. Objetivo General.....	4
I.2.2. Objetivo Específicos .....	5
I.3. Alcance .....	6
I.4. Limitaciones .....	14
I.5. Justificación .....	15
CAPÍTULO II. Marco Teórico .....	17
II.1. Sistemas Distribuidos .....	17
II.2. Sockets .....	19
II.3. Raspberry Pi .....	21
II.4. Agente .....	22
CAPÍTULO III. Metodología .....	24
III.1. Historias de Usuario .....	24
III.2. Plan de iteraciones .....	25
III.3. Iteraciones .....	25
III.4. Pruebas de aceptación .....	26
III.5. Pequeña liberación .....	26
CAPÍTULO IV. Desarrollo .....	29
IV.1. Base de Datos.....	30
IV.2. Scripts de Sistemas Operativo .....	31
IV.3. Primera Iteración: Librería de Mensajes .....	34
IV.3.1. Historias de Usuarios .....	34

IV.3.2. Diseño (Tarjetas CRC).....	35
IV.3.3. Codificación.....	35
IV.3.4. Pruebas.....	37
IV.4. Segunda Iteración: Agente de Configuración.....	37
IV.4.1. Historias de Usuario.....	37
IV.4.2. Diseño (Tarjetas CRC).....	38
IV.4.3. Codificación.....	38
IV.4.4. Pruebas.....	41
IV.5. Tercera Iteración: Servicio de Recepción (Servidor Central).....	41
IV.5.1. Historias de Usuario.....	41
IV.5.2. Diseño (Tarjetas CRC).....	42
IV.5.3. Codificación.....	43
IV.5.4. Pruebas.....	45
IV.6. Cuarta Iteración: Aplicación Web.....	46
IV.6.1. Historias de usuario .....	47
IV.6.2. Diseño (Tarjetas CRC).....	49
IV.6.3. Codificación.....	52
IV.6.4. Pruebas.....	56
IV.7. Quinta Iteración: Aplicaciones de Sistema Distribuido.....	56
IV.7.1. Historias de usuario .....	57
IV.7.2. Diseño (Tarjetas CRC).....	57
IV.7.3. Codificación.....	58
IV.7.4. Pruebas.....	59
IV.8. Documentación .....	59
IV.8.1. Manual de uso.....	60
IV.8.2. Guía de diseño.....	60
IV.8.3. Guía de configuración.....	60
CAPÍTULO V. Resultados.....	61
V.1. Aplicaciones de sistemas distribuidos:.....	61
V.2. Agente de configuración.....	61
V.3. Servicio de recepción (Servidor central) .....	62
V.4. Librería de mensajes.....	62
V.5. Base de datos .....	62

V.6. Scripts de sistemas operativos.....	63
V.7. Módulo de administración de ciclo de vida.....	63
V.8. Módulo de monitoreo.....	63
V.9. Módulo de gestión .....	64
V.10. Aplicación web .....	64
V.11. Documentación .....	64
<b>CAPÍTULO VI. Conclusiones y Recomendaciones .....</b>	<b>65</b>
VI.1. Conclusiones .....	65
VI.2. Recomendaciones.....	67
<b>REFERENCIAS BIBLIOGRÁFICAS.....</b>	<b>70</b>
<b>ANEXOS.....</b>	<b>72</b>
Anexo 1. Backlogs.....	72
Anexo 2. Documento de Diseño.....	76
Anexo 2.1. Librería de mensajes.....	76
Anexo 2.1.1. Historias de usuario .....	76
Anexo 2.1.2. Diagrama de clases .....	77
Anexo 2.2. Agente de configuración .....	77
Anexo 2.2.1. Historias de usuario .....	77
Anexo 2.2.2. Diagrama de clases .....	78
Anexo 2.3. Servidor Central .....	78
Anexo 2.3.1. Historia de usuarios .....	78
Anexo 2.3.2. Diagrama de clases: .....	79
Anexo 2.4. Aplicación Web.....	80
Anexo 2.4.1. Historias de usuario: .....	80
Anexo 2.4.2. Diagrama de clases .....	82
Anexo 2.5. Aplicaciones Sistemas Distribuidos .....	85
Anexo 2.5.1. Historias de usuario .....	85
Anexo 2.5.2. Características de los sistemas distribuidos .....	86
Anexo 2.5.2.1. Historias de usuario.....	86
Anexo 2.5.2.2. Arquitectura .....	87
Anexo 2.5.2.3. Diagrama de clases servidor .....	87
Anexo 2.5.2.4. Diagrama de clases cliente .....	88
Anexo 2.5.2.5. Parámetros propios .....	88

Anexo 2.5.3. Desafíos de los Sistemas Distribuidos.....	88
Anexo 2.5.3.1. Historias de usuario.....	88
Anexo 2.5.3.2. Arquitectura .....	89
Anexo 2.5.3.3. Diseño (Tarjetas CRC) - Servidor.....	89
Anexo 2.5.3.4. Diagrama de clases servidor .....	90
Anexo 2.5.3.5. Diseño (Tarjetas CRC) - Cliente.....	90
Anexo 2.5.3.6. Diagrama de clases cliente .....	90
Anexo 2.5.3.7. Parámetros propios .....	91
Anexo 2.5.4. Arquitectura Cliente / Servidor .....	91
Anexo 2.5.4.1. Historias de usuario.....	91
Anexo 2.5.4.2. Arquitectura .....	92
Anexo 2.5.4.3. Diagrama de clases servidor .....	92
Anexo 2.5.4.4. Diagrama de clases cliente .....	92
Anexo 2.5.4.5. Parámetros propios .....	93
Anexo 2.5.5. Peer to peer (P2P) .....	93
Anexo 2.5.5.1. Historias de usuario.....	93
Anexo 2.5.5.2. Arquitectura .....	93
Anexo 2.5.5.3. Diagrama de clases.....	94
Anexo 2.5.5.4. Parámetros propios .....	94
Anexo 2.5.6. Sockets .....	94
Anexo 2.5.6.1. Historias de usuario.....	94
Anexo 2.5.6.2. Arquitectura .....	95
Anexo 2.5.6.3. Diagrama de clases.....	95
Anexo 2.5.6.4. Parámetros propios .....	95
Anexo 2.5.7. RMI .....	96
Anexo 2.5.7.1. Historias de usuario.....	96
Anexo 2.5.7.2. Arquitectura .....	96
Anexo 2.5.7.3. Diseño (Tarjetas CRC) - Servidor.....	96
Anexo 2.5.7.4. Diagrama de clases servidor .....	97
Anexo 2.5.7.5. Tarjetas CRC - Cliente .....	97
Anexo 2.5.7.6. Diagrama de clases cliente .....	97
Anexo 2.5.7.7. Parámetros propios .....	98
Anexo 2.5.8. Comunicación en grupo .....	98

Anexo 2.5.8.1. Historias de usuario.....	98
Anexo 2.5.8.2. Arquitectura .....	98
Anexo 2.5.8.3. Diagrama de clases servidor .....	98
Anexo 2.5.8.4. Diagrama de clases cliente .....	99
Anexo 2.5.8.5. Parámetros propios .....	99
<b>Anexo 2.5.9. Lamport.....</b>	<b>99</b>
Anexo 2.5.9.1. Historias de usuario.....	99
Anexo 2.5.9.2. Arquitectura .....	100
Anexo 2.5.9.3. Diseño (Tarjetas CRC) - Cliente.....	100
Anexo 2.5.9.4. Diagrama de clases.....	100
Anexo 2.5.9.5. Parámetros únicos.....	101
<b>Anexo 2.5.10. Algoritmo Cristian.....</b>	<b>101</b>
Anexo 2.5.10.1. Historias de usuario.....	101
Anexo 2.5.10.2. Arquitectura .....	102
Anexo 2.5.10.3. Diagrama de clases servidor .....	102
Anexo 2.5.10.4. Diseño (Tarjetas CRC) - Cliente.....	102
Anexo 2.5.10.5. Diagrama de clases cliente .....	103
Anexo 2.5.10.6. Parámetros propios .....	103
<b>Anexo 2.5.11. Algoritmo Berkeley.....</b>	<b>103</b>
Anexo 2.5.11.1. Historias de usuario.....	103
Anexo 2.5.11.2. Arquitectura .....	103
Anexo 2.5.11.3. Diseño (Tarjetas CRC) - Servidor.....	104
Anexo 2.5.11.4. Diagrama de clases.....	104
Anexo 2.5.11.5. Diagrama de clases cliente .....	105
Anexo 2.5.11.6. Parámetros propios .....	105
<b>Anexo 2.5.12. Algoritmo con Promedio.....</b>	<b>105</b>
Anexo 2.5.12.1. Historias de usuario.....	105
Anexo 2.5.12.2. Arquitectura .....	105
Anexo 2.5.12.3. Diseño (Tarjetas CRC) - Cliente.....	106
Anexo 2.5.12.4. Diagrama de clases.....	106
Anexo 2.5.12.5. Parámetros propios .....	107
<b>Anexo 2.5.13. Algoritmo Centralizado.....</b>	<b>107</b>
Anexo 2.5.13.1. Historias de usuario.....	107

Anexo 2.5.13.2. Arquitectura .....	107
Anexo 2.5.13.3. Diagrama de clases servidor .....	108
Anexo 2.5.13.4. Diagrama de clases cliente .....	109
Anexo 2.5.13.5. Parámetros propios .....	109
Anexo 2.5.14. Algoritmo distribuido .....	109
Anexo 2.5.14.1. Arquitectura .....	110
Anexo 2.5.14.2. Diseño (Tarjeta CRC) .....	110
Anexo 2.5.14.3. Diagrama de clases .....	110
Anexo 2.5.14.4. Parámetros propios .....	111
Anexo 2.5.15. Algoritmo Grandulón .....	111
Anexo 2.5.15.1. Arquitectura .....	112
Anexo 2.5.15.2. Diseño (Tarjetas CRC) .....	112
Anexo 2.5.15.3. Diagrama de clases .....	112
Anexo 2.5.15.4. Parámetros propios .....	113
Anexo 2.5.16. Algoritmo de Anillo .....	113
Anexo 2.5.16.1. Arquitectura .....	114
Anexo 2.5.16.2. Diagrama de clases .....	114
Anexo 2.5.16.3. Parámetros propios .....	114
Anexo 2.5.17. Tipos de Fallas .....	115
Anexo 2.5.17.1. Historias de usuario .....	115
Anexo 2.5.17.2. Arquitectura .....	115
Anexo 2.5.17.3. Diagrama de clases - servidor .....	116
Anexo 2.5.17.4. Diagrama de clases - cliente .....	116
Anexo 2.5.17.5. Parámetros propios .....	116
Anexo 2.5.18. Fallas Bizantinas .....	117
Anexo 2.5.18.1. Historias de usuario .....	117
Anexo 2.5.18.2. Arquitectura .....	117
Anexo 2.5.18.3. Diagrama de clases .....	118
Anexo 2.5.18.4. Parámetros propios .....	119
Anexo 2.5.19. Modelo Acceso Remoto .....	119
Anexo 2.5.19.1. Historias de usuario .....	119
Anexo 2.5.19.2. Arquitectura .....	119
Anexo 2.5.19.3. Diagrama de clases - servidor .....	119

Anexo 2.5.19.4. Diagrama de clases - cliente .....	120
Anexo 2.5.19.5 .Parámetros propios .....	120
Anexo 2.5.20. Modelo Carga y Descarga .....	120
Anexo 2.5.20.1. Historias de usuario.....	120
Anexo 2.5.20.2. Arquitectura .....	121
Anexo 2.5.20.3. Diseño (Tarjetas CRC) - Servidor.....	121
Anexo 2.5.20.4. Diagrama de clases servidor.....	121
Anexo 2.5.20.5. Diseño (Tarjetas CRC) - Cliente.....	122
Anexo 2.5.20.6. Diagrama de clases.....	122
Anexo 2.5.20.7. Parámetros propios: .....	123
Anexo 2.5.21. Sistemas de archivos basados en Clúster .....	123
Anexo 2.5.21.1. Historias de usuario.....	123
Anexo 2.5.21.2. Arquitectura .....	124
Anexo 2.5.21.3. Diseño (Tarjetas CRC) - Servidor.....	124
Anexo 2.5.21.4. Diagrama de clases servidor .....	125
Anexo 2.5.21.5. Diseño (Tarjetas CRC) - Cliente.....	125
Anexo 2.5.21.6. Diagrama de clases cliente .....	125
Anexo 2.5.21.7. Parámetros propios .....	126
Anexo 2.5.22. DNS .....	126
Anexo 2.5.22.1. Historias de usuario.....	126
Anexo 2.5.22.2. Arquitectura .....	127
Anexo 2.5.22.3. Diagrama de clases servidor .....	127
Anexo 2.5.22.4. Diagrama de clases cliente .....	127
Anexo 2.5.22.5. Parámetros propios .....	128
Anexo 2.5.23. LDAP .....	128
Anexo 2.5.23.1. Historias de usuario.....	128
Anexo 2.5.23.2. Arquitectura .....	128
Anexo 2.5.23.3. Diseño (Tarjetas CRC) - Servidor.....	129
Anexo 2.5.23.4. Diagrama de clases servidor .....	129
Anexo 2.5.23.5. Diagrama de clases cliente .....	129
Anexo 2.5.23.6. Parámetros propios .....	129
Anexo 2.5.24. RMI .....	130
Anexo 2.5.24.1. Historias de usuario.....	130

Anexo 2.5.24.2. Arquitectura .....	130
Anexo 2.5.24.3. Diseño (Tarjetas CRC) - Servidor.....	130
Anexo 2.5.24.4. Diagrama de clases servidor. ....	131
Anexo 2.5.24.5. Diseño (Tarjetas CRC) - Cliente .....	131
Anexo 2.5.24.6. Diagrama de clases cliente.....	131
Anexo 2.5.24.7. Parámetros propios .....	132
<b>Anexo 2.5.25. EJB.....</b>	<b>132</b>
Anexo 2.5.25.1. Historias de usuario .....	132
Anexo 2.5.25.2. Arquitectura .....	132
Anexo 2.5.25.3. Diagrama de clases .....	133
Anexo 2.5.25.4. Parámetros propios .....	133
<b>Anexo 2.5.26. SOAP .....</b>	<b>133</b>
Anexo 2.5.26.1. Historias de usuario .....	133
Anexo 2.5.26.2. Arquitectura .....	133
Anexo 2.5.26.3. Diseño (Tarjetas CRC) - Servidor.....	134
Anexo 2.5.26.4. Diagrama de clases servidor .....	134
Anexo 2.5.26.5. Diagrama de clases cliente.....	135
Anexo 2.5.26.6. Parámetros propios .....	135
<b>Anexo 2.5.27. REST.....</b>	<b>135</b>
Anexo 2.5.27.1. Historias de usuario .....	135
Anexo 2.5.27.2. Arquitectura .....	136
Anexo 2.5.27.3. Diseño (Tarjetas CRC) - Servidor.....	136
Anexo 2.5.27.4. Diagrama de clases servidor .....	137
Anexo 2.5.27.5. Diagrama de clases cliente.....	137
Anexo 2.5.27.6. Parámetros únicos.....	137
<b>Anexo 3. Manual de Usuario.....</b>	<b>138</b>
<b>Anexo 4. Guía de configuración.....</b>	<b>153</b>
Anexo 4.1. Requisitos Hardware / Software .....	153
Anexo 4.2. Configuración .....	154
Anexo 4.3. Para desplegar aplicaciones de sistemas distribuidos: .....	155
<b>Anexo 5. Scripts de Sistema Operativos .....</b>	<b>156</b>

## INDICE DE TABLAS

---

Tabla 1: Aplicaciones Sistemas Distribuidos .....	7
Tabla 2: Aplicación metodología XP.....	28
Tabla 3: Tarjetas CRC Librería de mensajes.....	35
Tabla 4: Tarjetas CRC Agente de configuración.....	38
Tabla 5: Tarjetas CRC Servidor Central.....	43
Tabla 6: Tarjetas CRC Módulo de gestión.....	50
Tabla 7: Tarjetas CRC Módulo de administración del ciclo de vida.....	50
Tabla 8: Tarjetas CRC Módulo de monitoreo.....	51
Tabla 9: Tarjetas CRC Aplicación web.....	52
Tabla 10: Tarjetas CRC Aplicaciones Sistemas Distribuidos.....	57
Tabla 11: Backlogs historias de usuario.....	76
Tabla 12: Historias de usuario Librería de mensajes.....	76
Tabla 13: Historias de usuario Agente de configuración.....	78
Tabla 14: Historias de usuario Servidor central.....	79
Tabla 15: Historias de usuario Módulo de gestión.....	80
Tabla 16: Historias de usuario Módulo de administración del ciclo de vida.....	81
Tabla 17: Historias de usuario Módulo de monitoreo.....	81
Tabla 18: Historias de usuario Aplicación web.....	81
Tabla 19: Historias de usuario Aplicaciones de sistemas distribuidos.....	86
Tabla 20: Historias de usuario Características de los sistemas distribuidos..	87
Tabla 21: Historias de usuario Desafíos de los sistemas distribuidos.....	88
Tabla 22: Tarjetas CRC Desafíos de los sistemas distribuidos - Servidor.....	89
Tabla 23: Tarjetas CRC Desafíos de los sistemas distribuidos - Cliente.....	90
Tabla 24: Historias de usuario Arquitectura Cliente / Servidor.....	91
Tabla 25: Historias de usuario Peer to peer.....	93
Tabla 26: Historias de usuario Sockets.....	95
Tabla 27: Historias de usuario RMI 1.....	96
Tabla 28: Tarjetas CRC RMI 1 - Servidor.....	96
Tabla 29: Tarjetas CRC RMI 1 - Cliente.....	97
Tabla 30: Historias de usuario Comunicación en grupo.....	98
Tabla 31: Historias de usuario Algoritmo de Lamport.....	99
Tabla 32: Tarjetas CRC Algoritmo de Lamport.....	100
Tabla 33: Historias de usuario Algoritmo de Cristian.....	101
Tabla 34: Tarjetas CRC Algoritmo de Cristian - Cliente.....	102
Tabla 35: Historias de usuario Algoritmo de Berkeley.....	103
Tabla 36: Tarjetas CRC Algoritmo de Berkeley - Servidor.....	104
Tabla 37: Historias de usuario Algoritmo con promedio.....	105
Tabla 38: Tarjetas CRC Algoritmo con promedio.....	106
Tabla 39: Historias de usuario Algoritmo centralizado.....	107

Tabla 40: Historias de usuario Algoritmo distribuido.....	110
Tabla 41: Tarjetas CRC Algoritmo distribuido.....	110
Tabla 42: Historias de usuario Algoritmo grandulón.....	111
Tabla 43: Tarjetas CRC Algoritmo grandulón.....	112
Tabla 44: Historias de usuario Algoritmo de anillo.....	113
Tabla 45: Historias de usuario Tipos de fallas.....	115
Tabla 46: Historias de usuario Fallas Bizantinas.....	117
Tabla 47: Historias de usuario Modelo acceso remoto.....	119
Tabla 48: Historias de usuario Modelo carga y descarga.....	120
Tabla 49: Tarjetas CRC Modelo carga y descarga - Servidor.....	121
Tabla 50: Tarjetas CRC Modelo carga y descarga - Cliente.....	122
Tabla 51: Historias de usuario Sistemas de archivos basados en clúster.....	123
Tabla 52: Tarjetas CRC Sistemas de archivos basados en clúster - Servidor .....	124
Tabla 53: Tarjetas CRC Sistema de archivos basados en cluster - Cliente.....	125
Tabla 54: Historias de usuario DNS.....	126
Tabla 55: Historias de usuario LDAP.....	128
Tabla 56: Tarjetas CRC LDAP - Servidor.....	129
Tabla 57: Historias de usuario RMI 2.....	130
Tabla 58: Tarjetas CRC RMI 2 - Servidor.....	130
Tabla 59: Tarjetas CRC RMI 2 - Cliente.....	131
Tabla 60: Historias de usuario EJB.....	132
Tabla 61: Historias de usuario SOAP.....	133
Tabla 62: Tarjetas CRC SOAP - Servidor.....	134
Tabla 63: Historias de usuario REST.....	136
Tabla 64. Tarjetas CRC REST - Servidor.....	136

## INDICE DE ILUSTRACIONES

---

Ilustración 1: Arquitectura de la infraestructura. ....	13
Ilustración 2: Petición Cliente – Servidor. ....	20
Ilustración 3: Respuesta Servidor – Cliente. ....	20
Ilustración 4: Componentes de un Raspberry Pi. ....	22
Ilustración 5: Ciclo de vida en XP. ....	24
Ilustración 6: Diagrama Entidad Relación. ....	30
Ilustración 7: Autenticación llave pública - privada. ....	32
Ilustración 8: Diagrama de actividades para la ejecución de <i>scripts</i> . ....	33
Ilustración 9: Diagrama de clases Librería de mensajes. ....	77
Ilustración 10: Diagrama de clases Agente de configuración. ....	78
Ilustración 11: Diagrama de clases Servidor central. ....	79
Ilustración 12: Diagrama de clases Módulo ciclo de vida y Módulo de gestión. ....	82
Ilustración 13: Diagrama de clases Aplicación web. ....	83
Ilustración 14: Diagrama de clases Módulo de monitoreo. ....	84
Ilustración 15: Diagrama de clases relaciones Aplicación Web y Módulos. ....	85
Ilustración 17: Diagrama de clases características de los sistemas distribuidos - Servidor. ....	87
Ilustración 16: Arquitectura Características de los sistemas distribuidos. ....	87
Ilustración 18: Diagrama de clases Características de los sistemas distribuidos - Cliente. ....	88
Ilustración 19: Arquitectura Desafíos de los sistemas distribuidos. ....	89
Ilustración 20: Diagrama de clases Desafíos de los sistemas distribuidos - Servidor. ....	90
Ilustración 21: Diagrama de clases Desafíos de los sistemas distribuidos - Cliente. ....	91
Ilustración 22: Arquitectura Arquitectura Cliente / Servidor. ....	92
Ilustración 23: Diagrama de clases Arquitectura Cliente / Servidor - Servidor. ....	92
Ilustración 24: Diagrama de clases Arquitectura Cliente / Servidor - Cliente. ....	93
Ilustración 25: Arquitectura Peer to peer. ....	94
Ilustración 26: Diagrama de clases Peer to peer. ....	94
Ilustración 27: Arquitectura Sockets. ....	95
Ilustración 28: Diagrama de clases Sockets. ....	95
Ilustración 29: Arquitectura RMI 1. ....	96
Ilustración 30: Diagrama de clases RMI 1 - Servidor. ....	97
Ilustración 31: Diagrama de clases RMI 1 - Cliente. ....	97
Ilustración 32: Arquitectura Comunicación en grupo. ....	98
Ilustración 33: Diagrama de clases Comunicación en grupo - Servidor. ....	99
Ilustración 34: Diagrama de clases Comunicación en grupo - Cliente. ....	99

Ilustración 35: Arquitectura Algoritmo de Lamport. ....	100
Ilustración 36: Diagrama de clases Algoritmo de Lamport. ....	100
Ilustración 37: Arquitectura Algoritmo de Cristian. ....	102
Ilustración 38: Diagrama de clases Algoritmo de Cristian - Servidor. ....	102
Ilustración 39: Diagrama de clases Algoritmo de Cristian - Cliente. ....	103
Ilustración 40: Arquitectura Algoritmo de Berkeley. ....	104
Ilustración 41: Diagrama de clases Algoritmo de Berkeley - Servidor. ....	104
Ilustración 42: Diagrama de clases Algoritmo de Berkeley - Cliente. ....	105
Ilustración 43: Arquitectura Algoritmo con promedio. ....	106
Ilustración 44: Diagrama de clases Algoritmo con promedio. ....	107
Ilustración 45: Arquitectura Algoritmo centralizado. ....	108
Ilustración 46: Diagrama de clases Algoritmo centralizado - Servidor. ....	108
Ilustración 47: Diagrama de clases Algoritmo centralizado. ....	109
Ilustración 48: Arquitectura Algoritmo distribuido. ....	110
Ilustración 49: Diagrama de clases Algoritmo distribuido. ....	111
Ilustración 50: Arquitectura Algoritmo grandulón. ....	112
Ilustración 51: Diagrama de clases Algoritmo grandulón. ....	113
Ilustración 52: Arquitectura Algoritmo de anillo. ....	114
Ilustración 53: Diagrama de clases Algoritmo de anillo. ....	114
Ilustración 54: Arquitectura Tipos de fallas. ....	116
Ilustración 55: Diagrama de clases Tipos de fallas - Servidor. ....	116
Ilustración 56: Diagramas de clases Tipos de Fallas - Cliente. ....	116
Ilustración 57: Arquitectura Fallas Bizantinas. ....	118
Ilustración 58: Diagrama de clases Fallas Bizantinas. ....	118
Ilustración 59: Arquitectura Modelo acceso remoto. ....	119
Ilustración 60: Diagrama de clases Modelo acceso remoto - Servidor. ....	120
Ilustración 61: Diagrama de clases Modelo acceso remoto - Cliente. ....	120
Ilustración 62: Arquitectura Modelo carga y descarga. ....	121
Ilustración 63: Diagrama de clases Modelo carga y descarga - Servidor. ....	122
Ilustración 64: Diagrama de clases Modelo carga y descarga - Cliente. ....	123
Ilustración 65: Arquitectura Sistemas de archivos basados en clúster. ....	124
Ilustración 66: Diagrama de clases Sistemas de archivos basados en cluster - Servidor. ....	125
Ilustración 67: Diagrama de clases Sistemas de archivos basados en cluster - Cliente. ....	126
Ilustración 68: Arquitectura DNS. ....	127
Ilustración 69: Diagrama de clases DNS - Servidor. ....	127
Ilustración 70: Diagrama de clases DNS - Cliente. ....	127
Ilustración 71: Arquitectura LDAP. ....	128
Ilustración 72: Diagrama de clases LDAP - Servidor. ....	129
Ilustración 73: Diagrama de clases LDAP - Cliente. ....	129
Ilustración 74: Arquitectura RMI 2. ....	130
Ilustración 75: Diagrama de clases RMI 2 - Servidor. ....	131
Ilustración 76: Diagrama de clases RMI 2 - Cliente. ....	131
Ilustración 77: Arquitectura EJB. ....	132

Ilustración 78: Diagrama de clases EJB.	133
Ilustración 79: Arquitectura SOAP.	134
Ilustración 80: Diagrama de clases SOAP - Servidor.	135
Ilustración 81: Diagrama de clases SOAP - Cliente.	135
Ilustración 82: Arquitectura REST.	136
Ilustración 83: Diagrama de clases REST - Servidor.	137
Ilustración 84: Diagrama de clases REST - Cliente.	137
Ilustración 85: Página principal.	138
Ilustración 86: Funcionalidades página principal.	139
Ilustración 87: Tópicos de la materia.	139
Ilustración 88: Tópicos disponibles.	140
Ilustración 89: Tópicos disponibles 2.	140
Ilustración 90: Seleccionar aplicación.	141
Ilustración 91: Página de la aplicación - Descripción.	141
Ilustración 92: Página de la aplicación - Instrucciones.	142
Ilustración 93: Información del agente.	143
Ilustración 94: Escenarios aplicación.	144
Ilustración 95: Sección preguntas.	144
Ilustración 96: PDF preguntas.	145
Ilustración 97: Operaciones en base de datos.	145
Ilustración 98: <i>Login</i> .	146
Ilustración 99: Desabilitar botón módulo de gestión.	146
Ilustración 100: Módulo de gestión.	147
Ilustración 101: Gestión usuarios.	147
Ilustración 102: Gestión tópicos 1.	148
Ilustración 103: Gestión tópicos 2.	149
Ilustración 104: Gestión preguntas 1.	150
Ilustración 105: Gestión preguntas 2.	150
Ilustración 106: Gestión aplicación 1.	151
Ilustración 107: Gestión aplicación 2.	152
Ilustración 108: Script ejecutar.sh.	156
Ilustración 109: Script eliminarNodo.sh.	156
Ilustración 110: Script eliminarTodosNodos.sh.	156

## SIPNOSIS

---

El presente trabajo de grado titulado "Diseño e implementación de una infraestructura tecnológica con fines educativos sobre sistemas distribuidos" tiene como propósito ayudar a estudiantes de ingeniería informática a reforzar los conocimientos de la cátedra sistemas distribuidos, mediante aplicaciones orientadas a los tópicos de la materia. Dichas aplicaciones fueron desplegadas en computadoras *Raspberry Pi* que simulan los nodos del sistema y se controlan a través de una aplicación web.

Como esquema metodológico para el desarrollo de la solución se adoptó *eXtreme Programming* (XP) que permite obtener los requerimientos a través de historias de usuarios y la colaboración del cliente, enfocándose en entregar software funcional mediante iteraciones que conllevan a la liberación de una porción del sistema al finalizar cada una de ellas.

El producto final consiste en: Una aplicación web donde se pueden ejecutar pequeñas unidades de sistemas distribuidos divididas por tópico, un servidor central que despliega las aplicaciones en los nodos *Raspberry Pi* y una base de datos para obtener la información correspondiente a los tópicos y aplicaciones.

## INTRODUCCIÓN

---

Sistemas distribuidos, es una cátedra perteneciente al octavo semestre de la carrera Ingeniería Informática en la Universidad Católica Andrés Bello. En ella los estudiantes aprenden sobre conceptos relacionados con un conjunto de computadoras interconectadas entre sí para realizar una tarea en común.

Actualmente en la Escuela de Ingeniería Informática no existe un mecanismo que le permita a los estudiantes reforzar los conocimientos acerca de un tópico o tema de sistemas distribuidos. Se debe recurrir a los libros recomendados por el profesor o al internet para buscar información.

Este trabajo especial de grado pretende servir como una herramienta para reforzar los conocimientos de la cátedra, para ello, se utilizarán pequeñas computadoras *Raspberry Pi* que permitan simular los nodos de un sistema distribuido. Se desarrollarán aplicaciones orientadas a los tópicos de la cátedra y se asociaran escenarios que permitan comparar lo indicado por la teoría y lo obtenido por la aplicación, todo esto mediante una aplicación web donde el usuario podrá interactuar con los nodos a través de acciones generadas por componentes visuales, enviando mensajes, iniciando o deteniendo un nodo particular.

# CAPÍTULO I. Problema

---

## I.1. Planteamiento del Problema

Desde el año 2008 la Escuela de Ingeniería Informática de la Universidad Católica Andrés Bello dicta la cátedra de Sistemas Distribuidos, anteriormente llamada Sistemas de Operación II, en ella, se profundizan temas en relación a diferentes nodos conectados entre sí para proporcionar una funcionalidad específica. Es una materia teórico – práctica que posee proyecto y se necesita obtener un mínimo de 4.75 puntos en cada parte para ser aprobada.

Muchas materias de Ingeniería Informática cuentan con prácticas para reforzar los conocimientos adquiridos en la teoría, estas son realizadas dentro de un laboratorio, donde los estudiantes forman equipos de trabajo y logran realizar una serie de actividades que permiten asociar los conocimientos de la materia de forma directa. Normalmente estas prácticas poseen una duración de dos horas y en algunos casos se entregan informes que validan que los estudiantes concluyeron con los objetivos de la clase.

Hoy en día la materia de sistemas distribuidos se dicta con cuatro horas semanales, dónde los estudiantes aprenden todo lo relacionado con procesos e hilos, comunicación, sincronización, replicación, sistemas de archivos, servicios de nombre, objetos distribuidos y servicios web. Los

estudiantes atienden a la explicación de los profesores para un tema particular. Algunos temas pueden tardar más tiempo que otros debido a su complejidad. Por lo que a veces no pueden ser captados por los estudiantes de forma continua, es decir, en una misma semana de clases. Los profesores emiten los proyectos de la materia con la finalidad de que los estudiantes puedan aplicar los conocimientos teóricos adquiridos en clases para resolver el problema planteado en el enunciado.

Para realizar el proyecto, los alumnos se organizan en equipos, dos o tres personas como máximo. Dependiendo de las exigencias del proyecto, ellos deben buscar información acerca de los temas que se relacionan con el problema a resolver. Los apuntes de cada estudiante y las láminas por el profesor no siempre son suficientes para reforzar los conceptos, el uso del internet ha permitido que muchos estudiantes lo utilicen como su principal fuente de información para resolver dudas en relación a un concepto o tema que su significado no se encuentre muy claro.

Se propone diseñar e implementar una infraestructura tecnológica con fines educativos que permita reforzar los conocimientos en relación a los sistemas distribuidos mediante aplicaciones orientadas a los tópicos de la cátedra. Cada aplicación será desplegadas en pequeñas computadoras llamadas *Raspberry Pi* para simular los nodos de un sistemas distribuido. Las aplicaciones contendrán parámetros de configuración que serán almacenados en una base de datos al igual que la información teórica

asociada al tópico. Los usuarios controlarán los nodos a través de un módulo de ciclo de vida encargado de enviar instrucciones utilizando *scripts* de sistema operativo que se conecten a los nodos mediante el protocolo *Secure Shell* (SSH).

Cada nodo contendrá un agente de configuración que permite recolectar información sobre el estado del *Raspberry Pi* y las aplicaciones en ejecución, posteriormente se envía la información a un módulo de monitoreo que le permite al usuario visualizar los mensajes intercambiados entre los nodos. Los mensajes son enviados mediante una librería de registro de mensajes. La comunicación dentro de la infraestructura se realizará mediante sockets. Por último, los usuarios pueden crear sus propias aplicaciones con sus respectivos parámetros de configuración e información teórica y almacenarlas en la base de datos para utilizarlas consecutivamente dentro de la infraestructura tecnológica.

## I.2. Objetivos

### I.2.1. Objetivo General

Diseñar e implementar una infraestructura tecnológica con fines educativos que permita la consolidación de conocimientos sobre la cátedra de Sistemas Distribuidos mediante la simulación de nodos utilizando *Raspberry Pi* para estudiantes de la Escuela de Ingeniería Informática de la Universidad Católica Andrés Bello.

### I.2.2. Objetivo Específicos

- Desarrollar un conjunto de aplicaciones educativas que permitan la simulación de nodos de un sistema distribuido en función a los tópicos de la cátedra.
- Desarrollar un agente de configuración que permita recolectar información del estado actual de los nodos que forman parte del sistema distribuido.
- Desarrollar un servicio de recepción capaz de procesar la información proveniente de los agentes de configuración instalados en los nodos.
- Desarrollar una librería que encapsule un componente para el registro de mensajes de entrada y/o salida que será incorporada en los componentes de la infraestructura y en las aplicaciones de sistemas distribuidos.
- Diseñar e implementar una base de datos que permita el almacenamiento de los parámetros de configuración y la información teórica correspondiente a las aplicaciones que forman parte de los tópicos del contenido programático de la cátedra de Sistemas Distribuidos.
- Crear *scripts* de sistema operativo que serán desplegados en los nodos para controlar aspectos como: parametrización, ejecución y detención de las aplicaciones de sistemas distribuidos.
- Desarrollar un módulo de administración que permita controlar el ciclo de vida de los nodos que forman parte de la infraestructura tecnológica.

- Implementar un módulo de monitoreo que permita visualizar el estado de cada nodo así como los mensajes capturados por la librería de gestión de mensajes.
- Desarrollar un módulo de gestión que permita la creación, modificación y eliminación de tópicos y preguntas así como la incorporación de aplicaciones pertenecientes a la cátedra de Sistemas Distribuidos.
- Integrar los módulos de ciclo de vida, monitoreo y gestión a través de una aplicación web que permita la interacción del usuario con la infraestructura tecnológica.
- Elaborar la documentación de los componentes desarrollados pertenecientes a la infraestructura tecnológica.

### I.3. Alcance

Las aplicaciones educativas sobre sistemas distribuidos se basarán en el contenido programático de la cátedra Sistemas Distribuidos de la Escuela de Ingeniería Informática de la UCAB, cada aplicación estará asociada a un tópico y permiten consolidar los conocimientos de acuerdo a escenarios definidos. Un escenario es un posible resultado que se puede obtener por un evento generado por el usuario. La siguiente tabla indica los tópicos que fueron seleccionados para desarrollarse como aplicaciones:

Tema	Aplicaciones
Introducción a los Sistemas Distribuidos	<b>Características de los Sistemas Distribuidos</b> <b>Desafíos de los Sistemas Distribuidos</b> <b>Arquitecturas:</b> Cliente / Servidor Punto a Punto (P2P)
Comunicación en Sistemas Distribuidos	<b>Sockets</b> <b>RMI</b> <b>Comunicación en grupo</b>
Sincronización en ambientes distribuidos	<b>Relojes lógicos:</b> Algoritmo de Lamport <b>Relojes físicos:</b> Algoritmo de Cristian Algoritmo de Berkeley Algoritmo con promedio (distribuido) <b>Exclusión mutua:</b> Algoritmo Centralizado Algoritmo Distribuido (Ricart y Agrawala) <b>Algoritmos de selección:</b> Grandulón Anillo
Replicación	<b>Tipos de Fallas</b> <b>Fallas Bizantinas</b>
Sistemas de archivos distribuidos	<b>Arquitectura Cliente Servidor:</b> Modelo de acceso remoto Modelo de carga y descarga <b>Sistemas de archivos basados en Clúster</b>
Servicios de nombre	<b>Domain Name System (DNS)</b> <b>Lightweight Directory Access Protocol (LDAP)</b>
Objetos Distribuidos	<b>Remote Method Invocation (RMI)</b> <b>Enterprise JavaBeans (EJB)</b>
Servicios Web	<b>Simple Object Access Protocol (SOAP) (Servidor / Cliente)</b> <b>Representational State Transfer (REST) (Servidor / Cliente)</b>

Tabla 1: Aplicaciones Sistemas Distribuidos. Elaborado por: Héctor Sam

Las aplicaciones se desarrollarán utilizando Java como lenguaje de programación, contarán con patrones de diseño para garantizar las buenas prácticas del desarrollo de software, pruebas unitarias que validen que el

comportamiento de cada método es correcto y estándares de codificación para permitir que cualquier desarrollador pueda comprender el código sin problemas. Adicionalmente, poseen parámetros de configuración que son obtenidos desde una base de datos.

Los nodos se simularán utilizando *Raspberries Pi*, consisten en pequeñas computadoras que ejecutan el sistema operativo Linux y poseen la máquina virtual de Java. Se encontrarán conectadas a una red local para permitir la comunicación entre ellas y los elementos que conforman la infraestructura tecnológica.

La comunicación entre el usuario y las aplicaciones de sistemas distribuidos se realizará a través de un servidor central, quien será el encargado de enviar y recibir datos por parte de los nodos. Los ejecutables de cada aplicación se encontrarán alojados en un repositorio local dentro del servidor, permitiendo enviar las aplicaciones a cada nodo con sus respectivos parámetros en el momento que el usuario lo indique desde el módulo de ciclo de vida.

El agente de configuración consiste en una aplicación java que se encontrará alojada en todos los nodos *Raspberries*. Al encender el nodo, el agente se iniciará automáticamente y recopilará información del sistema operativo como por ejemplo: configuración de red, procesos en ejecución, memoria disponible, entre otros. Adicionalmente, indicará la aplicación activa

que se estará ejecutando (Nombre de la aplicación, estado actual, número de nodo, información propia de la aplicación). El agente tendrá configurado la dirección de red donde se encuentra en ejecución el servidor central para el envío de la información recopilada.

El servicio de recepción será el encargado de procesar la información enviada por los agentes de configuración y por las aplicaciones de los nodos para posteriormente ser visualizada por los usuarios. El servicio será ejecutado en el servidor central, este recibe los mensajes enviados a través de la librería de recepción de mensajes y los redirige al módulo de monitoreo para que el usuario pueda observar la información correspondiente.

La librería para el registro de los mensajes de entrada y salida se desarrollará en Java. Tendrá atributos configurables dónde se indicará la dirección ip del remitente y de los destinatarios, el mensaje a enviar, fecha y hora del mensaje, entre otros. Se encontrará en todos los componentes de la infraestructura.

Se creará una base de datos que almacene la información de los tópicos como por ejemplo: definiciones, puntos a tratar, imágenes, preguntas, entre otros. Permitirá almacenar los parámetros de configuración necesarios para ejecutar cada aplicación en los nodos (dirección ip, puertos, parámetros propios de las aplicaciones). Guardará las rutas del repositorio local (path) donde se encontrarán ubicados los ejecutables de las aplicaciones.

Los *scripts* de sistema operativo (*Shell scripts*) se encontrarán almacenados dentro del servidor central, permiten que el usuario pueda controlar las aplicaciones en cualquier momento desde el módulo de ciclo de vida. Cada script posee instrucciones de la acción a realizar (ejecutar la aplicación, detener un nodo particular, detener todos los nodos, entre otros). Mediante el protocolo ssh se accede a los nodos para realizar la acción deseada.

El módulo de administración de ciclo de vida le permite al usuario desplegar y controlar las aplicaciones sobre sistemas distribuidos en los nodos *Raspberry Pi*. Mediante un conjunto de botones se mostrarán los eventos disponibles para la aplicación actual (ejecutar, detener un nodo particular, detener todos los nodos, entre otros). El usuario selecciona un evento específico y el módulo envía un mensaje a través de la librería al servidor central para localizar el script correspondiente para su ejecución.

El módulo de monitoreo permite que el usuario pueda observar los eventos que ocurran en las aplicaciones distribuidas, los mensajes enviados entre cada nodo y su estado actual. Mediante la librería de mensajes se recibirá la información enviada por los nodos y por los agentes de configuración al servicio de recepción, este último reenvía la información al módulo de monitoreo separándola en dos tipos: información de los nodos (mensajes entre ellos), información del ambiente distribuido (cantidad de nodos, estado, aplicación activa, entre otros).

El módulo de gestión permitirá que se puedan agregar nuevos tópicos a la infraestructura con su debida aplicación, existirán formularios que permitan introducir la información del tópico que se está creando, adicionalmente se tiene la opción de subir los ejecutables de las nuevas aplicaciones y un formulario de parámetros de configuración que posteriormente será guardado en la base de datos. Se requiere un registro previo del usuario para poder utilizar el módulo de gestión. Cuando un usuario almacene una aplicación dentro del repositorio local, tendrá la opción de colocar la cantidad de parámetros necesarios a través de un formulario dinámico. Donde cada uno tendrá la siguiente sintaxis: Parámetro1 = Valor1, Parámetro2 = Valor2.

