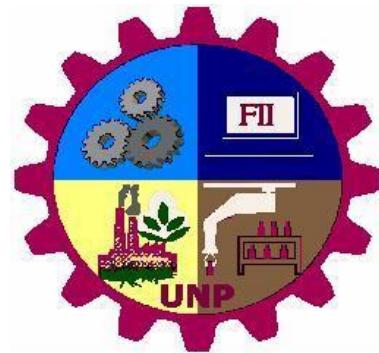


**UNIVERSIDAD NACIONAL DE PIURA**  
**FACULTAD DE INGENIERÍA INDUSTRIAL**  
**ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA**



**“ESTUDIO COMPARATIVO DEL TIEMPO DE EJECUCIÓN  
ENTRE FRAMEWORKS DE PERSISTENCIA ORM Y ODM  
BASADOS EN LOS LENGUAJES DE PROGRAMACIÓN  
JAVA Y PHP APLICADO A UN MÓDULO WEB”**

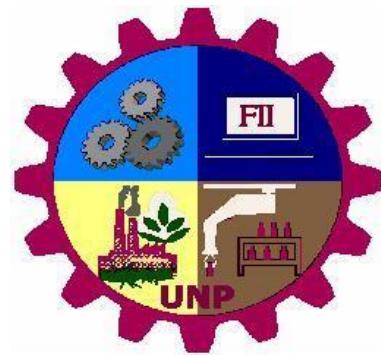
**PRESENTADO POR:  
ARÉVALO GARAVITO, ALEXANDER PAÚL**

**PARA OPTAR EL TÍTULO PROFESIONAL DE  
INGENIERO INFORMÁTICO**

**PIURA, PERÚ**

**2018**

**UNIVERSIDAD NACIONAL DE PIURA**  
**FACULTAD DE INGENIERÍA INDUSTRIAL**  
**ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA**



**"ESTUDIO COMPARATIVO DEL TIEMPO DE EJECUCIÓN  
ENTRE FRAMEWORKS DE PERSISTENCIA ORM Y ODM  
BASADOS EN LOS LENGUAJES DE PROGRAMACIÓN  
JAVA Y PHP APLICADO A UN MÓDULO WEB"**

**PRESENTADO POR:**  
**ARÉVALO GARAVITO, ALEXANDER PAÚL**

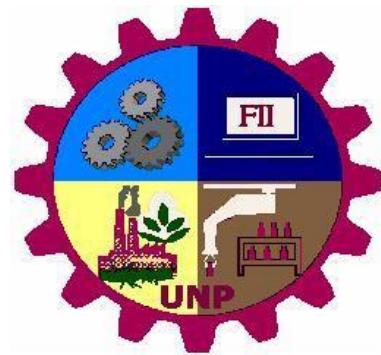
**PARA OPTAR EL TÍTULO PROFESIONAL DE  
INGENIERO INFORMÁTICO**

**LÍNEA DE INVESTIGACIÓN  
INGENIERÍA DE SOFTWARE**

**PIURA, PERÚ**

**2018**

**UNIVERSIDAD NACIONAL DE PIURA**  
**FACULTAD DE INGENIERÍA INDUSTRIAL**  
**ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA**



**TESIS PRESENTADA COMO REQUISITO PARA OPTAR EL  
TÍTULO PROFESIONAL DE INGENIERO INFORMÁTICO**

**“ESTUDIO COMPARATIVO DEL TIEMPO DE EJECUCIÓN  
ENTRE FRAMEWORKS DE PERSISTENCIA ORM Y ODM  
BASADOS EN LOS LENGUAJES DE PROGRAMACIÓN  
JAVA Y PHP APLICADO A UN MÓDULO WEB”**

---

**ARÉVALO GARAVITO  
ALEXANDER PAÚL  
TESISTA**

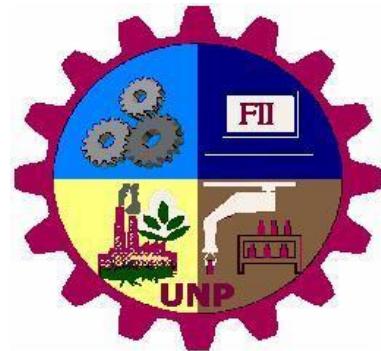
---

**DR. CRUZ VILCHEZ  
FRANCISCO JAVIER  
ASESOR**

---

**DR. SAAVEDRA ARANGO  
MOISÉS DAVID  
CO-ASESOR**

**UNIVERSIDAD NACIONAL DE PIURA**  
**FACULTAD DE INGENIERÍA INDUSTRIAL**  
**ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA**



**TESIS PRESENTADA COMO REQUISITO PARA OPTAR EL  
TÍTULO PROFESIONAL DE INGENIERO INFORMÁTICO**

**“ESTUDIO COMPARATIVO DEL TIEMPO DE EJECUCIÓN  
ENTRE FRAMEWORKS DE PERSISTENCIA ORM Y ODM  
BASADOS EN LOS LENGUAJES DE PROGRAMACIÓN  
JAVA Y PHP APLICADO A UN MÓDULO WEB”**

**JURADO**

---

**DR. REUCHER CORREA MOROCHO**  
**PRESIDENTE**

---

**DR. RIGO FELIX  
REQUENA FLORES**  
**VOCAL**

---

**MBA. PERSY WILLIANSH  
CABRERA ANTÓN**  
**SECRETARIO**

## **DECLARACIÓN JURADA DE ORIGINALIDAD DE LA TESIS**

Yo: \_\_\_\_\_ identificado con DNI N° \_\_\_\_\_, Bachiller de la Escuela Profesional de \_\_\_\_\_, de la Facultad de \_\_\_\_\_ y domiciliado en \_\_\_\_\_ del Distrito de \_\_\_\_\_, Provincia de \_\_\_\_\_, Departamento de \_\_\_\_\_, Celular: \_\_\_\_\_. Email: \_\_\_\_\_.

**DECLARO BAJO JURAMENTO:** que la tesis que presento es original e inédita, no siendo copia parcial ni total de una tesis desarrollada, y/o realizada en el Perú o en el Extranjero, en caso contrario de resultar falsa la información que proporciono, me sujeto a los alcances de lo establecido en el Art. N° 411, del código Penal concordante con el A11. 32º de la Ley N° 27444, y Ley del Procedimiento Administrativo General y las Normas Legales de Protección a los Derechos de Autor.

En fe de lo cual firmo la presente.

Piura \_\_\_\_ de \_\_\_\_\_ de \_\_\_\_\_.  
\_\_\_\_\_  
DNI N° \_\_\_\_\_

Artículo 411.- El que, en un procedimiento administrativo, hace una falsa declaración en relación con hechos o circunstancias que le corresponde probar, violando la presunción de veracidad establecida por ley, será reprimido con pena privativa de libertad no menor de uno ni mayor de cuatro años.

Art. 4. Inciso 4.12 del Reglamento del Registro Nacional de Trabajos de Investigación para optar grados académicos y títulos profesionales - RENATI Resolución de Consejo Directivo N° 033-2016-SUNEDU/CD.

## **DEDICATORIA**

A mi Familia, en especial a mi madre Santos Isabel, por el gran apoyo que me ha brindado durante todo este tiempo, por el gran esfuerzo que realizó día a día para ayudarme a lograr mi meta.

## **AGRADECIMIENTOS**

A mi madre por brindarnos apoyo económico y emocional tanto a mí como a mi hermano, por todo el esfuerzo realizado durante tantos años para ayudarnos a salir adelante, por sus consejos y su sabiduría para afrontar los obstáculos de la vida.

A mi padre Armando, mi hermano Jonathan y a la familia Garavito Mogollón por estar siempre presentes en los momentos más difíciles.

A mi asesor el Dr. Cruz Vílchez por aceptar ser mi asesor y por su apoyo incondicional en la presente investigación.

A mi co-asesor el Dr. Moisés Saavedra por ser la persona que me incentivó a empezar esta investigación, por sus consejos y su apoyo para culminar.

A mis primos, compañeros de clases y amigos que me brindaron su apoyo emocional, que estuvieron siempre preguntando cuando culminaba la tesis y que ahora ya no será necesario.

A los docentes de la Universidad Nacional de Piura, en especial a la Magister Carmen Quito que me ayudaron de una u otra forma en la presente investigación.

A todas las personas que conocí en el trámite del proyecto y desarrollo de tesis, gracias por su apoyo desinteresado.