El módulo de gestión permite generar preguntas acerca de un tópico, cada pregunta se encuentra relacionada con los conceptos teóricos o las aplicaciones educativas. Poseen el esquema de selección simple, donde se muestran varias opciones y solo una es la respuesta correcta. Existirán un conjunto de preguntas por cada tópico almacenadas en la base de datos y se podrán generar aleatoriamente, permitiendo que se puedan crear diferentes modelos de pruebas para un mismo tópico. Se creará un archivo en formato PDF por cada modelo para ser descargado por los usuarios de la infraestructura. El módulo permite realizar las operaciones de agregar, modificar o eliminar preguntas a la base de datos.

Los módulos desarrollados se integrarán mediante una aplicación web compuesta por una interfaz gráfica amigable, intuitiva y de fácil uso para todos los usuarios. Al acceder a la aplicación web, el usuario observa una lista con todos los temas de la materia y los tópicos asociados. Cuando se selecciona un tópico se muestra una descripción sobre el objetivo del mismo, las aplicaciones disponibles y la opción de generar preguntas. El usuario al hacer clic sobre la aplicación del tópico es redirigido a otra página donde se visualiza una pequeña introducción de la aplicación, las instrucciones de despliegue, los posibles escenarios que pueden surgir al generarse un evento, los módulos de administración de ciclo de vida y de monitoreo.

Las aplicaciones sobre sistemas distribuidos tendrán asociadas unos escenarios de acuerdo a los eventos generados por el usuario. Cuando se finalice la ejecución de la aplicación, el usuario compara el resultado obtenido con los escenarios predefinidos para comprobar que se haya cumplido con el objetivo del tópico.

La documentación se encuentra relacionada con cada componente desarrollado, se realizará un manual de uso de la infraestructura que puedan orientar al usuario al utilizar los *Raspberries Pi*, de igual forma se crearán documentos de creación que indiquen como se realizó cada uno de los componentes, incluyendo diagramas de diseño (historias de usuario, entidad relación, de clases, entre otros). Por último, se creará una guía de configuración que permita instalar y configurar toda la infraestructura para su

posterior funcionamiento, indicando los requerimientos de hardware, versiones del software, pasos para configurar el ambiente y como asociar aplicaciones realizadas por otros usuarios.

La ilustración 1 muestra la arquitectura de la infraestructura planteada, donde se obtienen cuatro componentes: Una aplicación web a la que accede el usuario de la infraestructura, un servidor central, un servidor de base de datos y cuatro *Raspberries Pi* que funcionarán como nodos del sistema distribuido:

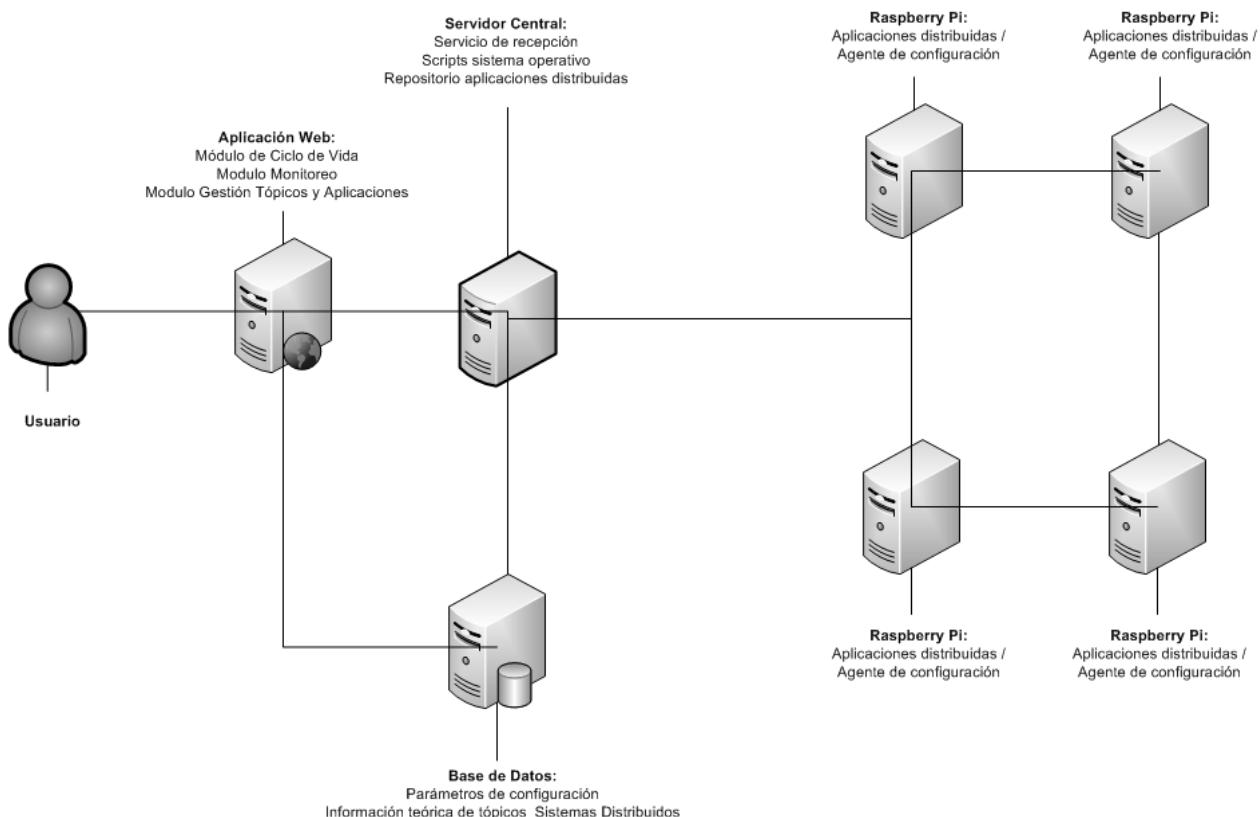


Ilustración 1: Arquitectura de la infraestructura. Elaboración propia

#### I.4. Limitaciones

- Los tópicos a desarrollar estarán basados únicamente en los temas del programa de sistemas distribuidos para estudiantes de ingeniería informática de la UCAB.
- Los *scripts* a realizar sólo podrán ser ejecutados bajo sistemas operativos Unix / Linux.
- Existirán un máximo de cuatro *Raspberries Pi* que funcionen como nodos.
- Las aplicaciones a desarrollar sobre sistemas distribuidos no poseen interfaz gráfica, por lo que todas las acciones que ocurran dentro de ellas solo podrán ser visualizadas por salidas enviadas por la librería desarrollada. En caso que se requiera enviar una información a un nodo particular el usuario lo podrá indicar desde el módulo de ciclo de vida.
- Las aplicaciones a desarrollar sobre sistemas distribuidos se enfocarán en resolver el objetivo del tópico, no poseen un alto grado de complejidad en términos de programación (líneas de código, cantidad de clases utilizadas, entre otros) más que el necesario para mantener las buenas prácticas del desarrollo de software.
- Existirá una base de diez preguntas relacionadas con cada tópico de la cátedra sistema distribuido en función a las aplicaciones a desarrollar, dejando a criterio del profesor la creación de nuevas preguntas.

## I.5. Justificación

El motivo de realizar este trabajo especial de grado se debe a que muchos de los elementos descritos en la materia (arquitecturas, servicios web, sincronización, servicio de nombre, entre otros) son muy utilizados fuera del ámbito académico, conceptos que todo ingeniero informático debe manejar independientemente de la especialización que desea aspirar. En ingeniería se ha demostrado que las prácticas le permiten a los alumnos visualizar los conceptos que en teoría pueden parecer complicados o muy extensos. El hecho de que el estudiante pueda observar que está ocurriendo con los nodos cuando ocurra algún evento ayuda a comprender los conceptos teóricos de la cátedra.

Utilizar *Raspberries Pi* como nodos permite que estudiantes de ingeniería informática que no posean los recursos necesarios para comprar una computadora con al menos cuatro núcleos (tres nodos + servidor web y base de datos) puedan aprovecharlas debido a sus bajos precios, sólo 35 \$. Debido a que los *Raspberries Pi* utilizan Linux como sistema operativo, se pueden desarrollar no sólo aplicaciones Java como en este trabajo, sino también bajos otros lenguajes muy utilizados en la actualidad.

A través de la infraestructura planteada se obtiene un laboratorio portátil, debido a que se requiere poco espacio físico para desplegar cada componente: *Raspberries Pi* y sus elementos (cables de red, de corriente, memoria sd), un *Switch* o *Router* para establecer la conexión entre cada

nodo y una computadora central que funcione como controladora de toda la infraestructura. Los estudiantes puedan comprender los conceptos relacionados con los sistemas distribuidos mediante los resultados obtenidos por las aplicaciones desplegadas y comparándolos con los escenarios planteados para cada aplicación, logrando cumplir con el objetivo del tópico.

Se permite que los usuarios de la infraestructura (Profesores o estudiantes) puedan generar sus propias aplicaciones, pudiéndole agregar mayor complejidad o enfocarlas desde otro punto de vista, almacenarlas en la infraestructura con sus parámetros de ejecución, instrucciones de despliegue, posibles escenarios y archivos ejecutables. La infraestructura permite generar preguntas de tipo examen que puedan servir como una guía de estudio para los estudiantes. Por otro parte, se permite que se puedan crear, modificar o eliminar tópicos a la infraestructura en caso de que el programa para Sistemas Distribuidos de la Escuela de Ingeniería Informática en la UCAB sea modificado.

## CAPÍTULO II. Marco Teórico

---

### II.1. Sistemas Distribuidos

Los sistemas distribuidos se pueden definir como “Aquel sistema en el que los componentes localizados en computadores conectados en red, comunican y coordinan sus acciones únicamente mediante el paso de mensajes” [1]. La principal razón para la construcción de sistemas distribuidos es compartir recursos, un recurso puede estar dirigido a hardware: Discos duros, impresoras, entre otros o a software pudiendo incluir elementos como base de datos, archivos o programas [1].

Otra definición sobre sistema distribuidos es: ‘Una colección de computadoras independientes que dan al usuario la impresión de constituir un único sistema coherente’ [2]. De estas dos definiciones se puede resaltar dos términos importantes: computadoras y red. Es requisito fundamental poseer más de una computadora y una conexión entre ellas que determinen una relación de intercambios de mensajes, balanceo de carga o compartimiento de recursos. Por ello, una definición básica es: Sistema de cómputo conectado en una red de alta velocidad [3].

Un sistema distribuido está constituido por dos elementos principales: La red y los nodos, la red es por donde viaja la información, es el canal de comunicación que permite enviar los mensajes desde un computador a otro.

Un nodo es un punto de conexión en una red, es cualquier elemento que posea una dirección única de red como IP o MAC. En los sistemas distribuidos, un nodo puede ser un computador o un servidor.

Las principales ventajas de los sistemas distribuidos son [3]:

- Tienen una proporción precio / desempeño mucho mayor a la de un sistema centralizado.
- Varias computadoras trabajan de manera conjunta con un mismo propósito.
- Poseen una mayor confiabilidad, si falla una máquina el sistema puede seguir funcionando a pesar de tener una perdida, a diferencia de un sistema centralizado, donde si falla la máquina se pierde la operatividad del sistema por completo.
- El crecimiento por incremento es más satisfactorio, ya que al tener un sistema distribuido sólo se debe agregar nuevos nodos (procesadores) que cuando se tiene un sistema centralizado y se requiera agregar otra computadora, debido a que en esta última las aplicaciones deberían reconfigurarse o reprogramarse.
- Permite balancear carga dividiendo las peticiones de los usuarios realizadas hacia el sistema logrando distribuirla en varios computadores con la finalidad de no saturar los recursos de hardware de cada máquina.

Entre las desventajas de los sistemas distribuidos se obtiene [3]:

- Aplicaciones distribuidas: La complejidad de desarrollar aplicaciones distribuidas es mayor a las centralizadas, debido a que deben funcionar en todos los nodos del sistema, algunos lenguajes de programación son más eficientes que otros.
- La red de comunicación entre los nodos del sistema se puede saturar de mucha información y puede generar pérdidas de datos. Si la red falla en un nodo, este queda aislado de la comunicación con el sistema distribuido.
- La seguridad es desventaja en algunos casos, debido a que pueden surgir escenarios donde los datos puedan ser vistos por otras personas sin necesidad de requerir permisos o autorización específica.

## II.2. Sockets

Una definición de *sockets* es “Un punto de un enlace bidireccional entre dos aplicaciones que se encuentra en ejecución dentro de una red” [4]. El punto o *endpoint* es una combinación de dirección IP y puerto, cada conexión TCP está únicamente identificada por ambos elementos, de ahí se puede tener múltiples conexiones entre el host y el servidor.

El servidor ejecuta una aplicación en un puerto específico y se queda en estado de escucha, esperando por las peticiones de los clientes. Al conocer

la dirección IP y el puerto por donde se encuentra escuchando el servidor, el cliente puede realizar las peticiones para conectarse; Se requiere que el cliente se identifique ante el servidor y para ello el sistema asigna un puerto de salida para lograr la comunicación [4].

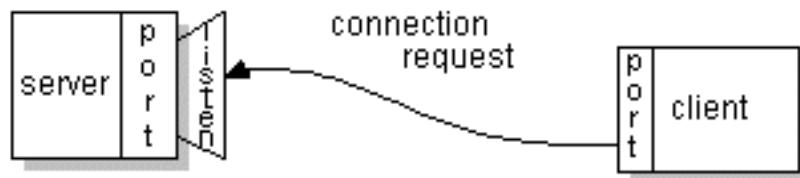


Ilustración 2: Petición Cliente – Servidor. Obtenido de:  
<http://docs.oracle.com/javase/tutorial/figures/networking/5connect.gif>

Si el servidor acepta la conexión, este obtiene la dirección y puerto del cliente y redirige el socket local a otro puerto para poder seguir escuchando por nuevas peticiones.

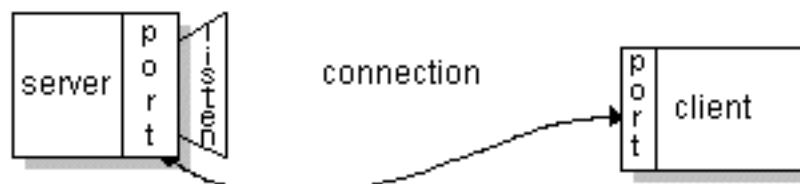


Ilustración 3: Respuesta Servidor – Cliente. Obtenida de:  
<http://docs.oracle.com/javase/tutorial/figures/networking/6connect.gif>

En el cliente, si la conexión es aceptada, se crea el socket entre ambas partes y se puede empezar a intercambiar información a través del canal [4].

### II.3. Raspberry Pi

Es una computadora del tamaño de un tarjeta de crédito que permite ejecutar las tareas básicas como navegar por internet, procesar documentos de textos, hojas de cálculos, juegos, programar, entre otros. El *Raspberry Pi* fue creado con la intención de enseñar a los niños a programar, tiene un costo de 35 \$ por lo que lo hace la computadora más barata que se puede conseguir en el mercado permitiéndole a países en vías de desarrollo adquirir una computadora accesible a personas con bajos recursos. Existen dos modelos: A y B, el modelo B posee un procesador ARMv6 de 700 MHz, 512mb de memoria RAM, dos puertos USB 2.0, puerto Ethernet, entrada para audio, rca video y HDMI. No contienen disco duro, por lo que el sistema operativo es almacenado dentro de una memoria SD [5].

Entre los sistemas operativos disponibles para los *Raspberries Pi* se encuentra una variante de Debian llamada Raspbian que se encuentra compilado para trabajar con procesadores la versión 6 del ARM contenido en el *Raspberry Pi* [6]. Otros sistemas operativos Linux que pueden utilizar son: ArchLinux y Fedora. Como lenguajes de programación tiene soporte principalmente para Python el cual es el recomendado por la fundación Raspberry para la educación, Perl, C y Java.

## RASPBERRY PI MODEL B

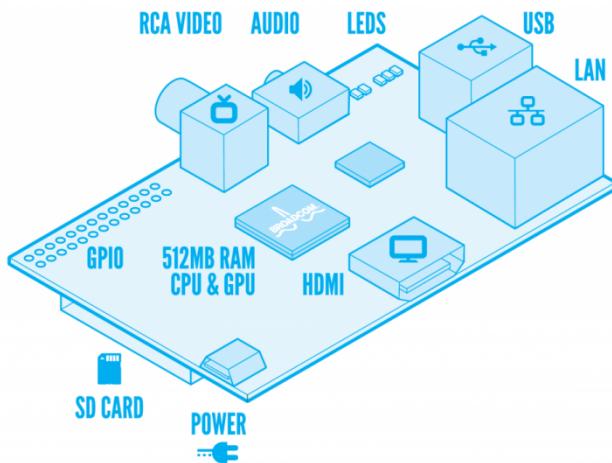


Ilustración 4: Componentes de un Raspberry Pi. Obtenido de: <http://www.raspberrypi.org/wp-content/uploads/2011/07/RaspiModelB-1024x902.png>

### II.4. Agente

Un agente es un software que realiza acciones continuas y automáticas dentro de una computadora, se ejecutan en *background*, por lo que el usuario puede realizar su trabajo sin notar la presencia del agente [7]. Pueden realizar tareas dependiendo de su objetivo: recolectar información, monitorear una aplicación o proceso, enviar datos hacia otro computador o agentes, entre otros.

La idea de desarrollar agentes es generar un software que funcione como un ‘robot’ que permita realizar tareas dentro del sistema sin la interrupción del usuario. Entre los atributos que definen a un agente se tienen [8]:

- Reactividad: El agente posee la capacidad de actuar de forma selectiva
- Autonomía: Debido a que se inicia por sí mismo y posee un propósito.
- Comportamiento Colaborativo: Puede actuar con otros agentes para completar una meta en común.
- Capacidad inferencial: Puede actuar sobre tareas utilizando conocimiento previo de los objetivos generales y los métodos preferidos para lograr flexibilidad.
- Continuidad temporal: Persistencia de su estado durante largo periodos de tiempo.
- Movilidad: La capacidad de migrarlo de manera auto dirigida de un host a otro.

## CAPÍTULO III. Metodología

---

Para este trabajo especial de grado se utilizó una adaptación de la metodología *eXtreme Programming* (XP) en el desarrollo de los componentes de la infraestructura. XP es una metodología ágil que permite desarrollar software a través de historias de usuario, programación en parejas, la cooperación del cliente y el constante uso de pruebas para garantizar que el código generado cumpla con sus objetivos [9]. Se utilizan iteraciones para entregar pequeñas versiones del software que satisfacen un conjunto de requerimientos.

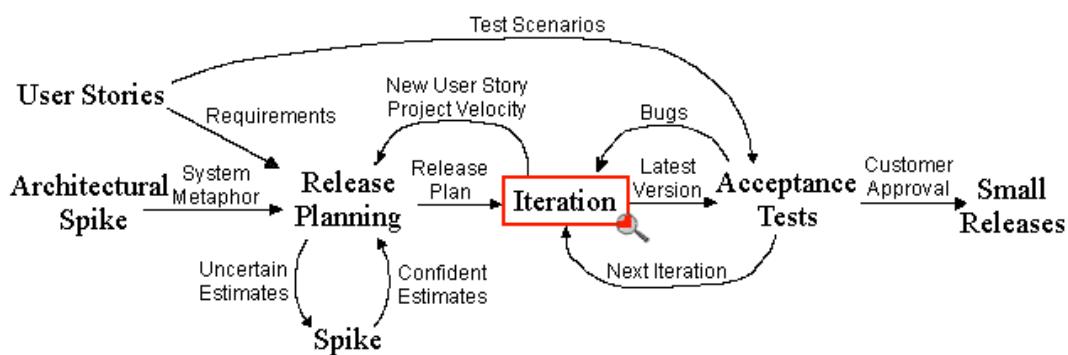


Ilustración 5: Ciclo de vida en XP. Obtenido de:  
<http://www.extremeprogramming.org/map/images/project.gif>

### III.1. Historias de Usuario

Permiten conocer los requerimientos necesarios para cumplir con los objetivos del proyecto. Una historia de usuario es un texto breve que indica una funcionalidad del proyecto, se escriben por parte del cliente con un

lenguaje natural, sin necesidad de agregar sintaxis técnica. Las historias poseen una estimación de desarrollo que puede ser una, dos o tres semanas. Más de tres semanas indica que la historia es muy compleja y debe dividirse en sub-historias, menos de una semana indica que la historia es muy simple y se debe combinar varias historias para lograr una estimación mayor. Las historias de usuario se enfocan en las necesidades del usuario, por ello, se debe evitar colocar algoritmos o especificaciones de alguna tecnología.

### III.2. Plan de iteraciones

Se coloca la prioridad a las historias de usuarios y se crea el plan de iteraciones que divide el proyecto en partes, colocando en cada iteración un conjunto de historias de usuarios relacionadas. Normalmente las iteraciones poseen una duración de una a tres semanas y al finalizar la iteración se obtiene un entregable que el cliente debe aprobar.

### III.3. Iteraciones

Cada iteración del proyecto contempla las siguientes actividades:

- **Diseño:** Se elabora el diseño de cada una de las historias de usuario que fueron acordadas en la iteración actual. Se crean las tarjetas CRC (clase, responsabilidad y colaboración) [10] que permiten conocer que clases interactuarán en el software, están compuestas por: Nombre de la clase, clases hijo (si aplica),

responsabilidad (descripción de los métodos) y colaboración (relaciones con otras clases) [11].

- **Codificación:** Se codifican las historias de usuarios de la iteración, respetando las siguientes normas: estándares de codificación [12] [13], creación de las pruebas unitarias antes que el código, reutilización de código, cooperación del cliente.
- **Pruebas Unitarias:** Se ejecutan luego de codificar las historias de usuario y deben ser exitosas para culminar la iteración y liberar la siguiente versión del software.

#### III.4. Pruebas de aceptación

Al terminar la iteración se realizan las pruebas de aceptación por parte del cliente, permiten verificar si el entregable obtenido logró cumplir con las especificaciones indicadas por las historias de usuario. En caso negativo, se debe realizar una nueva iteración para corregir errores o defectos que puedan haber surgido en el desarrollo, por el contrario, si es aprobado, entonces se libera una pequeña versión del software final

#### III.5. Pequeña liberación

Si el software es aceptado por el cliente se habrá liberado una versión del software final y se continua con la siguiente iteración del plan de iteraciones [14].

La adaptación de la metodología consiste en primer lugar en establecer el plan de iteraciones de acuerdo a los componentes necesarios en la infraestructura: Cada iteración representa un componente de la infraestructura (aplicaciones sistemas distribuidos, aplicación web, agente, librería, servidor central), adicionalmente, el desarrollo será ejecutado por un solo programador, por lo que se obtiene una mayor duración en el desarrollo de las historias de usuario debido a que una persona debe tener en cuenta la legibilidad, el uso de estándares y la optimización del código mientras se programa la historia de usuario.

Se favoreció la elección de esta metodología gracias a la preponderancia que se da al producto final y al interés de satisfacer las necesidades del cliente. XP se enfoca específicamente en crear software que funcione más allá de elaborar documentación extensa que no es útil cuando el software no realiza su propósito correctamente. Utilizando historias de usuario se obtienen los requerimientos de cada componente de la infraestructura de forma precisa. Se planifican iteraciones para dividir el proyecto en partes, cada iteración se encuentra dirigida a un componente de la infraestructura y permite desarrollarlo de manera independiente. A través del uso de pruebas unitarias se garantiza que cada método de las aplicaciones a desarrollar funciona de forma correcta y sin errores. La reutilización de código estará presente en los componentes de la infraestructura pero principalmente en

las aplicaciones educativas de sistemas distribuidos, donde en muchas existirán el esquema cliente y servidor.

La Tabla 2 muestra un sumario de la forma en que fue aplicada la metodología XP así como los productos obtenidos en cada una de sus etapas. En el anexo 1 se encuentra el compendio de *Backlogs* generados para cada iteración donde se detalla las historias de usuario completadas y su prioridad.

Etapa XP	Aplicación en el TEG	Productos Obtenidos.
Historias de usuario	Se elaboraron las historias de usuarios que satisfacen los requerimientos de los componentes de la infraestructura tecnológica.	Historias de Usuario identificadas para cada componente.
Plan de iteraciones	Se le asignó prioridad y estimación a cada historia de usuario. El proyecto se dividió en cinco iteraciones.	Cantidad de iteraciones necesarias para cumplir con los requerimientos del proyecto.
Iteraciones	Se dividió el proyecto en cinco iteraciones: 1. Librería de mensajes 2. Agente de configuración 3. Servidor Central 4. Aplicación Web 5. Aplicaciones de Sistemas Distribuidos.	Se obtuvo un componente de la infraestructura en cada iteración.
Pruebas de Aceptación	Al concluir una iteración se consultó con el cliente (profesor) para validar si el producto obtenido cumple con los requerimientos	Aprobación del cliente (profesor) sobre el producto resultante de la iteración actual.
Pequeña liberación	Se obtiene un componente de la infraestructura que puede ser utilizado.	Una porción de software que se encuentra aprobado por el cliente

Tabla 2: Aplicación metodología XP. Elaboración propia

## CAPÍTULO IV. Desarrollo

---

Para resolver el problema planteado se establece el desarrollo de los componentes de la infraestructura en cinco iteraciones a través de la metodología XP. La librería de mensajes se encuentra en la primera iteración de desarrollo debido a que es la pieza fundamental de comunicación entre las distintas partes que conforman la infraestructura. La segunda iteración consiste en el desarrollo del agente de configuración ya que se requiere que se encuentre ejecutándose en todo momento dentro de cada *Raspberry Pi*.

Como tercera iteración se encuentra el desarrollo del servidor central, este se encarga de conectar la aplicación web con los nodos del sistema distribuido. La cuarta iteración corresponde a la aplicación web ya que a través de ella se pueden ejecutar, controlar y visualizar el comportamiento de los nodos para una determinada aplicación de sistema distribuido. Por último, la iteración número cinco, consiste en el desarrollo de las aplicaciones de sistemas distribuidos. Se decidió desarrollarlas al final debido a que se pueden realizar pruebas de integración con los componentes de la infraestructura al momento de ejecutar cada aplicación y así descubrir y reparar posibles errores que puedan haber surgido en el desarrollo de cada elemento de la infraestructura.

## IV.1. Base de Datos

El diseño de la base de datos se inicia con la necesidad de almacenar la información correspondiente a la infraestructura: Tópicos, preguntas y las aplicaciones a desarrollar. Se elaboró un diagrama entidad relación que permitiera relacionar dicha información.

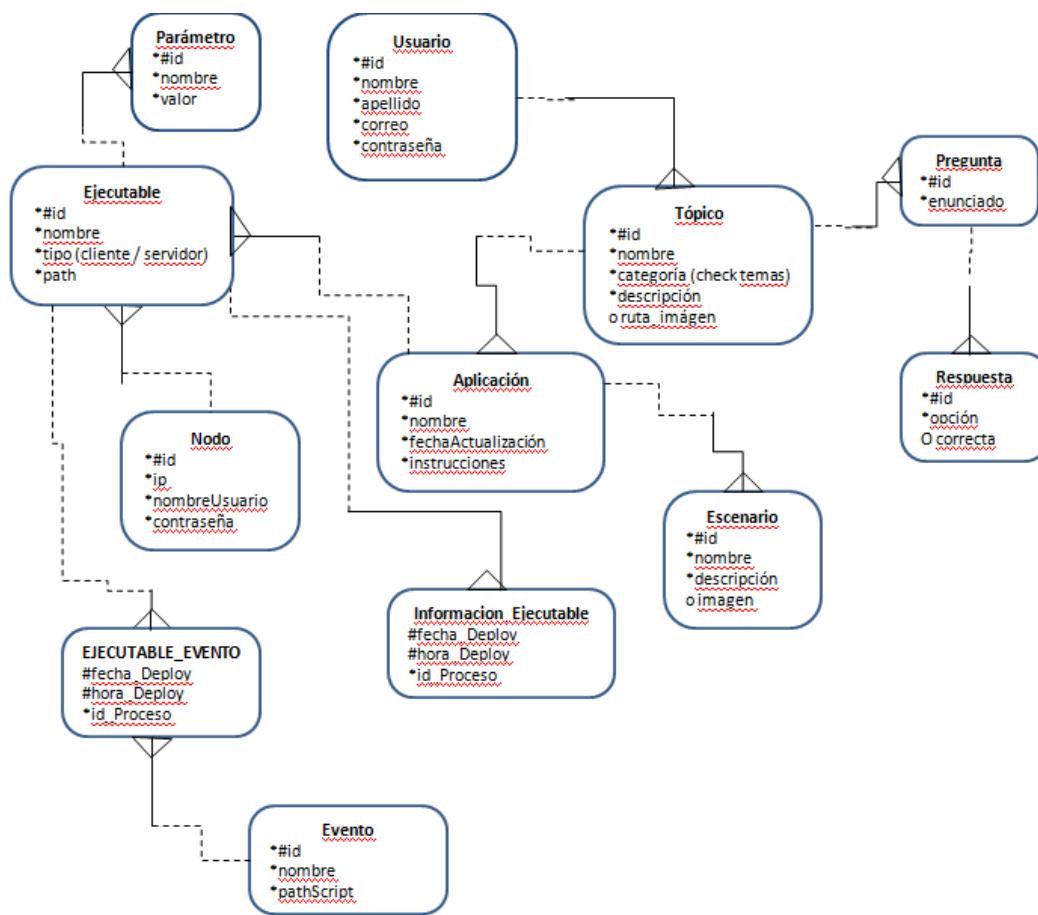


Ilustración 6: Diagrama Entidad Relación. Elaboración propia.

Un tópico se encuentra asociado con un grupo de preguntas de estudio, cada pregunta posee sus respectivas respuestas, los tópicos contienen aplicaciones de sistemas distribuidos que poseen ejecutables y parámetros

de ejecución. Las aplicaciones se encuentran relacionadas con escenarios que se orientan al objetivo del tópico. Cada ejecutable se encuentra asociado a un nodo. Por último, poseen eventos asociados para indicar las acciones que puede realizar el usuario en ese nodo.

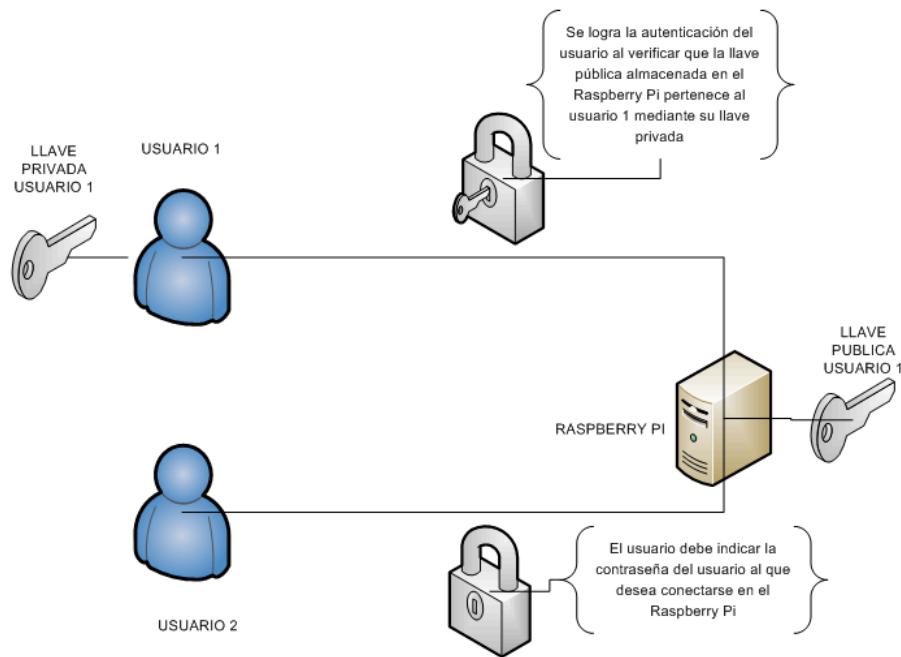
La base de datos solo puede ser accedida a través del módulo de gestión. Cada tópico se encuentra relacionado con un usuario, el cual puede ser: un profesor, un ayudante de cátedra o una persona que posea una cuenta de ingreso al módulo.

## IV.2. Scripts de Sistemas Operativo

La infraestructura requiere de un mecanismo que permita acceder a cada nodo desde el servidor central para controlar la ejecución y detención de los ejecutables asociados a cada aplicación de sistema distribuido. Para ello, se elaboraron *scripts* de sistema operativo [15] que se conectan a un nodo particular permitiendo realizar estas tareas en un ejecutable específico.

La conexión entre el servidor central y los nodos se realizó a través del protocolo SSH (*Secure Shell*). SSH permite conectarse remotamente a un host de manera segura, cifrando la comunicación entre dos hosts en una red insegura [16]. Por seguridad, al ejecutarse el comando SSH se solicita la contraseña del usuario de la máquina remota. Para evitar esta solicitud, se generaron un par de llaves (pública y privada) en el host donde se encuentra el servidor central de manera que la conexión remota hacia un nodo no

requiera autenticar al usuario mediante su contraseña. La llave pública se agrega a archivo de llaves autorizadas (*authorized\_keys*) en cada *Raspberry Pi*.



**Ilustración 7: Autenticación Llave pública - privada. Elaboración propia.**

Se elaboraron los siguientes *scripts*:

- **Ejecutar.sh:** Copia un ejecutable (.jar) dentro de un nodo y lo ejecuta. Requiere el nombre de usuario del nodo, su dirección ip, la ruta, el nombre y los parámetros del ejecutable.
- **EliminarNodo.sh:** Detiene el proceso de un ejecutable dentro de un nodo. Requiere el número de proceso del ejecutable para su detención utilizando el comando `kill -9 <número_de_proceso>`, nombre de usuario del nodo y dirección ip.

- **EliminarTodos.sh:** Detiene el proceso del ejecutable en cada nodo utilizando el comando `kill -9 <número_de_proceso>`. Requiere el número de proceso, nombre de usuario y la dirección ip de los nodos asociados a la aplicación.

La ilustración 8 indica el diagrama de actividades para la ejecución de los *scripts*:

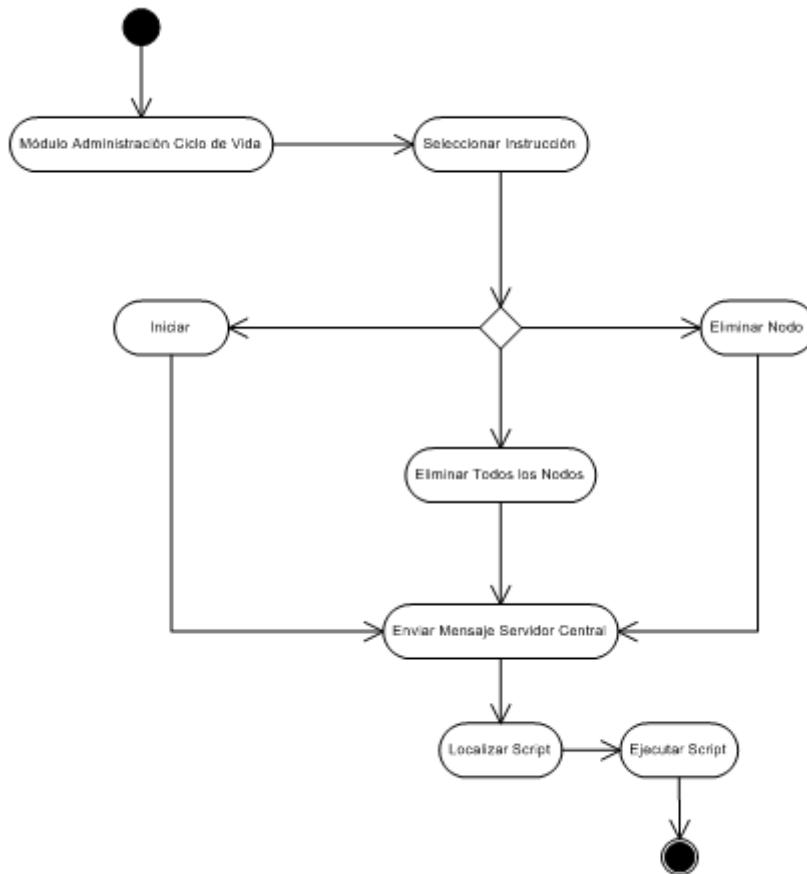


Ilustración 8: Diagrama de actividades para la ejecución de *scripts*. Elaboración propia.

### IV.3. Primera Iteración: Librería de Mensajes

La librería de mensajes es la pieza fundamental de la infraestructura, ya que permite el envío y recepción de mensajes entre los distintos componentes. Se definieron dos tipos de mensajes: Mensajes del agente de configuración y mensajes generales.

#### IV.3.1. Historias de Usuarios

Se crearon las siguientes historias de usuario que permiten obtener los requerimientos principales: Envío y recepción de mensajes:

1. **Configurar atributos:** La librería posee atributos que pueden ser configurados en cualquier momento: Puerto de escucha, cantidad máxima de mensajes a recibir, dirección ip de origen del mensaje y dirección ip de los destinatarios.
2. **Enviar mensajes simples:** Debe permitir el envío de mensajes a través de un método donde se indique el mensaje a enviar, la dirección ip y el puerto.
3. **Recibir detalles del mensaje:** Los mensajes recibidos deben contener la fecha y hora del momento en el que fue enviado.
4. **Enviar información del agente:** Permite enviar mensajes provenientes de los agentes de configuración para ser capturados por el servidor central o por la aplicación web.

#### IV.3.2. Diseño (Tarjetas CRC)

Para cumplir con los requerimientos de la librería se crearon cuatro (4) clases:

LibreríaMensajes	
<ul style="list-style-type: none"> <li>Almacenar los mensajes recibidos desde otro host.</li> <li>Envío de mensajes hacia uno o más destinatarios.</li> </ul>	EsuchaMensajes
EscuchaMensajes	
<ul style="list-style-type: none"> <li>Establecer el puerto de escucha de los mensajes.</li> <li>Recibir mensajes provenientes de una aplicación o agente de configuración.</li> </ul>	Mensaje InformaciónAgente.
Mensaje	
<ul style="list-style-type: none"> <li>Almacenar la información que se enviará hacia el destinatario(Fecha, hora y contenido)</li> </ul>	
InformacionAgente	
<ul style="list-style-type: none"> <li>Almacenar la información reco-pilada por los agentes de configuração</li> </ul>	

Tabla 3: Tarjetas CRC Librería de mensajes. Elaboración propia.

#### IV.3.3. Codificación

La librería se inicia a través de la clase LibreríaMensajes, en ella se indica el puerto de escucha donde se recibirán los mensajes, por defecto posee el valor de: 1337, sin embargo, este valor puede cambiarse por el indicado por el usuario o elegir iniciar sin puerto (únicamente para enviar mensajes). Se obtiene la dirección ip del *host* local buscando en las interfaces de red del sistema operativo la dirección base (192.168...).

Existen dos tipos de mensajes, el primero consiste en los mensajes generales. Un mensaje general posee los atributos: Fecha (día / mes / año), hora (hora: minutos : segundos) y el contenido con la información a enviar. El otro tipo de mensajes son los mensajes del agente de configuración, poseen los atributos que se recolectarán en los nodos: Procesos activos, memoria disponible, uso del CPU, dirección ip, puertos disponibles aplicación activa, numero de nodo y número de proceso del agente.

La clase EscucharMensajes consiste en un hilo que se encuentra en ejecución en todo momento. Al llegar un mensaje, el hilo verifica el tipo de mensaje recibido para almacenarlo en la lista correspondiente. La librería contiene dos listas para almacenar los mensajes recibidos de acuerdo al tipo de mensaje. El máximo número de mensajes que pueden almacenarse en ellas por defecto es de cinco, si se recibe un mensaje adicional se eliminará el mensaje más antiguo, esta cantidad puede ser cambiada por el usuario.

La librería posee los métodos para enviar mensajes, se pueden enviar ambos tipos de mensajes. A través de la sobrecarga de métodos se puede enviar un mensaje indicando: el contenido a enviar, un objeto Mensaje o InformacionAgente. Por último, la librería posee los métodos de buscar un mensaje en las listas, consultar el último mensaje recibido y eliminar un mensaje particular.

#### IV.3.4. Pruebas

- **Unitarias:** Se realizaron pruebas unitarias de los métodos pertenecientes a la clase LibreriaMensajes, Escenarios probados: Envío de mensaje general y envío de mensajes del agente.
- **De aceptación:** La librería de mensajes cumple con el envío y recepción de mensajes entre varios hosts y permite configurar sus principales atributos: Puerto de escucha, dirección de ip origen, cantidad de mensajes máximos que puede recibir.

### IV.4. Segunda Iteración: Agente de Configuración

El agente de configuración permite conocer como se encuentran los recursos de un nodo *Raspberry Pi* en un momento determinado, Se requiere que cumpla con las características principales de un agente: Reactividad, autonomía, continuidad y movilidad.

#### IV.4.1. Historias de Usuario

Para el agente de configuración se elaboraron las siguientes historias de usuario:

- **Recolectar información sobre el sistema operativo:** Se requiere que el agente puede obtener información del sistema: Uso del procesador (cpu), memoria disponible y procesos activos.

- **Recolectar información de red:** El agente obtiene información de la red donde se encuentra conectado el nodo *Raspberry Pi*: (dirección ip del *host* y puertos disponibles).
- **Recolectar información de la aplicación actual:** El agente adquiere la información de la aplicación en ejecución: Nombre de la aplicación y número de nodo.
- **Ejecutar agente:** Se requiere que el agente se inicie automáticamente al encender el nodo *Raspberry Pi* de manera que pueda recolectar y enviar la información correspondiente.
- **Enviar información:** Al terminar de recolectar la información en el nodo se debe enviar el mensaje al servidor central.

#### IV.4.2. Diseño (Tarjetas CRC)

Se crearon dos clases para el desarrollo del agente de configuración:

AgenteConfiguracion	
• Inicializa la configuración del agente, librería de mensajes y dirección ip del servidor central	Monitoreo
Monitoreo	
• Contiene los métodos necesarios para la recolección de información de diferente tipo (Sistema operativo, red y aplicación actual).	InformacionAgente (Librería de Mensajes)

Tabla 4: Tarjetas CRC Agente de configuración. Elaboración propia.

#### IV.4.3. Codificación

El agente de configuración se desarrolló mediante una aplicación en Java que debe ejecutarse al iniciar cada nodo *Raspberry Pi*, para ello, el

ejecutable compilado del agente (.jar) es almacenado en el nodo y su ejecución se inicia al arrancar el sistema operativo del dispositivo mediante el *script* de inicio *rc.local*<sup>1</sup> que permite la ejecución de comandos al iniciarse el sistema. Posee los parámetros: Puerto de escucha de mensajes, dirección ip del servidor central e interfaz de red activa del *Raspberry Pi*.