## ÍNDICE GENERAL

DEDICATORIA .....	II
AGRADECIMIENTOS .....	III
ÍNDICE GENERAL .....	IV
ÍNDICE DE TABLAS .....	IX
ÍNDICE DE GRÁFICOS .....	XIII
ÍNDICE DE FIGURAS .....	XV
ÍNDICE DE ANEXOS .....	XVIII
RESUMEN .....	XIX
ABSTRACT .....	XX
INTRODUCCIÓN .....	1
CAPÍTULO I .....	3
EL PROBLEMA DE LA INVESTIGACIÓN .....	3
1.1. PLANTEAMIENTO DEL PROBLEMA .....	4
1.2. FORMULACIÓN DEL PROBLEMA .....	4
1.3. JUSTIFICACIÓN DEL ESTUDIO .....	5
1.4. OBJETIVOS DE LA INVESTIGACIÓN .....	9
1.4.1. Objetivo general .....	9
1.4.2. Objetivos específicos .....	9
CAPÍTULO II .....	10
MARCO TEÓRICO .....	10
2.1. ANTECEDENTES DEL ESTUDIO .....	11
2.2. GLOSARIO DE TÉRMINOS BÁSICOS .....	12
2.2.1. Sistema Gestor de Base de Datos .....	12
2.2.2. Base de datos relacional .....	13
2.2.3. Base de datos NoSQL .....	13

2.2.4.	Persistencia .....	13
2.2.5.	<i>Framework</i> de persistencia.....	13
2.2.6.	ORM .....	14
2.2.7.	ODM.....	14
2.2.8.	Registro.....	14
2.2.9.	Consulta.....	14
2.2.10.	Tiempo de ejecución.....	14
2.2.11.	Módulo <i>Web</i> .....	15
2.2.12.	Trámite Documentario.....	15
2.2.13.	FURPS .....	15
2.2.14.	TrendySkills .....	15
2.2.15.	Álgebra relacional.....	16
2.2.16.	Regla ACID .....	16
2.2.17.	Normalización .....	16
2.2.18.	CurrentTimeMillis .....	16
2.2.19.	MicroTime .....	16
2.2.20.	Clase Math.....	16
2.3.	MARCO CONCEPTUAL .....	17
2.3.1.	Bases de datos relacionales.....	17
2.3.1.1.	Elementos de una base de datos .....	17
2.3.1.2.	Normalización del esquema relacional .....	17
2.3.2.	MariaDB .....	18
2.3.2.1.	Características .....	19
2.3.2.2.	Arquitectura.....	20
2.3.3.	PostgreSQL.....	21
2.3.3.1.	Características .....	22
2.3.3.2.	Arquitectura.....	22
2.3.4.	MongoDB .....	23
2.3.4.1.	Características .....	24
2.3.4.2.	Arquitectura.....	26
2.3.5.	CouchDB .....	27
2.3.5.1.	Características .....	27
2.3.5.2.	Arquitectura.....	28
2.3.6.	Persistencia de objetos .....	29
2.3.7.	<i>Framework</i> (marco de trabajo) .....	30

2.3.7.1.	Arquitectura de un <i>Framework</i> .....	32
2.3.7.2.	Tipos de <i>frameworks</i> .....	33
2.3.7.3.	<i>Framework</i> de persistencia.....	34
2.3.8.	ORM .....	34
2.3.9.	Java Database Connectivity (JDBC) .....	35
2.3.9.1.	Arquitectura de JDBC .....	36
2.3.9.2.	Características de JDBC.....	37
2.3.9.3.	Ventajas de JDBC .....	38
2.3.10.	Hibernate .....	38
2.3.11.	Doctrine .....	40
2.3.12.	ODM.....	42
2.3.13.	Hibernate OGM .....	42
2.3.14.	Doctrine ODM.....	45
2.3.15.	Metodología XP.....	46
2.4.	HIPÓTESIS .....	47
2.4.1.	Hipótesis general .....	47
2.4.2.	Hipótesis específicas.....	47
2.4.3.	Identificación y Operacionalización de variables.....	47
2.4.3.1.	Variables independientes.....	47
2.4.3.2.	Variable dependiente .....	47
2.4.3.3.	Operacionalización de variables.....	48
CAPÍTULO III .....		50
METODOLOGÍA.....		50
3.1.	TIPO DE INVESTIGACIÓN .....	51
3.2.	NIVEL DE INVESTIGACIÓN .....	51
3.3.	DISEÑO DE LA INVESTIGACIÓN .....	52
3.3.1.	Métodos estadísticos de procesamiento de datos .....	55
3.4.	TIPOS Y TÉCNICAS DE MUESTREO .....	56
3.4.1.	Universo poblacional .....	56
3.4.2.	Tamaño de muestra.....	56
3.5.	TÉCNICAS E INSTRUMENTOS PARA LA RECOLECCIÓN DE DATOS .....	56
3.5.1.	Encuestas .....	56
3.5.2.	Investigación bibliográfica .....	57
3.5.3.	Guía de observación .....	57

3.5.4.	Programa cronómetro .....	57
3.5.5.	Validez de la encuesta .....	57
3.5.6.	Aplicación de la encuesta .....	58
3.5.7.	Procesamiento de la encuesta .....	58
3.5.7.1.	Módulo Web más usado .....	58
3.5.7.2.	Operaciones más frecuentes .....	60
3.5.8.	Generación de datos para las pruebas .....	61
3.5.9.	Ambiente para desarrollo y simulación .....	62
3.6.	CASO DE ESTUDIO .....	63
3.6.1.	Caso de estudio: Sistema de Trámite Documentario.....	63
3.6.2.	Obtención de los requerimientos .....	64
3.6.2.1.	Requerimientos del usuario.....	64
3.6.2.2.	Especificación de los requerimientos .....	67
3.6.3.	Diagramas.....	75
3.6.3.1.	Diagrama de Casos de Uso del negocio .....	75
3.6.3.2.	Diagramas de Objetos del Negocio (DON).....	78
3.6.3.3.	Diagramas de Casos de Uso de Requerimientos (DUC).....	78
3.6.3.4.	Diagramas de Colaboración (DC) .....	80
3.6.3.5.	Diagramas de Secuencia (DS).....	82
3.6.3.6.	Diagrama de Clases .....	84
3.6.3.7.	Diagramas de Componentes .....	85
3.6.3.8.	Diagramas de Despliegue .....	89
3.6.3.9.	Diagramas físico de la base de datos.....	94
	CAPÍTULO IV .....	96
	RESULTADOS .....	96
4.1.	Presentación de resultados .....	97
4.2.	Análisis de resultados .....	102
4.3.	Discusión de resultados .....	129
4.4.	Validación de hipótesis.....	131
	CAPÍTULO V .....	132
	CONCLUSIONES Y RECOMENDACIONES .....	132
5.1.	Conclusiones .....	133
5.2.	Recomendaciones .....	134

REFERENCIAS BIBLIOGRÁFICAS .....	135
ANEXO .....	140

## ÍNDICE DE TABLAS

Tabla 1. Identificación de variables, indicadores e índices .....	49
Tabla 2. Tabla comparativa del tiempo de ejecución para los frameworks de persistencia ORM .....	53
Tabla 3. Tabla comparativa del tiempo de ejecución para los frameworks de persistencia ODM.....	54
Tabla 4. Instituciones encuestadas de la ciudad de Piura .....	59
Tabla 5. Hardware y Software utilizados para el desarrollo y simulación de las aplicaciones desarrolladas .....	63
Tabla 6. Requerimientos del usuario para el Trámite Documentario.....	65
Tabla 7. Requerimientos del sistema correspondiente al requerimiento de usuario 1 .....	66
Tabla 8. Requerimientos del sistema correspondiente al requerimiento de usuario 2 .....	66
Tabla 9. Requerimientos del sistema correspondiente al requerimiento de usuario 3 .....	67
Tabla 10. Especificación del requerimiento 1 .....	69
Tabla 11. Especificación del requerimiento 2 .....	70
Tabla 12. Especificación del requerimiento 3 .....	72
Tabla 13. Especificación del requerimiento 4 .....	74
Tabla 14. Actores de Negocio .....	76
Tabla 15. Especificación del caso de uso “Consultar expediente” .....	76
Tabla 16. Especificación del caso de uso “Gestión Expediente” .....	77
Tabla 17. Especificación del caso de uso “Gestión Persona”.....	77
Tabla 18. Resultados en milisegundos de la operación “Registro de expedientes” tomados de los módulos desarrollados con los frameworks de persistencia ORM. ....	97
Tabla 19. Resultados en milisegundos para la operación “Consulta de bandeja de entrada” tomados de los módulos desarrollados con los frameworks de persistencia ORM.....	98
Tabla 20. Resultados en milisegundos para la operación “Registro de Expediente” tomados de los módulos desarrollados con los frameworks de persistencia ODM. ....	98