Al iniciarse, se abre un puerto para la espera de mensajes de la aplicación (Nombre de la aplicación y número de nodo). Se verifica que exista conexión con el servidor central y con el puerto de escucha de las aplicaciones (el puerto por defecto de la librería), si existe, entonces el agente empieza la recolección de la información, en caso contrario se mantiene esperando hasta que una aplicación se ejecute en el nodo.

Para la recolección de la información se utiliza el método 'exec' perteneciente a la clase *Runtime* que permite la ejecución de comandos del sistema operativo a través de Java, la información a recolectar se divide en:

#### **Información del sistema operativo:**

- **Cantidad de procesos:** El comando *ps -e* permite obtener los procesos que se encuentran en el sistema operativo. Esta cantidad es totalizada y se retorna el número final de procesos.
- **Memoria disponible:** Se utiliza el comando *free -m* que indica la cantidad de memoria disponible en *Megabytes* del sistema operativo.

---

<sup>1</sup> Información sobre el script *rc.local* en: [http://wiki.vpslink.com/Linux\\_File\\_Reference:\\_/etc/rc.local](http://wiki.vpslink.com/Linux_File_Reference:_/etc/rc.local)

- **Uso del cpu:** El comando `ps -eo pcpu` muestra la cantidad de procesador utilizada por cada proceso, esta cantidad es totalizada y se devuelve el total de uso del cpu.

#### Información de red:

- **Dirección ip:** Se utiliza el comando `ifconfig` para obtener la dirección ip del nodo a través de la interfaz de red indicada por parámetro al iniciarse el agente.
- **Puertos Disponibles:** El comando `nmap` realiza una exploración de los puertos abiertos en el computador. Se obtiene el número de puerto y el protocolo utilizado.

#### Información de la aplicación:

- **Aplicación activa:** El agente envía un mensaje al puerto de la aplicación (por defecto 1337) para solicitar el nombre de la aplicación.
- **Número de nodo:** Adicionalmente se envía un mensaje para solicitar el número de nodo de la aplicación.

Al recolectar la información se construye un objeto de tipo `InformacionAgente` para ser enviado al servidor central. Este proceso ocurre por defecto cada diez segundos.

#### IV.4.4. Pruebas

- **Unitarias:** Se realizaron pruebas unitarias de los métodos pertenecientes a la clase Monitoreo. Escenarios probados: Recolección de la información de sistema operativo, red y aplicación.
- **De aceptación:** El agente cumple con la recolección de información para un nodo *Raspberry Pi*. Se inicia automáticamente al arrancar el sistema operativo del dispositivo y permite el envío de información hacia el servidor central.

### IV.5. Tercera Iteración: Servicio de Recepción (Servidor Central)

El servidor central es una aplicación en *Java* que permite recibir y redirigir los mensajes enviados por los nodos, agentes de configuración y módulo de administración de ciclo de vida, adicionalmente, debe poseer la capacidad de ejecución y eliminación de los ejecutables en cada nodo dependiendo de la aplicación de sistema distribuido.

#### IV.5.1. Historias de Usuario

- **Reenviar mensajes:** El servidor permite recibir los mensajes por parte de las aplicaciones de sistema distribuidos y de los agentes de configuración para ser reenviados al módulo de monitoreo.
- **Gestionar repositorio Local:** Contiene un repositorio local donde se almacenen los ejecutables de las aplicaciones y los *scripts* de sistemas operativos.

- **Ejecutar Scripts:** Permite recibir los mensajes pertenecientes al módulo de ciclo de vida y ejecutar los *scripts* de sistemas operativos de manera que pueda ejecutar o eliminar una aplicación en un nodo *Raspberry Pi*
- **Conectar a la base de datos:** Permite conectarse a la base de datos para obtener la información de un ejecutable.
- **Obtener información de los nodos del sistema distribuido:** Posee información de los nodos que se encuentran activos de manera que se pueda detener su ejecución cuando el usuario lo indique desde el módulo de administración de ciclo de vida.

#### IV.5.2. Diseño (Tarjetas CRC)

Se crearon tres clases:

<b>ConexionBD</b>	<ul style="list-style-type: none"> <li>• Realizar las llamadas a la base de datos para generar consultas asociadas a la ejecución o detención de las aplicaciones de sistemas distribuidos</li> <li>• Insertar la información perteneciente a los mensajes recibidos por parte de las aplicaciones, agentes de configuración</li> </ul>
<b>NodoActivo</b>	<ul style="list-style-type: none"> <li>• Mantener la información perteneciente al nodo que se encuentra activo en una aplicación de sistema distribuido</li> </ul>

GestionarInfraestructura	
<ul style="list-style-type: none"> <li>Realizar las operaciones de ejecución y eliminación de las aplicaciones.</li> <li>Redirigir los mensajes enviados por los nodos y los agentes de configuración.</li> <li>Consultar la base de datos para obtener los parámetros de las aplicaciones, la información de cada nodo (ip, usuario, contraseña).</li> <li>Insertar en base de datos la información correspondiente a los mensajes entre los nodos, el ejecutable actual en el nodo y los agentes de configuración</li> </ul>	ConexionBD NodoActivo LibreriaMensajes

Tabla 5: Tarjetas CRC Servidor Central. Elaboración Propia.

#### IV.5.3. Codificación

El servidor central se inicia como un hilo que se encuentra en espera de los mensajes enviados desde el módulo de ciclo de vida, los agentes de configuración o los nodos de una aplicación de sistemas distribuidos.

Posee parámetros relacionados con los datos de conexión de la base de datos: Nombre de usuario, contraseña, dirección ip y puerto de conexión. El servidor contiene una lista de nodos activos para tener control de las aplicaciones que se encuentren en ejecución. Tiene configurado la dirección ip y el puerto de la aplicación web para el reenvío de la información recibida.

Al llegar un mensaje se verifica si el mensaje es de tipo agente o general, si el mensaje proviene del agente entonces:

- **Reenvía el mensaje:** Se envía el mensaje hacia la aplicación web para ser visualizado por el usuario a través del módulo de monitoreo.
- **Eliminación del mensaje:** Se elimina el mensaje de la lista de mensajes recibidos y se espera por un nuevo mensaje proveniente de un agente de configuración.

Si se trata de un mensaje general entonces puede provenir de los siguientes componentes:

1. **Mensaje ciclo de vida:** Un mensaje del módulo de ciclo de vida contiene el carácter '>' y una de las siguientes instrucciones:
  - **EliminarTodos:** Esta instrucción permite deshacerse de todos los nodos activos de la aplicación a través del script EliminarTodos.sh y la lista de nodos activos.
  - **Ejecutar:** Permite iniciar un ejecutable dentro de un nodo específico, el mensaje contiene el nombre del ejecutable y la dirección ip del nodo. El servidor consulta el nombre de usuario, los parámetros y la ruta del ejecutable en la base de datos de manera que se pueda correr el script EjecutarNodo.sh y agregar el nodo a la lista de nodos activos.
  - **EliminarAplicación:** Permite detener un ejecutable en un nodo. Se obtiene la dirección ip del nodo a partir del complemento del mensaje recibido, se localiza en la lista de nodos activos el número de proceso del ejecutable para el ip recibido y se ejecuta el script

EliminarNodo.sh, por último, se elimina el nodo de la lista de nodos activos.

- **MensajeNodo:** En este caso el módulo de ciclo de vida envió un mensaje a un nodo particular, el servidor redirige el complemento del mensaje (dirección ip y contenido a enviar) hacia el nodo indicado.

Los *scripts* se invocan utilizando el método 'exec' perteneciente a la clase 'Runtime' desde Java.

1. **Id de proceso:** Si el mensaje recibido contiene el carácter '<' significa que se recibió el número de proceso de un nodo. El servidor busca en la lista de nodos activos el nodo que contiene la dirección ip enviada en el mensaje y se asocia el número de proceso, la fecha y la hora de recepción, esta información adicionalmente es almacenada en la base de datos.
2. **Mensaje de nodos:** Si el mensaje recibido no contiene ninguno de los caracteres mencionados anteriormente entonces se trata de un mensaje enviado por un nodo, se reenvía el mensaje hacia la aplicación web para ser visualizado en el módulo de monitoreo.

#### IV.5.4. Pruebas

- **Unitarias:** Se realizaron pruebas unitarias de los métodos pertenecientes a la clase GestionarInfraestructura.

- **De aceptación:** El servidor central permite que se puedan recibir mensajes desde el módulo de administración de ciclo de vida, de los agentes de configuración y de los nodos del sistema distribuido. Por lo que se puede ejecutar o eliminar aplicaciones, reenviar la información correspondiente y registrarla en base de datos.

Al terminar de desarrollar el servidor central se concluye la tercera iteración.

#### IV.6. Cuarta Iteración: Aplicación Web.

La cuarta iteración consiste en la aplicación web, en ella se encuentran los tres módulos propuestos: Administración de ciclo de vida, monitoreo y gestión. Se desarrolló utilizando *JavaServer Pages* (JSP<sup>2</sup>) que permite crear rápidamente contenido web dinámico y ofrece aplicaciones multiplataforma. Las páginas JSP a menudo envían información hacia los *Servlets*<sup>3</sup>. Un *servlet* se ejecuta del lado del servidor, permite obtener acceso a la API de Java para aprovechar sus beneficios: Portabilidad, rendimiento, reusabilidad y protección contra errores. Los *servlets* requieren de un contenedor web<sup>4</sup>, estos manejan las peticiones que se realizan en los archivos jsp, *servlets* o cualquier archivo que ejecute código del lado del servidor. El contenedor web

---

<sup>2</sup> Información sobre JSP disponible en: <http://www.oracle.com/technetwork/java/overview-138580.html>

<sup>3</sup> Información sobre Servlets disponible en: <http://www.oracle.com/technetwork/java/overview-137084.html>

<sup>4</sup> ¿Qué es un contenedor web? disponible en:  
[http://publib.boulder.ibm.com/infocenter/wsdock400/v6r0/index.jsp?topic=/com.ibm.websphere.iseries.doc/info/ae/ae/cweb\\_aov4.html](http://publib.boulder.ibm.com/infocenter/wsdock400/v6r0/index.jsp?topic=/com.ibm.websphere.iseries.doc/info/ae/ae/cweb_aov4.html)

crea instancias de *servlets*,, carga y descarga de *servlets*, crea y maneja las peticiones y respuestas de objetos y otras tareas de administración.

Se utilizó las librerías de etiquetas estándar de JSP (JSTL<sup>5</sup>) [17] que permite manipular funcionalidades como: Condicionales, iteraciones, consultas, entre otros. La aplicación se desarrolló utilizando el patrón de arquitectura Modelo Vista Controlador (MVC) que permite separar funcionalidades [18] [19].

La aplicación web contiene dos secciones: Tópicos de la materia y Módulo de Gestión, la sección de tópicos es de libre acceso, cualquier usuario puede acceder a ella y seleccionar una aplicación de sistema distribuido para su ejecución. El módulo de gestión, por otro lado, requiere de una cuenta de usuario para su acceso. Los módulos de monitoreo y de administración de ciclo de vida se visualizan en la pantalla principal de la aplicación seleccionada por el usuario. Adicionalmente, se muestra la información del tema, instrucciones de la aplicación, los escenarios planteados, eventos disponibles y el botón para generar preguntas.

#### IV.6.1. Historias de usuario

Se elaboraron las siguientes historias de usuario:

##### Módulo de gestión:

---

<sup>5</sup> Información sobre las etiquetas JSTL disponible en: <http://www.oracle.com/technetwork/java/index-jsp-135995.html>

- **Acceder módulo:** Se requiere que el módulo solo pueda ser accedido por usuarios que posean una cuenta asociada a la infraestructura.
- **Ejecutar operaciones de base de datos:** Debe permitir realizar las operaciones típicas de base de datos: Insertar, consultar, modificar y eliminar los datos almacenados en las tablas pertenecientes a la infraestructura.
- **Cargar aplicaciones:** El módulo debe permitir crear una aplicación de sistemas distribuido y cargar el ejecutable al servidor central junto a sus parámetros de ejecución.
- **Utilizar formularios dinámicos:** Debe poseer formularios dinámicos para mostrar la información en aquellos casos donde puede variar la cantidad de campos a llenar por el usuario.

#### Módulo de administración de ciclo de vida:

- **Consultar eventos:** Se requiere consultar los eventos asociados a un ejecutable de la aplicación para ser generados por el usuario.
- **Generar botones de eventos:** Debe existir botones asociados a los eventos del ejecutable de manera que el usuario al hacer clic pueda activarlos.
- **Informar al servidor central:** El módulo debe enviar mensajes al servidor central en relación al evento seleccionado por el usuario.

**Módulo de monitoreo:**

- **Recibir información:** Permite recibir la información proveniente del servidor central para mostrarla al usuario.
- **Separar información:** El módulo separa la información recibida en dos tipos: información del agente e información de los nodos de la aplicación.
- **Verificar mensajes en un lapso de tiempo :** Cada determinado tiempo el módulo debe chequear si se ha recibido nuevos mensajes para ser visualizados por el usuario.

**IV.6.2. Diseño (Tarjetas CRC)****Módulo de gestión:**

<b>Consultar&lt;Tabla&gt;Servlet1</b>	
• Permite realizar una consulta a la base de datos para obtener el registro solicitado por el usuario	ConexionBD
<b>Crear&lt;Tabla&gt;Servlet</b>	
• Permite agregar un nuevo registro a la base de datos	ConexionBD
<b>Modificar&lt;Tabla&gt;Servlet1</b>	
• Consulta a la base de datos para mostrar los datos de un registro solicitado por el usuario para ser modificado	ConexionBD
<b>Modificar&lt;Tabla&gt;Servlet2</b>	
• Actualiza el registro solicitado por el usuario	ConexionBD

Eliminar<Tabla>Servlet1	
<ul style="list-style-type: none"> <li>Consulta a la base de datos para mostrar los datos de un registro solicitado por el usuario para ser eliminado</li> </ul>	ConexionBD
<hr/>	
Eliminar<Tabla>Servlet2	
<ul style="list-style-type: none"> <li>Borra el registro solicitado por el usuario.</li> </ul>	ConexionBD
<hr/>	
SubirArchivoServlet	
<ul style="list-style-type: none"> <li>Permite cargar un archivo (ejecutable, imagen de tópico o de escenario) desde la aplicación web.</li> </ul>	ConexionBD Directorios

Tabla 6: Tarjetas CRC Módulo de gestión. Elaboración propia

#### Módulo de administración del ciclo de vida:

CicloDeVidaServlet	
<ul style="list-style-type: none"> <li>Evalúa el evento seleccionado por el usuario y envía el mensaje hacia el servidor central para su ejecución.</li> </ul>	ConexionBD LibreriaMensajes

Tabla 7: Tarjetas CRC Módulo de administración del ciclo de vida. Elaboración propia.

#### Módulo de monitoreo:

Archivo	
<ul style="list-style-type: none"> <li>Recibe los mensajes enviados por el servidor central</li> <li>Guarda los mensajes separándolos en archivos XML para el agente o para los nodos.</li> </ul>	ConexionBD
<hr/>	
MonitoreoServlet	
<ul style="list-style-type: none"> <li>Lee la información guardada en los archivos de los nodos y agente para ser visualizado por el usuario</li> </ul>	Archivo
<hr/>	
InformacionAgenteServlet	
<ul style="list-style-type: none"> <li>Obtiene los parámetros del agente por la clase MonitoreoServlet para</li> </ul>	MonitoreoServlet

ser visualizados en una nueva ventana por el usuario.	
---	--

Tabla 8: Tarjetas CRC Módulo de monitoreo. Elaboración propia.

**Aplicación Web:**

<b>Aplicacion</b>	
<ul style="list-style-type: none"> <li>• Permite iniciar la aplicación seleccionada por el usuario</li> <li>• Obtiene la información de la aplicación (Tópicos, escenarios, ejecutables)</li> <li>• Inicia el hilo de escucha para recibir los mensajes provenientes del servidor central.</li> </ul>	ConexionBD Archivo
<b>FiltroUsuario</b>	
<ul style="list-style-type: none"> <li>• Permite restringir el acceso a las páginas donde se requiera iniciar sesión (módulo de gestión)</li> </ul>	
<b>GenerarPreguntaServlet</b>	
<ul style="list-style-type: none"> <li>• Permite generar aleatoriamente preguntas para un tópico específico.</li> <li>• Crea un archivo en formato PDF para visualizar las preguntas del tópico.</li> </ul>	ConexionBD Pregunta
<b>IniciarSesion</b>	
<ul style="list-style-type: none"> <li>• Permite que un usuario pueda iniciar o cerrar sesión en la página.</li> </ul>	ConexionBD
<b>MostrarEscenarioServlet</b>	
<ul style="list-style-type: none"> <li>• Permite visualizar la información de un escenario de la aplicación activa.</li> </ul>	
<b>TerminarAplicacionServlet</b>	
<ul style="list-style-type: none"> <li>• Elimina el puerto de escucha de mensajes al cerrar la aplicación activa</li> </ul>	

<b>Directarios</b>	
<ul style="list-style-type: none"> <li>• Contiene las rutas de los directorios asociados a: Ejecutables, tópicos y escenarios</li> </ul>	
<b>Preguntas</b>	
<ul style="list-style-type: none"> <li>• Contiene los atributos relacionados con una pregunta y sus posibles respuestas</li> </ul>	

Tabla 9: Tarjetas CRC Aplicación web. Elaboración propia.

#### IV.6.3. Codificación

Se inicia en la página principal, en ella existen dos opciones que se pueden seleccionar: Tópicos y Gestión. Adicionalmente, existe la opción de inicio de sesión del usuario mediante su dirección de correo y contraseña para acceder al módulo de gestión. Al hacer clic sobre el módulo se redirige a la página de gestión, en ella se visualiza las tablas: Usuarios, tópicos, preguntas, aplicaciones, ejecutables, nodos y eventos. Cada una contiene las opciones: Crear, modificar, eliminar o consultar. Cuando se selecciona una opción, la aplicación web carga el formulario de la página seleccionada utilizando *Ajax*<sup>6</sup> y el usuario completa los campos del formulario presentado y los envía al servidor para solicitar la siguiente página o esperar el mensaje de confirmación. Los *servlets* asociados a cada tabla y operación poseen los siguientes métodos:

- **ObtenerDatos:** Permite recolectar los parámetros del formulario enviado por el usuario o inicializar los objetos de la clase.

---

<sup>6</sup> Técnica de programación que permite intercambiar datos con el servidor y actualizar partes de una página sin necesidad de recargarla. <http://www.w3schools.com/ajax/>

- **EjecutarQuery:** Realiza la consulta SQL asociada al tipo de *servlet*: Consultar, eliminar, crear o modificar.
- **EnviarInformación:** Si se trata de un *servlet* informativo (aquellos que poseen el número uno, ejemplo: EliminarUsuarioServlet1 entonces envía los datos obtenidos por la consulta SQL, en caso contrario envía la respuesta al usuario acerca de la operación realizada.

La opción de tópicos en la página de inicio le permite al usuario seleccionar diferentes tópicos de la cátedra sistemas distribuidos divididos en temas. Al hacer clic sobre un tema se carga vía *Ajax* las aplicaciones disponibles. El usuario puede escoger la aplicación a ejecutar y enviar el formulario para dar inicio a la página principal de la infraestructura. El *servlet* Aplicación realiza varias tareas al recibir la solicitud por parte del usuario:

- Obtiene el número de la aplicación (id) seleccionada por el usuario.
- Consulta los datos de la aplicación en base de datos: Tópico, escenarios asociados y del ejecutable.
- Iniciar la espera de mensajes provenientes del servidor central a través de la clase Archivo.
- Envío de la información hacia la página de la aplicación.

La página de la aplicación le permite al usuario interactuar con toda la infraestructura, está compuesta por los siguientes elementos:

- La información de la aplicación elegida: Nombre de la aplicación, fecha de creación, instrucciones de la aplicación.
- Nombre del tópico, descripción del tópico e imagen asociada (opcional).
- Los nodos pertenecientes a la aplicación junto a los botones (Información del agente de configuración y eventos asociados).
- Las pantallas de cada nodo que indican el número de nodo y si se trata de un nodo cliente o servidor.
- Los escenarios asociados a la aplicación mediante enlaces (*links*).
- La opción de generar preguntas acerca del tópico de la aplicación.
- El botón de regresar para elegir otra aplicación y la opción de eliminar todos los nodos.

Los ejecutables de los nodos se inician automáticamente al cargar la página con una diferencia de tres segundos, el servidor central ejecuta el script en el nodo número uno, luego continua con el siguiente nodo hasta que todos los ejecutables asociados a la aplicación se hayan iniciado, el usuario podrá observar por pantalla el mensaje 'Ejecutable inicializado'.

Para mostrar por pantalla los mensajes enviados por los nodos o por el agente de configuración se realiza una petición *Ajax* cada cinco segundos para consultar el contenido de los archivos de cada nodo y del agente de configuración a través de la clase MonitoreoServlet.

Los eventos asociados son cargados debajo de la pantalla de cada nodo a través de botones, cada uno indica la acción a realizar, por defecto todos los nodos poseen los eventos de: Iniciar y detener, en algunos casos (dependiendo del ejecutable) existe el evento de enviar mensaje al nodo de acuerdo a las instrucciones de la aplicación. Al hacer clic sobre un botón se envía la información mediante *Ajax* a la clase CicloDeVidaServlet indicando el número del ejecutable y la instrucción, esta clase envía un mensaje a través de la librería de mensajes al servidor central para realizar la solicitud.

Los escenarios de la aplicación se encuentran desplegados mediante enlaces (*links*), al hacer clic sobre él se abre una nueva ventana indicando el nombre del escenario y su descripción.

La información del agente de configuración para un nodo particular se obtiene al hacer clic en el botón Información el cual despliega una ventana y muestra los valores recolectados.

Las preguntas acerca del tópico se generan mediante el botón de generar preguntas, al hacer clic, la aplicación elabora preguntas aleatoriamente sobre el tópico para cargarlas en un archivo PDF descargable, se generan un total de cinco preguntas, este procedimiento se puede realizar varias veces para obtener distintas preguntas.

El usuario en cualquier momento puede detener los ejecutables, para ello puede hacer clic sobre el botón eliminar todos los nodos o en el botón de

regresar, este último permite retornar a la página de tópicos para elegir una nueva aplicación.

#### IV.6.4. Pruebas

- **Unitarias:** Se realizaron pruebas unitarias de los métodos contenidos en los *servlets* pertenecientes al módulo de gestión módulo de monitoreo y de ciclo de vida. Escenarios probados: Comportamiento de los métodos al recibir una petición HTTP.
- **De aceptación:** La aplicación web cumple con lo acordado: Le permite a un usuario administrador, profesor o ayudante de cátedra agregar nuevos tópicos, aplicaciones, preguntas a través del módulo de gestión y a los estudiantes les permite ejecutar aplicaciones en los nodos *Raspberry Pi* para reforzar sus conocimientos acerca de un tema particular.

### IV.7. Quinta Iteración: Aplicaciones de Sistema Distribuido.

Las aplicaciones de sistemas distribuidos tienen como propósito cumplir con el objetivo de un tema mediante la simulación de nodos, cada aplicación contiene varios ejecutables que serán almacenados en un repositorio dentro del servidor central y enviados a cada nodo mediante los *scripts* de sistema operativo.

#### IV.7.1. Historias de usuario

- **Poseer información sobre la aplicación:** Las aplicaciones contienen los siguientes datos: Nombre de la aplicación, número de nodo y número de proceso.
- **Recibir mensajes del agente:** Cada aplicación debe permitir recibir los mensajes del agente de configuración para que este pueda recolectar la información de la aplicación.
- **Enviar información al servidor central:** Deben permitir enviar los mensajes remitidos hacia otros nodos o mensajes propios de la aplicación al servidor central de manera que puedan ser redirigidos al modulo de monitoreo y ser visualizados por el usuario.

#### IV.7.2. Diseño (Tarjetas CRC)

Se elaboraron las siguientes clases "base" en las aplicaciones:

DatosAplicacion	
• Almacena la información de la aplicación para ser consultada por el agente de configuración	
<hr/>	
EscucharClientes	
• Verifica el mensaje recibido en la librería de mensajes y lo envía a la lógica para realizar la acción solicitada	LibreríaMensajes LogicaAplicacion
<hr/>	
LogicaAplicacion	
• Clase principal de la aplicación • Almacena los métodos que permiten cumplir con el objetivo del tópico	LibreriaMensajes DatosAplicacion

Tabla 10: Tarjetas CRC Aplicaciones Sistemas Distribuidos. Elaboración propia

#### IV.7.3. Codificación

Las aplicaciones de sistemas distribuidos se inician con los siguientes parámetros de configuración en común:

- Nombre de la aplicación
- Número de nodo
- Dirección ip del servidor central.
- Puerto del agente de configuración

Cada aplicación adicionalmente posee parámetros que son propios a la aplicación, pudiendo ser la dirección ip de los demás nodos, un parámetro de tiempo, nombre de un archivo, entre otros. Depende del tópico y de su propósito.

La aplicación se inicia creando los objetos de la clase DatosAplicación y LibreriaMensajes, se instancia la clase LogicaAplicacion mediante el patrón de diseño *Singleton* que permite crear una sola ocurrencia del objeto [20]. Se inicializa un objeto de la clase EscucharClientes que consiste en un hilo que se encuentra verificando si se han recibido mensajes en la librería de mensajes.

Al llegar un mensaje se compara si se trata de un mensaje del agente de configuración, en caso afirmativo se responde con el nombre de la aplicación y el número de nodo, por el contrario, si el mensaje no pertenece al agente entonces puede tratarse de un mensaje proveniente de otro nodo o del

módulo de administración ciclo de vida, en ambos casos se evalúa el mensaje recibido y se realiza la tarea indicada.

Algunas aplicaciones poseen clases adicionales que permiten dar soporte al propósito del tópico, en el anexo 2 se puede observar las clases desarrolladas para cada aplicación, junto a sus historias de usuario, arquitectura, una breve descripción y parámetros propios.

#### IV.7.4. Pruebas

- **Unitarias:** Se realizaron pruebas unitarias de los métodos pertenecientes a la clase LogicaAplicación. Escenarios probados: Recepción de mensajes por parte del agente de configuración y del servidor central dependiendo del objetivo de la aplicación.
- **De aceptación:** Las aplicaciones cumplen con el objetivo del tópico de sistema distribuido, son controladas mediante la aplicación web y le permiten a los usuarios reforzar sus conocimientos de acuerdo a los escenarios planteados, las preguntas tipo examen y la descripción del tópico.

#### IV.8. Documentación

La documentación perteneciente a la infraestructura se divide en tres documentos: Manual de uso, guía de diseño y guía de configuración.

#### **IV.8.1. Manual de uso**

El manual de uso es un documento que permite conocer cómo se puede interactuar con la infraestructura a través de la aplicación web, indicando en cada página sus elementos y las acciones que el usuario puede realizar en ella.

#### **IV.8.2. Guía de diseño**

La guía de diseño es un documento que contiene toda la información referente a los componentes de la infraestructura: Aplicación web , servidor central, librería de mensajes, agente de configuración y aplicaciones de sistemas distribuidos. Cada componente posee sus historias de usuarios, diagrama de clases y arquitectura.

#### **IV.8.3. Guía de configuración**

La guía de configuración es un documento que contiene los pasos necesarios para instalar la infraestructura, contiene las versiones de software y hardware que se utilizaron en el desarrollo de cada componente.

## CAPÍTULO V. Resultados

---

A continuación se detallan los resultados obtenidos luego de la elaboración de este trabajo especial de grado:

### V.1. Aplicaciones de sistemas distribuidos:

Se elaboraron las aplicaciones de sistemas distribuidos de manera que puedan ayudar a los estudiantes de ingeniería informática a reforzar sus conocimientos adquiridos en clases a través de aplicaciones enfocadas a los tópicos de la cátedra junto a sus escenarios y la generación de preguntas de tipo examen.

### V.2. Agente de configuración

Se desarrolló una aplicación en Java que actúa como agente de configuración dentro de cada nodo *Raspberry Pi* para recolectar información correspondiente al sistema operativo, red y del ejecutable activo. Se encuentra en ejecución en todo momento y reenvía la información recolectada al detectar conexión con el servidor central y con el ejecutable local. Cuenta con las características de reactividad, autonomía, continuidad temporal y movilidad.

### V.3. Servicio de recepción (Servidor central)

Se creó el servicio de recepción a través de un servidor que permite recibir los mensajes enviados por los agentes de configuración o aplicaciones de sistema distribuido y redirigirlos al módulo de monitoreo para ser visualizado por el usuario de la infraestructura, de igual forma, se reciben las instrucciones enviadas por el módulo de administración de ciclo de vida y las ejecuta en uno o más nodos del sistema distribuido. Este servicio actúa como un “bus de comunicación” entre los nodos y el módulo de monitoreo.

### V.4. Librería de mensajes

Se desarrolló una librería en *Java* que permite enviar mensajes de manera simple hacia un host a través de un método donde se indica el mensaje a enviar, la dirección ip y el puerto de recepción. Adicionalmente, recibe mensajes por parte de otros host y los almacena localmente. Puede ser utilizada en cualquier aplicación *Java*, *JSP* o *Framework* para el envío y recepción de mensajes, lo cual lo convierte en una herramienta agnóstica de cualquier *framework* o implementación concreta usada dentro de la tecnología *Java*.

### V.5. Base de datos

Se elaboró una base de datos que permitiera almacenar toda la información correspondiente a la infraestructura tecnológica: tópicos, preguntas, aplicaciones, ejecutables, parámetros, eventos y nodos. La

aplicación web accede a ella a través del módulo de gestión para realizar las operaciones: Agregar, modificar, eliminar o consultar.

### V.6. Scripts de sistemas operativos

Se crearon *scripts* de sistemas operativos UNIX / Linux que le permiten al usuario controlar los nodos desde el módulo de administración de ciclo de vida. Las instrucciones pueden ser: ejecutar o eliminar un nodo y eliminar todos los nodos del sistema distribuido. Estos *scripts* dada su configuración y codificación, se convierten en herramientas portables para diferentes distribuciones UNIX / Linux reutilizables según las necesidades del proyecto.

### V.7. Módulo de administración de ciclo de vida

Se desarrolló un módulo web con interacciones asíncronas y una interfaz gráfica con una tendencia minimalista, para el envío de instrucciones a los nodos de una aplicación determinada. Cada componente gráfico (botones) contiene una instrucción específica que permite obtener información del agente de configuración o ejecutar acciones para alterar el ciclo de vida de las aplicaciones desplegadas en los nodos.

### V.8. Módulo de monitoreo

Se implementó un módulo de monitoreo web donde se reciben los mensajes reenviados por el servidor central para ser almacenados en un archivo local para cada nodo del sistema distribuido. El módulo verifica cada determinado tiempo si existen mensajes en los archivos y los envía a la

pantalla de cada nodo para ser visualizado por el usuario de la infraestructura.

### V.9. Módulo de gestión

Se desarrolló un módulo web que le permite a un usuario registrado realizar las operaciones “clásicas” en un ambiente de datos relacional como lo son: Agregar, eliminar, modificar y consultar datos del repositorio utilizado.

### V.10. Aplicación web

Los módulos desarrollados se integraron mediante una aplicación web para facilitar la comunicación entre las aplicaciones de sistemas distribuidos y el usuario. La aplicación posee la sección de tópicos para seleccionar y ejecutar las aplicaciones y la sección de gestión para administrar la base de datos. Esta aplicación actúa como el punto de entrada para el uso de la infraestructura tecnológica, siendo este parte del objetivo principal del presente trabajo especial de grado.

### V.11. Documentación

Se creó el manual de uso para indicarle al usuario como interactuar con la infraestructura. Se elaboró el documento diseño que contiene los diagramas asociados a la creación de cada componente de la infraestructura y un documento de configuración que contiene información referente al ambiente requerido para instalar y configurar la infraestructura.

## CAPÍTULO VI. Conclusiones y Recomendaciones

---

### VI.1. Conclusiones

Luego del desarrollo de los componentes de la infraestructura y en base a los resultados obtenidos se concluye que:

- La infraestructura tecnológica permite que un estudiante pueda reforzar sus conocimientos de forma práctica al interactuar con los diferentes componentes de la aplicación web (módulos, instrucciones escenarios y preguntas) para poder cumplir con el objetivo de la aplicación de sistemas distribuidos seleccionada.
- Los *Raspberries Pi* son computadoras que permiten realizar diferentes tareas, en este trabajo especial permitieron la simulación de nodos de un sistema distribuido, adicionalmente, se pueden utilizar como servidor de correo, de archivos o web. En el área académica, sirven como estaciones de trabajo para estudiantes de ingeniería informática en materias como redes, seguridad o sistemas operativos. Sus bajos costos los hacen accesibles para las personas ya que utilizan software libre con el cual se evita realizar gastos de licenciamiento en programas que pueden ser más costosos que el *hardware* y su portabilidad permite que se pueda transportar de un lugar a otro sin mayor complicación.

- El agente de configuración permite visualizar de manera automática los recursos de un nodo *Raspberry Pi*, cada información que recolecta equivale a un comando del sistema operativo que el usuario no puede ejecutarlo simultáneamente, y en caso de poderse, se requiere una conexión remota al nodo mientras se observa el comportamiento de la aplicación desde el módulo de monitoreo. Asimismo, el agente puede utilizarse como depurador (*debugger*) de la aplicación en ejecución debido a que la información recolectada puede ser útil al momento de detectar posibles errores a nivel de código relacionados con: el uso del procesador, el consumo de memoria o puertos que no se encuentren abiertos en un momento determinado.
- La librería de mensajes permite el envío y recepción de mensajes de manera simple librando al usuario de implementar métodos que permiten abrir un puerto o socket para transmitir o recibir una información particular.
- El patrón de arquitectura modelo vista controlador (MVC) junto a la librería JSTL permiten el desarrollo de aplicaciones en JSP con separación de funcionalidades: La vistas (páginas) solo poseen los elementos de presentación que el usuario puede observar, mientras que el controlador (*servlets*) contiene la lógica de negocio que permite atender a las solicitudes enviadas por el usuario apoyándose del modelo y retornar los resultados nuevamente en la vista.

- Los componentes de la infraestructura fueron desarrollados pensando en la interoperabilidad entre sistemas UNIX / Linux, esto permite su funcionamiento bajo cualquier sistema operativo que cumpla con el estándar POSIX, por ello, se requiere de dos elementos fundamentales: La máquina virtual de Java (JVM) para ejecutar el servidor central, agente y aplicaciones de sistemas distribuidos y un contenedor web (*GlassFish*, *Tomcat*, entre otros) que permita ejecutar la aplicación web.
- El uso de elementos de software libre para el desarrollo de la infraestructura permite que se pueda liberar el código de los componentes para realizar nuevas versiones, ya sea agregando nuevas funcionalidades, corrigiendo posibles errores o actualizando el software utilizado.

## VI.2. Recomendaciones

En base a las conclusiones obtenidas se recomienda lo siguiente:

- Utilizar un *framework* de desarrollo para la aplicación web permitiría acotar tiempo de programación debido a que la mayoría de ellos poseen la técnica de *Object-relational-mapping* (ORM) que permite convertir las tablas de una base de datos en objetos dentro del *framework* facilitando la creación de páginas relacionadas con las operaciones CRUD (*Create, read, update, delete*).

- Elaborar aplicaciones de sistema distribuidos enfocadas a los tópicos de la cátedra que no fueron abarcados en este trabajo especial de grado como por ejemplo: Algoritmo *token ring*, transacciones, replicación, modelo de consistencia centrada en los datos, arquitecturas simétricas, entre otros.
- Desarrollar las aplicaciones de sistemas distribuidos e implementar la librería de mensajes en lenguaje Python de manera que los recursos en el *Raspberry Pi* no se saturen.
- Extender la aplicación web a un entorno adaptativo en términos de elementos, apariencia y contenido para dispositivos móviles (teléfonos inteligentes o *tablets*) de manera que los usuarios puedan conectarse a la infraestructura desde cualquier medio.
- Agregar un módulo administrativo que gestione los nodos *Raspberry Pi* desde la aplicación web que contenga funcionalidades como: Apagar un nodo particular, agregar o eliminar archivos a un directorio, configurar direcciones ip, entre otros.
- Generar preguntas de tipo examen a través de formularios en la aplicación web que le permitan al profesor de la cátedra la generación de gráficos estadísticos que proporcionen información acerca de los tópicos o temas donde los estudiantes requieran reforzar sus conocimientos.

- Elaborar aplicaciones de sistemas distribuidos que posean una complejidad superior, mediante la combinación de algoritmos, tópicos, generación de nuevos eventos por parte del usuario, guardar el estado de los nodos, entre otros, de manera que los estudiantes puedan interactuar con un sistema distribuido más real.

## REFERENCIAS BIBLIOGRÁFICAS

---

1. Coulouris, G. (2012). *Distributed Systems: Concepts and Design* (5ta ed.).
2. Tanenbaum, A., & Van Steen, M. (2008). *Sistemas Distribuidos: Principios y Paradigmas*.
3. Tanenbaum, A. (1995). *Sistemas Operativos Distribuidos*.
4. Oracle. (1995 - 2013). *What is a socket?* Obtenido de Oracle Documentation:  
<http://docs.oracle.com/javase/tutorial/networking/sockets/definition.html>
5. The Raspberry Pi Foundation. (2013). *Raspberry Pi*. Obtenido de  
<http://www.raspberrypi.org/>
6. Raspbian. (2013). *Raspbian*. Obtenido de <http://www.raspbian.org/>
7. Zwass, V. (2013). *Encyclopædia Britannica: Agent (Computer Science)*. Obtenido de <http://www.britannica.com/EBchecked/topic/705121/agent>
8. Bradshaw, J. M. (1997). *Software Agent: An introduction to software agents*. Obtenido de <http://agents.umbc.edu/introduction/01-Bradshaw.pdf>
9. Jeffries, R. E. (1999 - 2013). *XProgramming: An agile software development resource*. Obtenido de Xprogramming:  
<http://xprogramming.com/>
10. Ambler, S. W. (2003-2012). *Introduction to Class Responsibility Collaborator (CRC) Models*. Obtenido de Agile Modeling:  
<http://www.agilemodeling.com/artifacts/crcModel.htm>
11. Stotts, D. (2000). *CRC Cards*. Obtenido de University of North Carolina Department of Computer Science:  
<http://www.cs.unc.edu/~stotts/145/homes/crypt/CRC.html>
12. *Code Conventions for the Java Programming Language*. (20 de Abril de 1999). Obtenido de Oracle:  
<http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html>

13. Fung, K. H. (Febrero de 2003). *Code Conventions for the JavaServer Pages Technology*. Obtenido de Oracle: <http://www.oracle.com/technetwork/articles/javase/code-convention-138726.html>
14. Wells, D. (2009). *Extreme Programming: A gentle introduction*. Obtenido de Extreme programming: <http://www.extremeprogramming.org/>
15. Saternos, C. (Noviembre de 2005). *An Introduction to Linux Shell Scripting for DBAs*. Obtenido de Oracle: <http://www.oracle.com/technetwork/articles/linux/saternos-scripting-088882.html>
16. University of Edinburgh. (25 de September de 1999). *UNIX man pages: ssh (1)*. Obtenido de <http://unixhelp.ed.ac.uk/CGI/man-cgi?ssh+1>
17. *JSTL Introduction*. (s.f.). Obtenido de Apekshit: <http://www.apekshit.com/t/83/JSTL-Introduction>
18. Eckstein, R. (Marzo de 2007). *Java SE Application Design With MVC*. Obtenido de Oracle: <http://www.oracle.com/technetwork/articles/javase/index-142890.html>
19. *JSP - MVC Tutorial*. (s.f.). Obtenido de Data Disk: [http://www.datadisk.co.uk/html\\_docs/jsp/jsp\\_mvc\\_tutorial.htm](http://www.datadisk.co.uk/html_docs/jsp/jsp_mvc_tutorial.htm)
20. *Singleton Pattern*. (s.f.). Obtenido de Object Oriented Design: <http://www.oodesign.com/singleton-pattern.html>

## ANEXOS.

---

### Anexo 1. Backlogs.

Nombre Historia de Usuario	Iteración	Prioridad	Tiempo de desarrollo (horas)
Configurar atributos (Librería de mensajes)	I	Must	2
Enviar mensajes simples (Librería de mensajes)	I	Must	5
Recibir detalles del mensaje (Librería de mensajes)	I	Must	1
Enviar información del agente (Librería de mensajes)	I	Must	1
Recolectar información sobre el sistema operativo (Agente de configuración)	II	Must	5
Recolectar información de red (Agente de configuración)	II	Must	2
Recolectar información de la aplicación actual (Agente de configuración)	II	Must	2
Ejecutar agente (Agente de configuración)	II	Must	3
Enviar información (Agente de configuración)	II	Must	1
Reenviar mensajes (Servidor central)	III	Must	3
Gestionar repositorio local (Servidor central)	III	Must	4
Ejecutar scripts (Servidor central)	III	Must	1
Conectar a la base de datos (Servidor central)	III	Must	1
Obtener información de los nodos del sistema distribuido (Servidor central)	III	Must	3
Acceder al módulo (Módulo gestión, aplicación web)	IV	Must	2
Ejecutar operaciones de base de datos (Módulo gestión, aplicación web)	IV	Must	10
Cargar aplicaciones (Módulo gestión, aplicación web)	IV	Must	5
Utilizar formularios dinámicos (Módulo gestión, aplicación web)	IV	Must	3

Consultar eventos (Módulo de ciclo de vida, aplicación web)	IV	Must	2
Generar botones de eventos (Módulo de ciclo de vida, aplicación web)	IV	Must	3
Informar al servidor central (Módulo de ciclo de vida, aplicación web)	IV	Must	4
Recibir información (Módulo de monitoreo, aplicación web)	IV	Must	2
Separar información (Módulo de monitoreo, aplicación web)	IV	Must	3
Verificar mensajes en un lapso de tiempo (Módulo de monitoreo, aplicación web)	IV	Must	4
Generar preguntas (Aplicación web)	IV	Must	6
Mostrar escenarios (Aplicación web)	IV	Must	3
Poseer información sobre la aplicación (Aplicaciones sistemas distribuidos)	V	Must	1
Recibir mensajes del agente (Aplicaciones sistemas distribuidos)	V	Must	2
Enviar información al servidor central (Aplicaciones sistemas distribuidos)	V	Must	1
Almacenar archivos de lectura (Características de los sistemas distribuidos)	V	Must	2
Iniciar servidor web (Características de los sistemas distribuidos)	V	Must	1
Acceder archivo (Características de los sistemas distribuidos)	V	Must	2
Iniciar segundo nodo servidor (Características de los sistemas distribuidos)	V	Must	1
Crear cliente (Características de los sistemas distribuidos)	V	Must	2
Redirigir mensajes (Características de los sistemas distribuidos)	V	Must	1
Cifrar conexión (Desafíos de los sistemas distribuidos)	V	Must	3
Almacenar archivos descargables (Desafíos de los sistemas distribuidos)	V	Must	2
Utilizar semáforos (Desafíos de los sistemas distribuidos)	V	Must	2
Crear cliente (Desafíos de los sistemas)	V	Must	2

distribuidos)			
Crear servidor (Arquitectura cliente / servidor)	V	Must	1
Crear cliente (Arquitectura cliente / servidor)	V	Must	1
Crear cliente (Peer to peer)	V	Must	2
Crear cliente (Sockets)	V	Must	1
Implementar métodos remotos (RMI 1)	V	Must	2
Invocar métodos remotos (RMI 1)	V	Must	1
Asociar nodo a grupo receptor (Comunicación en grupo)	V	Must	1
Crear grupo emisor (Comunicación en grupo)	V	Must	2
Enviar marcas de reloj ( Algoritmo de Lamport)	V	Must	1
Verificar marca de reloj ( Algoritmo de Lamport)	V	Must	2
Enviar hora (Algoritmo de Cristian)	V	Must	1
Obtener hora de envío (Algoritmo de Cristian)	V	Must	1
Recibir hora (Algoritmo de Cristian)	V	Must	3
Preguntar hora (Algoritmo de Berkeley)	V	Must	1
Calcular promedio (Algoritmo de Berkeley)	V	Must	1
Recibir y enviar hora (Algoritmo de Berkeley)	V	Must	3
Enviar hora (Algoritmo con promedio)	V	Must	1
Calcular promedio (Algoritmo con promedio)	V	Must	2
Recibir mensajes solicitud (Algoritmo centralizado)	V	Must	1
Verificar región (Algoritmo centralizado)	V	Must	1
Solicitar región (Algoritmo centralizado)	V	Must	1
Enviar mensaje solicitud (Algoritmo distribuido)	V	Must	1
Enviar mensaje de aceptacion (Algoritmo distribuido)	V	Must	1
Guardar peticiones (Algoritmo distribuido)	V	Must	2
Comparar marca de reloj (Algoritmo distribuido)	V	Must	3
Comparar mensajes recibidos (Algoritmo distribuido)	V	Must	2
Enviar mensajes a los mayores (Algoritmo grandulón)	V	Must	1
Responder mensaje (Algoritmo grandulón)	V	Must	1
Enviar mensaje de coordinador ( Algoritmo grandulón)	V	Must	1
Enviar mensaje al siguiente nodo (Algoritmo de anillo)	V	Must	1
Recibir mensaje de elección (Algoritmo de anillo)	V	Must	1

Elegir coordinador (Algoritmo de anillo)	V	Must	1
Generar falla de congelación (Tipos de fallas)	V	Must	1
Generar falla de omisión (Tipos de fallas)	V	Must	1
Generar falla de tiempo (Tipos de fallas)	V	Must	1
Generar falla de respuesta (Tipos de fallas)	V	Must	1
Crear cliente (Tipos de fallas)	V	Must	1
Enviar número de nodo (Fallas bizantinas)	V	Must	1
Generar nodo corrupto (Fallas bizantinas)	V	Must	2
Recibir número de nodo (Fallas bizantinas)	V	Must	1
Enviar vector (Fallas bizantinas)	V	Must	1
Recibir vector (Fallas bizantinas)	V	Must	3
Almacenar archivo (Modelo acceso remoto)	V	Must	1
Acceder archivo (Modelo acceso remoto)	V	Must	1
Almacenar archivo descargable (Modelo carga y descarga)	V	Must	2
Acceder archivo local (Modelo carga y descarga)	V	Must	1
Almacenar archivo en partes (Sistemas de archivos basados en clúster)	V	Must	5
Almacenar archivo completo (Sistemas de archivos basados en clúster)	V	Must	2
Descargar archivos (Sistemas de archivos basados en clúster)	V	Must	2
Almacenar dominios y direcciones ip (Domain name system)	V	Must	2
Consultar dominios (Domain name system)	V	Must	1
Agregar dominios (Domain name system)	V	Must	1
Almacenar dominios con usuarios (Lightweight Directory Access Protocol)	V	Must	3
Consultar usuarios (Lightweight Directory Access Protocol)	V	Must	2
Agregar usuario (Lightweight Directory Access Protocol)	V	Must	2
Implementar métodos remotos (RMI 2)	V	Must	1
Invocar métodos remotos (RMI 2)	V	Must	1
Almacenar beans de sesión (Enterprise Java Beans)	V	Must	3
Consumir beans de sesión (Enterprise Java Beans)	V	Must	2

Almacenar servicios web (Simple object access protocol)	V	Must	3
Consumir servicios (Simple object access protocol)	V	Must	2
Almacenar servicios (Representational State Transfer)	V	Must	4
Consumir servicios (Representational State Transfer)	V	Must	2

Tabla 11: Backlogs historias de usuario. Elaboración propia.