Tabla 21. Resultados en milisegundos para la operación “Consulta de bandeja de entrada” tomados de los módulos desarrollados con los frameworks de persistencia ODM.....	99
Tabla 22. Resultados transformados de la operación “Registro de expedientes” tomados de los módulos desarrollados con los frameworks de persistencia ORM. ....	100
Tabla 23. Resultados transformados de la operación “Consulta de bandeja de entrada” tomados de los módulos desarrollados con los frameworks de persistencia ORM.....	100
Tabla 24. Resultados transformados de la operación “Registro de Expediente” tomados de los módulos desarrollados con los frameworks de persistencia ODM. ....	101
Tabla 25. Resultados transformados de la operación “Consulta de bandeja de entrada” tomados de los módulos desarrollados con los frameworks de persistencia ODM.....	101
Tabla 26. Datos para el análisis de los resultados obtenidos para los frameworks de persistencia ORM. ....	103
Tabla 27. Prueba de hipótesis respecto al efecto de los factores principales. ....	104
Tabla 28. Prueba de hipótesis respecto a la interacción de los factores principales.....	105
Tabla 29. Tiempos de ejecución transformados de la operación “Registro de operación” para los frameworks de persistencia ORM.....	106
Tabla 30. Resultados del análisis de varianza para la operación 1 .....	106
Tabla 31. Resultados del Test de Kruskal – Wallis para la operación 1 .....	106
Tabla 32. Tiempos de ejecución en milisegundos de la operación “Consulta de Bandeja de Entrada” para los frameworks de persistencia ORM.....	107
Tabla 33. Resultados del análisis de varianza para la operación 2 .....	108
Tabla 34. Resultados del Test de Kruskal – Wallis para la operación 2 .....	108
Tabla 35. Resumen del análisis estadístico de los tiempos de ejecución de cada operación para los módulos desarrollados con frameworks de persistencia ORM .....	109
Tabla 36. Tiempo promedio de ejecución en milisegundos para SGBDR .....	111
Tabla 37. Tiempo promedio de ejecución en milisegundos para frameworks de persistencia ORM. ....	111
Tabla 38. Tiempo promedio de ejecución en milisegundos para Cantidades de datos ....	111

Tabla 39. Tiempo promedio de ejecución en milisegundos para la interacción SGBDR y framework de persistencia ORM.....	112
Tabla 40. Tiempo promedio de ejecución en milisegundos para la interacción SGBDR y Cantidad de datos. ....	112
Tabla 41. Tiempo promedio de ejecución en milisegundos para la interacción framework de persistencia y Cantidad de datos .....	113
Tabla 42. Tiempo promedio de ejecución en milisegundos para la interacción entre SGBDR, framework de persistencia y Cantidad de datos .....	114
Tabla 43. Tiempo promedio de ejecución en milisegundos para la interacción entre SGBDR, Framework de Persistencia y Cantidad de datos.....	115
Tabla 44. Datos para el análisis de los resultados obtenidos para los frameworks de persistencia ODM .....	116
Tabla 45. Prueba de hipótesis respecto a la interacción de los factores principales.....	117
Tabla 46. Prueba de hipótesis respecto a la interacción de los factores principales.....	118
Tabla 47. Tiempos de ejecución en milisegundos de la operación “Registro de Expediente” para los frameworks de persistencia ODM.....	119
Tabla 48. Resultados del análisis de varianza para la operación 1 .....	119
Tabla 49. Resultados del Test de Kruskal – Wallis para la operación 1 .....	119
Tabla 50. Tiempos de ejecución en milisegundos de la operación “Consulta de Bandeja de Entrada” para los frameworks de persistencia ODM .....	121
Tabla 51. Resultados del análisis de varianza para la operación 2 .....	121
Tabla 52. Resultados del Test de Kruskal – Wallis para la operación 2 .....	121
Tabla 53. Resumen del análisis estadístico de los tiempos de ejecución de cada operación para los módulos desarrollados con frameworks de persistencia ODM.....	122
Tabla 54. Tiempo promedio de ejecución en milisegundos para SGBD NoSQL .....	124
Tabla 55. Tiempo promedio de ejecución en milisegundos para frameworks de persistencia ODM.....	124
Tabla 56. Tiempo promedio de ejecución en milisegundos para Cantidad de datos ....	125

Tabla 57. Tiempo promedio de ejecución en milisegundos para la interacción SGBD NoSQL y Framework de persistencia ODM .....	125
Tabla 58. Tiempo promedio de ejecución en milisegundos para la interacción SGBD NoSQL y Cantidad de datos.....	126
Tabla 59. Tiempo promedio de ejecución en milisegundos para la interacción Framework de Persistencia ODM y Cantidad de datos. ....	126
Tabla 60. Tiempo promedio de ejecución en milisegundos para la interacción entre SGBD NoSQL, framework de persistencia ODM y Cantidad de datos.....	127
Tabla 61. Tiempo promedio de ejecución en milisegundos para la interacción entre SGBD NoSQL, Framework de persistencia ODM y Cantidad de datos.....	128
Tabla 62. Resumen de análisis de resultados .....	129
Tabla 63. Resultados del test de Kolmogorov – Smirnov .....	158

## ÍNDICE DE GRÁFICOS

Gráfico 1. Comparación por tipo de aplicación.....	6
Gráfico 2. Uso de los frameworks ORM.....	7
Gráfico 3. Uso de los frameworks ORM.....	8
Gráfico 4. Resultados de la encuesta sobre el módulo Web más utilizado .....	60
Gráfico 5. Resultados de la encuesta sobre las operaciones más frecuentes.....	61
Gráfico 6. Medias marginales estimadas en tiempo de ejecución del factor ORM.....	159
Gráfico 7. Medias marginales estimadas en tiempo de ejecución del factor SGBDR. ....	159
Gráfico 8. Medias marginales estimadas en tiempo de ejecución del factor Cantidad de datos.....	160
Gráfico 9. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ORM y SGBDR. ....	160
Gráfico 10. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ORM y Cantidad de datos.....	161
Gráfico 11. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: SGBDR y la Cantidad de datos.....	161
Gráfico 12. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ORM, SGBDR para la operación consulta de datos. ....	162
Gráfico 13. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ORM, SGBDR para la operación inserción de datos. ....	162
Gráfico 14. Medias marginales estimadas en tiempo de ejecución del factor ODM. ....	163
Gráfico 15. Medias marginales estimadas en tiempo de ejecución del factor SGBD NoSQL.....	163
Gráfico 16. Medias marginales estimadas en tiempo de ejecución del factor cantidad de datos.....	164
Gráfico 17. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ODM y SGBD NoSQL. ....	164

Gráfico 18. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ODM y cantidad de datos. ....	165
Gráfico 19. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: SGBD NoSQL y cantidad de datos.....	165
Gráfico 20. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ODM y SGBD NoSQL para la operación consulta de datos. ....	166
Gráfico 21. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ODM y SGBD NoSQL para la operación inserción de datos.....	166

## ÍNDICE DE FIGURAS

Figura 1. Logo de MariaDB .....	18
Figura 2. Arquitectura de MariaDB.....	20
Figura 3. Logo de PostgreSQL.....	21
Figura 4. Arquitectura de PostgreSQL .....	23
Figura 5. Logo de MongoDB .....	23
Figura 6. Arquitectura de MongoDB.....	26
Figura 7. Logo de CouchDB .....	27
Figura 8. Arquitectura de CouchDB.....	29
Figura 9. Arquitectura de JDBC, según Ricardo .....	36
Figura 10. Arquitectura de JDBC, según Coronel.....	37
Figura 11. Logo del framework Hibernate .....	38
Figura 12. Arquitectura del framework Hibernate .....	40
Figura 13. Logo del framework Doctrine .....	40
Figura 14. Logo del framework Hibernate OGM.....	42
Figura 15. Arquitectura del framework Hibernate OGM. ....	44
Figura 16. Logo del framework Doctrine ODM.....	45
Figura 17. Arquitectura del framework Doctrine y su partición en ODM. ....	46
Figura 18. Caso de uso de negocio .....	75
Figura 19. Don Consultar Expediente .....	78
Figura 20. Don Gestión Expediente .....	78
Figura 21. Don Gestión Persona.....	78
Figura 22. DUC Consultar Expediente.....	79
Figura 23. DUC Gestión Expediente .....	79
Figura 24. DUC Gestión Persona .....	80
Figura 25. DC Consultar Expediente.....	80

Figura 26. DC Mantenedor Expediente .....	81
Figura 27. DC Mantenedor Persona .....	81
Figura 28. DS Consulta Expediente .....	82
Figura 29. DS Mantenedor Expediente .....	83
Figura 30. DS Mantenedor Persona.....	83
Figura 31. Diagrama de clases de entidades del módulo desarrollado .....	84
Figura 32. Diagrama de componentes de los módulos de Trámite Documentario desarrollados con Hibernate ORM .....	85
Figura 33. Diagrama de componentes de los módulos de Trámite Documentario desarrollados con Doctrine ORM .....	86
Figura 34. Diagrama de componentes de los módulos de Trámite Documentario desarrollados con Hibernate OGM .....	87
Figura 35. Diagrama de componentes de los módulos de Trámite Documentario desarrollados con Doctrine ODM.....	88
Figura 36. Diagrama de despliegue de los módulos Web desarrollados con el framework de persistencia Hibernate ORM .....	90
Figura 37. Diagrama de despliegue de los módulos Web desarrollados con el framework de persistencia Doctrine ORM .....	91
Figura 38. Diagrama de despliegue de los módulos Web desarrollados con el framework de persistencia Hibernate OGM .....	92
Figura 39. Diagrama de despliegue de los módulos Web desarrollados con el framework de persistencia Doctrine ODM .....	93
Figura 40. Diagrama físico relacional de la base de datos. ....	94
Figura 41. Diagrama físico no relacional de la base de datos. ....	95
Figura 42. Constancia de validación de instrumento.....	148
Figura 43. Primera constancia de validación de guía de observación .....	152
Figura 44. Segunda constancia de validación de guía de observación .....	153
Figura 45. Tercera constancia de validación de guía de observación .....	154

Figura 46. Registro de expediente en el Módulo de Trámite Documentario .....	155
Figura 47. Bandeja de Entrada en el Módulo de Trámite Documentario.....	155
Figura 48. Bandeja de Recepcionados en el Módulo de Trámite Documentario.....	156
Figura 49. Bandeja de por Derivar en el Módulo de Trámite Documentario.....	156
Figura 50. Búsqueda de expedientes en el Módulo de Trámite Documentario.....	157

## **ÍNDICE DE ANEXOS**

Anexo 1: Matriz de consistencia .....	141
Anexo 2: Constancia de validación de instrumento .....	142
Anexo 3: Encuesta para desarrolladores para obtener el módulo <i>Web</i> más utilizado .....	143
Anexo 4: Constancias de validación de encuesta .....	148
Anexo 5: Constancia de validación de guía de observación de encuesta .....	149
Anexo 6: Guía de observación para la toma de tiempos de ejecución del módulo <i>Web</i> de Trámite Documentario.....	150
Anexo 7: Constancia de Validación de Guía de Observación .....	152
Anexo 8: Capturas de pantalla del módulo de Trámite Documentario desarrollado.....	155
Anexo 9: Test de Kolmogorov-Smirnov para los tiempos de ejecución .....	158
Anexo 10: Gráficos obtenidos del análisis lineal univariante para los <i>frameworks</i> de persistencia ORM .....	159
Anexo 11: Gráficos obtenidos del análisis lineal univariante para los <i>framewoks</i> de persistencia ODM .....	163

## RESUMEN

La presente investigación se basa en el análisis de comparación de los tiempos de ejecución que emitieron las aplicaciones desarrolladas con *frameworks* de persistencia ORM y ODM de manera conjunta con los gestores de base de datos relacionales y no relacionales, respectivamente, y de este modo determinar los *frameworks* de persistencia ORM y ODM que emitieron los menores tiempos de ejecución.

El módulo que se desarrolló para realizar la comparación fue el módulo de Trámite Documentario, que fue seleccionado como el módulo *Web* más utilizado de la ciudad de Piura, a través de una encuesta aplicada a instituciones y entidades de dicha ciudad. Del módulo se desarrollaron cuatro réplicas con las tecnologías ORM y cuatro con las tecnologías ODM, tecnologías que fueron seleccionadas bajo varios criterios, además se utilizaron las operaciones “Registro de Expediente” y “Consulta de la Bandeja de Entrada de Expedientes” para realizar la experimentación, obtener los tiempos de ejecución y el análisis comparativo.

Los resultados obtenidos en la presente investigación concluyeron que las tecnologías que mejor se combinan para ofrecer los mejores tiempos de ejecución tanto en inserción como en consulta de datos son: Hibernate ORM para las bases de datos relacionales y Doctrine ODM para las bases de datos no relacionales.

**Palabras Clave:** *frameworks*, persistencia, tiempo de ejecución, módulo *Web*, ANOVA, Trámite Documentario, base de datos relacional, base de datos no relacional.

## **ABSTRACT**

The present investigation is based on the analysis of comparison of execution times that issued the applications developed with ORM and ODM persistence frameworks jointly with relational and non-relational database managers, respectively, and in this way determine the ORM and ODM persistence frameworks that issued the shortest execution times.

The module that was developed to make the comparison was the module of Documentary Processing, which was selected as the most used Web module in the city of Piura, through a survey applied to institutions and entities of said city. Four replicas with ORM technologies and four with ODM technologies were developed from the module, technologies that were selected under several criteria, in addition operations "Registering Files" and "consultation Inbox Records" were used for experimentation, obtain the execution times and the comparative analysis.

The results obtained in the present investigation concluded that the technologies that best combine to offer the best execution times in both insertion and data query are: Hibernate ORM for relational databases and Doctrine ODM for non-relational databases.

**Keywords:** frameworks, persistence, execution time, Web module, ANOVA, Documentary procedure, relational database, non-relational database.

## INTRODUCCIÓN

En el campo del desarrollo de *software* aparecen casi a diario nuevas tecnologías que ayudan a agilizar este proceso, es decir, ayudan a disminuir la cantidad de tiempo que un desarrollador invierte en crear una aplicación. Una de estas tecnologías es el *framework* de persistencia, más conocido como ORM, que funciona como una capa intermediaria entre la base de datos y la aplicación. La tarea principal de un ORM es el mapeo de datos, es decir, la forma de transformar los objetos a datos primitivos que se guardan en la base de datos, y viceversa. Las mismas funciones que cumplen los ORM son realizadas por los ODM, pero al contrario de estas, operan con las bases de datos NoSQL orientadas a documentos. Se debe tener en cuenta que los ODM son relativamente nuevos en el mercado del desarrollo del *software*.

Como sucede con todas las tecnologías existentes en el campo de la informática, el hacer uso de ellas trae consigo una serie de ventajas pero también desventajas, los ORM y los ODM no son ajeno a ello, y pues una de sus desventajas es el aumento del tiempo de ejecución, provocando un efecto negativo en el rendimiento de la aplicación, entonces, es una tarea obligatoria saber cuál el efecto que marcará el uso de uno de estos *frameworks* dentro de la aplicación, así se tomará la decisión de usarlos o no. Para saber que se tomó una buena decisión se debe tener un buen respaldo de la elección realizada, y nada mejor que tener a disposición una serie de estudios que ofrezcan los mejores *frameworks* de persistencia teniendo como factor principal el tiempo de ejecución.

En la presente investigación se hace uso de los términos Sistema Gestor de Base de Datos Relacional y Sistema Gestor de Base de Datos NoSQL para distinguir de esta manera los gestores de base de datos relacionales de los gestores de bases de datos no relacionales orientadas a documentos, a partir de este párrafo en adelante se utiliza la reducción de términos utilizando SGBDR y SGBD NoSQL, esto con el propósito de simplificar la redacción en la investigación.

El presente estudio realizó un análisis comparativo de los *frameworks* de persistencia ORM Hibernate y Doctrine basados en los lenguajes de programación Java y PHP, respectivamente, haciendo uso de los SGBDR MariaDB y PostgreSQL, y para el caso de ODM se hizo elección de los *frameworks* de persistencia homólogos a los ORM: Hibernate OGM y Doctrine ODM, que trabajarán en conjunto con los SGBD NoSQL MongoDB y CouchDB.