## Anexo 2. Documento de Diseño.

### Anexo 2.1. Librería de mensajes

#### Anexo 2.1.1. Historias de usuario

Configurar atributos
Como usuario de la infraestructura quiero que la librería posea atributos configurables como (ip del remitente y de los destinatarios entre otros) de manera que puedan ser definidos en cualquier momento.
Enviar mensajes simples
Como usuario de la infraestructura quiero que la librería posea métodos para enviar mensajes de manera que solo se indiquen parámetros básicos (mensaje a enviar, dirección ip y puerto).
Recibir detalles del mensaje
Como usuario de la infraestructura quiero que los mensajes enviados por la librería posean la hora y fecha de su creación, de manera que se pueda conocer en qué momento fueron enviados.
Enviar información del agente
Como usuario de la infraestructura quiero que la librería de mensajes contenga un clase para referenciar a los mensajes del agente de manera que puedan ser recibidos por el servidor central y la aplicación web.

Tabla 12: Historias de usuario Librería de mensajes. Elaboración propia

### Anexo 2.1.2. Diagrama de clases

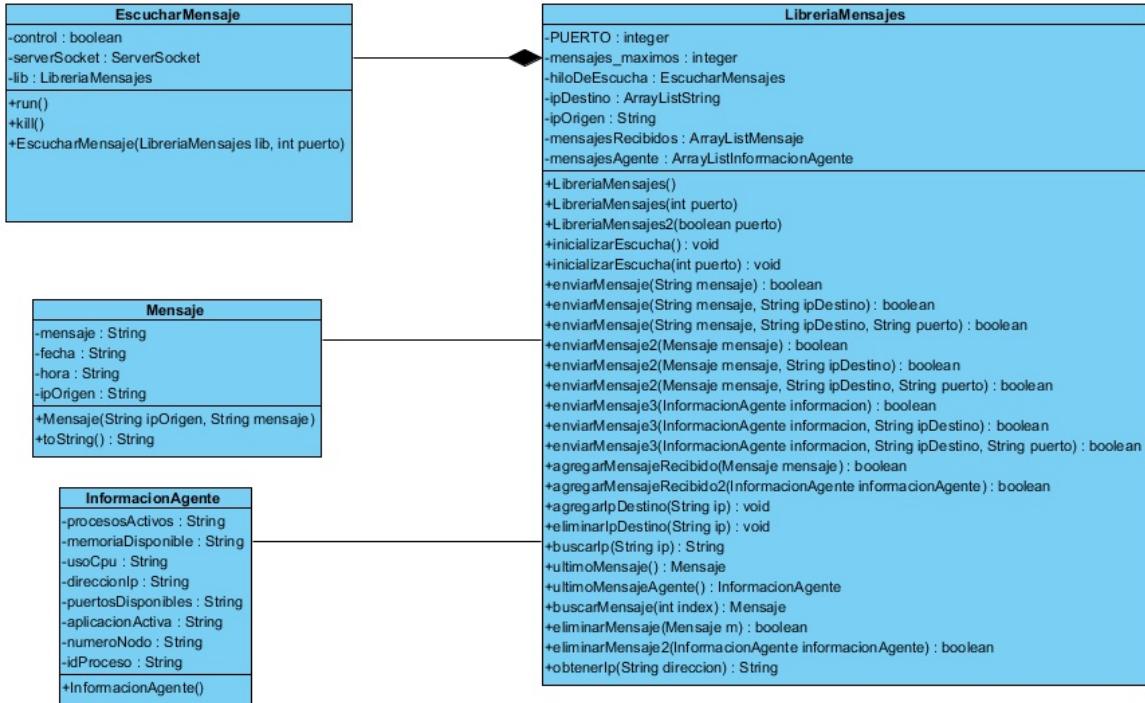


Ilustración 9: Diagrama de clases Librería de mensajes. Elaboración propia.

### Anexo 2.2. Agente de configuración

#### Anexo 2.2.1. Historias de usuario

Recolectar información sobre el sistema operativo
Como usuario de la infraestructura quiero que el agente de configuración obtenga información del sistema operativo (uso CPU, memoria disponibles, procesos) del <i>Raspberry Pi</i> de manera que pueda enviarse hacia el servicio de recepción.
Recolectar información de red
Como usuario de la infraestructura quiero que el agente obtenga la configuración de red (dirección IP, puertos disponibles) del <i>Raspberry Pi</i> de manera que pueda enviarse hacia el servicio de recepción
Recolectar información de la aplicación actual
Como usuario de la infraestructura quiero que el agente obtenga la información de la aplicación que se esté ejecutando (nombre de la aplicación, número de nodo) de manera que se pueda enviar hacia el servicio de recepción.

<b>Ejecutar agente</b>
Como usuario de la infraestructura quiero que el agente de configuración se inicie automáticamente al encender el nodo <i>Raspberry Pi</i> de manera que pueda recolectar y enviar la información.
<b>Enviar información</b>
Como usuario de la infraestructura quiero que el agente de configuración envíe la información recolectada a través de la librería de mensajes de manera que pueda ser visualizada por el usuario.

Tabla 13: Historias de usuario Agente de configuración. Elaboración propia

### Anexo 2.2.2. Diagrama de clases

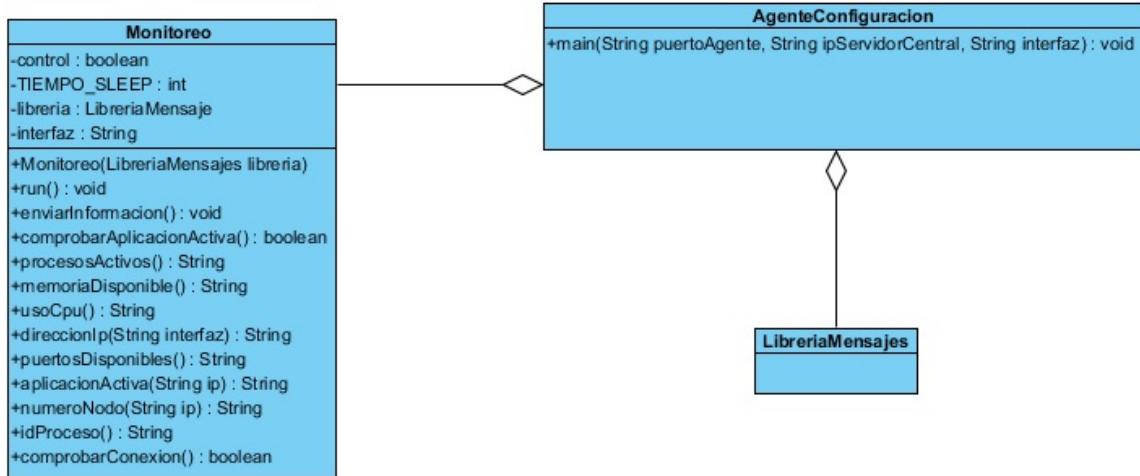


Ilustración 10: Diagrama de clases Agente de configuración. Elaboración propia.

### Anexo 2.3. Servidor Central

#### Anexo 2.3.1. Historia de usuarios

<b>Reenviar mensajes</b>
Como usuario de la infraestructura quiero que el servidor central reciba los mensajes de los agentes y aplicaciones de sistema distribuidos de manera que puedan ser reenviados hacia el módulo de monitoreo.
<b>Gestionar repositorio local</b>

Como usuario de la infraestructura quiero que exista un repositorio local que permita almacenar los ejecutables de las aplicaciones de sistemas distribuidos y los scripts de ejecución de manera que puedan ser llamados por el módulo de ciclo de vida.
<b>Ejecutar scripts</b>
Como usuario de la infraestructura quiero que el servidor central pueda recibir los mensajes pertenecientes al módulo de ciclo de vida y ejecutar los <i>scripts</i> de sistemas operativos de manera que pueda ejecutar o eliminar una aplicación en un nodo <i>Raspberry Pi</i>
<b>Conectar a la base de datos</b>
Como usuario de la infraestructura quiero que el servidor central pueda realizar consultas a la base de datos de manera que pueda localizar la información correspondiente a un ejecutable.
<b>Obtener información de los nodos del sistema distribuido</b>
Como usuario de la infraestructura quiero que el servidor central posea información de los nodos que se encuentran activos de manera que se pueda detener su ejecución cuando el usuario lo indique desde el módulo de administración de ciclo de vida.

Tabla 14: Historias de usuario Servidor central. Elaboración propia.

### Anexo 2.3.2. Diagrama de clases:

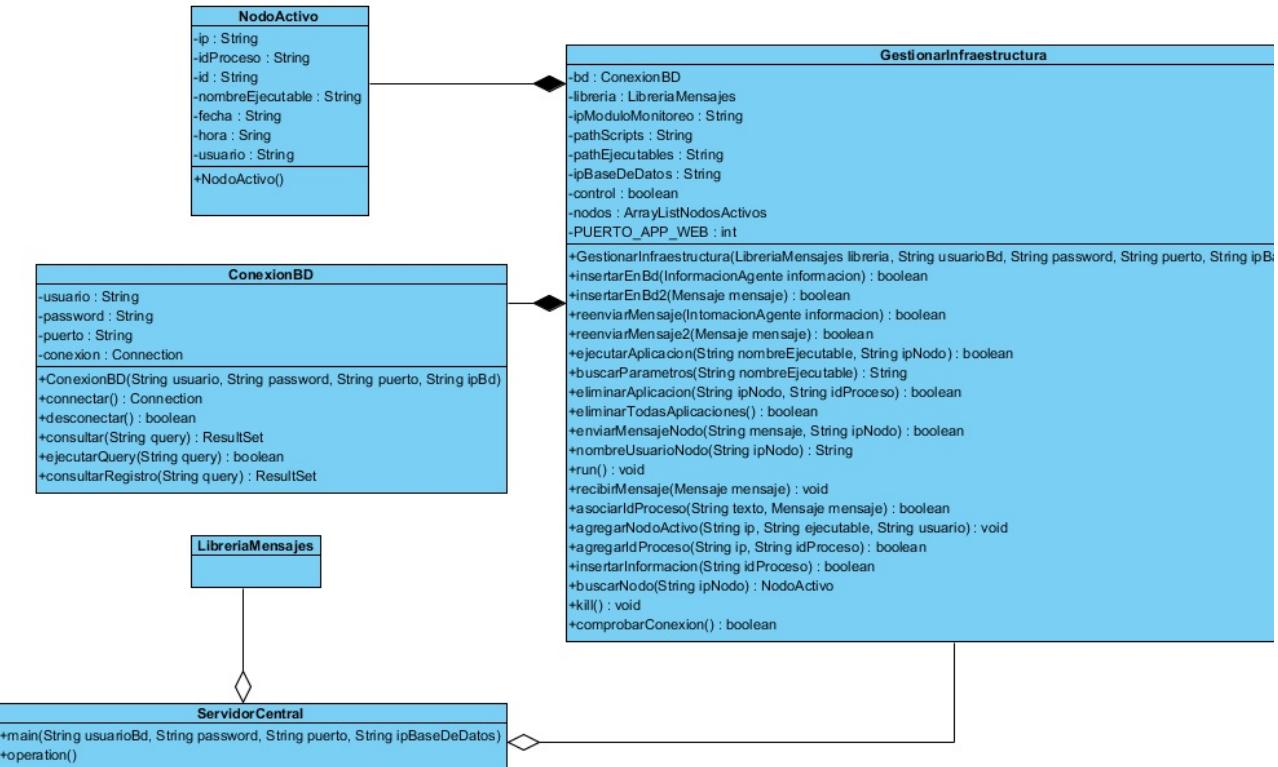


Ilustración 11: Diagrama de clases Servidor central. Elaboración propia.

## Anexo 2.4. Aplicación Web

### Anexo 2.4.1. Historias de usuario:

#### Módulo de gestión

<b>Acceder al módulo</b>
Como usuario de la infraestructura quiero el módulo de gestión sea accedido únicamente por usuarios que posean una cuenta de manera que se restrinja el acceso a los demás usuarios.
<b>Ejecutar operaciones de base de datos</b>
Como usuario de la infraestructura quiero que el módulo de gestión permita realizar las operaciones de crear, modificar, consultar y eliminar de cada tabla de la base de datos de manera que pueda ser modificada en cualquier momento.
<b>Cargar aplicaciones</b>
Como usuario de la infraestructura quiero que el módulo de gestión permita almacenar los ejecutables de las aplicaciones de sistemas distribuidos junto a sus parámetros de ejecución.
<b>Utilizar formularios dinámicos</b>
Como usuario de la infraestructura quiero que el módulo de gestión contenga formularios dinámicos para mostrar la información de manera que el usuario pueda indicar la cantidad de campos a llenar.

Tabla 15: Historias de usuario Módulo de gestión. Elaboración propia.

#### Módulo administración ciclo de vida:

<b>Consultar eventos</b>
Como usuario de la infraestructura quiero que el módulo obtenga los eventos asociados al ejecutable de la aplicación desde la base de datos de manera que el usuario los pueda generar en cualquier momento.
<b>Generar botones de eventos</b>
Como usuario de la infraestructura quiero que existan botones dentro del módulo de ciclo de vida asociados a los eventos de manera que el usuario pueda controlar los nodos a través de ellos.

Como usuario de la infraestructura quiero el módulo de ciclo de vida se comunique con el servidor central para enviar las instrucciones indicadas por el usuario de manera que puedan ejecutarse.

**Tabla 16: Historias de usuario Módulo de administración del ciclo de vida. Elaboración propia.**

### Módulo de monitoreo

<b>Recibir información</b>
Como usuario de la infraestructura quiero que el módulo de monitoreo reciba la información enviada por el servidor central de manera que pueda ser visualizada por el usuario.
<b>Separar información</b>
Como usuario de la infraestructura quiero que el módulo de monitoreo permita separar la información recibida de acuerdo al tipo de información: información de los agentes e información de los nodos de la aplicación.
<b>Verificar mensajes en un lapso de tiempo</b>
Como usuario de la infraestructura quiero que el módulo de monitoreo verifique cada cierto tiempo si han llegado mensajes de manera que se puedan mostrar en pantalla

**Tabla 17: Historias de usuario Módulo de monitoreo. Elaboración propia.**

### Otros:

<b>Generar preguntas</b>
Como usuario de la infraestructura quiero se puedan generar preguntas aleatoriamente en relación a un tópico de manera que los usuarios puedan descargarla para estudiar.
<b>Mostrar escenarios</b>
Como usuario de la infraestructura quiero que se pueda visualizar los escenarios asociados a la aplicación en ejecución de manera que se pueda comparar

**Tabla 18: Historias de usuario Aplicación web. Elaboración propia.**

### Anexo 2.4.2. Diagrama de clases

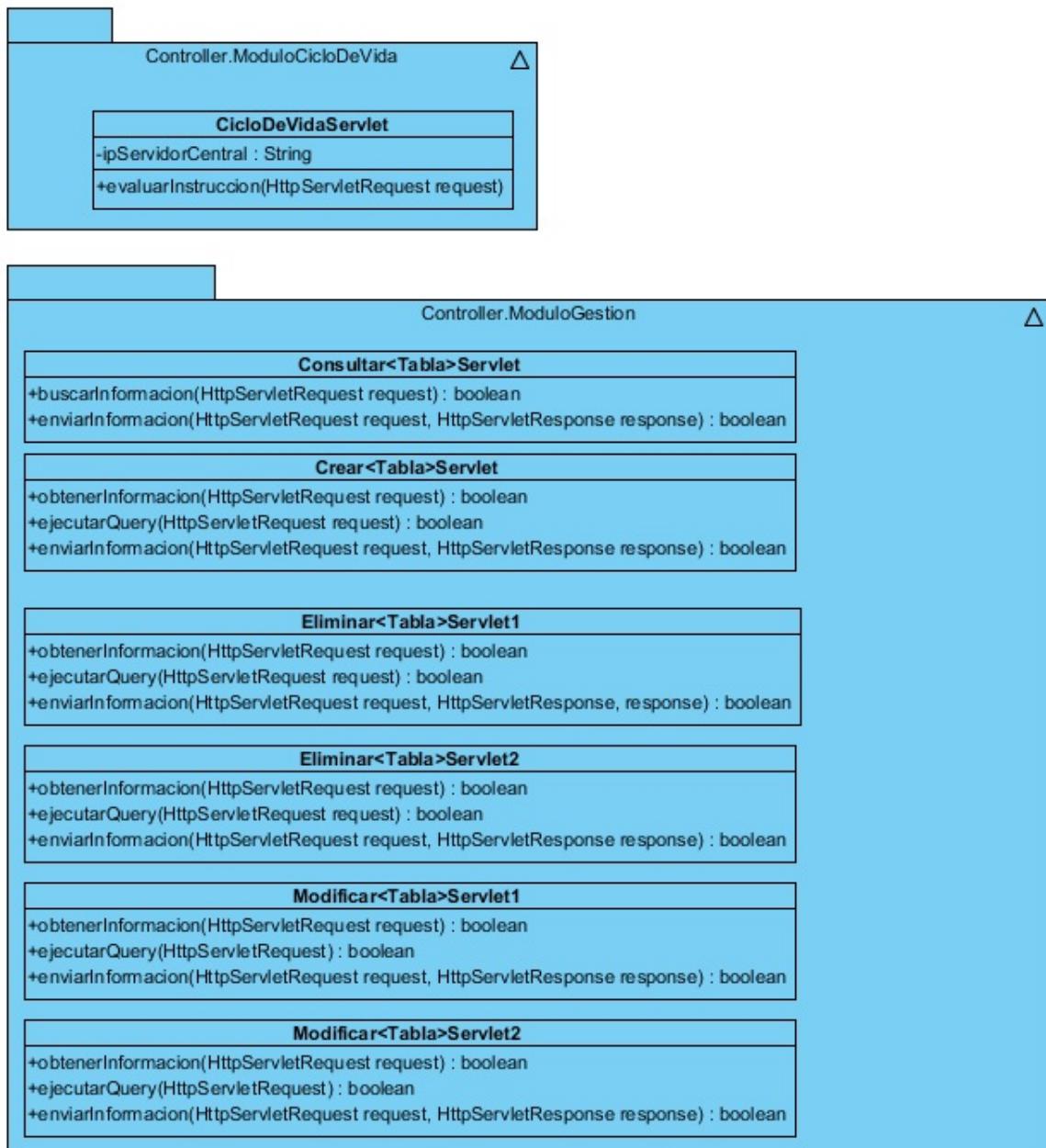


Ilustración 12: Diagrama de clases Módulo ciclo de vida y Módulo de gestión. Elaboración propia.

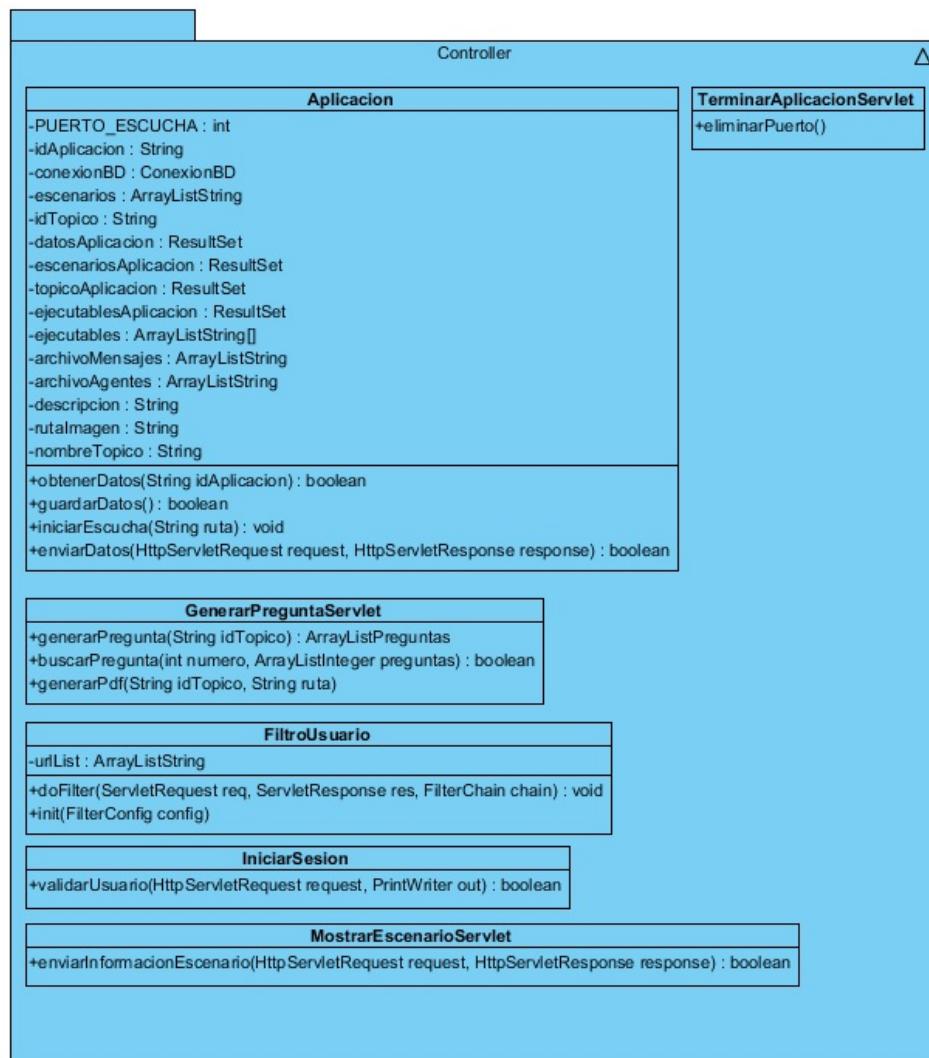


Ilustración 13: Diagrama de clases Aplicación web. Elaboración propia.

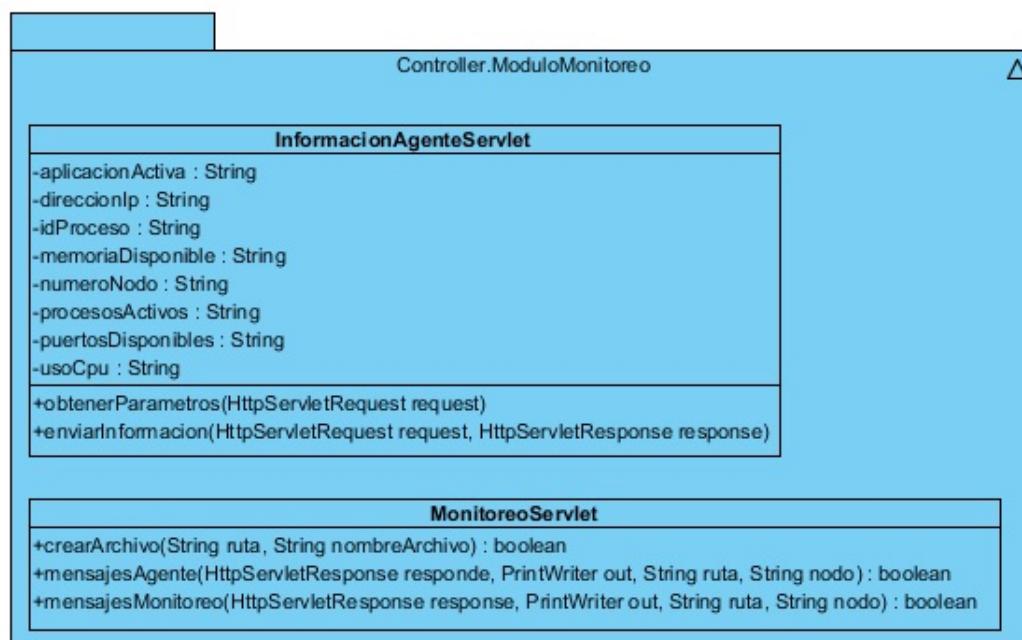


Ilustración 14: Diagrama de clases Módulo de monitoreo. Elaboración propia.

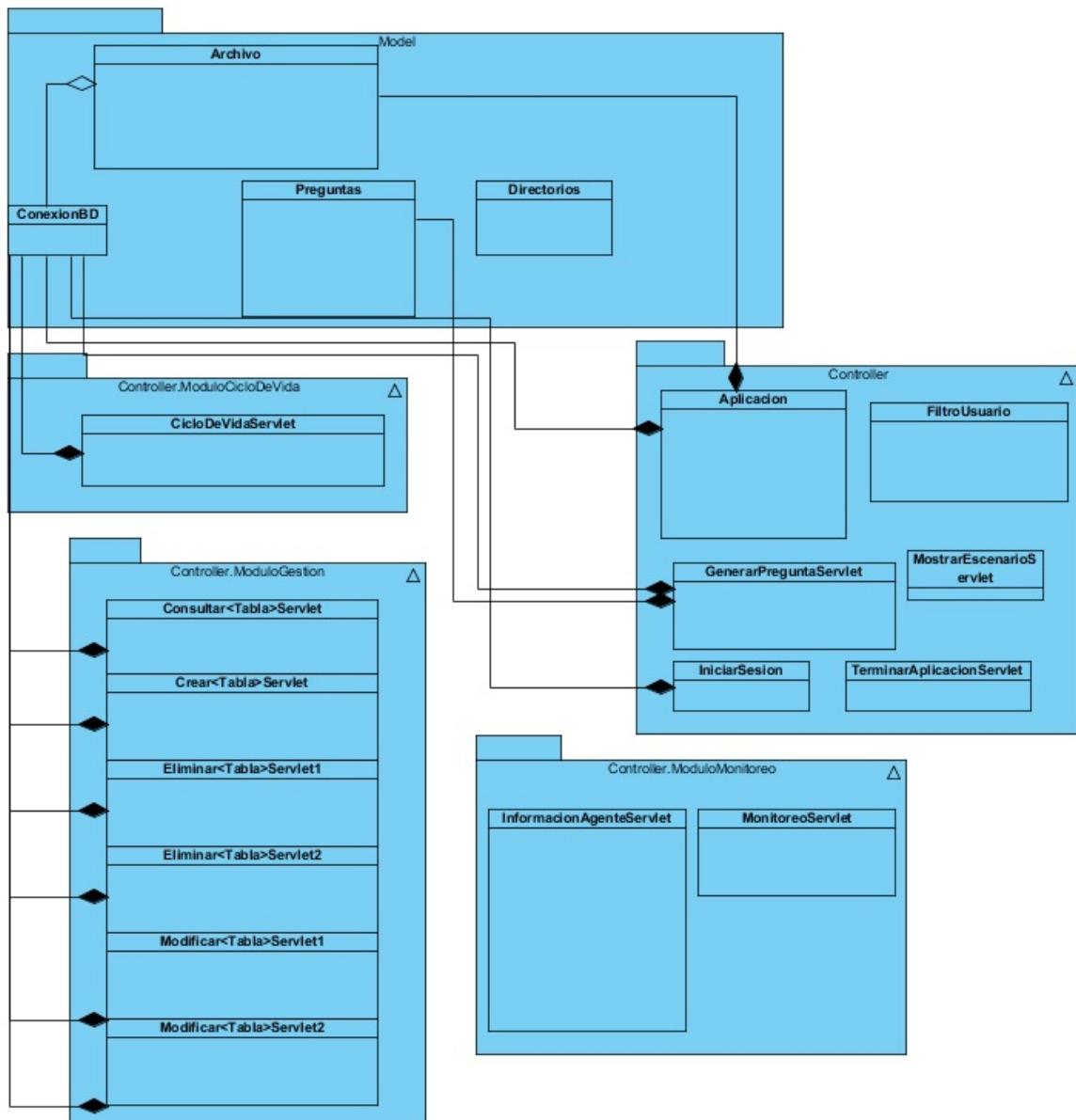


Ilustración 15: Diagrama de clases relaciones Aplicación Web y Módulos. Elaboración propia.

## Anexo 2.5. Aplicaciones Sistemas Distribuidos

### Anexo 2.5.1. Historias de usuario

Poseer información sobre la aplicación
Como usuario de la infraestructura quiero que las aplicaciones de sistemas distribuidos contengan los datos: nombre de la aplicación, número de nodo y número de proceso de

manera que pueda ser consultado por el agente de configuración.
<b>Recibir mensajes del agente</b>
Como usuario de la infraestructura quiero que las aplicaciones permitan recibir información del agente de configuración de manera que este pueda recolectar la información correspondiente a la aplicación.
<b>Enviar información al servidor central</b>
Como usuario de la infraestructura quiero que las aplicaciones envíen los mensajes remitidos hacia otros nodos o mensajes propios al servidor central de manera que puedan ser redirigidas al módulo de monitoreo para ser visualizada por el usuario.

**Tabla 19: Historias de usuario Aplicaciones de sistemas distribuidos. Elaboración propia.**

### Anexo 2.5.2. Características de los sistemas distribuidos

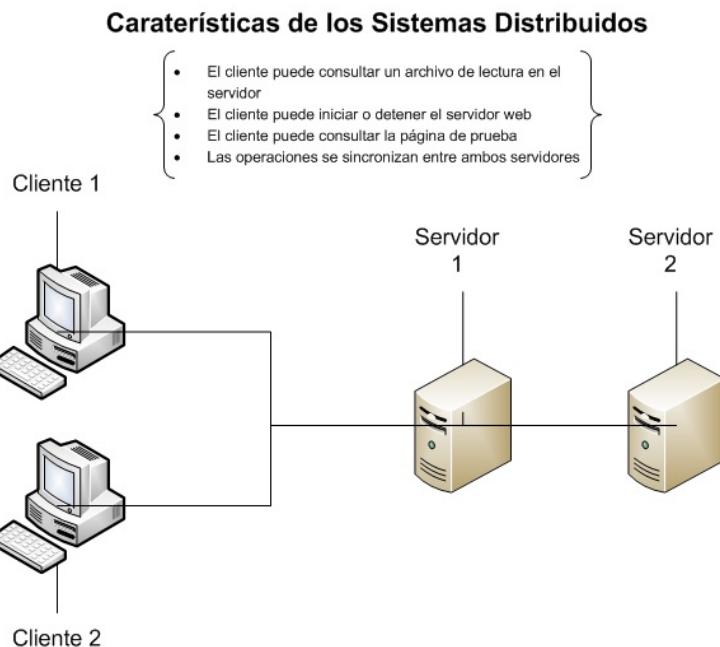
#### Anexo 2.5.2.1. Historias de usuario

<b>Almacenar archivos de lecturas</b>
Como usuario de la infraestructura quiero que exista una aplicación servidor que posea un conjunto de archivos almacenados localmente para ser accedidos por los usuarios en modo lectura de manera que se pueda comprobar la compartición de recursos en los sistemas distribuidos.
<b>Iniciar servidor web</b>
Como usuario de la infraestructura quiero que el servidor posea un servidor web para ser iniciado en un momento determinado para consultar una página de prueba por parte de los clientes de manera que se pueda comprobar la apertura de los sistemas distribuidos.
<b>Acceder archivo</b>
Como usuario de la infraestructura quiero que el servidor permita que otro nodo cliente pueda acceder al archivo al mismo tiempo de manera que se pueda comprobar la concurrencia de los sistemas distribuidos.
<b>Iniciar segundo nodo servidor</b>
Como usuario de la infraestructura quiero se puede iniciar otro nodo servidor de manera que se pueda comprobar la escalabilidad de los sistemas distribuidos.
<b>Crear cliente</b>
Como usuario de la infraestructura quiero exista una aplicación cliente que permita conectarse al servidor de manera que se pueda acceder a sus recursos y comprobar las características de los sistemas distribuidos.
<b>Redirigir mensajes</b>

Como usuario de la infraestructura quiero la aplicación cliente permita redirigir las peticiones a otro nodo al momento de eliminarse un nodo servidor de manera que se pueda seguir accediendo a los recursos compartidos y comprobar la tolerancia a fallas.

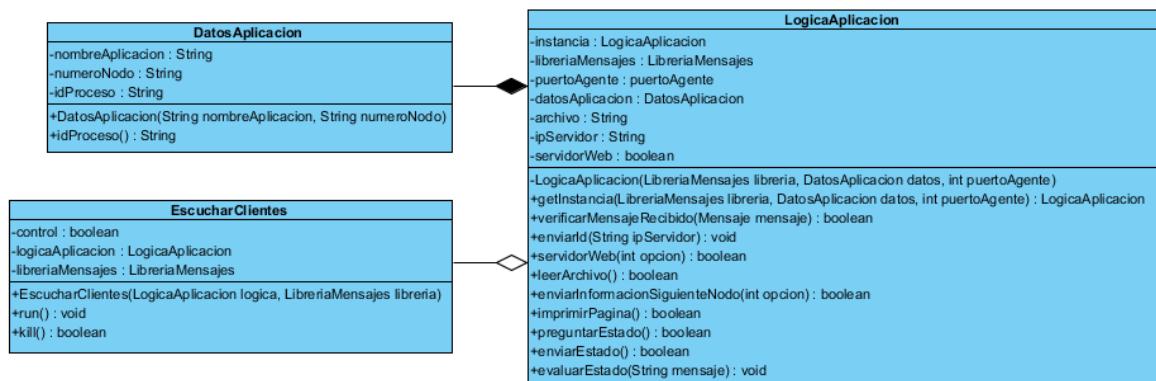
**Tabla 20: Historias de usuario Características de los sistemas distribuidos. Elaboración propia.**

#### Anexo 2.5.2.2. Arquitectura



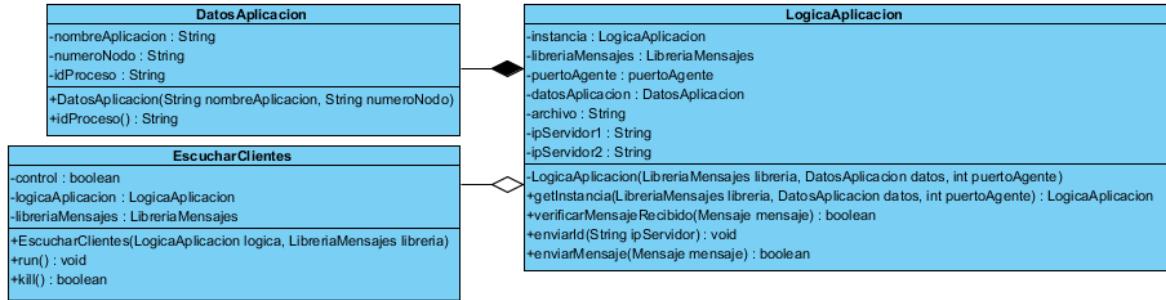
**Ilustración 16: Arquitectura Características de los sistemas distribuidos. Elaboración propia.**

#### Anexo 2.5.2.3. Diagrama de clases servidor



**Ilustración 17: Diagrama de clases características de los sistemas distribuidos - Servidor. Elaboración propia.**

#### Anexo 2.5.2.4. Diagrama de clases cliente



**Ilustración 18: Diagrama de clases Características de los sistemas distribuidos - Cliente.**  
Elaboración propia.

#### Anexo 2.5.2.5. Parámetros propios

- Servidor: Ip segundo servidor
- Cliente: Ip servidor 1, ip servidor 2

### Anexo 2.5.3. Desafíos de los Sistemas Distribuidos

#### Anexo 2.5.3.1. Historias de usuario

Cifrar conexión
Como usuario de la infraestructura quiero exista una aplicación servidor que permita la conexión con clientes a través de sockets seguros utilizando SSL de manera que se pueda comprobar la seguridad de los sistemas distribuidos.
Almacenar archivos descargables
Como usuario de la infraestructura quiero que el servidor posea archivos almacenado que permitan ser descargados por el cliente junto a su <i>checksum</i> de manera que se pueda comprobar la tolerancia a fallos.
Utilizar semáforos
Como usuario de la infraestructura quiero el servidor utilice semáforos que controle los accesos a los archivos por parte de los clientes de manera que se pueda comprobar la concurrencia de los sistemas distribuidos.
Crear cliente
Como usuario de la infraestructura quiero que exista una aplicación cliente que permita conectarse al servidor de manera que se puedan acceder a los recursos (archivos) y comprobar los desafíos de los sistemas distribuidos.

**Tabla 21: Historias de usuario Desafíos de los sistemas distribuidos. Elaboración propia.**

## Anexo 2.5.3.2. Arquitectura

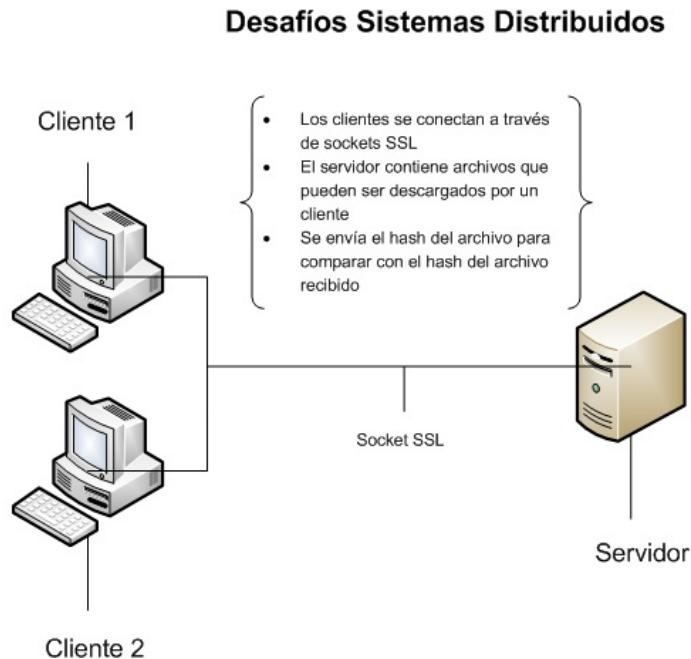


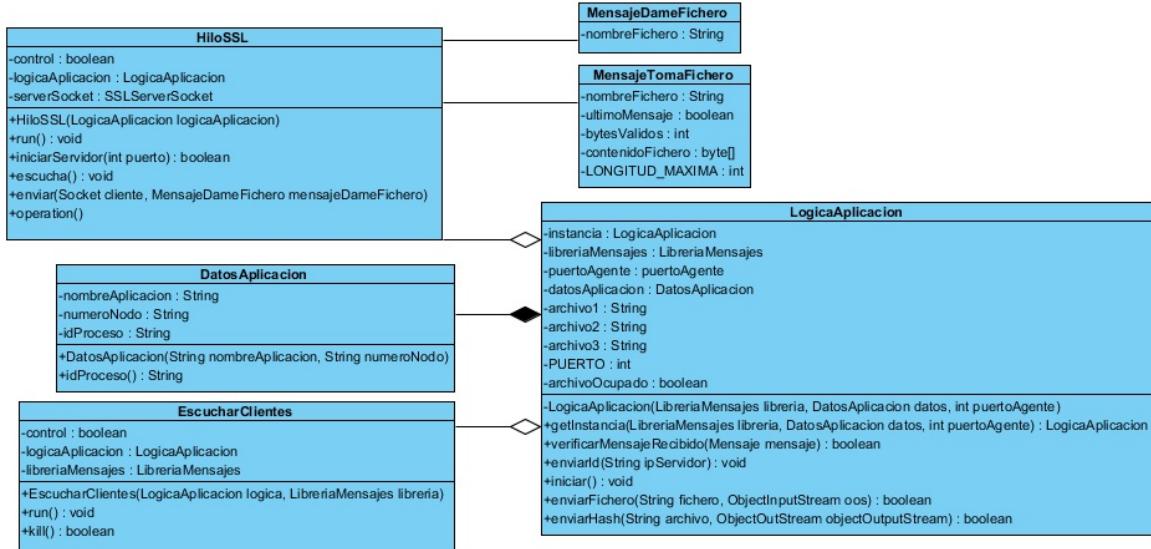
Ilustración 19: Arquitectura Desafíos de los sistemas distribuidos. Elaboración propia.

## Anexo 2.5.3.3. Diseño (Tarjetas CRC) - Servidor

HiloSSL	
<ul style="list-style-type: none"> <li>• Abre un puerto para permitir la conexión segura con los clientes a través de SSL.</li> <li>• Le indica a la lógica si se recibió una petición del usuario de archivo solicitado.</li> </ul>	LibreríaMensajes LogicaAplicacion MensajeDameFichero
<b>MensajeDameFichero</b>	
<ul style="list-style-type: none"> <li>• Posee el nombre de un archivo a transmitir por socket</li> </ul>	
<b>MensajeTomaFichero</b>	
<ul style="list-style-type: none"> <li>• Contiene los datos del contenido del archivo a enviarse por socket</li> </ul>	

Tabla 22: Tarjetas CRC Desafíos de los sistemas distribuidos - Servidor. Elaboración propia.

#### Anexo 2.5.3.4. Diagrama de clases servidor



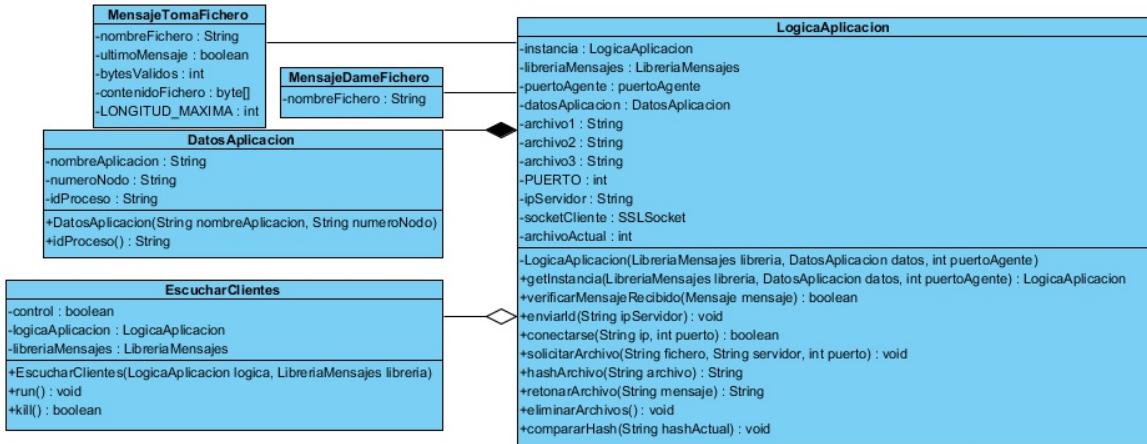
**Ilustración 20: Diagrama de clases Desafíos de los sistemas distribuidos - Servidor. Elaboración propia.**

#### Anexo 2.5.3.5. Diseño (Tarjetas CRC) - Cliente

MensajeDameFichero	
<ul style="list-style-type: none"> <li>Posee el nombre de un archivo a transmitir por socket</li> </ul>	
MensajeTomaFichero	
<ul style="list-style-type: none"> <li>Contiene los datos del contenido del archivo a enviarse por socket</li> </ul>	

**Tabla 23: Tarjetas CRC Desafíos de los sistemas distribuidos - Cliente. Elaboración propia.**

#### Anexo 2.5.3.6. Diagrama de clases cliente



**Ilustración 21: Diagrama de clases Desafíos de los sistemas distribuidos - Cliente. Elaboración propia.**

#### Anexo 2.5.3.7. Parámetros propios

- Cliente: Ip servidor

#### Anexo 2.5.4. Arquitectura Cliente / Servidor

##### Anexo 2.5.4.1. Historias de usuario

<b>Crear servidor</b>
Como usuario de la infraestructura quiero que exista una aplicación servidor capaz de recibir un mensaje de información por parte de un cliente de manera que pueda responder con mensajes inmediatamente y comprobar la comunicación entre ellos.
<b>Crear cliente</b>
Como usuario de la infraestructura quiero que exista una aplicación cliente capaz de enviar mensajes a un servidor específico de manera que se reciba un mensaje de respuesta para comprobar la comunicación entre ellos.

**Tabla 24: Historias de usuario Arquitectura Cliente / Servidor. Elaboración propia.**

## Anexo 2.5.4.2. Arquitectura

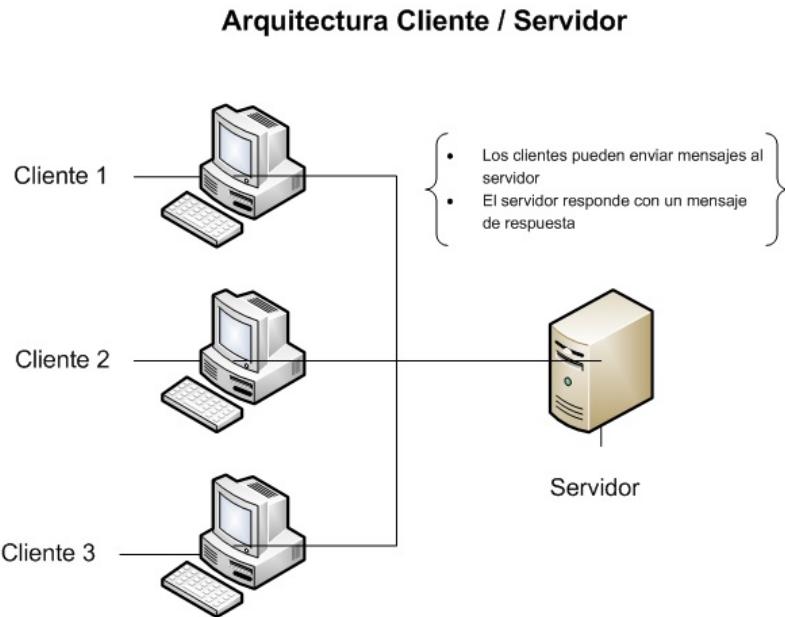


Ilustración 22: Arquitectura Arquitectura Cliente / Servidor. Elaboración propia.

## Anexo 2.5.4.3. Diagrama de clases servidor

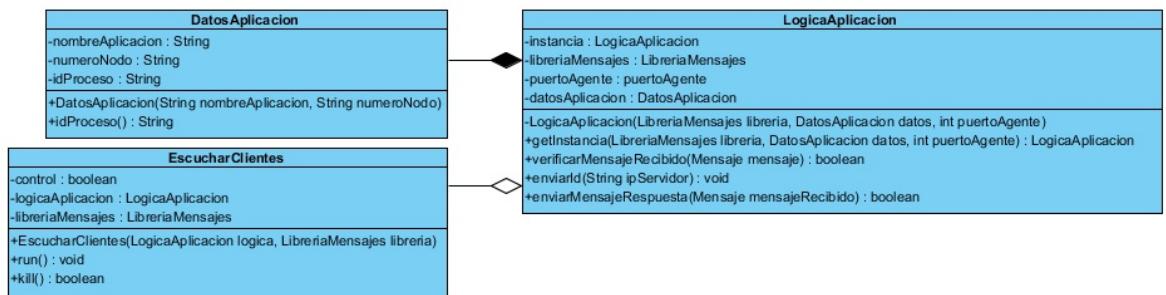
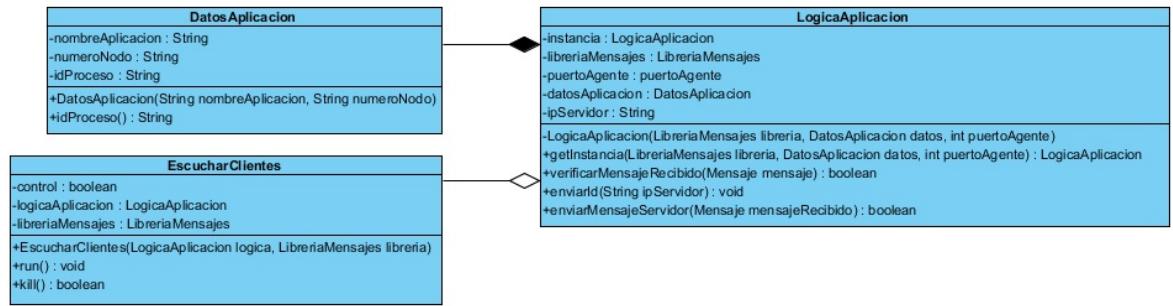


Ilustración 23: Diagrama de clases Arquitectura Cliente / Servidor - Servidor. Elaboración propia.

## Anexo 2.5.4.4. Diagrama de clases cliente

**Ilustración 24: Diagrama de clases Arquitectura Cliente / Servidor - Cliente. Elaboración propia.****Anexo 2.5.4.5. Parámetros propios**

- Cliente: Ip servidor.

**Anexo 2.5.5. Peer to peer (P2P)****Anexo 2.5.5.1. Historias de usuario**

<b>Crear cliente</b>
Como usuario de la infraestructura quiero que exista un aplicación que permita enviar y recibir mensajes a cualquier host activo de manera que puedan agregarse o eliminarse nodos sin afectar el envío de información.

**Tabla 25: Historias de usuario Peer to peer. Elaboración propia.****Anexo 2.5.5.2. Arquitectura**

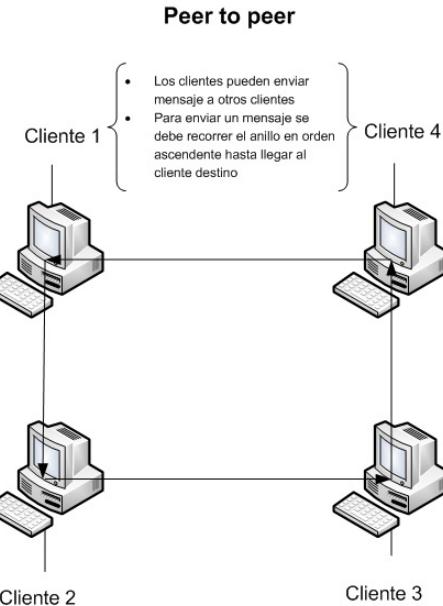


Ilustración 25: Arquitectura Peer to peer. Elaboración propia.

#### Anexo 2.5.5.3. Diagrama de clases

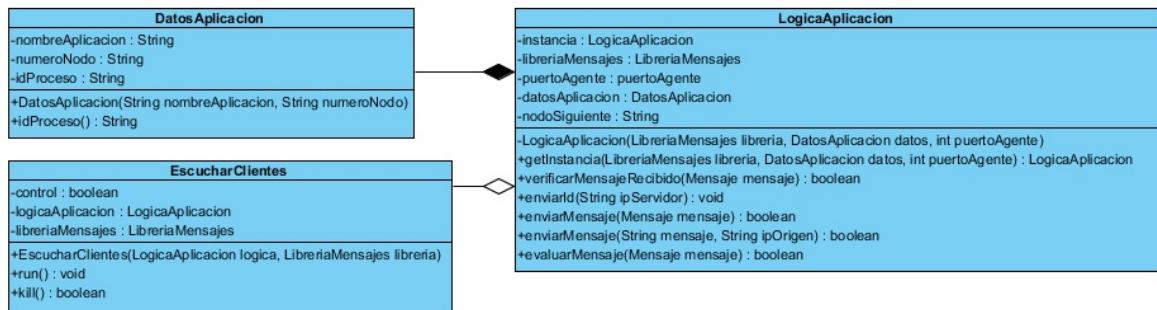


Ilustración 26: Diagrama de clases Peer to peer. Elaboración propia.

#### Anexo 2.5.5.4. Parámetros propios

- Cliente: Ip siguiente nodo

#### Anexo 2.5.6. Sockets

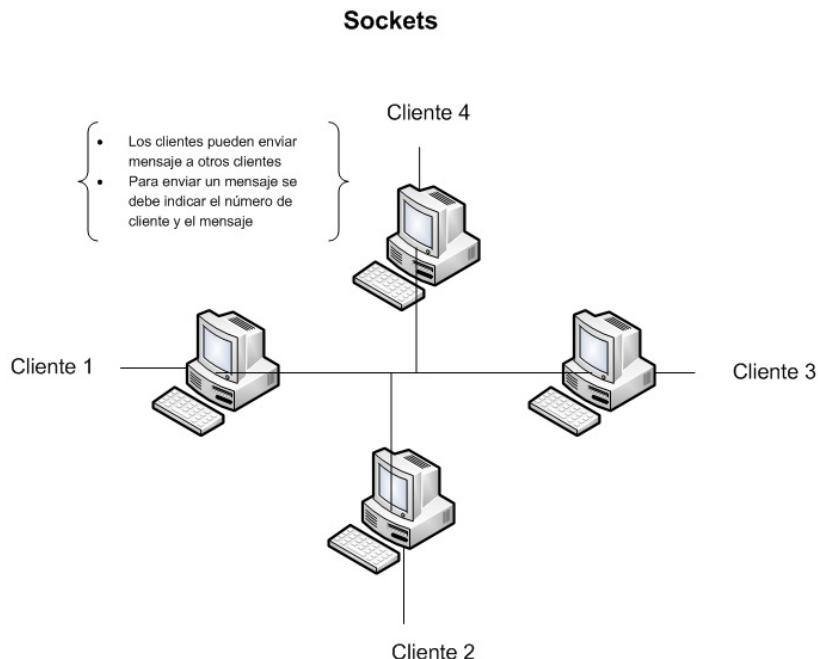
##### Anexo 2.5.6.1. Historias de usuario

Crear cliente

Como usuario de la aplicación quiero que exista una aplicación que permita enviar y recibir mensajes a un nodo particular de manera que se pueda comprobar el concepto de socket.

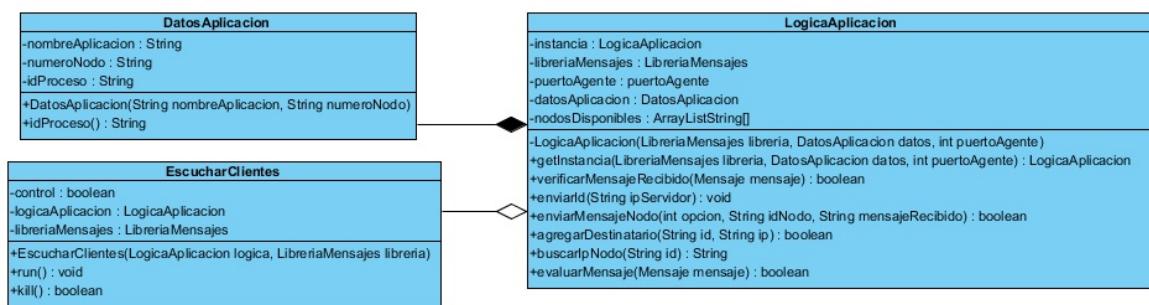
**Tabla 26: Historias de usuario Sockets. Elaboración propia.**

#### Anexo 2.5.6.2. Arquitectura



**Ilustración 27: Arquitectura Sockets. Elaboración propia.**

#### Anexo 2.5.6.3. Diagrama de clases



**Ilustración 28: Diagrama de clases Sockets. Elaboración propia.**

#### Anexo 2.5.6.4. Parámetros propios

- Cliente: Id nodo 1, id nodo 2, id nodo 3, id nodo 4.

## Anexo 2.5.7. RMI

### Anexo 2.5.7.1. Historias de usuario

<b>Implementar métodos remotos</b>
Como usuario de la infraestructura quiero que exista una aplicación servidor que implemente método de manera que puedan ser invocados por el cliente.
<b>Invocar métodos remotos</b>
Como usuario de la infraestructura quiero que exista una aplicación cliente que permita invocar los métodos remotos del servidor de manera que pueda comprobarse el concepto de RMI.

Tabla 27: Historias de usuario RMI 1. Elaboración propia.

### Anexo 2.5.7.2. Arquitectura

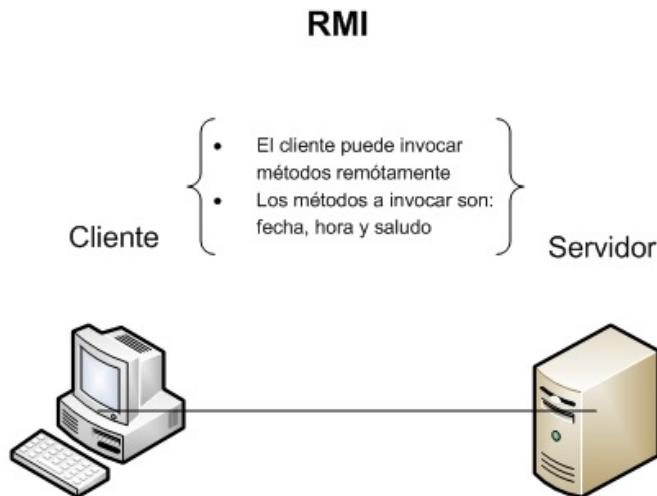


Ilustración 29: Arquitectura RMI 1. Elaboración propia.

### Anexo 2.5.7.3. Diseño (Tarjetas CRC) - Servidor

<b>MensajesRemotos</b>	
• Interfaz que posee los métodos que pueden ser invocados.	
<b>MensajesRemotosImpl</b>	
• Clase que implementa los métodos de la interfaz para ser invocados por el cliente.	

Tabla 28: Tarjetas CRC RMI 1 - Servidor. Elaboración propia.

#### Anexo 2.5.7.4. Diagrama de clases servidor

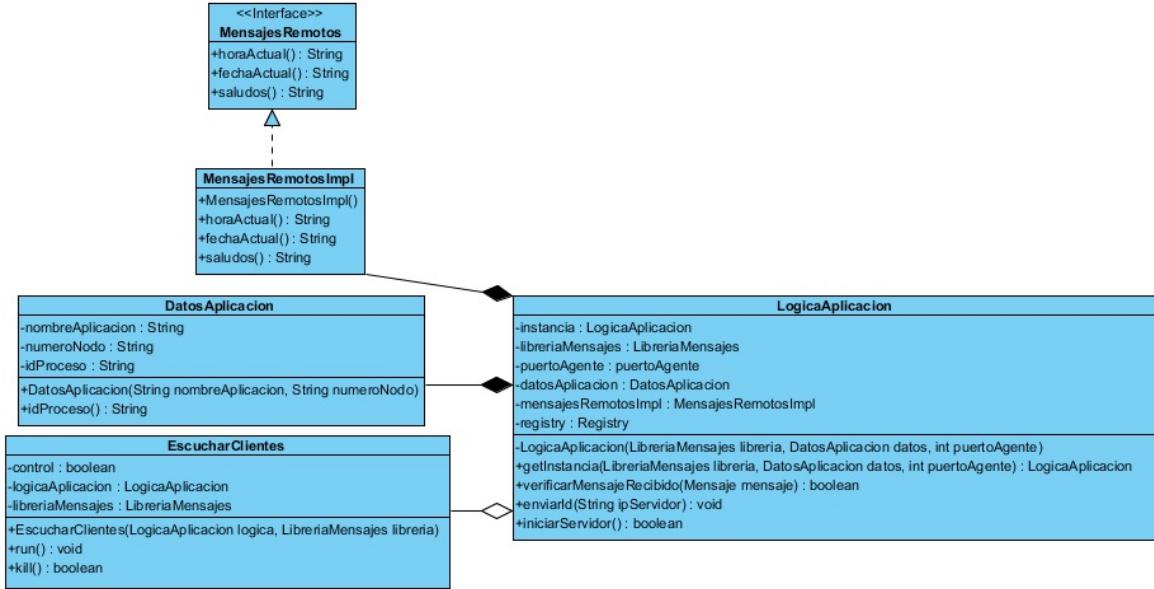


Ilustración 30: Diagrama de clases RMI 1 - Servidor. Elaboración propia.

#### Anexo 2.5.7.5. Tarjetas CRC - Cliente

MensajesRemotos	<ul style="list-style-type: none"> <li>La interfaz que posee los métodos que el cliente puede invocar remotamente</li> </ul>
-----------------	--

Tabla 29: Tarjetas CRC RMI 1 - Cliente. Elaboración propia.

#### Anexo 2.5.7.6. Diagrama de clases cliente

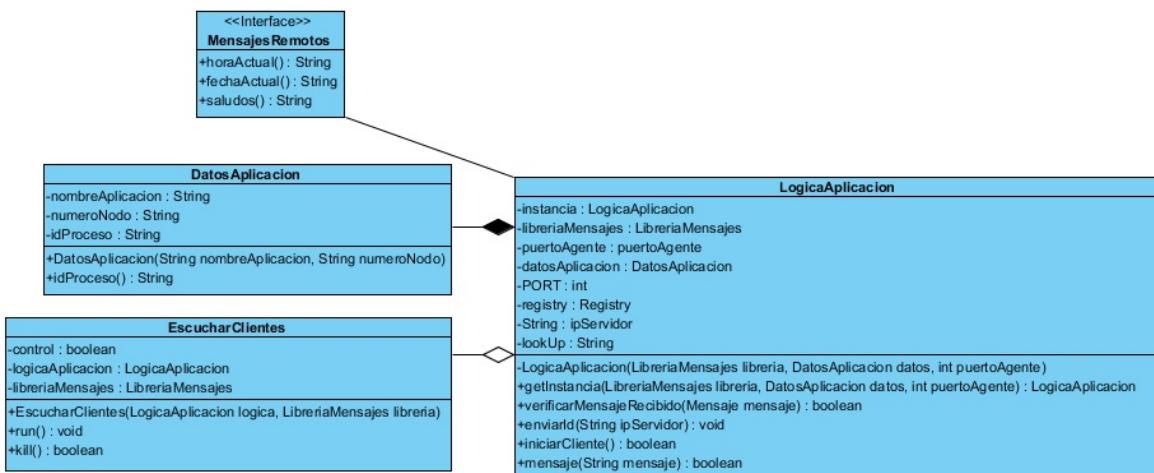


Ilustración 31: Diagrama de clases RMI 1 - Cliente. Elaboración propia.

#### Anexo 2.5.7.7. Parámetros propios

- Cliente: Ip servidor.

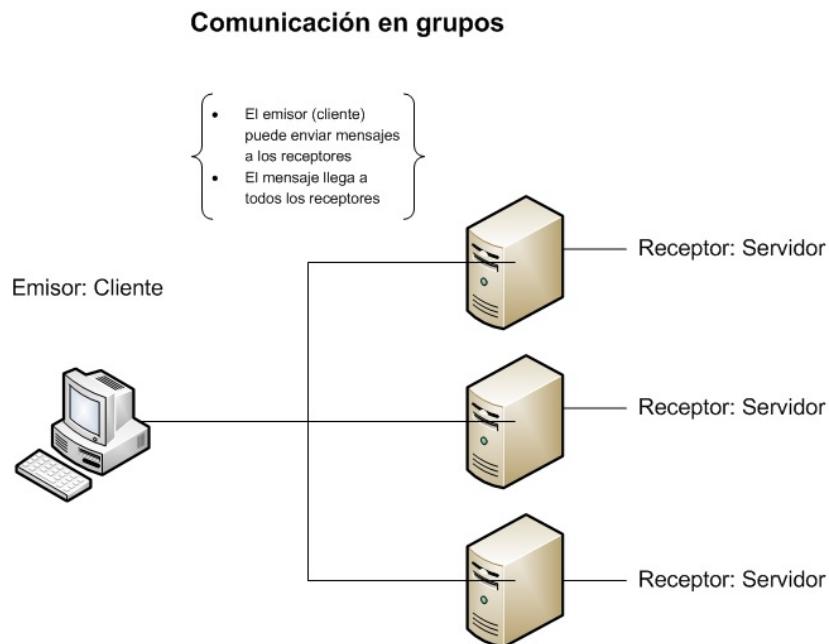
#### Anexo 2.5.8. Comunicación en grupo

##### Anexo 2.5.8.1. Historias de usuario

<b>Asociar nodo a grupo receptor</b>
Como usuario de la infraestructura quiero que exista una aplicación que permita asociar un nodo a un grupo receptor de manera que puedan recibir mensajes por parte del emisor.
<b>Crear grupo emisor</b>
Como usuario de la infraestructura quiero que exista una aplicación que funcione como grupo emisor de manera permita enviar mensajes al grupo receptor.

**Tabla 30: Historias de usuario Comunicación en grupo. Elaboración propia.**

##### Anexo 2.5.8.2. Arquitectura



**Ilustración 32: Arquitectura Comunicación en grupo. Elaboración propia.**

##### Anexo 2.5.8.3. Diagrama de clases servidor

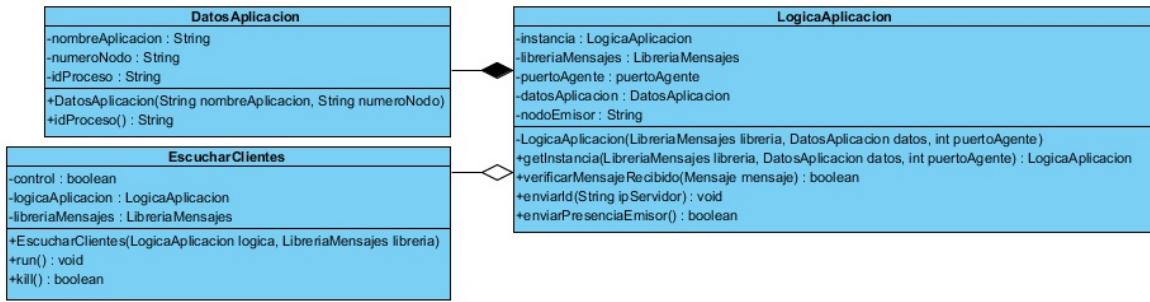


Ilustración 33: Diagrama de clases Comunicación en grupo - Servidor. Elaboración propia.

## Anexo 2.5.8.4. Diagrama de clases cliente

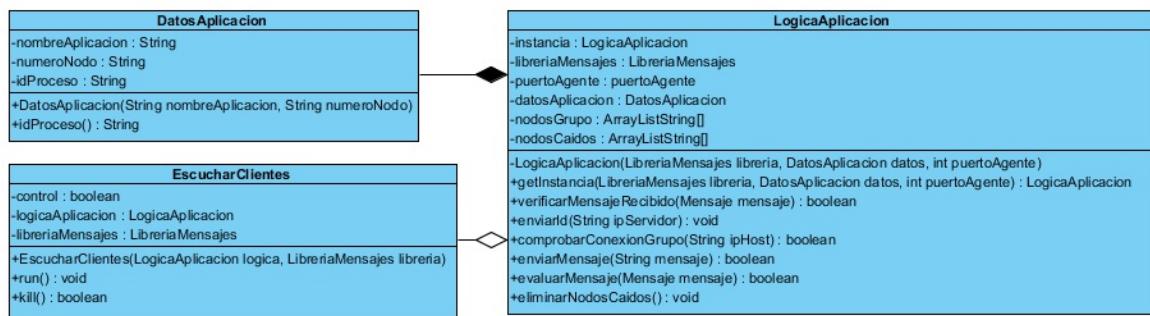


Ilustración 34: Diagrama de clases Comunicación en grupo - Cliente. Elaboración propia.

## Anexo 2.5.8.5. Parámetros propios

- Grupo Receptor : Ip servidor

## Anexo 2.5.9. Lamport

## Anexo 2.5.9.1. Historias de usuario

<b>Enviar marcas de reloj</b>
Como usuario de la infraestructura quiero que exista una aplicación que permita enviar marcas de reloj aleatorias a otros nodos de manera que pueda ser recibidas para su verificación.
<b>Verificar marcas de reloj</b>
Como usuario de la infraestructura quiero la aplicación permita verificar las marcas de reloj recibidas de manera que pueda ser comparada con la marca actual y arreglarla mediante el algoritmo de Lamport en caso de que sea menor a la actual.

Tabla 31: Historias de usuario Algoritmo de Lamport. Elaboración propia.

### Anexo 2.5.9.2. Arquitectura

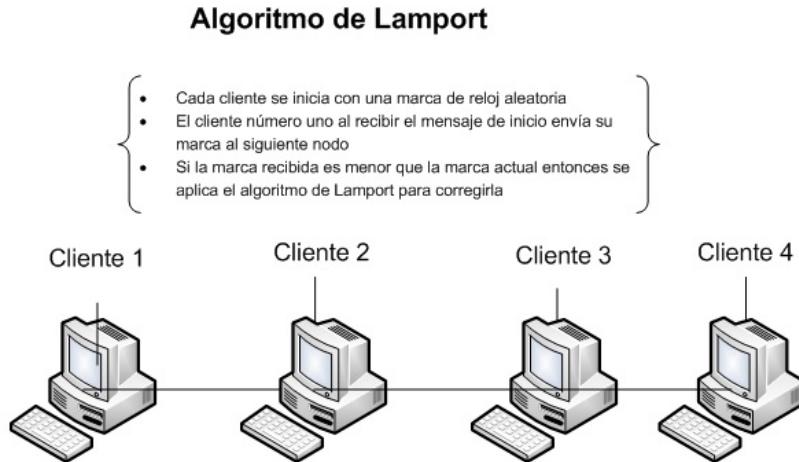


Ilustración 35: Arquitectura Algoritmo de Lamport. Elaboración propia.

### Anexo 2.5.9.3. Diseño (Tarjetas CRC) - Cliente

MarcaReloj	
<ul style="list-style-type: none"> <li>Permite iniciar una marca de reloj con un valor aleatorio para aumentarse cada segundo.</li> </ul>	<ul style="list-style-type: none"> <li>LogicaAplicacion</li> </ul>

Tabla 32: Tarjetas CRC Algoritmo de Lamport. Elaboración propia.

### Anexo 2.5.9.4. Diagrama de clases

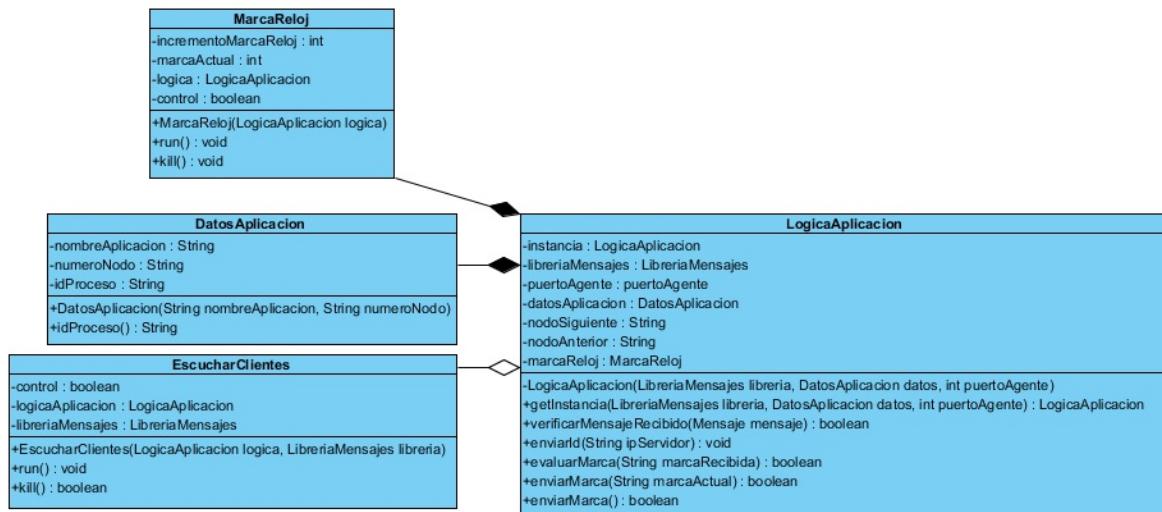


Ilustración 36: Diagrama de clases Algoritmo de Lamport. Elaboración propia.

#### Anexo 2.5.9.5. Parámetros únicos

- Cliente: Ip nodo siguiente, ip nodo anterior

#### Anexo 2.5.10. Algoritmo Cristian

##### Anexo 2.5.10.1. Historias de usuario

Enviar hora
Como usuario de la infraestructura quiero que exista una aplicación servidor que envíe su hora al recibir un mensaje por parte del cliente de manera que se pueda calcular el tiempo local.
Obtener hora de envío
Como usuario de la infraestructura quiero que exista una aplicación cliente que envíe mensajes al servidor cada cierto tiempo de manera que pueda guardar la hora de envío.
Recibir hora
Como usuario de la infraestructura quiero el cliente reciba el mensaje con la hora proveniente del servidor de manera que pueda sincronizar su reloj a través del algoritmo de Cristian.

Tabla 33: Historias de usuario Algoritmo de Cristian. Elaboración propia.

## Anexo 2.5.10.2. Arquitectura

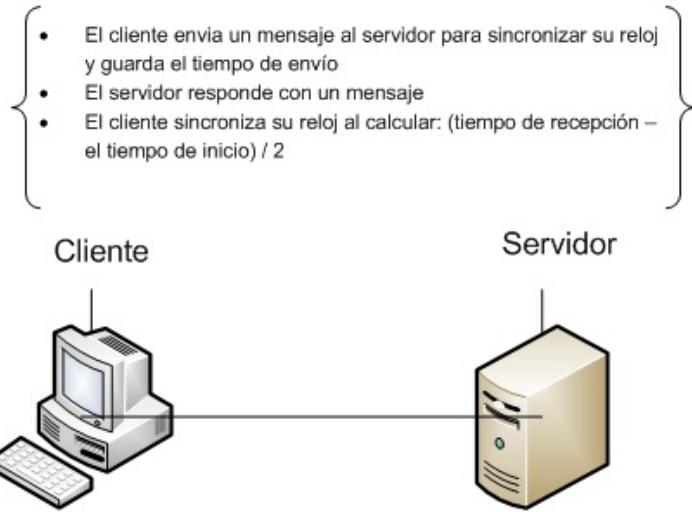
**Algoritmo de Cristian**

Ilustración 37: Arquitectura Algoritmo de Cristian. Elaboración propia.

## Anexo 2.5.10.3. Diagrama de clases servidor

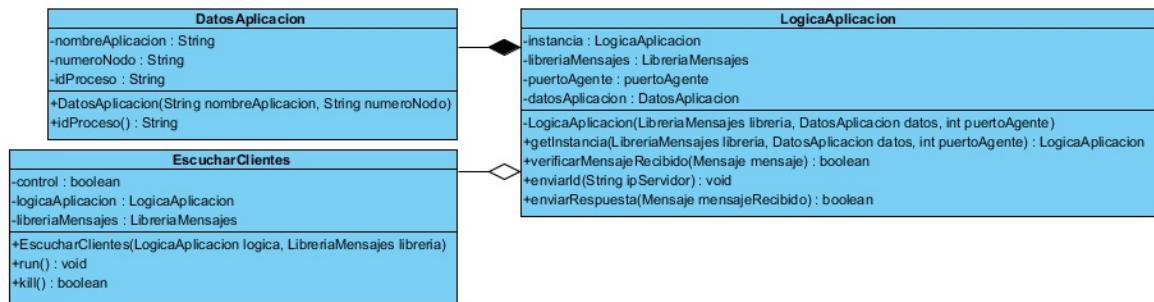


Ilustración 38: Diagrama de clases Algoritmo de Cristian - Servidor. Elaboración propia.

## Anexo 2.5.10.4. Diseño (Tarjetas CRC) - Cliente

Sincronización	
<ul style="list-style-type: none"> <li>Es un hilo que determina cuando se pueda enviar del tiempo actual al servidor.</li> </ul>	LogicaAplicacion

Tabla 34: Tarjetas CRC Algoritmo de Cristian - Cliente. Elaboración propia.

#### Anexo 2.5.10.5. Diagrama de clases cliente

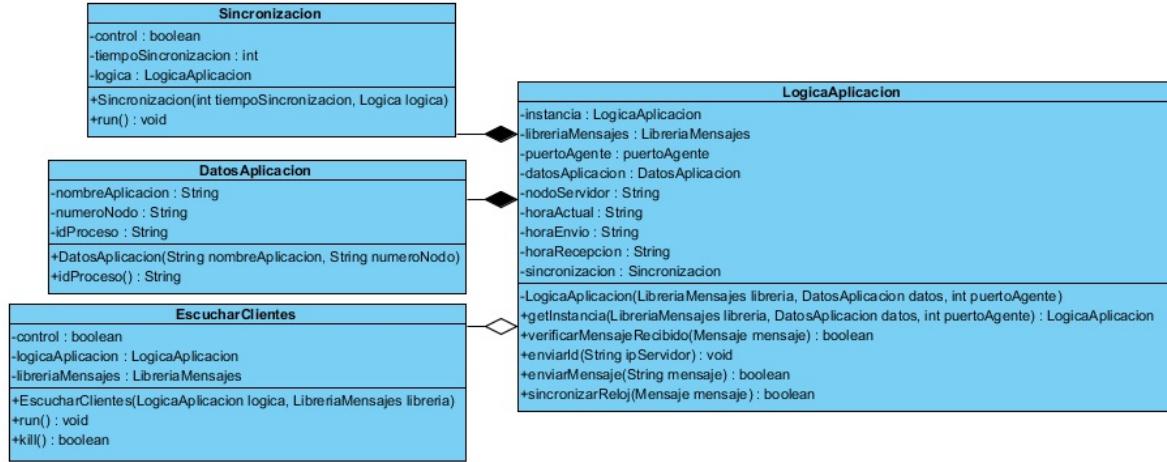


Ilustración 39: Diagrama de clases Algoritmo de Cristian - Cliente. Elaboración propia.

#### Anexo 2.5.10.6. Parámetros propios

- Cliente: Ip servidor, tiempo.

#### Anexo 2.5.11. Algoritmo Berkeley

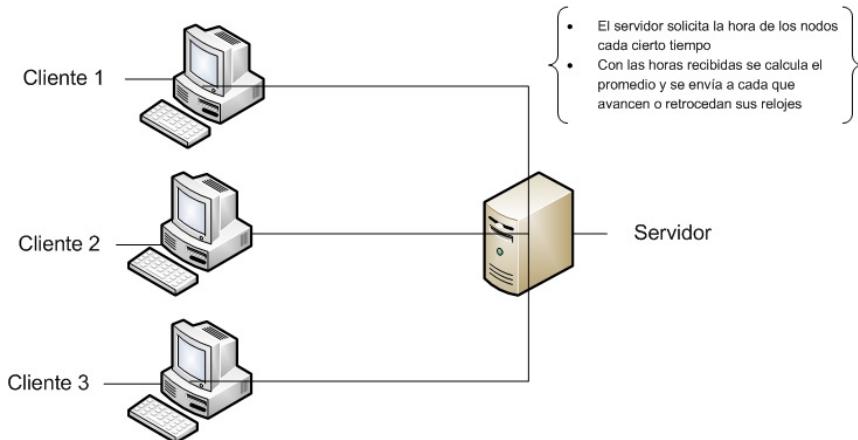
##### Anexo 2.5.11.1. Historias de usuario

Preguntar hora
Como usuario de la infraestructura quiero que exista una aplicación servidor que mida el tiempo de todos los clientes de manera que pueda calcular el tiempo promedio.
Calcular promedio
Como usuario de la infraestructura quiero que el servidor calcule el promedio del tiempo obtenido por los nodos de manera que pueda enviárselos de regreso y comprobar el algoritmo de Berkeley.
Recibir y enviar hora
Como usuario de la infraestructura quiero que exista una aplicación cliente que posea la hora actual de manera que pueda ser consultado por el servidor para calcular el tiempo promedio.

Tabla 35: Historias de usuario Algoritmo de Berkeley. Elaboración propia.