Para realizar el estudio comparativo se desarrollaron módulos *Web* del Trámite Documentario, haciendo uso de las tecnologías ORM y ODM que se mencionaron anteriormente; para luego utilizar estos módulos en la realización de pruebas de inserción y consulta de datos y por último determinar a través de un análisis estadístico el ORM y ODM que ofrecen los menores tiempos de ejecución.

La presente investigación se divide en los siguientes capítulos:

**En el capítulo 1,** se presenta la problemática y la justificación en la que se basa la investigación, el planteamiento del objetivo y la hipótesis general, así como también de los objetivos e hipótesis específicas, por último se identifica las variables dependiente e independiente de la investigación.

**El capítulo 2,** tiene como objetivo mostrar el marco teórico, el cual contiene los antecedentes al presente estudio, la definición de los términos que se deben conocer y algo que no debería faltar, los conceptos de las tecnologías que se utilizan en esta investigación.

**En el capítulo 3,** se muestra la metodología que se empleó para el presente estudio, entre lo más resaltante las técnicas e instrumentos para la recolección de datos, el procesamiento de la encuesta realizada, narración de cómo se realizó la experimentación, además de una sección donde se estudia el desarrollo del módulo de Trámite Documentario.

**En el capítulo 4,** se realiza la presentación, el análisis y la discusión de los resultados, además de la validación de la hipótesis general y de las específicas que se plantearon en la presente investigación.

**En el capítulo 5,** se muestran las conclusiones a las que se llegó con los resultados obtenidos. Además, las recomendaciones para futuros estudios.

**CAPÍTULO I**

**EL PROBLEMA DE LA INVESTIGACIÓN**

## **1.1. PLANTEAMIENTO DEL PROBLEMA**

Los desarrolladores de las entidades manejan tiempos cortos para hacer entrega de las aplicaciones solicitadas, y se preguntan si es óptimo hacer uso de las nuevas tecnologías que proponen agilizar el desarrollo de *software*, tal como los *frameworks* de persistencia ORM y ODM. Estas tecnologías funcionan como una capa que abstrae a la base de datos durante el desarrollo, ayudando al desarrollador a concentrarse más en la lógica de la aplicación que en los detalles de la programación de acceso a datos.

Algo que se debe tener en cuenta es que la utilización de los *frameworks* de persistencia puede traer como consecuencia el incremento del tiempo de ejecución de la aplicación, por lo que deben ser utilizados donde el factor tiempo de ejecución no sea un requerimiento imprescindible. Pero, los desarrolladores de las entidades no cuentan con el tiempo debido para realizar estudios, evaluar el efecto de los *frameworks* de persistencia en el tiempo de ejecución de las aplicaciones y determinar si es óptimo su uso. Además, existe una carencia de estudios de comparación de *frameworks* de persistencia donde se tome como principal factor el tiempo de ejecución, por lo que le es difícil a los desarrolladores optar por uno u otro.

Entonces, es un problema para los desarrolladores hacer uso de los *frameworks* de persistencia sin tener en cuenta cuál será el efecto en la aplicación a desarrollar, que se tenga la duda que tal vez exista otro u otros *frameworks* que afecten en menor medida el tiempo de ejecución y que maximicen la calidad de la aplicación. Hewlett-Packard (1987), desarrolló un conjunto de factores de calidad del *software* al cual se le conoce con el acrónimo de FURPS. Entre los factores de calidad se encuentra el rendimiento, y los criterios con los que cuenta son velocidad de procesamiento, consumo de recursos, rendimiento efectivo total, eficacia y tiempo de respuesta. Siendo entonces muy importante la medición del tiempo de respuesta para evaluar la calidad del *software*, este factor fue estudiado en la presente investigación a través de la comparación de los tiempos de las operaciones de los módulos *Web* desarrollados.

## **1.2. FORMULACIÓN DEL PROBLEMA**

¿Existen diferencias significativas en el tiempo de ejecución al hacer uso de los *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP?

### **1.3. JUSTIFICACIÓN DEL ESTUDIO**

La programación para el acceso de datos es una de las actividades más laboriosas dentro del desarrollo de *software*, y hacer uso de un *framework* de persistencia agilizaría el desarrollo y el mantenimiento de la aplicación, pero se debe tener en cuenta que el rendimiento de la aplicación puede ser afectado, pues al ser una capa que opera entre la base de datos y la aplicación, ralentizaría el funcionamiento. Es una necesidad conocer cuál será el efecto en el tiempo de ejecución de una aplicación al usar algún *framework* de persistencia, tal vez se opta por no usarse porque el factor tiempo de ejecución será uno de los factores imprescindibles de la aplicación.

Los desarrolladores de las entidades al no contar con el tiempo adecuado para realizar pruebas de comparación y justificar el uso de los *frameworks* de persistencia, deben recurrir a estudios previos que ayuden a tomar una buena decisión. En la actualidad existe una carencia de estudios comparativos que tomen como prioridad el tiempo de ejecución, lo que se puede encontrar son los comparativos de tipo cualitativo en los cuales las características de los *frameworks* es lo fundamental al momento de la comparación.

El presente estudio de comparación ofrecerá a los desarrolladores el ORM y ODM que ofrecen los menores tiempos de ejecución, además las mejores opciones de combinación entre estas tecnologías y los SGBDR y SGBD NoSQL, respectivamente. El conocimiento que resulte será de gran ayuda a los desarrolladores de las entidades que cuentan con plazos cortos para la entrega de las aplicaciones solicitadas, que deben cumplir con los requerimientos de *software* solicitados, y desean hacer uso de tecnologías que no afectan de manera significativa la calidad de la aplicación a desarrollar. Con ayuda del presente estudio los desarrolladores tendrán a disposición información valiosa a tomar en cuenta al momento de elegir un *framework* de persistencia.

Para llevar a cabo el estudio comparativo se desarrollará un módulo de tipo *Web*. Uno de los motivos de su elección es que necesitan menos requisitos que las aplicaciones de escritorio para empezar hacer uso de ellas. Además, al parecer las aplicaciones *Web* son las más desarrolladas actualmente y lo respalda la encuesta global “*Java Tools and Technologies Landscape Report 2016*”, realizada a 2040 profesionales

Java por la compañía *ZeroTurnaround* a través de RebelLabs, pues emite como uno de sus resultados que el 67% de los encuestados trabajan desarrollando con este tipo de aplicaciones. **Los lenguajes de programación que se elegirán son PHP y Java**, pues son los más usados para el desarrollo de aplicaciones, además, cuentan con *frameworks* de persistencia que son independientes de otros.

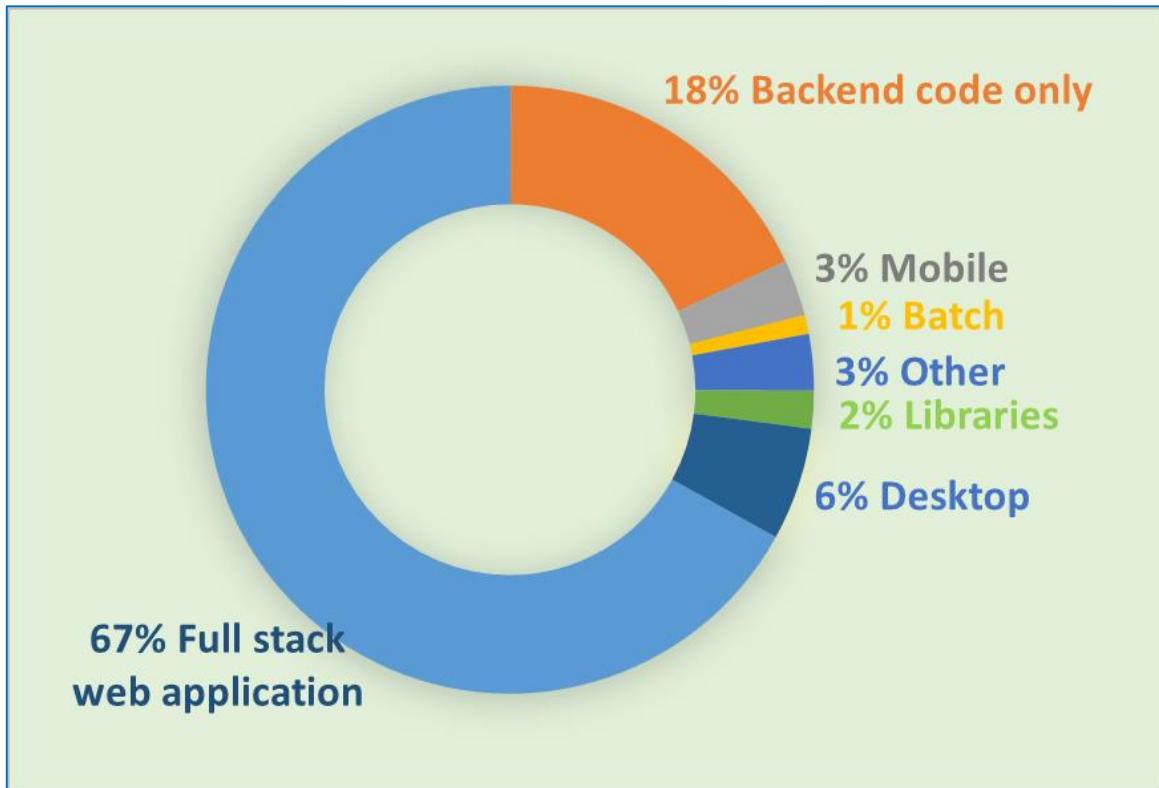


Gráfico 1. Comparación por tipo de aplicación  
Fuente: “Java Tools and Technologies Landscape Report 2016”

Los *frameworks* de persistencia ORM elegidos para el estudio comparativo son: **Hibernate** y **Doctrine**, y para el caso de ODM: **Hibernate OGM** y **Doctrine ODM**, los motivos que llevaron a tomar esa decisión son: los *frameworks* son independientes de otros *frameworks* de desarrollo (no forman parte de otra tecnología), son los más solicitados en el mercado, otro punto muy importante es que se encuentran basados en los lenguajes de programación Java y PHP, y por último cuentan con una licencia gratuita. Entonces, para ayudar en la elección de los *frameworks* se tomó en cuenta las estadísticas acerca de qué tan usadas son estas tecnologías, la primera es la encuesta global “*Java Tools and Technologies Landscape for 2014*” realizada a 2164 profesionales Java, la cual emite como parte de sus resultados que el *framework* de

persistencia ORM más solicitado (al menos por la comunidad Java) es *Hibernate* con un 67.5% del total, es decir, un gran porcentaje.

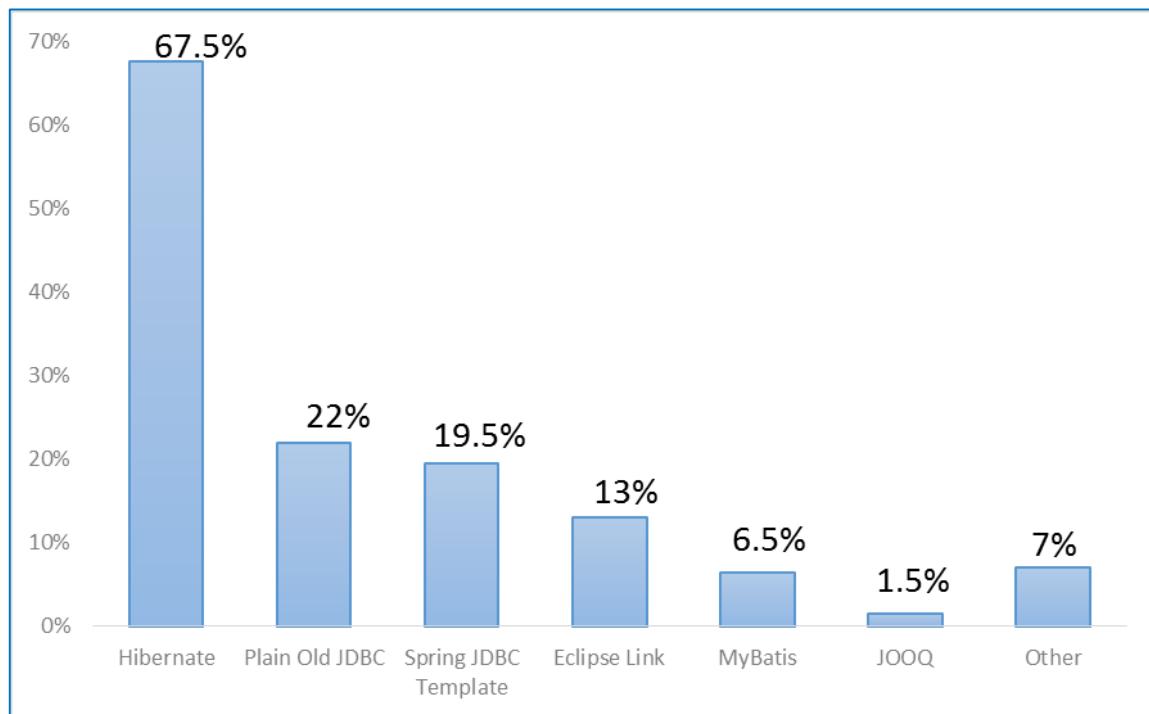
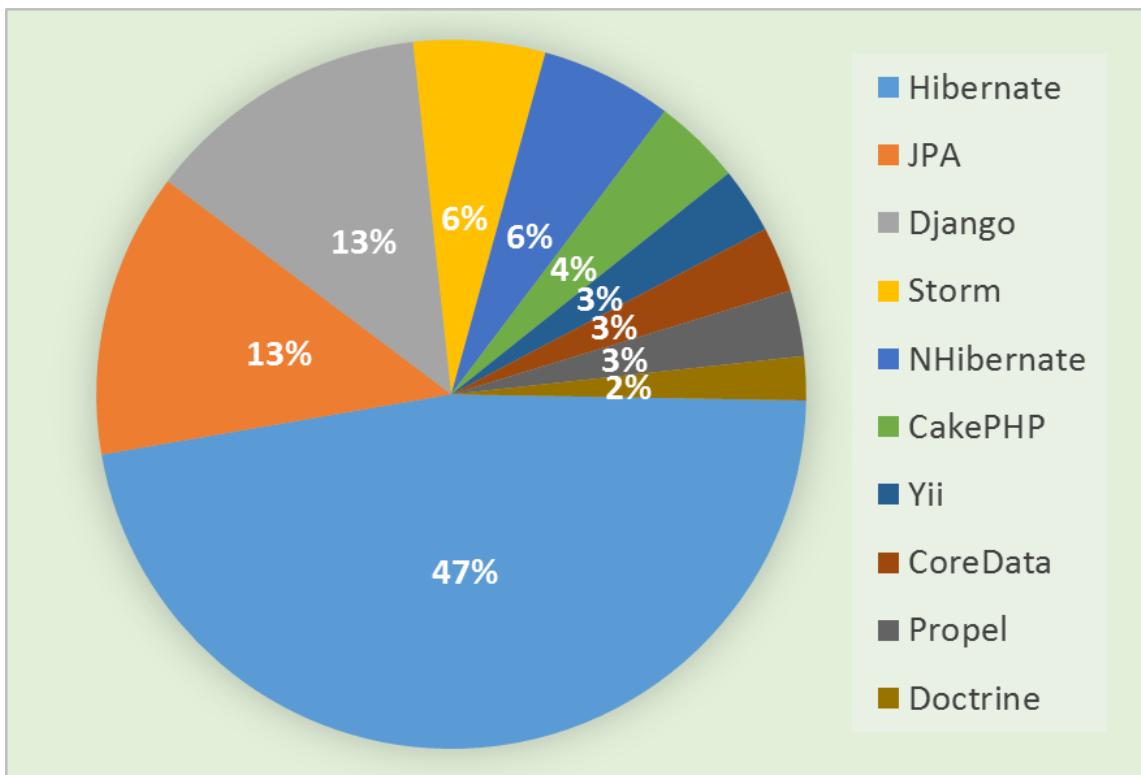


Gráfico 2. Uso de los frameworks ORM  
Fuente: “Java Tools and Technologies Landscape for 2014”

Y como segundo informe se presenta las tendencias emitidas por *TrendySkills*, las cuales indican que el framework de persistencia más solicitado por las empresas en países como los Estados Unidos, España y Gran Bretaña, es *Hibernate* con una representación del 47%.