#### Anexo 2.5.11.2. Arquitectura

### Algoritmo de Berkeley



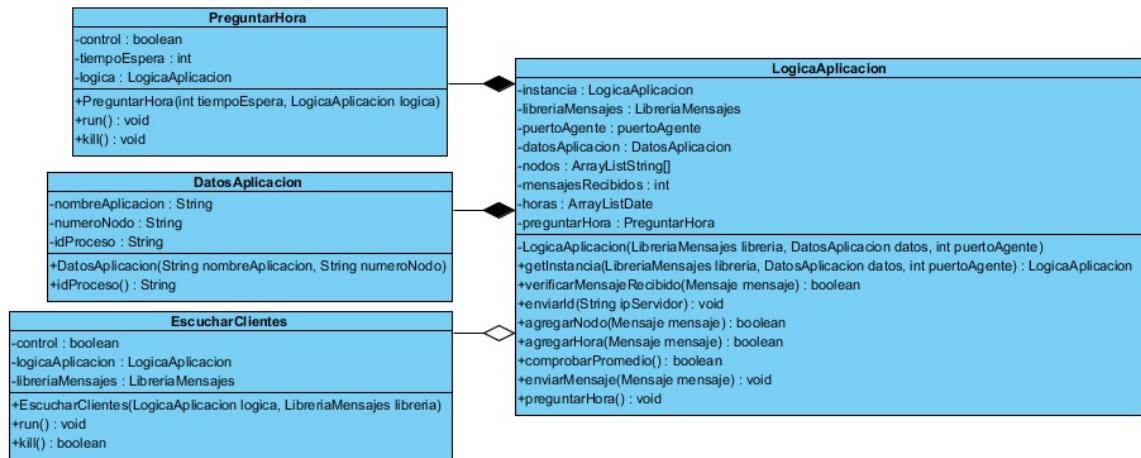
**Ilustración 40: Arquitectura Algoritmo de Berkeley. Elaboración propia.**

#### Anexo 2.5.11.3. Diseño (Tarjetas CRC) - Servidor

PreguntarHora	LogicaAplicacion
Es un hilo que realiza la petición de la hora a los nodos del sistema cada cierto tiempo	

**Tabla 36: Tarjetas CRC Algoritmo de Berkeley - Servidor. Elaboración propia.**

#### Anexo 2.5.11.4. Diagrama de clases



**Ilustración 41: Diagrama de clases Algoritmo de Berkeley - Servidor. Elaboración propia.**

#### Anexo 2.5.11.5. Diagrama de clases cliente

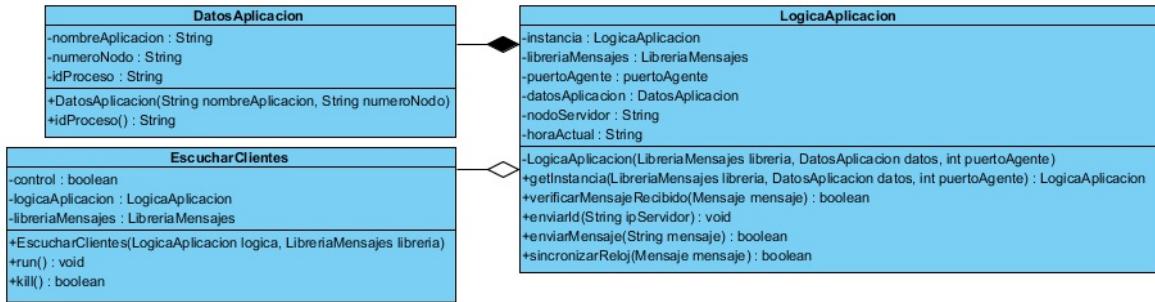


Ilustración 42: Diagrama de clases Algoritmo de Berkeley - Cliente. Elaboración propia.

#### Anexo 2.5.11.6. Parámetros propios

- Servidor: Tiempo.
- Cliente: Ip servidor.

#### Anexo 2.5.12. Algoritmo con Promedio

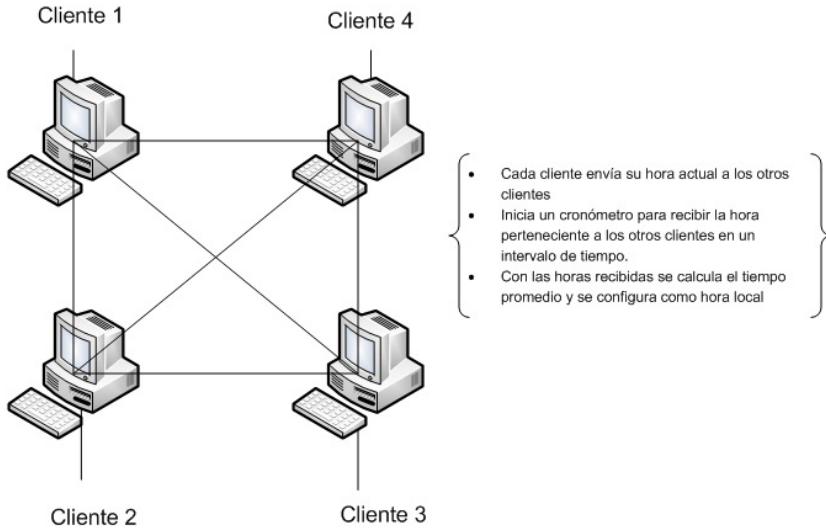
##### Anexo 2.5.12.1. Historias de usuario

Enviar hora
Como usuario de la infraestructura quiero que exista un aplicación que envía su hora cada cierto tiempo a los demás nodos de manera que se pueda recibir las horas de los demás nodos
Calcular promedio
Como usuario de la infraestructura quiero que la aplicación inicie un cronómetro local para recibir transmisiones del tiempo de manera que se pueda obtener el tiempo promedio como hora local

Tabla 37: Historias de usuario Algoritmo con promedio. Elaboración propia.

##### Anexo 2.5.12.2. Arquitectura

### Algoritmo con promedio (distribuido)



**Ilustración 43: Arquitectura Algoritmo con promedio. Elaboración propia.**

#### Anexo 2.5.12.3. Diseño (Tarjetas CRC) - Cliente

EscuchaHoras	LogicaAplicacion
<ul style="list-style-type: none"> <li>• Es un hilo que permite ejecutarse por cierto tiempo y luego se elimina</li> <li>• Se utiliza para contar el tiempo de espera de las horas provenientes de los demás nodos.</li> </ul>	

**Tabla 38: Tarjetas CRC Algoritmo con promedio. Elaboración propia.**

#### Anexo 2.5.12.4. Diagrama de clases

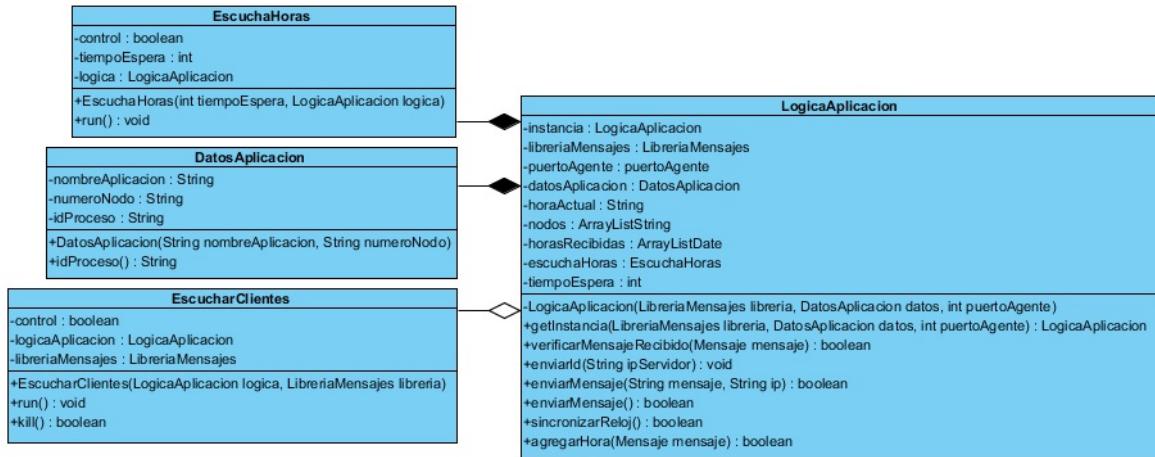


Ilustración 44: Diagrama de clases Algoritmo con promedio. Elaboración propia.

#### Anexo 2.5.12.5. Parámetros propios

- Cliente: Tiempo, ip nodo 1,i p nodo 2, ip nodo 3

#### Anexo 2.5.13. Algoritmo Centralizado

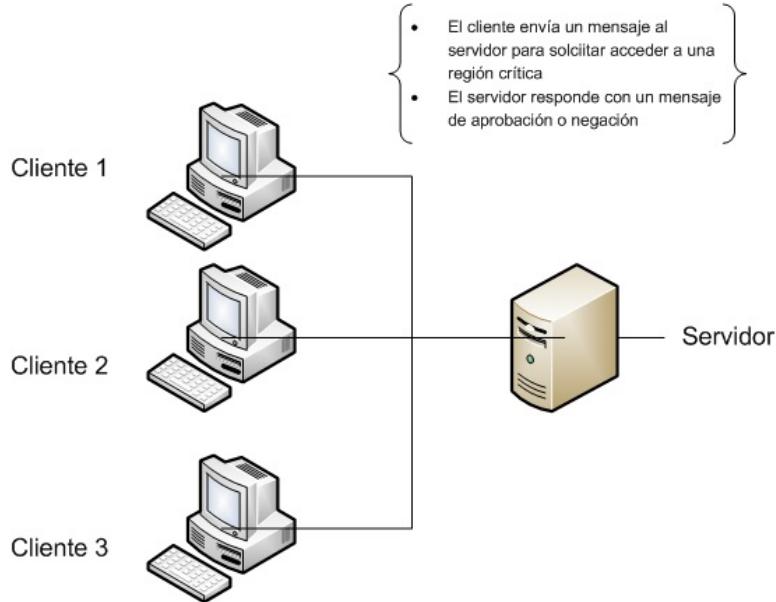
##### Anexo 2.5.13.1. Historias de usuario

<b>Recibir mensajes solicitud</b>
Como usuario de la infraestructura quiero que exista una aplicación servidor que reciba mensajes de otros nodos de manera que sepa cuando un nodo desea acceder a una región crítica.
<b>Verificar región</b>
Como usuario de la infraestructura quiero que el servidor chequee si un nodo tiene disponible su región crítica de manera que pueda aprobar la solicitud por parte de los nodos solicitantes.
<b>Solicitar región</b>
Como usuario de la infraestructura quiero que exista una aplicación cliente que permita enviar mensajes al coordinador cada vez que quiera entrar a la región crítica de otro proceso

Tabla 39: Historias de usuario Algoritmo centralizado. Elaboración propia.

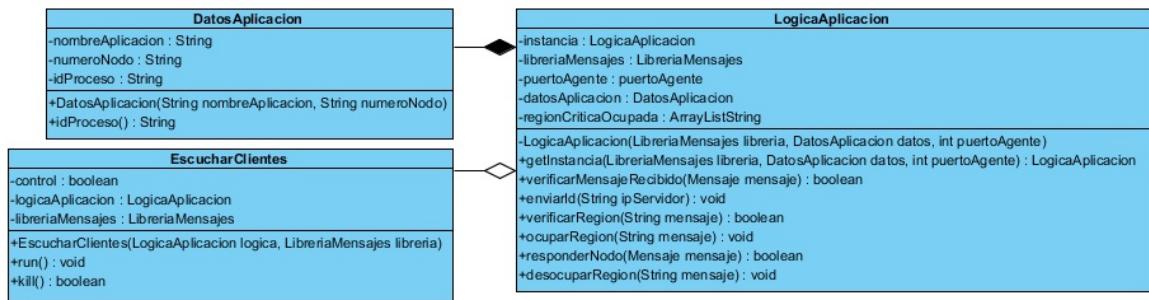
#### Anexo 2.5.13.2. Arquitectura

## Algoritmo Centralizado



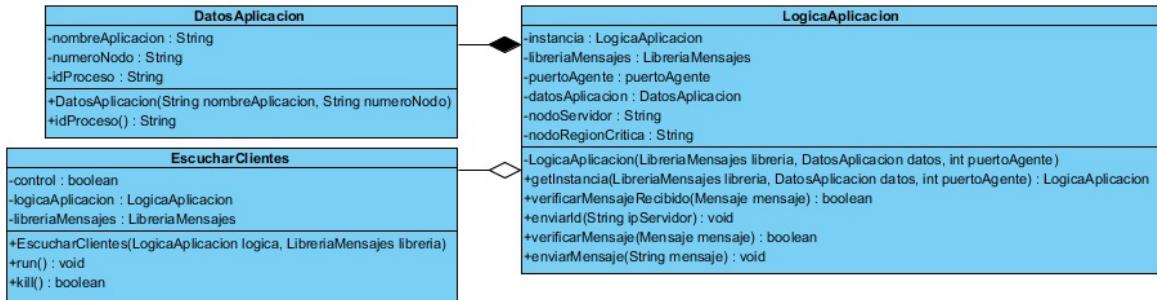
**Ilustración 45: Arquitectura Algoritmo centralizado. Elaboración propia.**

### Anexo 2.5.13.3. Diagrama de clases servidor



**Ilustración 46: Diagrama de clases Algoritmo centralizado - Servidor. Elaboración propia.**

#### Anexo 2.5.13.4. Diagrama de clases cliente



**Ilustración 47: Diagrama de clases Algoritmo centralizado. Elaboración propia.**

#### Anexo 2.5.13.5. Parámetros propios

- Cliente: Ip servidor.

#### Anexo 2.5.14. Algoritmo distribuido

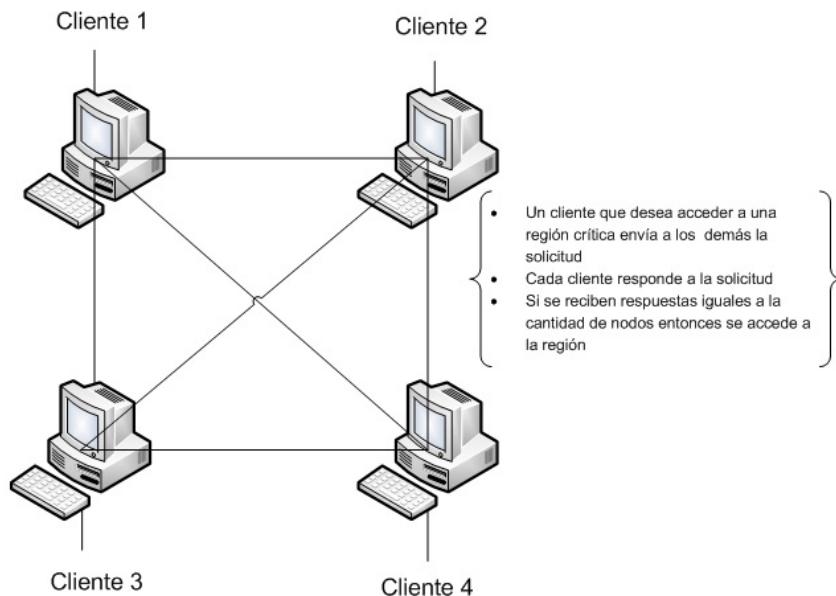
<b>Enviar mensaje solicitud</b>
Como usuario de la infraestructura quiero que exista una aplicación que envíe un mensaje con la hora actual, nombre de la región donde quiere acceder e número de nodo a todos los nodos (incluyéndolo) de manera que pueda saber si puede entrar a la región crítica
<b>Enviar mensaje de aceptación</b>
Como usuario de la aplicación quiero que la aplicación permita enviar un mensaje de aceptación cuando reciba la petición desde otro nodo y no requiera entrar en una región crítica
<b>Guardar peticiones</b>
Como usuario de la infraestructura quiero que la aplicación permita guardar los mensajes en una cola cuando esta se encuentre en una región crítica de manera de enviar mensaje de Ok cuando libere la región.
<b>Comparar marca de reloj</b>
Como usuario de la infraestructura quiero que la aplicación permita comparar la marca de reloj obtenida con la suya en caso de querer entrar en una región crítica de manera que si la marca recibida es menor entonces envía Ok, en caso contrario almacena el mensaje en una cola.
<b>Comparar mensajes recibidos</b>

Como usuario de la infraestructura quiero que la aplicación reciba los mensajes enviados por los demás nodos y compare la cantidad de Ok recibidos con la cantidad de nodos en el sistema de manera que si existen la misma cantidad de Ok y nodos se entra a la región crítica. Al terminar se envía mensaje al siguiente nodo en cola.

**Tabla 40: Historias de usuario Algoritmo distribuido. Elaboración propia.**

#### Anexo 2.5.14.1. Arquitectura

##### Algoritmo Distribuido



**Ilustración 48: Arquitectura Algoritmo distribuido. Elaboración propia.**

#### Anexo 2.5.14.2. Diseño (Tarjeta CRC)

EsperaMensaje	LogicaAplicacion
<ul style="list-style-type: none"> <li>Permite contar un tiempo específico para enviar el mensaje con la hora, numero de nodo y región crítica a acceder.</li> </ul>	

**Tabla 41: Tarjetas CRC Algoritmo distribuido. Elaboración propia.**

#### Anexo 2.5.14.3. Diagrama de clases

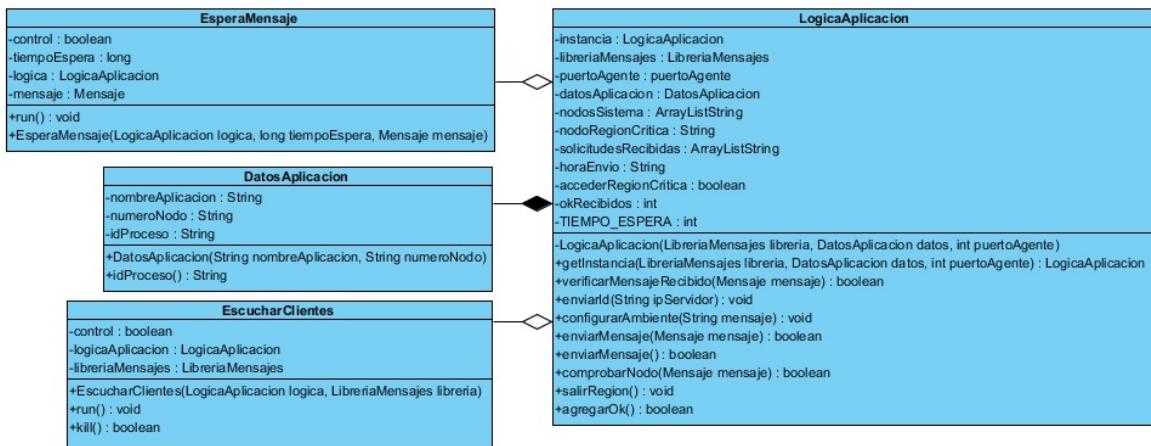


Ilustración 49: Diagrama de clases Algoritmo distribuido. Elaboración propia.

## Anexo 2.5.14.4. Parámetros propios

- Cliente: Ip nodo 1, ip nodo 2, ip nodo 3

## Anexo 2.5.15. Algoritmo Grandulón

<b>Enviar mensajes a los mayores</b>
Como usuario de la infraestructura quiero que exista una aplicación que envíe un mensaje de elección a los nodos mayores que el de manera de elegir el nodo coordinador.
<b>Responder mensaje</b>
Como usuario de la infraestructura quiero que la aplicación responda al mensaje de elección para indicar que ese nodo no puede ser coordinador.
<b>Enviar mensaje de coordinador</b>
Como usuario de la infraestructura quiero que la aplicación después de haber transcurrido un tiempo envíe un mensaje de coordinador a todos los nodos indicando que esta ganó la elección.

Tabla 42: Historias de usuario Algoritmo grandulón. Elaboración propia.

## Anexo 2.5.15.1. Arquitectura

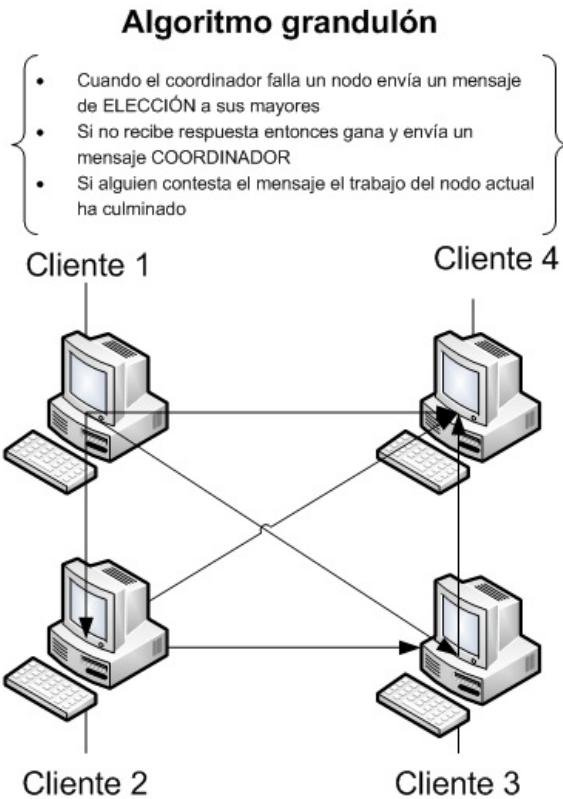


Ilustración 50: Arquitectura Algoritmo grandulón. Elaboración propia.

## Anexo 2.5.15.2. Diseño (Tarjetas CRC)

EsperaMensaje	LogicaAplicación
<ul style="list-style-type: none"> <li>• Permite contar un tiempo máximo para esperar por los mensajes de un nodo mayor</li> <li>• Si no se recibe mensaje, entonces se envía a todos los nodos que el nodo actual es coordinador</li> </ul>	

Tabla 43: Tarjetas CRC Algoritmo grandulón. Elaboración propia.

## Anexo 2.5.15.3. Diagrama de clases

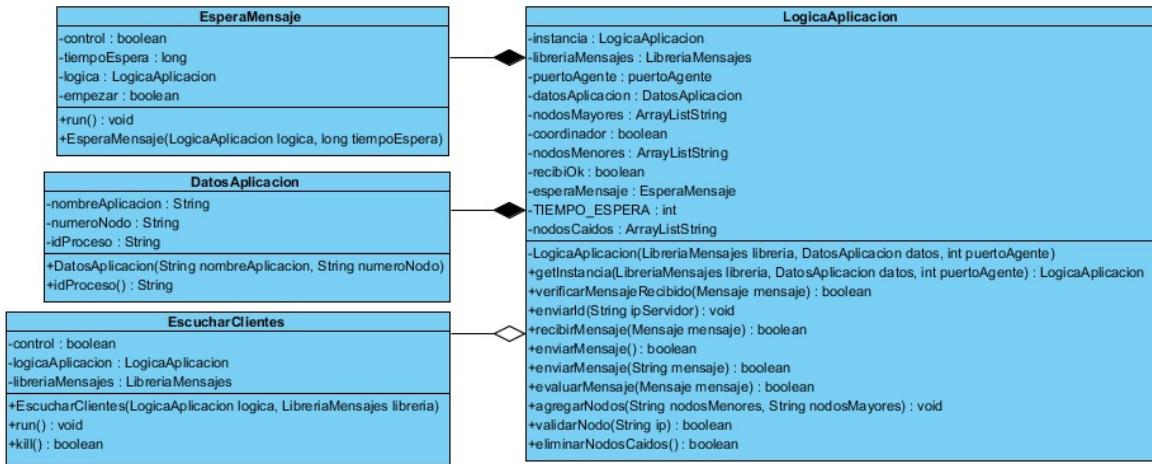


Ilustración 51: Diagrama de clases Algoritmo grandulón. Elaboración propia.

#### Anexo 2.5.15.4. Parámetros propios

- Cliente: Nodos menores, nodos mayores.

#### Anexo 2.5.16. Algoritmo de Anillo

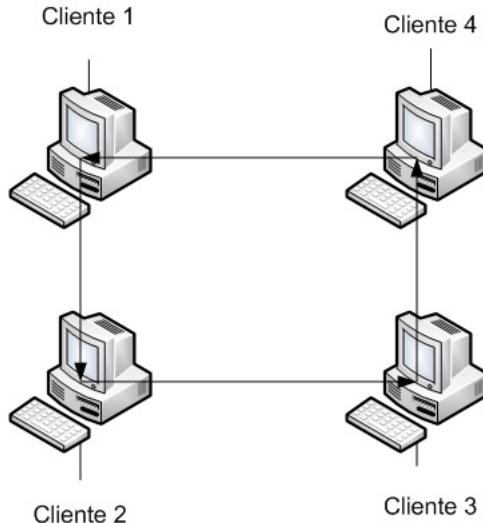
Enviar mensaje al siguiente nodo
Como usuario de la infraestructura quiero que exista una aplicación que envíe un mensaje de elección con el número de proceso y número de nodo de manera de elegir un nuevo coordinador.
Recibir mensaje de elección
Como usuario de la infraestructura quiero que la aplicación reciba el mensaje de elección de otro nodo y lo reenvíe al siguiente incluyendo su número de proceso y nodo de manera que al finalizar la ronda se pueda elegir el coordinador.
Elegir coordinador
Como usuario de la infraestructura quiero que la aplicación al finalizar la ronda elija el coordinador de acuerdo al nodo con el número de proceso más alto y lo notifique a través del anillo.

Tabla 44: Historias de usuario Algoritmo de anillo. Elaboración propia.

### Anexo 2.5.16.1. Arquitectura

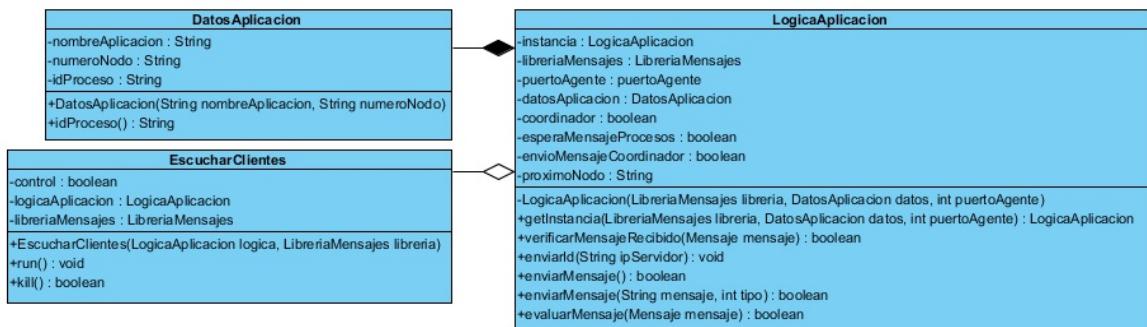
#### Algoritmo de anillo

- Un nodo envía mensaje de ELECCIÓN a través del anillo con su número de proceso
- Al concluir la ronda, el nodo que inició el mensaje elige el coordinador (mayor proceso) y lo notifica en el anillo



**Ilustración 52: Arquitectura Algoritmo de anillo. Elaboración propia.**

### Anexo 2.5.16.2. Diagrama de clases



**Ilustración 53: Diagrama de clases Algoritmo de anillo. Elaboración propia.**

### Anexo 2.5.16.3. Parámetros propios

- Cliente: Próximo nodo.

## Anexo 2.5.17. Tipos de Fallas

### Anexo 2.5.17.1. Historias de usuario

<b>Generar falla de congelación</b>
Como usuario de la infraestructura quiero que exista una aplicación servidor que posea una falla de congelación de manera que deje de funcionar deteniéndose luego de un tiempo funcionando normalmente.
<b>Generar falla de omisión</b>
Como usuario de la infraestructura quiero que el servidor posea una falla de omisión de manera que no responda a las peticiones del cliente.
<b>Generar falla de tiempo</b>
Como usuario de la infraestructura quiero el servidor posea una falla de tiempo de manera que la respuesta del servidor hacia el cliente sea entregada fuera del intervalo de tiempo especificado.
<b>Generar falla de respuesta</b>
Como usuario de la infraestructura quiero el servidor posea una falla de respuesta de manera que envíe una respuesta incorrecta al cliente luego de haber realizado una petición.
Crear cliente
Como usuario de la infraestructura quiero que exista una aplicación cliente que permita conectarse al servidor de manera que se puedan observar los diferentes tipos de fallas.

**Tabla 45: Historias de usuario Tipos de fallas. Elaboración propia.**

### Anexo 2.5.17.2. Arquitectura

### Tipos de fallas

- El servidor se inicia con una falla aleatoria (congelación, omisión, de tiempo y respuesta)
- El cliente envía mensajes al servidor para comprobar la falla generada.

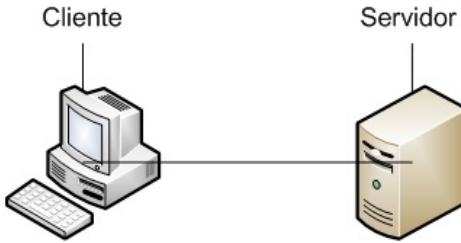


Ilustración 54: Arquitectura Tipos de fallas. Elaboración propia.

#### Anexo 2.5.17.3. Diagrama de clases - servidor

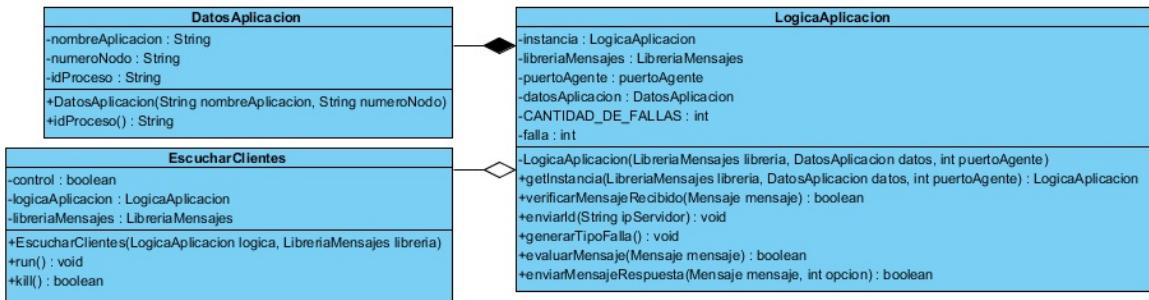


Ilustración 55: Diagrama de clases Tipos de fallas - Servidor. Elaboración propia.

#### Anexo 2.5.17.4. Diagrama de clases - cliente

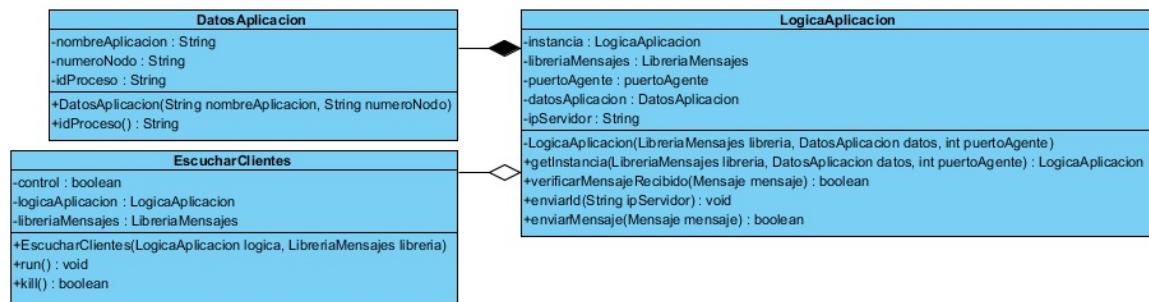


Ilustración 56: Diagramas de clases Tipos de Fallas - Cliente. Elaboración propia.

#### Anexo 2.5.17.5. Parámetros propios

- Cliente: Ip servidor.

## Anexo 2.5.18. Fallas Bizantinas

### Anexo 2.5.18.1. Historias de usuario

<b>Enviar número de nodo</b>
Como usuario de la infraestructura quiero que exista una aplicación que envíe su número de nodo a los demás nodos de manera que se puedan recibir los números de cada nodo
<b>Generar nodo corrupto</b>
Como usuario de la infraestructura quiero que exista la posibilidad de iniciar la aplicación en modo corrupto de manera que envíe un número de nodo distinto al real.
<b>Recibir número de nodo</b>
Como usuario de la infraestructura quiero que la aplicación permita recibir el número de nodo de los demás nodos de manera que se pueda almacenar en un vector local.
<b>Enviar vector</b>
Como usuario de la infraestructura quiero que la aplicación envíe el vector con los números de nodos recibidos incluyendo el suyo a todos los nodos.
<b>Recibir vector</b>
Como usuario de la infraestructura quiero que la aplicación reciba el vector perteneciente a todos los nodos de manera que se pueda comparar cada casilla y colocar el elemento que más se repite, en caso contrario se coloca UNKNOWN.

**Tabla 46: Historias de usuario Fallas Bizantinas. Elaboración propia.**

### Anexo 2.5.18.2. Arquitectura

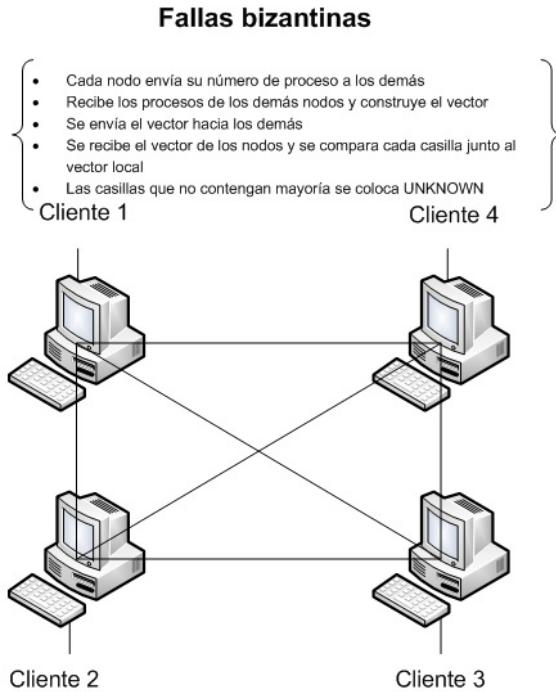


Ilustración 57: Arquitectura Fallas Bizantinas. Elaboración propia.

#### Anexo 2.5.18.3. Diagrama de clases

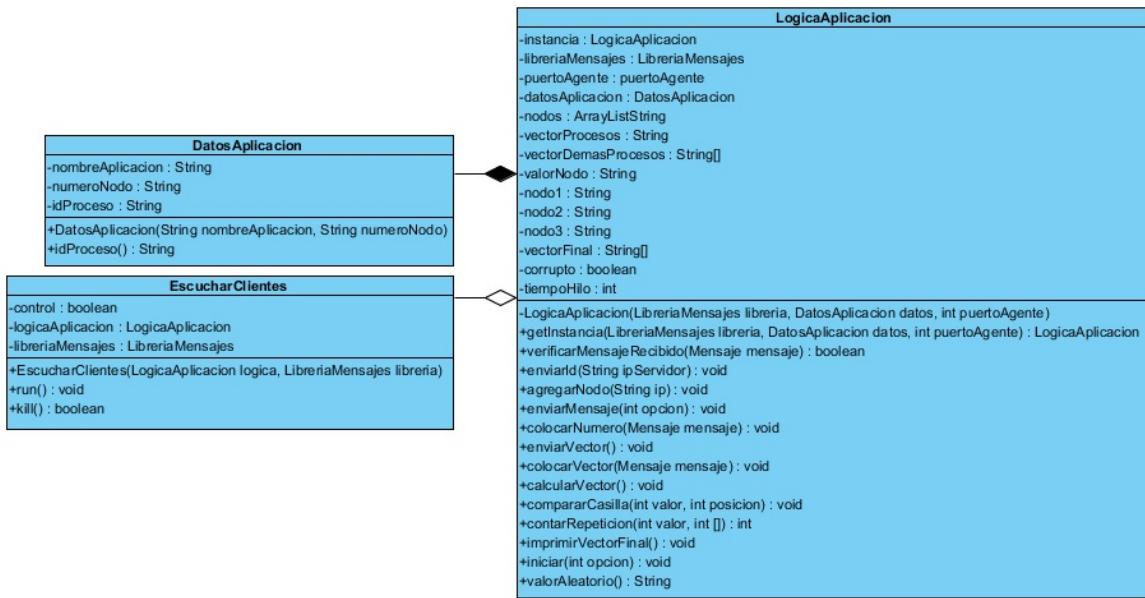


Ilustración 58: Diagrama de clases Fallas Bizantinas. Elaboración propia.

#### Anexo 2.5.18.4. Parámetros propios

- Cliente: Ip nodo 1, ip nodo 2, ip nodo 3, tiempo.

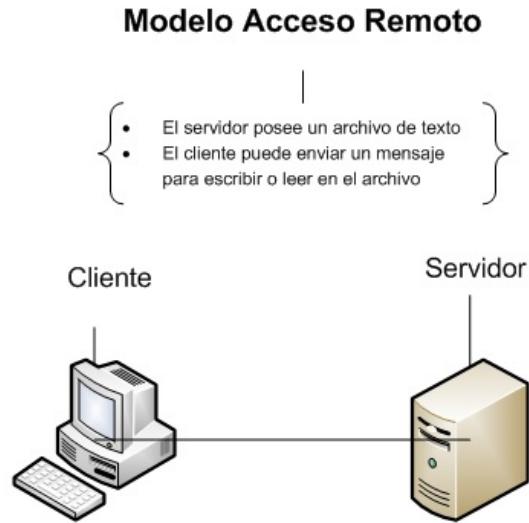
#### Anexo 2.5.19. Modelo Acceso Remoto

##### Anexo 2.5.19.1. Historias de usuario

Almacenar archivo
Como usuario de la infraestructura quiero que exista una aplicación servidor que posea un conjunto de archivos de manera que sea accedido por el cliente para lectura o escritura.
Acceder archivo
Como usuario de la infraestructura quiero que exista una aplicación cliente que pueda acceder al archivo en el servidor de manera que pueda leer o escribir en el.

**Tabla 47: Historias de usuario Modelo acceso remoto. Elaboración propia.**

##### Anexo 2.5.19.2. Arquitectura



**Ilustración 59: Arquitectura Modelo acceso remoto. Elaboración propia.**

##### Anexo 2.5.19.3. Diagrama de clases - servidor

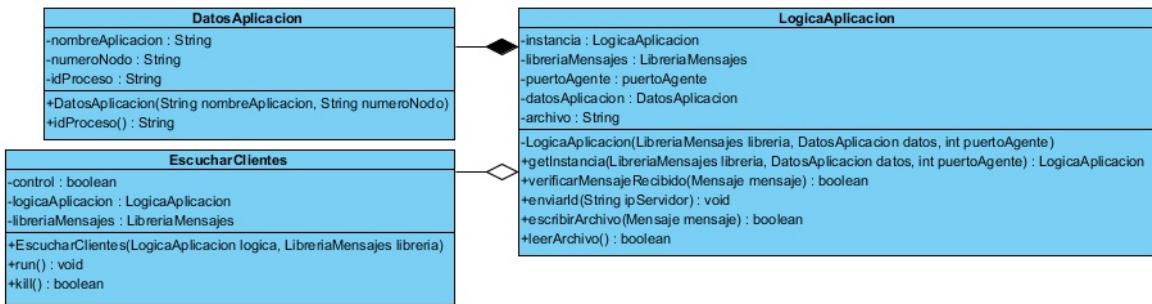


Ilustración 60: Diagrama de clases Modelo acceso remoto - Servidor. Elaboración propia.

## Anexo 2.5.19.4. Diagrama de clases - cliente

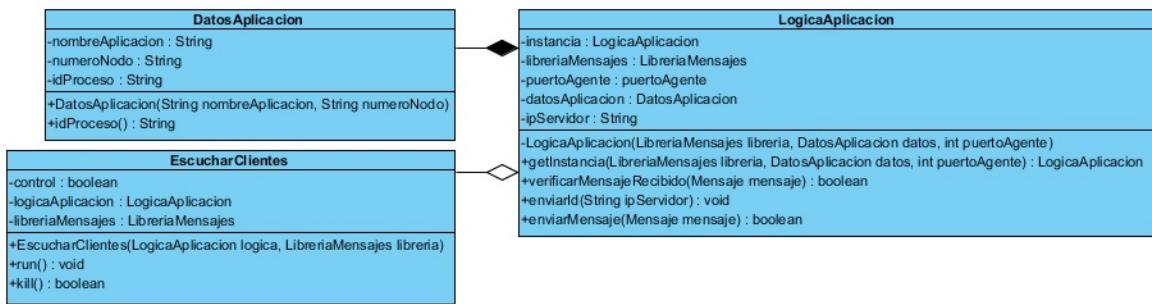


Ilustración 61: Diagrama de clases Modelo acceso remoto - Cliente. Elaboración propia.

## Anexo 2.5.19.5. Parámetros propios

- Cliente: Ip servidor.

## Anexo 2.5.20. Modelo Carga y Descarga

## Anexo 2.5.20.1. Historias de usuario

<b>Almacenar archivo descargable</b>
Como usuario de la infraestructura quiero que exista una aplicación servidor que contenga archivo de manera que pueda ser descargado por los clientes.
<b>Acceder archivo local</b>
Como usuario de la infraestructura quiero que exista una aplicación cliente que acceda al servidor y descargue un archivo específico de manera que pueda escribir y leer para posteriormente cargarlo nuevamente en el servidor.

Tabla 48: Historias de usuario Modelo carga y descarga. Elaboración propia.

## Anexo 2.5.20.2. Arquitectura

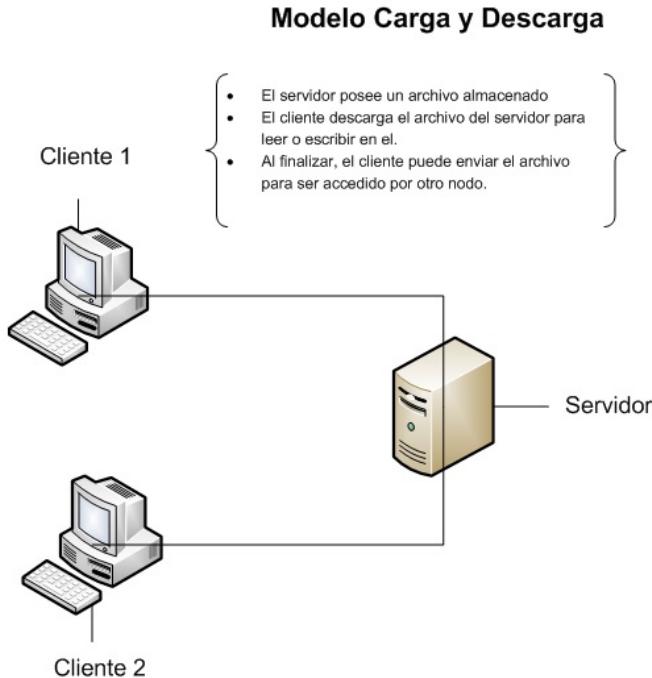


Ilustración 62: Arquitectura Modelo carga y descarga. Elaboración propia.

## Anexo 2.5.20.3. Diseño (Tarjetas CRC) - Servidor

<b>EsperaArchivo</b>	<ul style="list-style-type: none"> <li>Permite esperar por la petición del archivo por parte del usuario</li> <li>Envía el archivo solicitado por el usuario</li> </ul>	LogicaAplicacion MensajeDameFichero MensajeTomaFichero
<hr/>		
<b>MensajeDameFichero</b>	<ul style="list-style-type: none"> <li>Contiene el nombre del archivo a buscar</li> </ul>	
<hr/>		
<b>MensajeTomaFichero</b>	<ul style="list-style-type: none"> <li>Posee el contenido del archivo que se transmitirá por sockets.</li> </ul>	

Tabla 49: Tarjetas CRC Modelo carga y descarga - Servidor. Elaboración propia.