*Gráfico 3. Uso de los frameworks ORM*  
Fuente: “Java Tools and Technologies Landscape for 2014”

Estas mismas tendencias brindan como resultado que los *frameworks* de persistencia más usados y escritos para PHP son: *CakePHP*, *Yii*, *Propel* y *Doctrine*, de los cuales el único que es independiente de otro *framework* es *Doctrine*, además este cuenta con una buena documentación. Para el caso de ODM no se encontró estadística alguna para justificar que sean los más usados, así que se tomó en cuenta los *frameworks* homólogos a los ORM elegidos.

Para el manejo de la información los equipos de desarrollo de las entidades hacen uso en mayor parte de las bases de datos relacionales, que se basan en el álgebra relacional, cálculo relacional y, además, hacen uso de principios de diseño como la regla ACID, pero por lo general tienden a disminuir su rendimiento cuando se habla del tratamiento de gran cantidad de registros. En cambio los modelos de base de datos NoSQL cumplen estas expectativas de rendimiento, y para lograrlo hacen uso de características particulares: no son relacionales, son distribuidas y horizontalmente escalables, pero usualmente sacrifican algunas cuestiones como la normalización, las transacciones y la consistencia de los datos.

Se usará para el estudio comparativo los SGBDR de licencia *Open Source* más utilizados en el mercado, y que cumplan con los siguientes requisitos: compatibilidad con los lenguajes de programación Java y PHP, funcionamiento en las distintos sistemas operativos (multiplataforma), soporte para transacciones de tipo ACID y documentación extensa; siendo los SGBDR que se ajustan a los requisitos: MariaDB y PostgreSQL. Si bien es cierto MariaDB no es uno de los más utilizados en la actualidad, pero es derivado de MySQL por lo que tienen bastante compatibilidad (<https://mariadb.org/about/>), además, promete un mayor rendimiento frente a su antecesor, siendo una gran alternativa a tomar en cuenta. Otro punto a tener cuenta es que MySQL ahora pertenece a la empresa Oracle por lo que cuenta con doble licencia. Para el caso de los SGBD NoSQL se necesitan que cumplan los mismos requisitos mencionados para los SGBDR, siendo MongoDB y CouchDB los que se ajustan a ellos.

## 1.4. OBJETIVOS DE LA INVESTIGACIÓN

### 1.4.1. Objetivo general

Estudiar comparativamente el tiempo de ejecución entre *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP haciendo uso de un módulo *Web*.

### 1.4.2. Objetivos específicos

- ❖ Identificar el módulo *Web* más usado por las entidades de la ciudad de Piura en la actualidad.
- ❖ Desarrollar el módulo *Web* haciendo uso de los *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP.
- ❖ Determinar el *framework* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP que ofrecen el menor tiempo de ejecución.

**CAPÍTULO II**

**MARCO TEÓRICO**

## 2.1. ANTECEDENTES DEL ESTUDIO

- ❖ **Payá, M. (2010)**, realizó un proyecto para analizar y usar *frameworks* de persistencia de Java, con el objetivo de describir su funcionamiento e identificar las características más relevantes de cada una de las alternativas estudiadas. Se necesitó desarrollar una aplicación que consistió en un portal *Web* para uso cotidiano, que será accesible a través de la intranet en una pequeña empresa. El gestor de base de datos que se usó fue MySQL. Se llegó a la conclusión que la programación con Hibernate es mucho más sencilla y llevadera, ya que libera de tareas tan ásperas como el encapsulamiento y desencapsulamiento de los objetos, y pone a disposición una API completísima y fácil de usar. Sin embargo, después de concluir con la implantación del portal, ha quedado patente que los accesos con JDBC son siempre más rápidos que los gestionados por un *framework* como Hibernate, tal y como se había predicho en el estudio. Este estudio se relaciona con la investigación planteada debido a dos razones, la primera es la comparación entre distintas tecnologías, y la segunda es el desarrollo de una aplicación *Web* para la aplicación de pruebas. En el estudio la elección del mejor ORM se realiza a través de cualidades, no de datos cuantitativos como se piensa realizar en esta investigación.
- ❖ **Gavillanes, Glacio (2016)**, realizó un estudio comparativo de la productividad entre los *frameworks* de persistencia en Java: Hibernate y MyBatis, con el objetivo de elegir el que mejor se adapte al sistema informático de evaluación docente del IPEC (Instituto de Posgrado y Educación Continua). Las variables tentativas que se usaron para analizar la productividad de los *frameworks* fueron: simplicidad, desarrollo, producto y mapeo de objetos. Como una de las conclusiones a la que se llegó es que la utilización del *framework* Hibernate, permite mejorar la productividad que se destina a la abstracción de la capa de datos con respecto al *framework* MyBatis. Este estudio se relaciona con la investigación planteada, en que ambos proponen comparar para concluir que ORM se adecua mejor a una aplicación, pero en el caso del antecedente la mayoría de índices que se tomaron en cuenta son del tipo cuantitativo, pero se resalta que el índice cuantitativo “tiempo de respuesta de consultas” fue tomado en cuenta.

- ❖ **Leisalu, O. (2009)**, elaboró la tesis de licenciatura denominada “*Comparative Evaluation of two PHP Persistence Frameworks*”, la cual tiene como objetivo comparar dos *frameworks* de persistencia: Doctrine y DomAr en términos de funcionalidad, facilidad de uso y el rendimiento. Para realizar el estudio el autor eligió una base de datos preexistente y construyó un modelo de objetos de trabajo. La base de datos elegida se llama Sakila. Se hizo uso de Jounneau, que es un punto de referencia de tamaño, una prueba la velocidad de la selección de registros de base de datos, y para eso consta de 12 pruebas que no incluyen inserciones, actualizaciones ni eliminaciones. Se concluyó que el tiempo de ejecución de Doctrine crece más cuando se selecciona más filas de la base de datos y que además, utiliza aproximadamente el doble de memoria que DomAr. Otra conclusión es que los resultados de las pruebas de referencia de Doctrina y DomAr podrían haber estado sesgados debido a las características de la referencia Jounneau.
- ❖ **González, V. (2009)**, elaboró la tesis “*Frameworks para Mapeo Objeto Relacional: Un Análisis Comparativo*”, la cual tuvo por objetivo proporcionar una evaluación que compare los *frameworks* ORM más utilizados en el mercado, para guiar en la selección del ORM más conveniente en un proyecto en particular. Lo primero que se hizo fue estudiar las características principales de los ORM Hibérnate e Ibatis. Para luego realizar el estudio comparativo en el cual uno de los factores elegidos fue el rendimiento. Se llegó a la conclusión que la simplicidad es la mayor ventaja de Ibatis sobre otras herramientas de mapeo Objeto Relacional y, que es recomendable utilizarla cuando se tienen sistemas con muchas sentencias SQL y llamadas a procedimientos almacenados.

## 2.2. GLOSARIO DE TÉRMINOS BÁSICOS

### 2.2.1. Sistema Gestor de Base de Datos

Es un conjunto de datos interrelacionados y con herramientas computacionales específicas para acceder a dichos datos. Su lugar de almacenamiento se denomina base de datos, pues es aquella que contiene información relevante de una empresa u organización. El objetivo principal de un Sistema Gestor Bases de Datos es el de almacenar y recuperar la

información de la organización o empresa de una forma práctica y eficiente. (Silberschatz, A., Korth, H. & Sudarshan, S., 2002).

#### **2.2.2. Base de datos relacional**

Según Date (2001), una base de datos relacional es una base de datos que los usuarios perciben como un conjunto de variables de relación – es decir, *varrels* – o, de manera más informal, tablas. Un sistema relacional es aquel que maneja bases de datos y operaciones relacionales en dichas bases de datos, incluyendo las operaciones restringir, proyectar y juntar en particular. Todas estas operaciones, y otras parecidas, se encuentran en el nivel de conjunto.

#### **2.2.3. Base de datos NoSQL**

Grupo ecléctico y cada vez más familiar de los sistemas de gestión de datos no relacionales; donde las bases de datos no se construyen principalmente en tablas, y por lo general no utilizan SQL para la manipulación de datos. Los sistemas de gestión de bases de datos NoSQL son útiles cuando se trabaja con una gran cantidad de datos y cuando la naturaleza de estos no requiere un modelo relacional. (Moniruzzaman & Hossain, 2013).