## Anexo 2.5.20.4. Diagrama de clases servidor

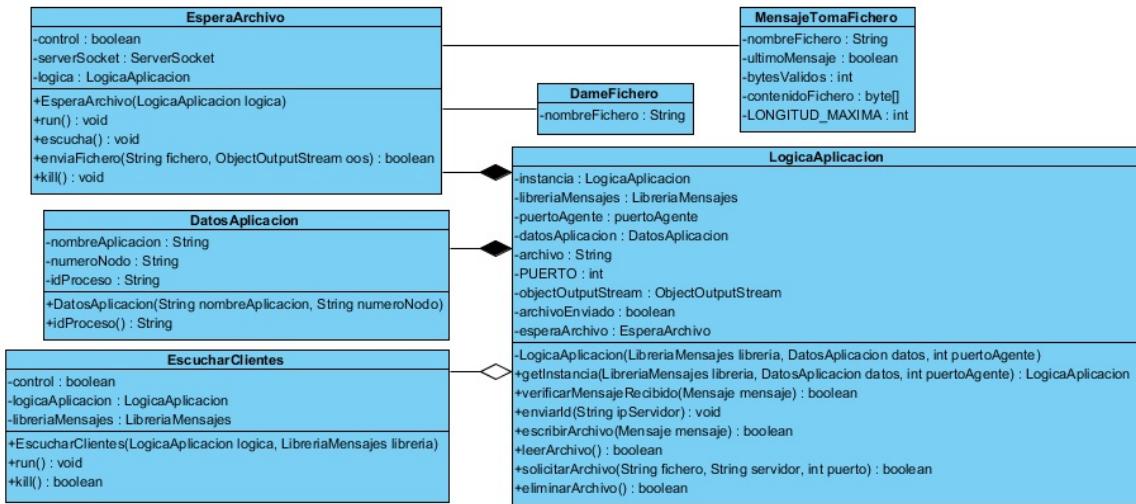


Ilustración 63: Diagrama de clases Modelo carga y descarga - Servidor. Elaboración propia.

## Anexo 2.5.20.5. Diseño (Tarjetas CRC) - Cliente

<b>EsperaArchivo</b>	<ul style="list-style-type: none"> <li>Permite esperar por la petición del archivo por parte del servidor</li> <li>Envía el archivo al servidor</li> </ul>	<b>LogicaAplicacion</b> <b>MensajeDameFichero</b> <b>MensajeTomaFichero</b>
<b>MensajeDameFichero</b>		
<ul style="list-style-type: none"> <li>Contiene el nombre del archivo a buscar</li> </ul>		
<b>MensajeTomaFichero</b>		
<ul style="list-style-type: none"> <li>Posee el contenido del archivo que se transmitirá por sockets.</li> </ul>		

Tabla 50: Tarjetas CRC Modelo carga y descarga - Cliente. Elaboración propia.

## Anexo 2.5.20.6. Diagrama de clases

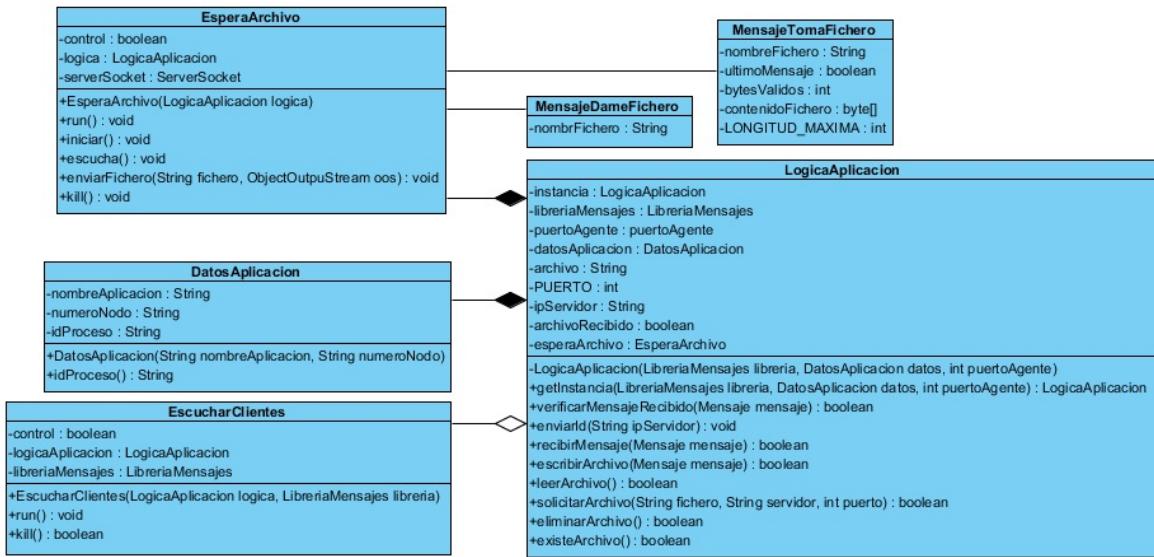


Ilustración 64: Diagrama de clases Modelo carga y descarga - Cliente. Elaboración propia.

## Anexo 2.5.20.7. Parámetros propios:

- Cliente: Ip servidor.

## Anexo 2.5.21. Sistemas de archivos basados en Clúster

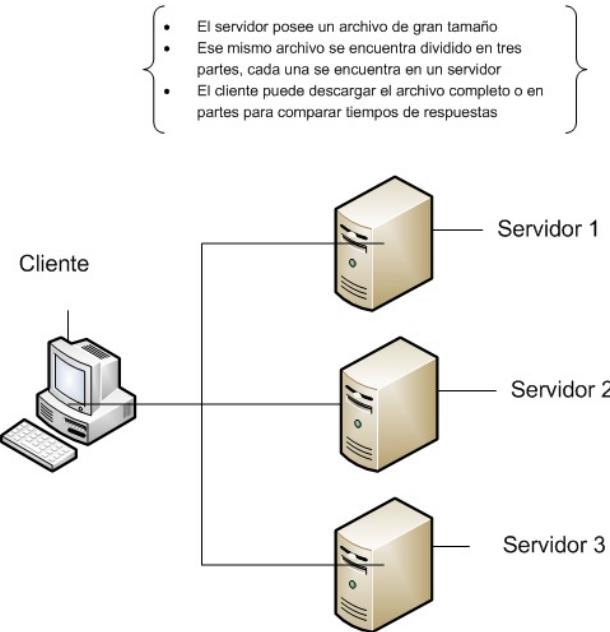
## Anexo 2.5.21.1. Historias de usuario

Almacenar archivo en partes
Como usuario de la infraestructura quiero que exista una aplicación servidor que almacene un archivo muy grande dividido en partes de manera que pueda ser enviado al cliente.
Almacenar archivo completo
Como usuario de la infraestructura quiero que la aplicación servidor almacene un archivo muy grande de manera que pueda ser enviado al cliente
Descargar archivos
Como usuario de la infraestructura quiero que exista una aplicación cliente que permita acceder a los archivos de los servidores de manera que se pueda comparar tiempo de respuesta entre el archivo completo y el archivo por partes.

Tabla 51: Historias de usuario Sistemas de archivos basados en clúster. Elaboración propia.

### Anexo 2.5.21.2. Arquitectura

#### Sistemas de archivos basados en Cluster



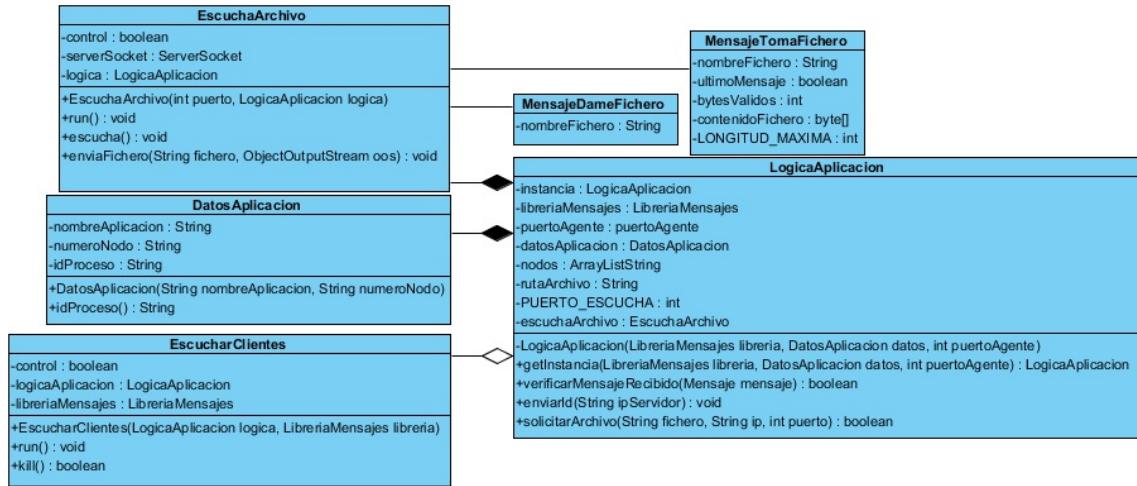
**Ilustración 65: Arquitectura Sistemas de archivos basados en clúster. Elaboración propia.**

### Anexo 2.5.21.3. Diseño (Tarjetas CRC) - Servidor

<b>EscuchaArchivo</b>	<ul style="list-style-type: none"> <li>Permite esperar por la petición del archivo por parte del servidor</li> <li>Envía el archivo al cliente</li> </ul>	<b>LogicaAplicacion</b> <b>MensajeDameFichero</b> <b>MensajeTomaFichero</b>
<b>MensajeDameFichero</b>	<ul style="list-style-type: none"> <li>Contiene el nombre del archivo solicitado</li> </ul>	
<b>MensajeTomaFichero</b>	<ul style="list-style-type: none"> <li>Posee el contenido del archivo que se transmitirá por sockets.</li> </ul>	

**Tabla 52: Tarjetas CRC Sistemas de archivos basados en clúster - Servidor. Elaboración propia.**

#### Anexo 2.5.21.4. Diagrama de clases servidor



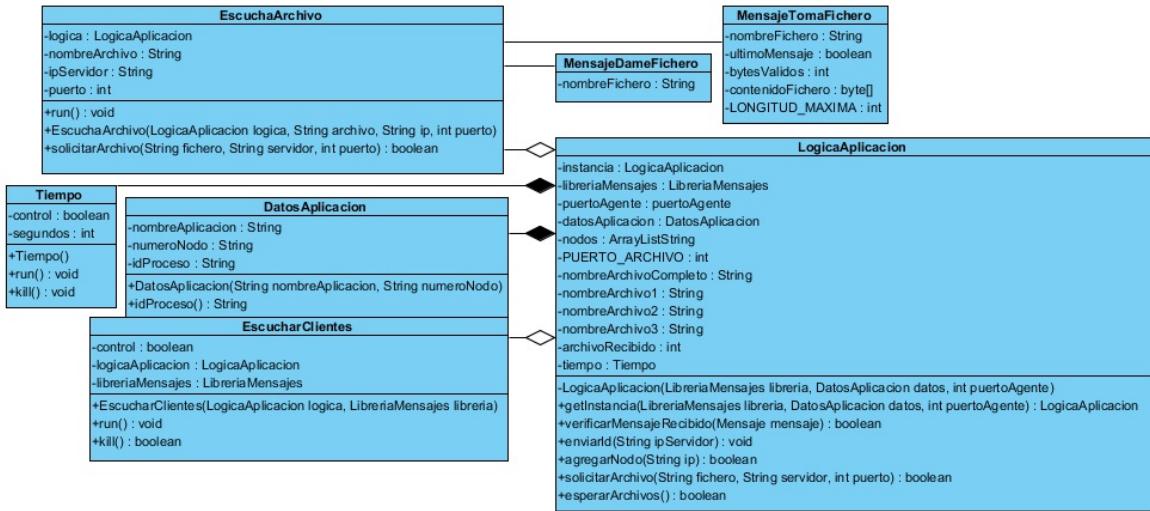
**Ilustración 66: Diagrama de clases Sistemas de archivos basados en cluster - Servidor.**  
Elaboración propia.

#### Anexo 2.5.21.5. Diseño (Tarjetas CRC) - Cliente

<b>EsperaArchivo</b>	
<ul style="list-style-type: none"> <li>Permite esperar por los archivos enviados por los nodos servidores</li> <li>Envía el archivo al servidor</li> </ul>	LogicaAplicacion MensajeDameFichero MensajeTomaFichero
<b>MensajeDameFichero</b>	
<ul style="list-style-type: none"> <li>Contiene el nombre del archivo a buscar</li> </ul>	
<b>MensajeTomaFichero</b>	
<ul style="list-style-type: none"> <li>Posee el contenido del archivo que se transmitirá por sockets.</li> </ul>	

**Tabla 53: Tarjetas CRC Sistema de archivos basados en cluster - Cliente.** Elaboración propia.

#### Anexo 2.5.21.6. Diagrama de clases cliente



**Ilustración 67: Diagrama de clases Sistemas de archivos basados en cluster - Cliente.**  
Elaboración propia.

#### Anexo 2.5.21.7. Parámetros propios

- Cliente: Ip servidor 1, ip servidor 2, ip servidor 3.

#### Anexo 2.5.22. DNS

##### Anexo 2.5.22.1. Historias de usuario

<b>Almacenar dominios y direcciones ip</b>
Como usuario de la infraestructura quiero que exista una aplicación servidor que contenga un conjunto de dominios asociados a direcciones ip de manera que puedan ser consultados por el cliente.
<b>Consultar dominios</b>
Como usuario de la infraestructura quiero que exista una aplicación cliente que permita enviar un dominio hacia el servidor de manera que pueda recibir el ip asociado al dominio enviado.
<b>Agregar dominios</b>
Como usuario de la infraestructura quiero que la aplicación cliente permita agregar nuevos dominios al servidor junto a su dirección ip.

**Tabla 54: Historias de usuario DNS. Elaboración propia.**

## Anexo 2.5.22.2. Arquitectura

**Domain Name System (DNS)**

- El servidor contiene un conjunto de dominios almacenados junto a sus direcciones ip
- El cliente puede agregar nuevos dominios o consultar por la dirección ip asociada a uno.

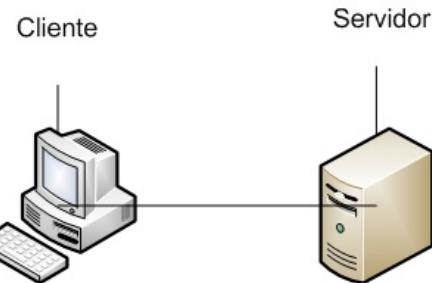


Ilustración 68: Arquitectura DNS. Elaboración propia.

## Anexo 2.5.22.3. Diagrama de clases servidor

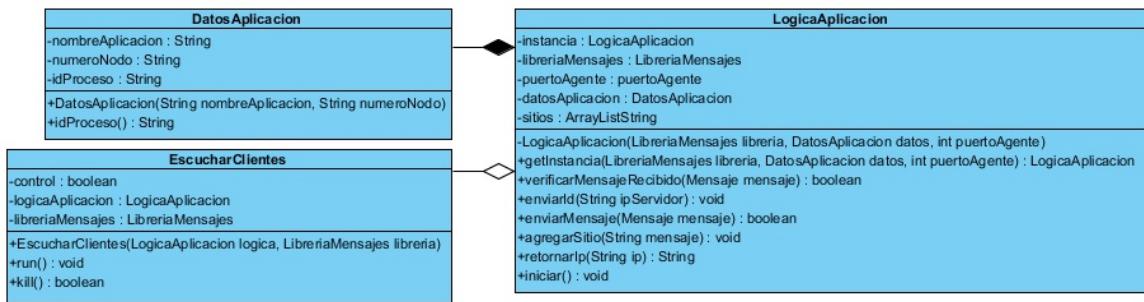


Ilustración 69: Diagrama de clases DNS - Servidor. Elaboración propia.

## Anexo 2.5.22.4. Diagrama de clases cliente

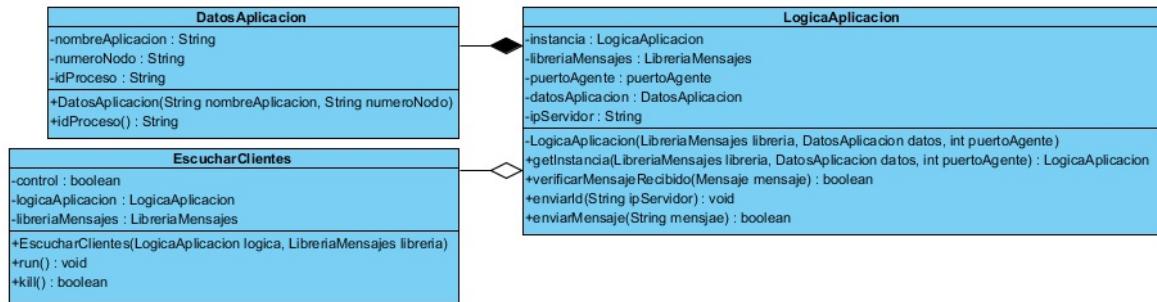


Ilustración 70: Diagrama de clases DNS - Cliente. Elaboración propia.

#### Anexo 2.5.22.5. Parámetros propios

- Cliente: Ip servidor.

### Anexo 2.5.23. LDAP

#### Anexo 2.5.23.1. Historias de usuario

<b>Almacenar dominios con usuarios</b>
Como usuario de la infraestructura quiero que exista una aplicación servidor con un conjunto de usuarios guardados en un dominio de manera que puedan ser consultados por los clientes.
<b>Consultar usuario</b>
Como usuario de la infraestructura quiero que exista una aplicación cliente que permita enviar el nombre de un usuario de manera que pueda recibir toda la información asociada a él.
<b>Agregar usuario</b>
Como usuario de la infraestructura quiero que el cliente pueda agregar un usuario de manera que se pueda adjuntar a un dominio específico.

Tabla 55: Historias de usuario LDAP. Elaboración propia.

#### Anexo 2.5.23.2. Arquitectura

### Lightweight Directory Access Protocol (LDAP)

- El servidor contiene un conjunto de usuarios asociados a dominios
- El cliente puede agregar nuevos usuarios o consultar por un usuario para visualizar sus datos.

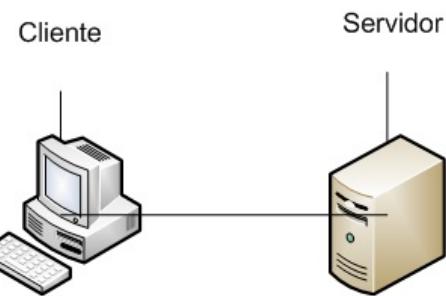


Ilustración 71: Arquitectura LDAP. Elaboración propia.

### Anexo 2.5.23.3. Diseño (Tarjetas CRC) - Servidor

<b>Cliente</b>	
<ul style="list-style-type: none"> <li>Clase que permite almacenar los datos personales pertenecientes a un usuario</li> </ul>	
<b>Dominio</b>	
<ul style="list-style-type: none"> <li>Clase que contiene un conjunto de clientes asociados</li> </ul>	Cliente

Tabla 56: Tarjetas CRC LDAP - Servidor. Elaboración propia.

### Anexo 2.5.23.4. Diagrama de clases servidor

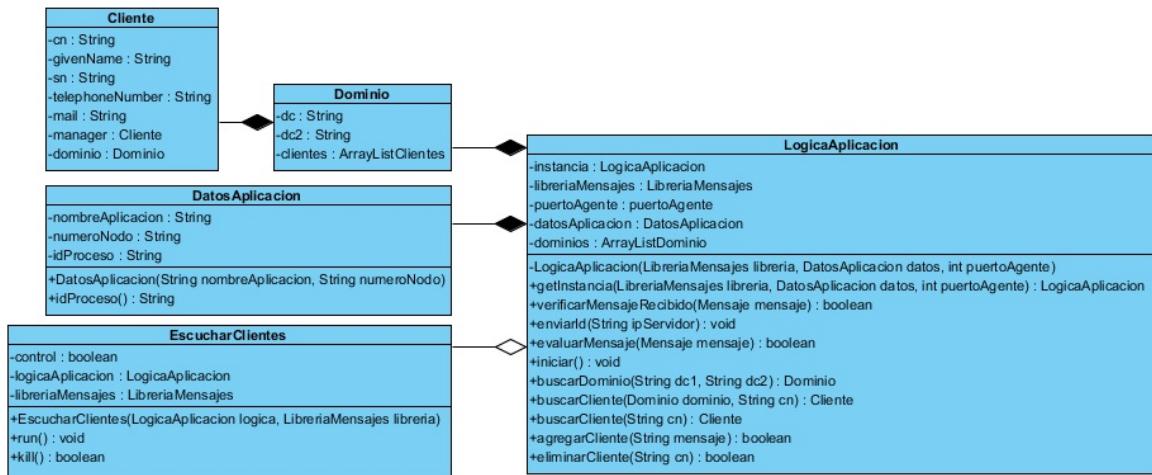


Ilustración 72: Diagrama de clases LDAP - Servidor. Elaboración propia.

### Anexo 2.5.23.5. Diagrama de clases cliente

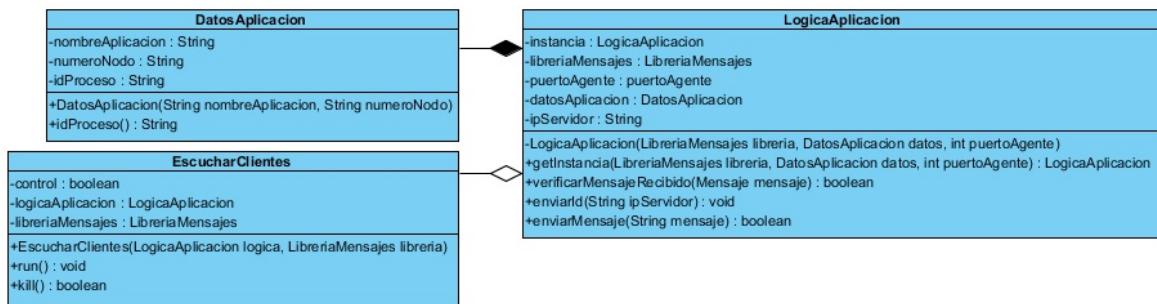


Ilustración 73: Diagrama de clases LDAP - Cliente. Elaboración propia.

### Anexo 2.5.23.6. Parámetros propios

- Cliente: Ip servidor.

## Anexo 2.5.24. RMI

### Anexo 2.5.24.1. Historias de usuario

<b>Implementar métodos remotos</b>
Como usuario de la infraestructura quiero exista una aplicación servidor que posea métodos remotos de manera que puedan ser invocados por el cliente.
<b>Invoker métodos remotos</b>
Como usuario de la infraestructura quiero exista una aplicación cliente que permita invocar los métodos del servidor de manera que se pueda comprobar el concepto de RMI.

Tabla 57: Historias de usuario RMI 2. Elaboración propia.

### Anexo 2.5.24.2. Arquitectura

#### Remote Method Invocation (RMI)



Ilustración 74: Arquitectura RMI 2. Elaboración propia.

### Anexo 2.5.24.3. Diseño (Tarjetas CRC) - Servidor

<b>MensajesRemotos</b>	
• Interfaz que contiene los métodos: listar mensajes, agregar mensaje y eliminar mensaje	
<b>MensajesRemotosImpl</b>	
• Clase que implementa los métodos de la interfaz	MensajesRemotos

Tabla 58: Tarjetas CRC RMI 2 - Servidor. Elaboración propia.

#### Anexo 2.5.24.4. Diagrama de clases servidor.

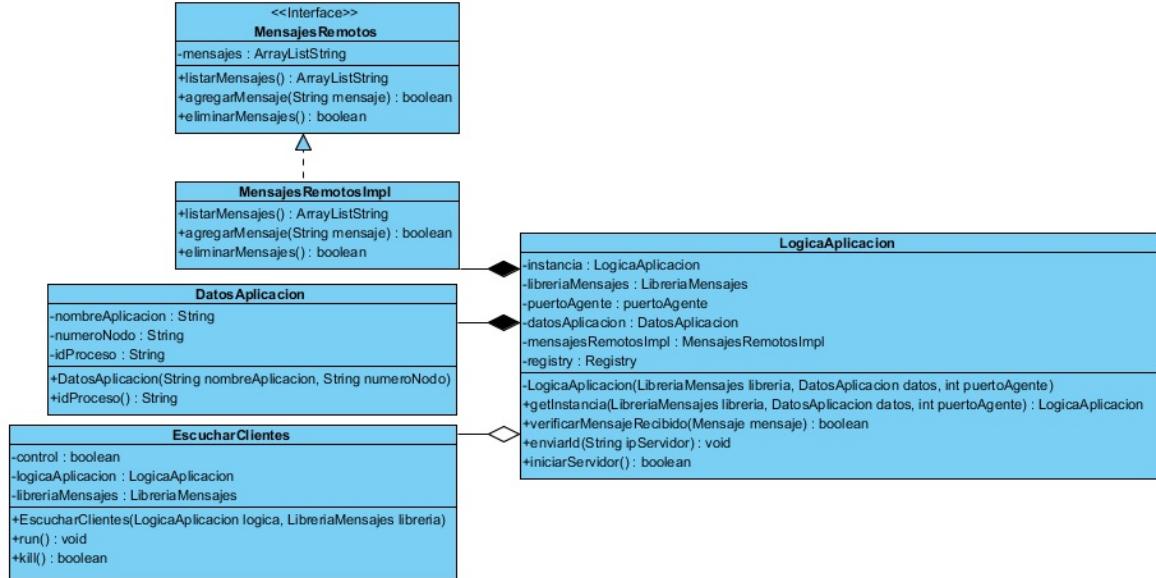


Ilustración 75: Diagrama de clases RMI 2 - Servidor. Elaboración propia.

#### Anexo 2.5.24.5. Diseño (Tarjetas CRC) - Cliente

MensajesRemotos	<ul style="list-style-type: none"> <li>• Interfaz que permite invocar los métodos remotos del servidor.</li> </ul>
-----------------	--

Tabla 59: Tarjetas CRC RMI 2 - Cliente. Elaboración propia.

#### Anexo 2.5.24.6. Diagrama de clases cliente

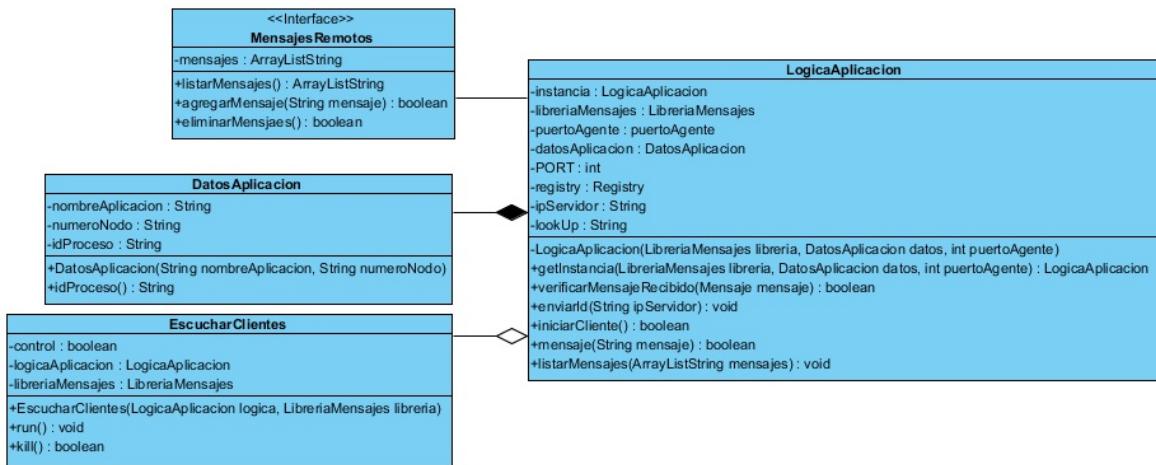


Ilustración 76: Diagrama de clases RMI 2 - Cliente. Elaboración propia.

#### Anexo 2.5.24.7. Parámetros propios

- Cliente: Ip servidor.

#### Anexo 2.5.25. EJB

##### Anexo 2.5.25.1. Historias de usuario

<b>Almacenar beans de sesión</b>
Como usuario de la infraestructura quiero exista una aplicación servidor que contenga el <i>bean</i> de sesión con estado ( <i>stateful</i> ) y sin estado ( <i>stateless</i> ) de manera que pueda ser utilizado por el cliente y comprobar diferencias.
<b>Consumir beans de sesión</b>
Como usuario de la infraestructura quiero que exista una aplicación cliente que permita acceder a los EJB almacenados en el servidor de manera que se pueda comparar ambos <i>beans</i> .

Tabla 60: Historias de usuario EJB. Elaboración propia.

##### Anexo 2.5.25.2. Arquitectura

#### Enterprise Java Beans (EJB)

- El servidor una aplicación web con los beans de sesión: con estado y sin estado.
- El cliente puede enviar mensajes para observar las diferencias entre cada bean.

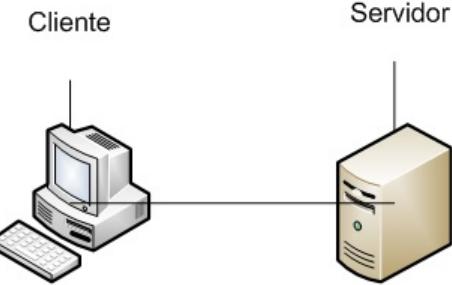


Ilustración 77: Arquitectura EJB. Elaboración propia.

### Anexo 2.5.25.3. Diagrama de clases

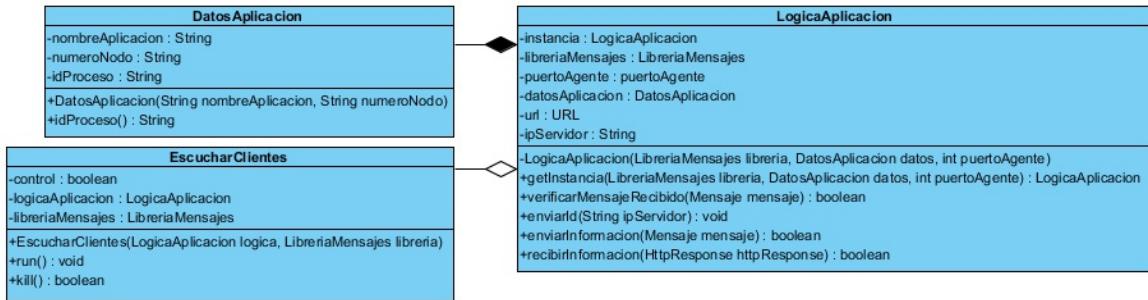


Ilustración 78: Diagrama de clases EJB. Elaboración propia.

### Anexo 2.5.25.4. Parámetros propios

- Cliente: Ip Servidor.

## Anexo 2.5.26. SOAP

### Anexo 2.5.26.1. Historias de usuario

Almacenar servicios web
Como usuario de la infraestructura quiero que exista una aplicación servidor que contenga los servicios web de manera que pueda ser accedido por los usuarios a través del WSDL
Consumir servicios
Como usuario de la infraestructura quiero que exista una aplicación que permita consumir los servicios web del servidor a través del WSDL

Tabla 61: Historias de usuario SOAP. Elaboración propia.

### Anexo 2.5.26.2. Arquitectura

### Simple Object Access Protocol (SOAP)



Ilustración 79: Arquitectura SOAP. Elaboración propia.

#### Anexo 2.5.26.3. Diseño (Tarjetas CRC) - Servidor

<b>MensajesServicio</b>	
<ul style="list-style-type: none"> <li>• Interfaz que contiene la firma de métodos web que pueden ser consumidos por el usuario</li> </ul>	
<b>MensajesServicioImpl</b>	
<ul style="list-style-type: none"> <li>• Clase que contiene la implementación de los métodos</li> </ul>	MensajesServicio

Tabla 62: Tarjetas CRC SOAP - Servidor. Elaboración propia.

#### Anexo 2.5.26.4. Diagrama de clases servidor

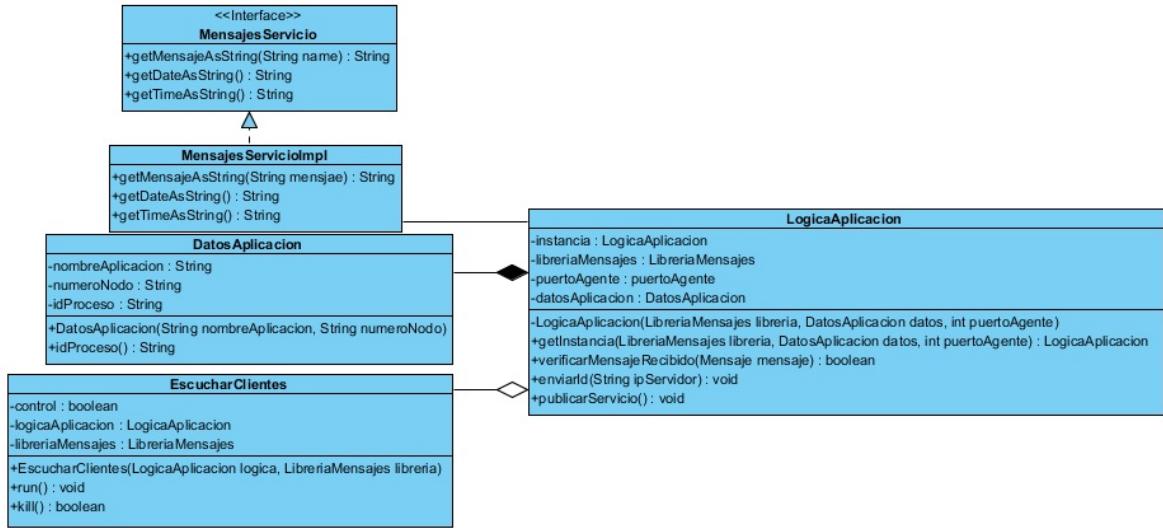


Ilustración 80: Diagrama de clases SOAP - Servidor. Elaboración propia.

#### Anexo 2.5.26.5. Diagrama de clases cliente

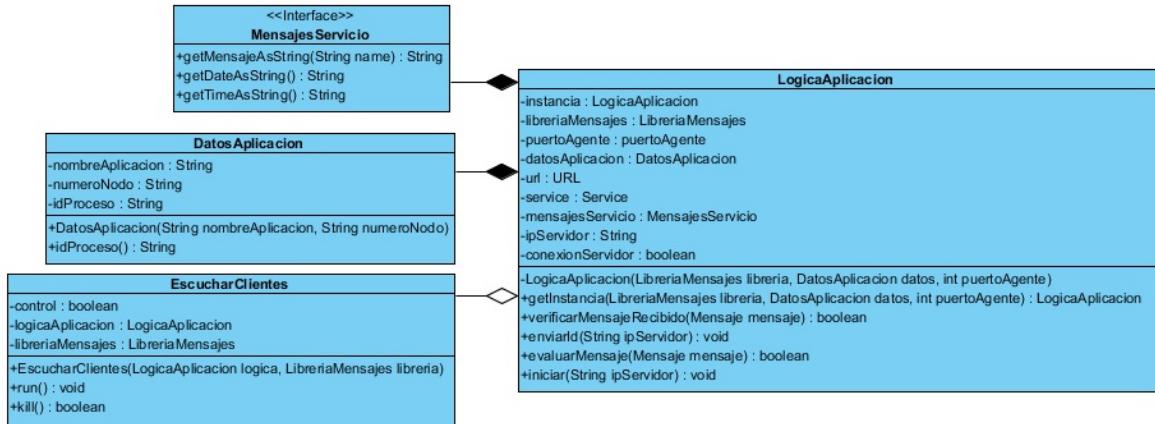


Ilustración 81: Diagrama de clases SOAP - Cliente. Elaboración propia.

#### Anexo 2.5.26.6. Parámetros propios

- Cliente Ip servidor.

#### Anexo 2.5.27. REST

##### Anexo 2.5.27.1. Historias de usuario

Almacenar servicios
Como usuario de la infraestructura quiero que exista una aplicación servidor que permita

contener varios servicios REST de manera que puedan ser consultado por los clientes.
Consumir servicios
Como usuario de la infraestructura quiero que exista una aplicación cliente que permita consultar los servicios a través de sus URL de manera que se pueda obtener la información correspondiente.

Tabla 63: Historias de usuario REST. Elaboración propia.

## Anexo 2.5.27.2. Arquitectura

**Representational State Transfer (REST)**

- El servidor contiene una aplicación web con una URL para cada servicio.
- Los servicios disponibles son: Hora, fecha y mensaje
- Se envía un mensaje al cliente para que acceda a una de las url para consumir el servicio

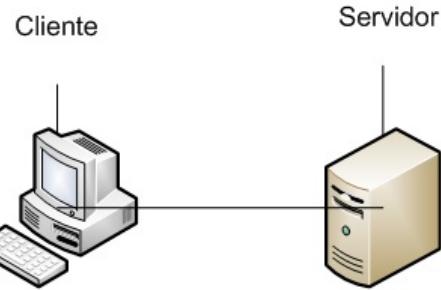


Ilustración 82: Arquitectura REST. Elaboración propia.

## Anexo 2.5.27.3. Diseño (Tarjetas CRC) - Servidor

<b>Fecha</b>	
• Servicio que devuelve la fecha actual del sistema	
<b>Hora</b>	
• Servicio que devuelve la hora actual del sistema	
<b>Saludo</b>	
• Servicio que devuelve un saludo general.	

Tabla 64. Tarjetas CRC REST - Servidor. Elaboración propia.

## Anexo 2.5.27.4. Diagrama de clases servidor

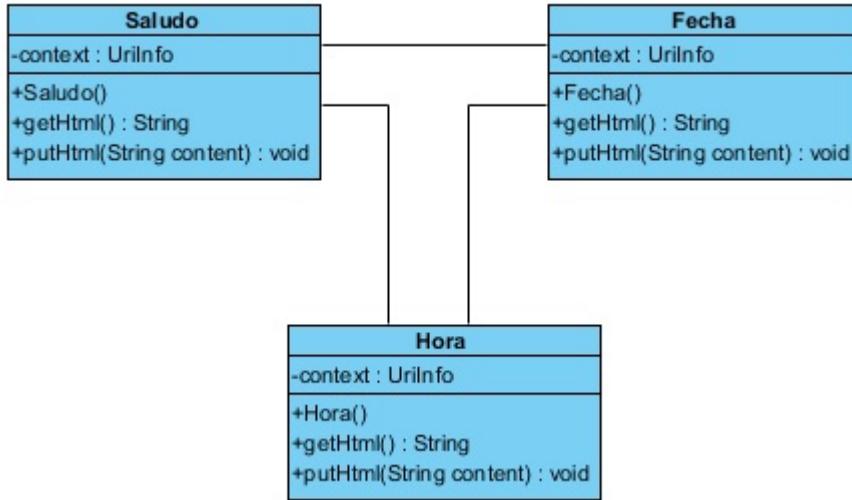


Ilustración 83: Diagrama de clases REST - Servidor. Elaboración propia.

## Anexo 2.5.27.5. Diagrama de clases cliente

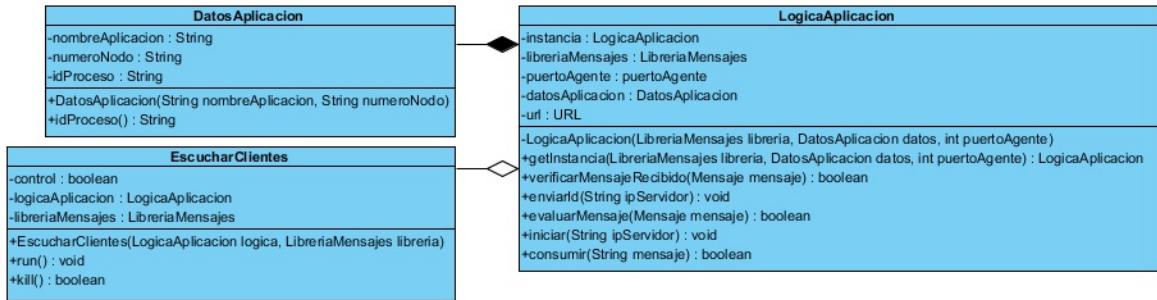


Ilustración 84: Diagrama de clases REST - Cliente. Elaboración propia.

## Anexo 2.5.27.6. Parámetros únicos

- Cliente: Ip servidor.

### Anexo 3. Manual de Usuario

Interfaz para la página de inicio del sistema.

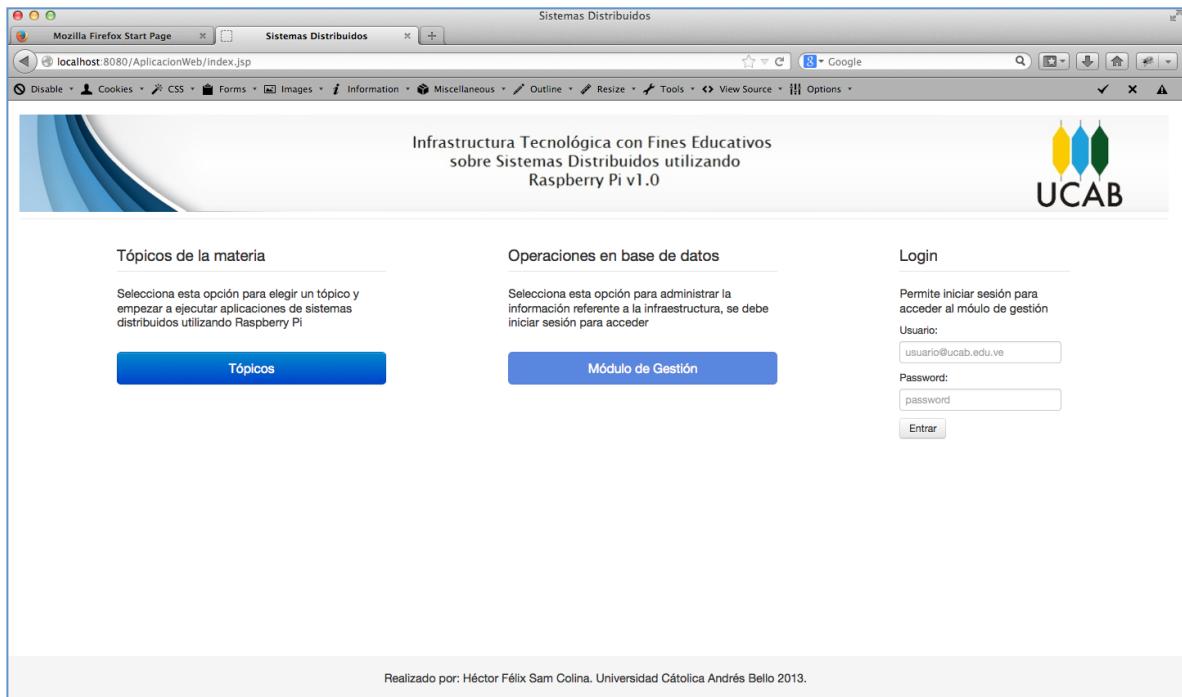


Ilustración 85: Página principal. Elaboración propia.

Funcionalidades de la página principal:

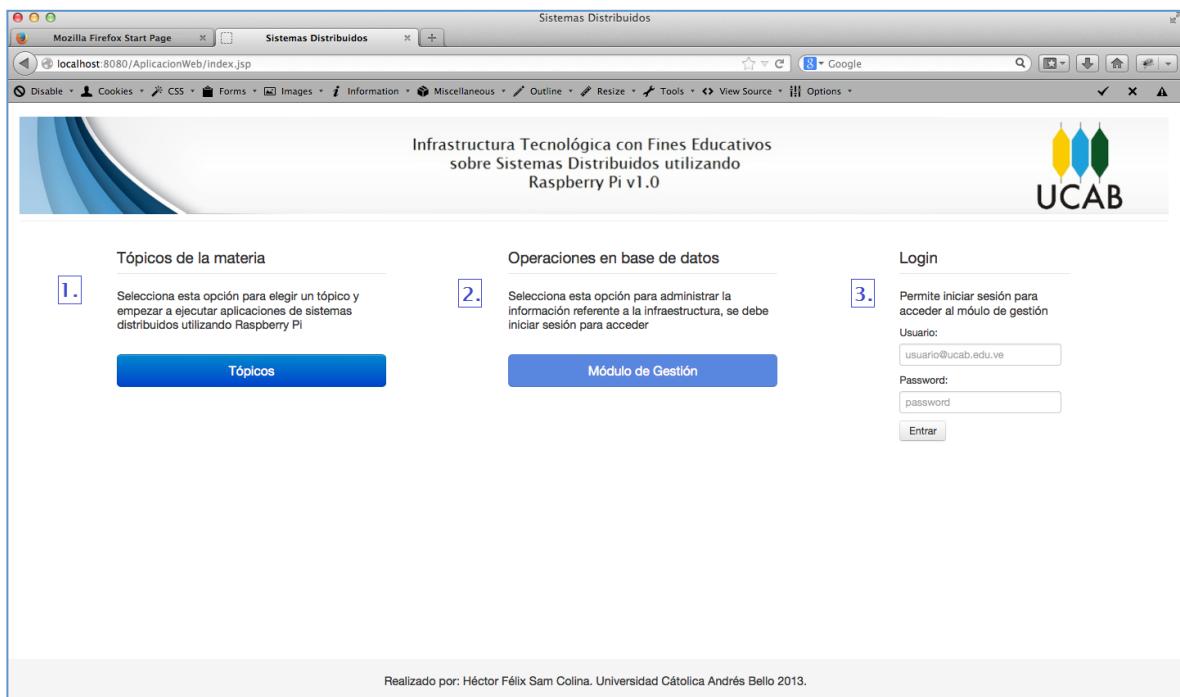


Ilustración 86: Funcionalidades página principal. Elaboración propia.

#### 1. Tópicos de la materia:

Esta opción permite seleccionar un tema de la cátedra sistemas distribuidos y elegir una de las aplicaciones disponibles para su ejecución.

**Tópicos de la materia**

Selecciona esta opción para elegir un tema y comenzar a ejecutar aplicaciones de sistemas distribuidos utilizando Raspberry Pi

**Tópicos**

Ilustración 87: Tópicos de la materia. Elaboración propia.

La página principal de Tópicos muestra los temas disponibles para un tema particular.

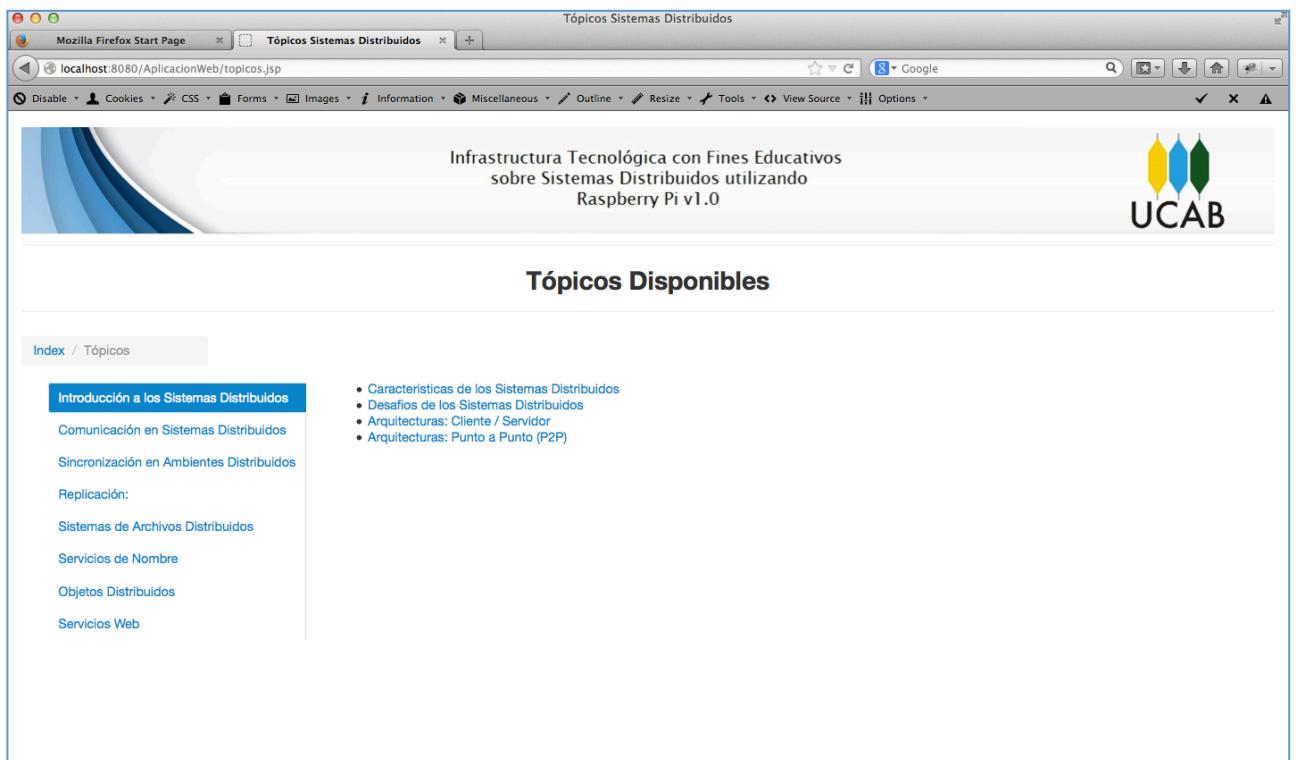


Ilustración 88: Tópicos disponibles. Elaboración propia.

Al seleccionar una de estas opciones se desplegará la lista de tópicos del tema como se muestra a continuación.

Ilustración 89: Tópicos disponibles 2. Elaboración propia.

Al elegir un tópico se muestran las aplicaciones disponibles

**Ilustración 90: Seleccionar aplicación. Elaboración propia.**

Al seleccionar y enviar la aplicación se da inicio a la página de la aplicación. En ella, se muestra la descripción del tópico y las instrucciones asociadas.

**Ilustración 91: Página de la aplicación - Descripción. Elaboración propia.**

The screenshot shows a Mozilla Firefox browser window with the title bar "Aplicacion Sistemas Distribuidos". The main content area displays the following information:

- Descripción Tópico:** Infrastructure Tecnológica con Fines Educativos sobre Sistemas Distribuidos utilizando Raspberry Pi v1.0.
- Escenarios:**
  - Leer Archivo
  - Iniciar Servidor Web
  - Página de Prueba
  - Detener Servidor Web
  - Escalabilidad
  - Tolerancia a Fallos
  - Transparencia
- Preguntas:** A button labeled "Generar Preguntas".

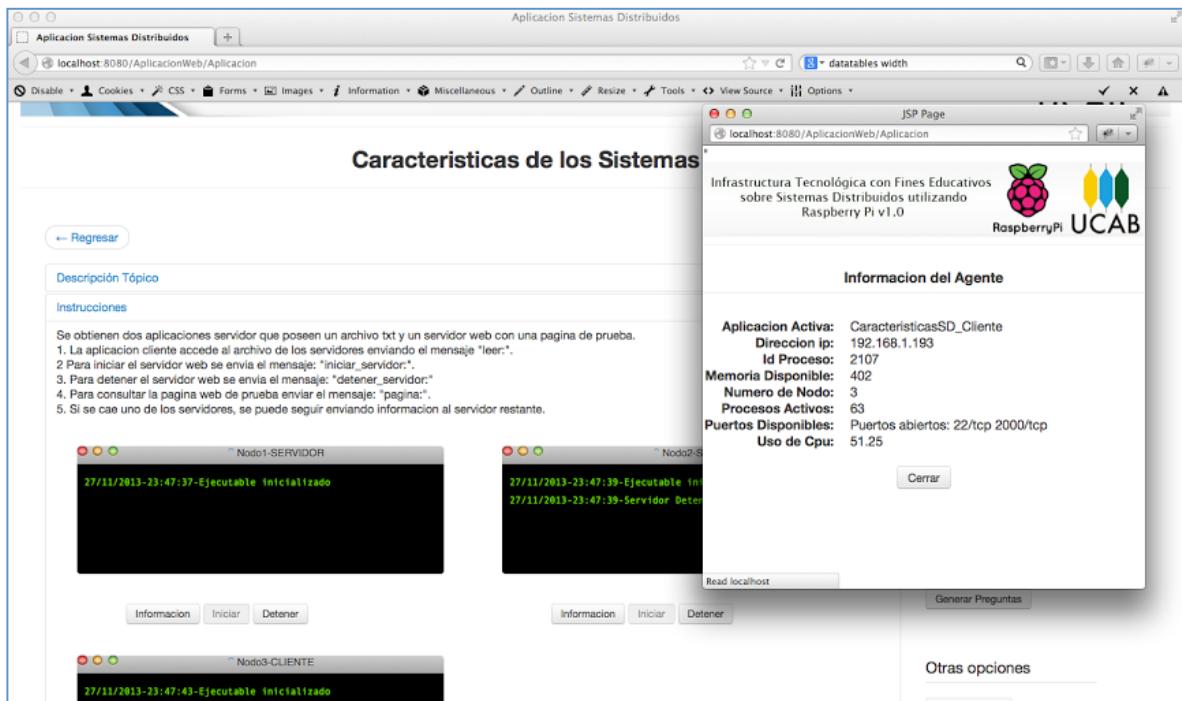
Below the main content, there are two small windows labeled "Nodo1-SERVIDOR" and "Nodo2-SERVIDOR", each with a black screen. At the bottom of the main window, there are buttons for "Información", "Iniciar", and "Detener". The URL in the address bar is "localhost:8080/AplicacionWeb/Aplicacion#collapseOne".

Ilustración 92: Página de la aplicación - Instrucciones. Elaboración propia.

### Nodos:

Los nodos permiten la simulación de sistemas distribuidos, cuentan con la opción de información y los eventos asociados: iniciar, detener o enviar un mensaje.

Al hacer clic en el botón de información se puede visualizar la información recolectada por el agente alojado en el nodo elegido.



**Ilustración 93: Información del agente. Elaboración propia.**

#### Escenarios:

Esta sección permite conocer los hechos que pueden ocurrir al seguir las instrucciones de la aplicación.

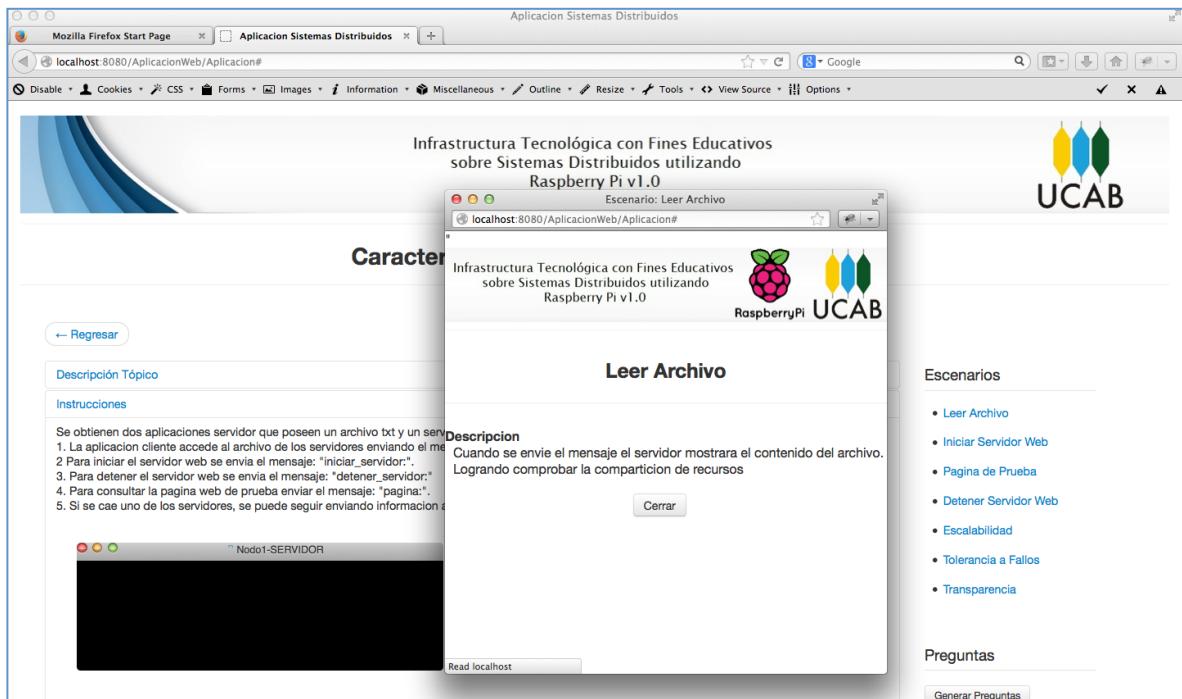


Ilustración 94: Escenarios aplicación. Elaboración propia.

### Preguntas:

Esta sección permite generar un conjunto de preguntas relacionadas con el tópico de aplicación seleccionada, se hace clic en generar preguntas y luego aparece la opción de descargar.



Ilustración 95: Sección preguntas. Elaboración propia.

Al hacer clic en descargar se abre una nueva ventana con un archivo PDF que contiene las preguntas generadas en ese momento. Este proceso puede repetirse varias veces para generar diferentes preguntas.

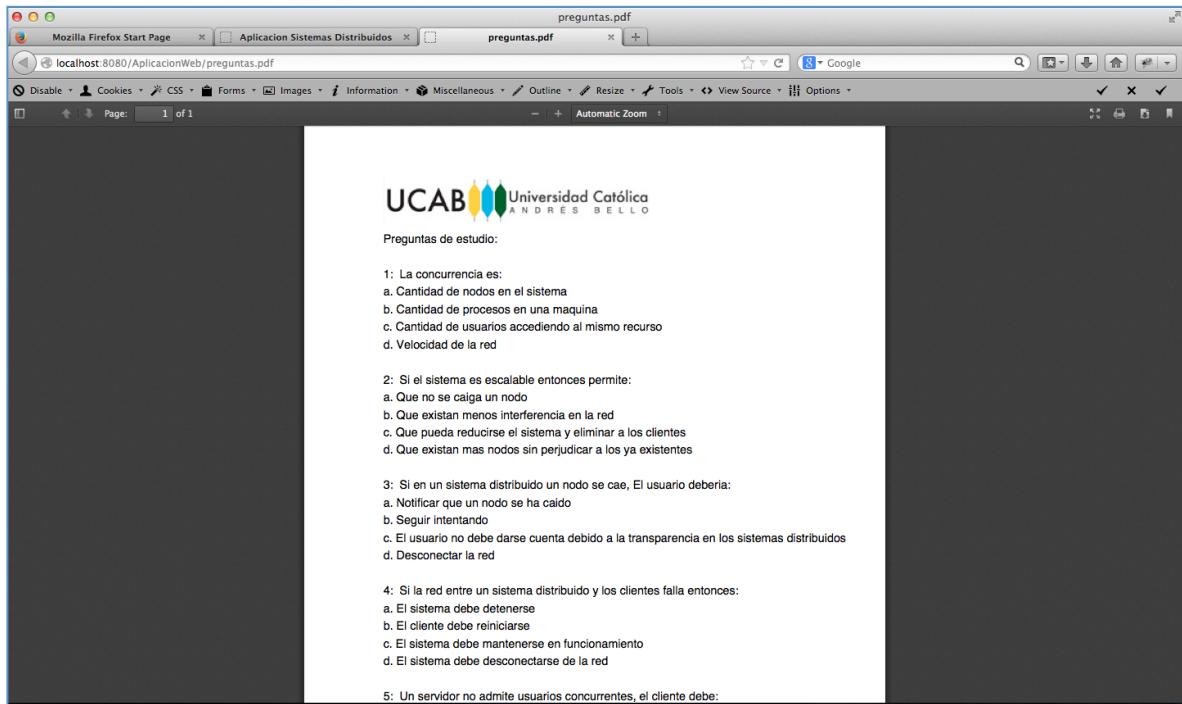


Ilustración 96: PDF preguntas. Elaboración propia.

**2. Operaciones en base de datos:** Esta opción permite realizar las operaciones: Insertar, modificar, eliminar y consultar en las tablas de la base de datos. Por defecto no se puede acceder.

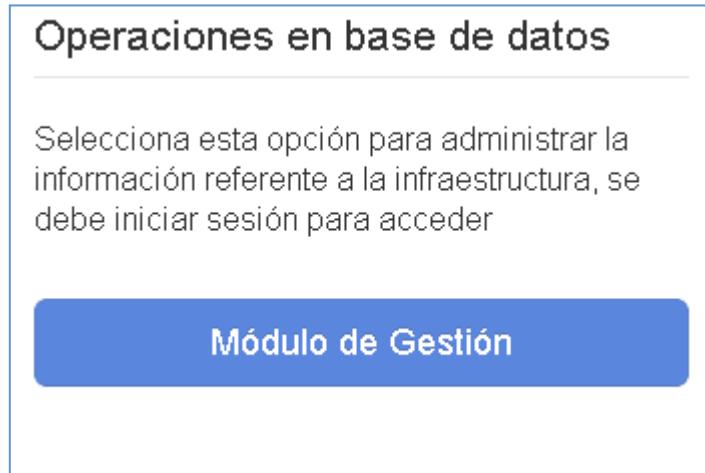


Ilustración 97: Operaciones en base de datos. Elaboración propia.

3. **Login o Área de ingreso:** Esta opción permite la validación del usuario a la infraestructura y permite el ingreso a las operaciones del módulo de gestión.

The illustration shows two side-by-side interface snippets. On the left, under 'Operaciones en base de datos', there is a note: 'Selecciona esta opción para administrar la información referente a la infraestructura, se debe iniciar sesión para acceder'. Below this is a blue button labeled 'Módulo de Gestión'. On the right, under 'Login', there is a note: 'Permite iniciar sesión para acceder al módulo de gestión'. It features input fields for 'Usuario' (with 'admin' typed) and 'Password' (with '\*\*\*\*' typed), followed by a 'Entrar' button.

Ilustración 98: *Login*. Elaboración propia.

Al ingresar exitosamente se mostrará el nombre y apellido del usuario y las opciones del módulo de gestión activadas.

The illustration shows the same two snippets as Illustration 98. On the left, the 'Operaciones en base de datos' section remains the same. On the right, under 'Login', the note 'Bienvenido, Hector Sam' is displayed above a 'Salir' button. The 'Módulo de Gestión' button from the left snippet is now shown in a grayed-out state, indicating it is disabled.

Ilustración 99: Deshabilitar botón módulo de gestión. Elaboración propia.

## Funcionalidades del módulo Gestión:

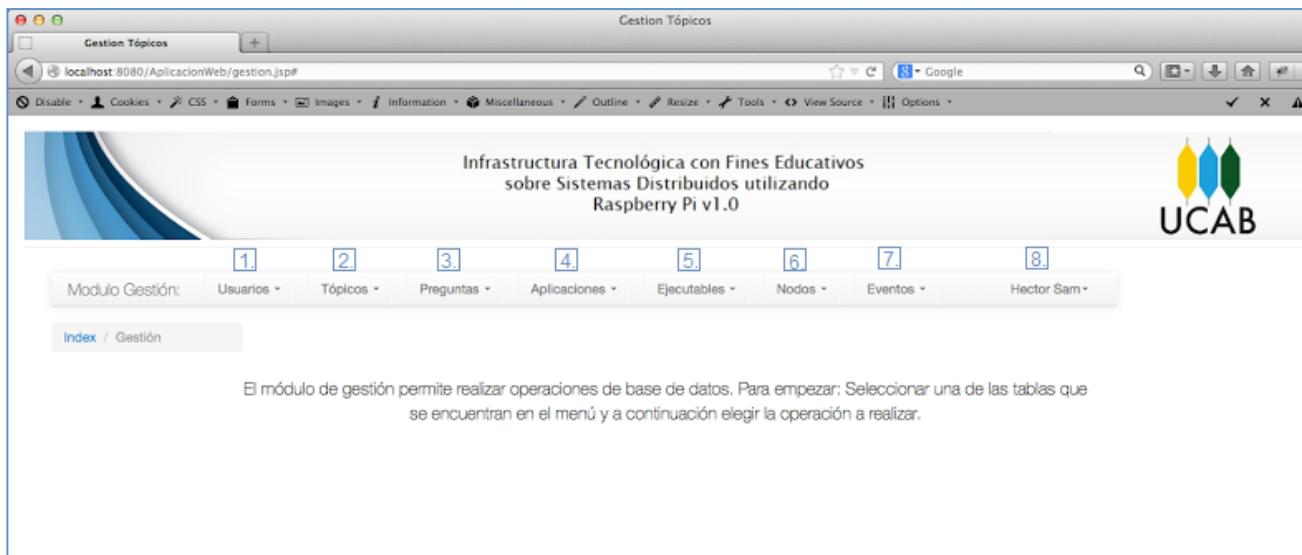


Ilustración 100: Módulo de gestión. Elaboración propia.

**1. Gestión de usuarios:** Esta sección cuenta con las opciones de crear, modificar, eliminar o consultar un usuario de la infraestructura.

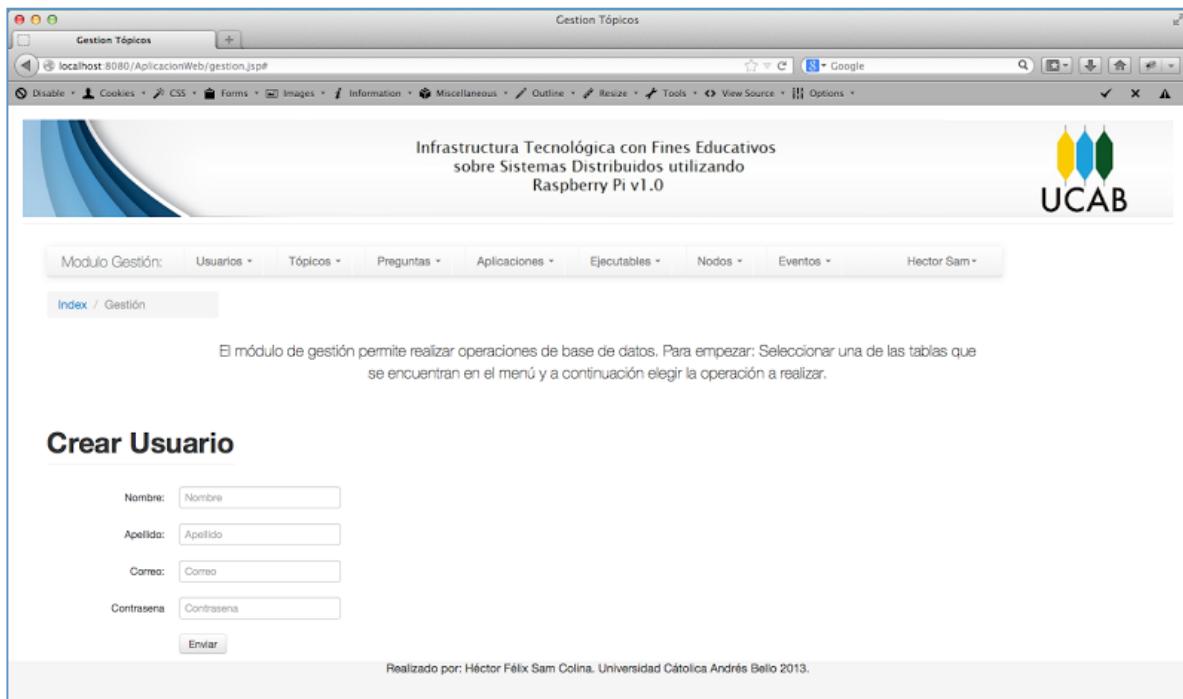


Ilustración 101: Gestión usuarios. Elaboración propia.

**2. Gestión de tópicos:** Esta sección cuenta con las opciones de crear, modificar , eliminar o consultar los tópicos de la cátedra.

Nombre	Categoría
Algoritmo de Selección: Anillo	SINCRONIZACION EN AMBIENTES DISTRIBUIDOS
Algoritmo de Selección: Gradulon	SINCRONIZACION EN AMBIENTES DISTRIBUIDOS
Arquitectura Cliente / Servidor: Modelo de Acceso Remoto	SISTEMAS DE ARCHIVOS DISTRIBUIDOS
Arquitectura Cliente / Servidor: Modelo de Carga y Descarga	SISTEMAS DE ARCHIVOS DISTRIBUIDOS
Arquitecturas: Cliente / Servidor	INTRODUCCION A LOS SISTEMAS DISTRIBUIDOS
Arquitecturas: Punto a Punto (P2P)	INTRODUCCION A LOS SISTEMAS DISTRIBUIDOS
Características de los Sistemas Distribuidos	INTRODUCCION A LOS SISTEMAS DISTRIBUIDOS
Comunicación en red	COMUNICACION EN SISTEMAS DISTRIBUIDOS

Mostrando registros del 1 al 26 de un total de 26 registros

Realizado por: Héctor Félix Sam Colina. Universidad Católica Andrés Bello 2013.

Ilustración 102: Gestión tópicos 1. Elaboración propia.

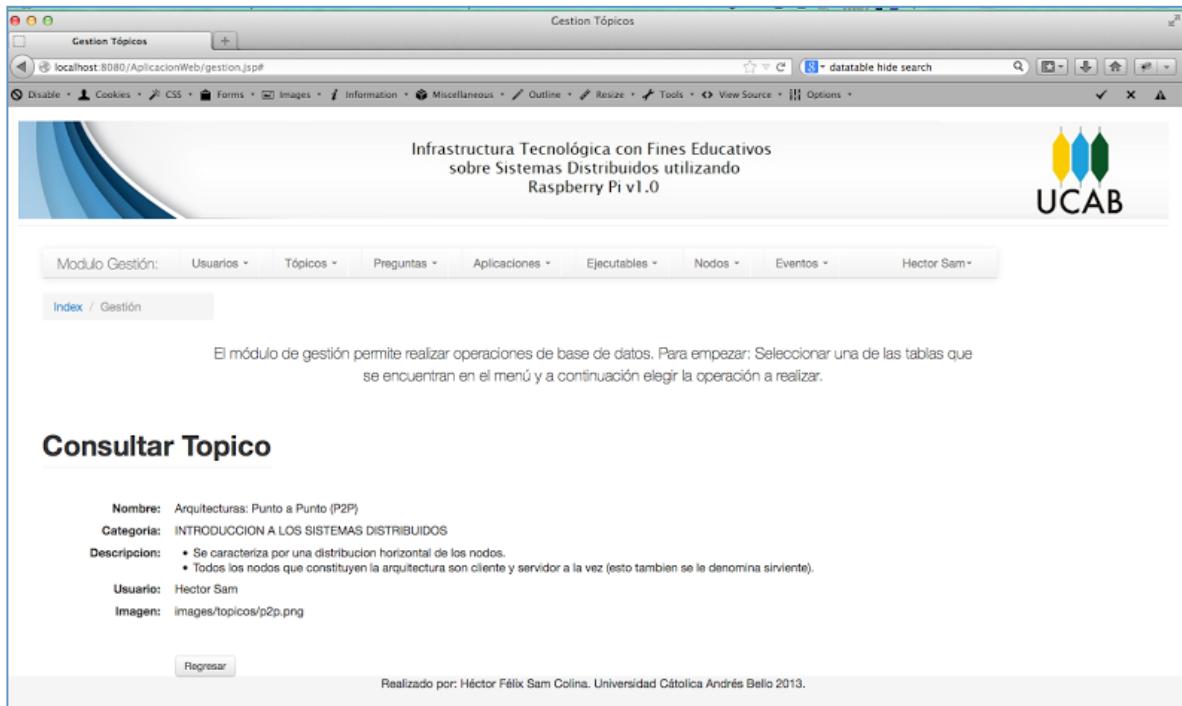


Ilustración 103: Gestión tópicos 2. Elaboración propia.

**3. Gestión de preguntas:** Cuenta con las opciones de agregar, modificar, eliminar o consultar las preguntas de los tópicos.

Ilustración 104: Gestión preguntas 1. Elaboración propia.

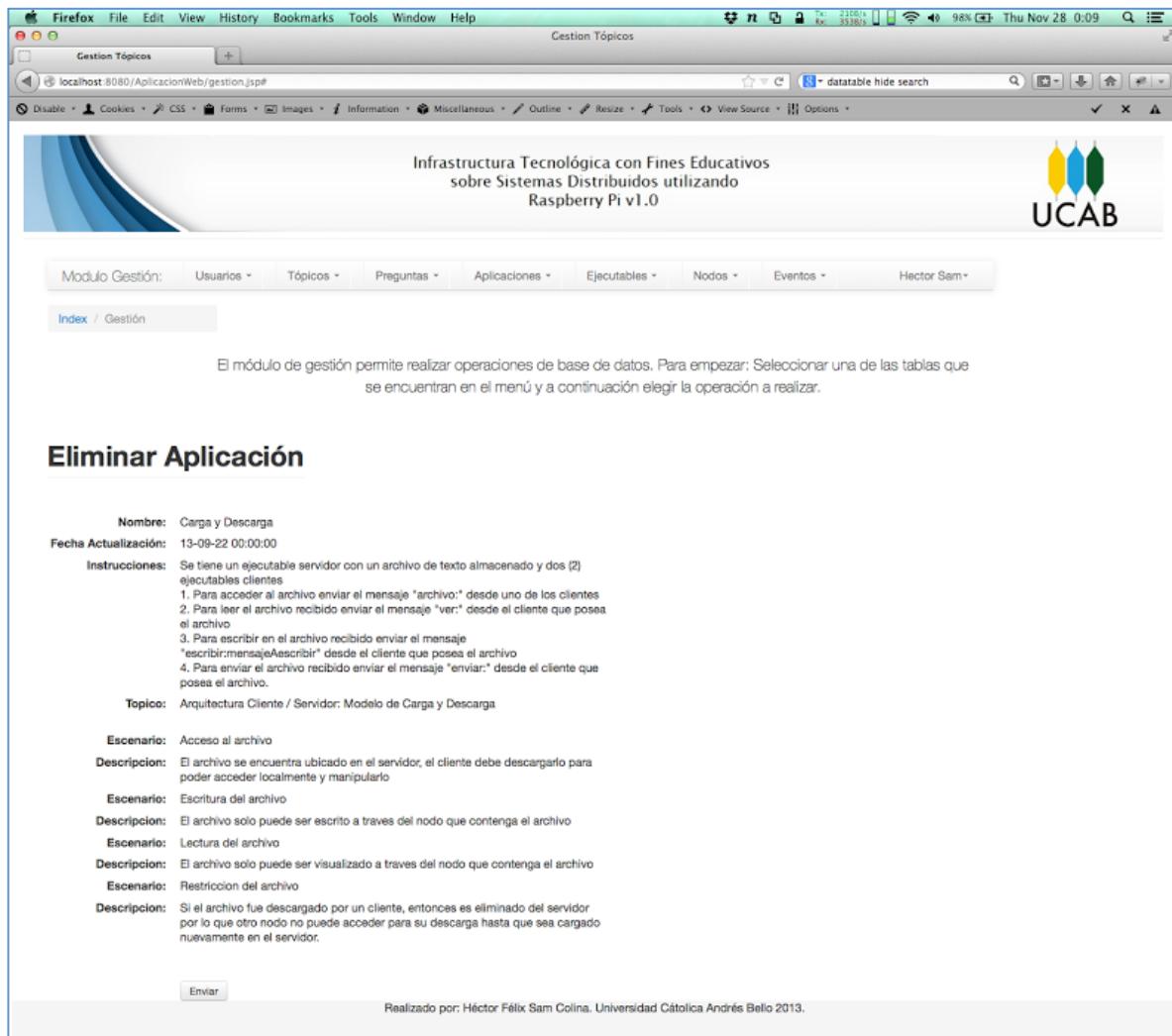
Ilustración 105: Gestión preguntas 2. Elaboración propia.

**4. Gestión de aplicaciones:** Cuenta con las opciones de agregar, modificar, eliminar o consultar las aplicaciones del sistema.

The screenshot shows a web browser window with the title "Gestion Tópicos". The URL is "localhost:8080/ApplicacionWeb/gestion.jsp#". The page header includes the text "Infraestructura Tecnológica con Fines Educativos sobre Sistemas Distribuidos utilizando Raspberry Pi v1.0" and the UCAB logo. A navigation bar at the top has items like "Modulo Gestión", "Usuarios", "Tópicos", "Preguntas", "Aplicaciones", "Ejecutables", "Nodos", "Eventos", and "Hector Sam...". Below the navigation is a breadcrumb trail: "Index / Gestión". A message states: "El módulo de gestión permite realizar operaciones de base de datos. Para empezar: Seleccionar una de las tablas que se encuentran en el menú y a continuación elegir la operación a realizar." The main content area is titled "Eliminar Aplicación" and contains a table with columns "Nombre aplicación" and "Topico". The table lists 26 entries, each with a link to its details. At the bottom of the table, it says "Mostrando registros del 1 al 26 de un total de 26 registros". A note at the bottom right says "Realizado por: Héctor Félix Sam Colina. Universidad Católica Andrés Bello 2013." The footer of the browser window shows various developer tools icons.

Nombre aplicación	Topico
Acceso Remoto	Arquitectura Cliente / Servidor: Modelo de Acceso Remoto
Anillo	Algoritmo de Selección: Anillo
Arquitectura_Cliente_Servidor	Arquitecturas: Cliente / Servidor
Berkeley	Relojes Físicos: Algoritmo de Berkeley
Características SD	Características de los Sistemas Distribuidos
Carga y Descarga	Arquitectura Cliente / Servidor: Modelo de Carga y Descarga
Centralizado	Exclusión Mutua: Algoritmo Centralizado
Cluster	Sistemas de Archivar Recursos en Cluster

Ilustración 106: Gestión aplicación 1. Elaboración propia.



**Ilustración 107: Gestión aplicación 2. Elaboración propia.**

5. **Gestión de ejecutables:** Posee las opciones de crear, modificar, eliminar o consultar los ejecutables asociados a una aplicación de sistemas distribuidos.
6. **Gestión de nodos:** Cuenta con las opciones de agregar, modificar, eliminar o consultar los nodos del sistema distribuido.
7. **Gestión de eventos:** Cuenta con las opciones de crear, modificar, eliminar o consultar los eventos asociados a una aplicación.
8. **Usuario:** Permite cerrar sesión para terminar con el trabajo o entrar con otra cuenta de usuario.

## Anexo 4. Guía de configuración

### Anexo 4.1. Requisitos Hardware / Software

1. Raspberry Pi:
  - a. Modelo B.
  - b. Cable de red o tarjeta inalámbrica.
  - c. Raspbian (Linux raspberry pi 3.6.11+ armv6l GNU/ Linux).
  - d. JDK 1.7.0\_03 o superior.
2. Base de datos:
  - a. Windows 7 Professional 64 bits.
  - b. 1GB de memoria RAM o superior.
  - c. Un procesador.
  - d. Oracle 11g Release 11.2.0.2.0.
  - e. SQL Developer (opcional, requiere JDK 1.7+).
3. Servidor Central:
  - a. Mac OS X 10.8+ o cualquier distribución Linux 64 bits.
  - b. 1GB de memoria RAM.
  - c. Un procesador
  - d. JDK 1.7.0\_45 o superior.
  - e. Ojdbc6.jar o superior
4. Agente de configuración:
  - a. JDK 1.7.0\_45 o superior.
5. Librería de mensajes:
  - a. JDK 1.7\_0\_45 o superior.
6. Aplicación Web:
  - a. Glassfish Server versión 3.12.2.
  - b. 2GB de memoria RAM.
  - c. Al menos dos procesadores.
  - d. JDK 1.7.0\_45 o superior.
  - e. Librerías:
    - i. Ojdbc6.jar
    - ii. Commons-fileupload-1.3.jar
    - iii. Commons-io-2.4.jar
    - iv. Jdom-2.0.5.jar
    - v. Itextpdf-5.1.3.jar
7. Otros:
  - a. Editor utilizado: Netbeans versión 7.4
  - b. Control de versiones: Git
  - c. Repositorio utilizado: Bitbucket, dirección de la infraestructura: <https://bitbucket.org/qckzr/teg-hs>

## Anexo 4.2. Configuración

### 1. Raspberry Pi:

- Cada Raspberry Pi posee una dirección ip estática. Para cada nodo se tiene la siguiente dirección:
  - Nodo 1: 192.168.1.191
  - Nodo 2: 192.168.1.192
  - Nodo 3: 192.168.1.193
  - Nodo 4: 192.168.1.194
- El usuario y contraseña de cada Raspberry Pi: pi / raspberry.
- Se instaló el jdk siguiendo el siguiente enlace <http://www.oracle.com/technetwork/articles/java/raspberrypi-1704896.html>
- Para permitir la comunicación entre el servidor central se habilitó el modulo de SSH.
- Los ejecutables de las aplicaciones de sistemas distribuidos son alojados en el directorio Desktop, es por ello que se debe crear un directorio lib en /home/pi/Desktop/ y almacenar el ejecutable de la librería de mensajes.
- Puerto de escucha de mensajes (aplicaciones sistemas distribuidos): 1337.

### 2. Base de datos :

- Posee dirección ip estática de valor: 192.168.1.200
- Debe estar en ejecución antes de iniciar la aplicación web.
- Los Scripts de la base de datos son: CREATE.SQL, DROPS.SQL y INSERTS.SQL.
- Se debe conocer el nombre de usuario y contraseña del usuario para conectarse a la base de datos.

### 3. Servidor Central:

- Posee dirección ip estática de valor: 192.168.1.199.
- Se debe iniciar el servidor central antes de iniciar la aplicación web.
- Iniciar el ejecutable del servidor central con los siguientes argumentos: Nombre de usuario, contraseña, puerto base de datos y dirección ip.
- Debe existir una carpeta llamada lib que contenga el ejecutable de la librería de mensajes y el ojdbc (librería para ejecutar la conexión con la base de datos) en donde se encuentre corriendo el ejecutable del servidor central.
- El directorio donde se encuentren los scripts de sistema operativo debe ser: <ruta\_del\_ejecutable\_servidor>/scripts/
- El directorio donde se encuentre el repositorio de los ejecutables debe ser: <ruta\_del\_ejecutable\_servidor>/ejecutables/
- Puerto de escucha de mensajes: 1337.

### 4. Librería de mensajes:

- No requiere de configuración, solo basta con obtener el ejecutable disponible para copiarlo.

**5. Agente de configuración:**

- Se debe copiar el ejecutable del agente de configuración en cada nodo Raspberry Pi en la ruta: /home/pi/Desktop/
- Editar el archivo /etc/rc.local en el Raspberry y colocar el comando: java -jar /home/pi/Desktop/AgenteConfiguracion.jar <Puerto 2000> <Direccion ip del servidor central: 192.168.1.199> <Interfaz de red del Raspberry Pi>
- Puerto de escucha de mensajes: 2000.

**6. Aplicación Web:**

- Por defecto, la aplicación web se encuentra configurada para trabajar en el mismo computador que el servidor central.
- Dirección ip: Igual al servidor central
- Ruta de imágenes por defecto:  
<path\_ejecutable\_aplicacion\_web>/images/<tópicos> y  
<path\_ejecutable\_aplicacion\_web>/images/<escenarios>
- Ejecutar la aplicación web desde netbeans (es más fácil que cargar el ejecutable dentro de glassfish server).

**Anexo 4.3. Para desplegar aplicaciones de sistemas distribuidos:**

1. Utilizar la librería de mensajes.
2. Puerto por defecto de espera de mensajes: 1337.
3. La aplicación debe permitir recibir mensajes del agente de configuración (Puerto 2000)
4. Debe poseer los parámetros base: Nombre de la aplicación. número de nodo, puerto del agente y dirección ip del servidor central.
5. Todo mensaje de información debe ser enviado al servidor central.
6. Los ejecutables deben estar almacenados en el servidor central

## Anexo 5. Scripts de Sistema Operativos

```
#! /bin/bash

# $1 es el .jar
# $2 es el ip destino
# $3 es el user
archivo=$1
ruta=$2
ipDestino=$3
usuario=$4
parametros=${@:5}

scp $ ruta $ archivo $ usuario@$ ipDestino :/home/$ usuario/Desktop
ssh $ usuario@$ ipDestino -t "cd Desktop; java -jar $ archivo $ parametros"
```

Ilustración 108: Script ejecutar.sh. Elaboración propia.

```
#! /bin/bash

ipDestino=$1
usuario=$2
proceso=$3

ssh $ usuario@$ ipDestino -t "kill -9 $ proceso"
```

Ilustración 109: Script eliminarNodo.sh. Elaboración propia.

```
#! /bin/bash

parametro=1
for parametros in ${@:1}
do
    if (( $parametro == 1 ))
    then
        usuario=$parametros
    elif (( $parametro == 2 ))
    then
        ip=$parametros
    elif (( $parametro == 3 ))
    then
        proceso=$parametros
        ssh $usuario@$ip -t "kill -9 $proceso"
        parametro=0
    fi
    ((parametro++))
done
```

Ilustración 110: Script eliminarTodosNodos.sh. Elaboración propia.