#### **2.2.4. Persistencia**

Según Carneiro (2008), la persistencia de los objetos, es el hecho de persistir la información de un objeto de forma permanente (guardar), pero también se refiere a recuperar y volver a ver la información introducida o guardada para poder ser utilizada de nuevo. La persistencia no es más que el mecanismo que se utiliza para persistir la información de un tipo determinado que puede serializar, guardar, etc. La persistencia permite al programador almacenar, transferir, recuperar el estado de los objetos a través de diferentes técnicas que se utilizan, como son: serialización, motores de persistencia, bases de datos orientadas a objetos, etc.

#### **2.2.5. Framework de persistencia**

Los *frameworks* de persistencia según Suarez (2011), actúan como capa intermedia entre la capa de negocios y la capa de datos, facilitando el

almacenamiento de la información proveniente de la aplicación en base de datos, especialmente relacionales. Según Carneiro (2008), un *framework* de persistencia es un conjunto de tipos de propósito general, reutilizable y extensible, que proporciona funcionalidad para dar soporte a los objetos persistentes.

#### **2.2.6. ORM**

Object Relational Mapping (Mapeo Relacional de Objetos). Es una herramienta de programación que consiste en la transformación de las tablas de una base de datos, en una serie de entidades que simplifiquen las tareas básicas de acceso a los datos para el programador (Amador & Johana, 2013).

#### **2.2.7. ODM**

Object Document Mapper (Mapeo Objeto Documento). Al igual que el ORM se encarga de transformar objetos, pero en este caso a documentos (lo equivalente a un registro en una base de datos relacional) de una base de datos NoSQL (Fernández, 2013).

#### **2.2.8. Registro**

Según Talledo (2015), el registro (también llamado tupla) representa un objeto único de datos implícitamente estructurados en una tabla. Los documentos dentro de una base de datos orientada a documentos son similar de algún modo a los registros, pero son menos rígidos.

#### **2.2.9. Consulta**

Una consulta es un objeto de la base de datos en forma de tabla que extrae datos de otra tabla, de otra consulta o de un conjunto de tablas y consultas, y los muestra al usuario (Aguilera et al., 2014).

#### **2.2.10. Tiempo de ejecución**

Tiempo de ejecución es definido por el autor a cargo del estudio como la diferencia de tiempo para que se muestre un resultado al realizar una consulta

(lectura, actualización, eliminación, búsqueda) a la base de datos, desde la interfaz gráfica del módulo.

#### **2.2.11. Módulo Web**

Un módulo *Web* es la unidad de recursos *Web* desplegable y utilizable más pequeña de un programa informático. Un módulo *Web* contiene componentes *Web* y archivos de contenido *Web* estáticos, como imágenes, que se denominan recursos *Web* (Eric et al., 2014).

#### **2.2.12. Trámite Documentario**

El Sistema de Trámite Documentario es un aplicativo de uso interno que tiene como fin el seguimiento de la documentación generada en la Institución y/o recepcionada en cada una de sus mesas de parte. Los documentos, ya sean de origen interno o externo que necesiten circular por cualquier Área de la Institución, son registrados en el Sistema, en donde se le aplica los movimientos de envío, recepción y archivo según corresponda. (INGEMMET, Oficina de Sistemas de Información – Manual de usuario. Diciembre 2011).

#### **2.2.13. FURPS**

Modelo desarrollado por Hewlett-Packard en el año 1987. En el cual se desarrollan un conjunto de factores de calidad de software, bajo el acrónimo de FURPS: funcionalidad (*Functionality*), usabilidad (*Usability*), confiabilidad (*Reliability*), desempeño (*Performance*) y capacidad de soporte (*Supportability*).

#### **2.2.14. TrendySkills**

Sitio *Web* que emite representaciones cuantitativas acerca de las tendencias de empleo que los empleadores buscan en la industria de TI (Tecnologías de la Información). Se inició en el año 2012 como parte de un proyecto de tesis de Licenciatura en Informática y Comunicaciones del estudiante Barzokas Vassilios con la supervisión del Dr. Petalidis Nicholaos que tenía como título “*Data mining on labor market of IT sector*”.

## **2.2.15. Álgebra relacional**

Es un lenguaje abstracto de consulta procedimental. Consta de un conjunto de operaciones que toman como entrada una o dos relaciones y producen como resultado una nueva relación.

## **2.2.16. Regla ACID**

Conjunto de propiedades que se aplican específicamente a las transacciones de bases de datos. Acrónimo de *Atomicity*, *Consistency*, *Isolation* y *Durability*. (Aislamiento, Consistencia, Atomicidad y Durabilidad en español).

## **2.2.17. Normalización**

La normalización de bases de datos es un proceso que consiste en analizar las relaciones para satisfacer los requisitos cada vez más estrictos que conducen a progresivamente a mejores agrupaciones o formas normales superiores.

## **2.2.18. CurrentTimeMillis**

Función del lenguaje de Programación Java que devuelve la hora actual en milisegundos; para ello calcula la diferencia, medida en milisegundos, entre la hora actual y la medianoche del 1 de enero de 1970 UTC.

## **2.2.19. MicroTime**

Función del lenguaje de Programación PHP que devuelve la fecha Unix actual con microsegundos. Esta función sólo está disponible en sistemas operativos que soportan la llamada al sistema de gettimeofday().

## **2.2.20. Clase Math**

La clase *Math* del lenguaje Java contiene métodos para realizar operaciones numéricas básicas, como la exponencial elemental, el logaritmo, la raíz cuadrada y las funciones trigonométricas.

## **2.3. MARCO CONCEPTUAL**

### **2.3.1. Bases de datos relacionales**

Los Sistemas manejadores de base de datos, en la actualidad son un elemento muy importante al tenerse en cuenta en la creación de productos *software*, ya que en ellos se almacena y administra una gran cantidad de datos, es por eso que debe elegirse una base de datos que permita tener integridad en los datos, administrar y que la información esté disponible cuando sea requerida, según Nevado (2010) define a un Sistema Gestor de Bases de Datos como *el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos*.

#### **2.3.1.1. Elementos de una base de datos**

Los elementos de una base de datos son cuatro y estos son los que la componen en su totalidad, tal como menciona Carmona, (2014). *Los cuatro elementos fundamentales que componen una base de datos son: Tablas, Vistas o Consultas, Formularios e Informes o reports.*

- *Tablas: es el único elemento cuyo fin es almacenar información, ya que los demás solo la administran, una tabla está compuesta a su vez por campos y registros.*
- *Vistas: las vistas no son más que consultas especializadas para la búsqueda.*
- *Formulario: es un objeto diseñado para introducir, visualizar y modificar los datos de las tablas.*
- *Informes: permiten mostrar los registros contenidos en las tablas que componen la base de datos de forma atractiva a fin de mostrar la información en pantalla.*

#### **2.3.1.2. Normalización del esquema relacional**

Oppel (2009), hace mención que *La normalización es una técnica para producir un conjunto de correlaciones (datos*

*representados de manera lógica en un formato bidimensional que emplee filas y columnas) que posea cierto grupo de propiedades.*

El esquema relacional debe ser normalizado para evitar inconsistencia de datos, duplicidad de los mismos, esto se ve reflejado en lo que define Gabillaud (2009), *cuando se define el esquema relacional para responder a todas las necesidades de los usuarios, es necesario normalizarlo para evitar redundancias de información y estructuras no conformes con el modelo relacional.*

Según Nevado (2010) *el proceso de normalización se hace necesario para:*

- *Evitar la redundancia de los datos y las inconsistencias.*
- *Evitar la incapacidad de almacenar ciertos datos.*
- *Evitar las ambigüedades y pérdida de información.*
- *Evitar problemas de actualización (anomalías de inserción, borrado y modificación) de los datos en las tablas.*
- *Proteger la integridad de los datos.*

### 2.3.2. MariaDB



*Figura 1. Logo de MariaDB  
Fuente: Twitter Oficial de MariaDB*

MariaDB al estar basada en MySQL, la cual es una de las bases de datos que muchas empresas y organizaciones utilizan; añade las mismas funcionalidades, algunas de las cualidades según Ben, F. (2012) posee son:

- *Costo: es de código abierto y libre para usar (e incluso modificar) sin pagar por ello.*
- *Rendimiento: es rápida.*
- *Confianza: al estar basada en MySQL, es utilizado por algunas de las más importantes y prestigiosas organizaciones.*

### 2.3.2.1.Características

MariaDB es sin duda una de las bases de datos relacionales de código abierto que ha tenido una gran aceptación, al estar basada en MySQL y desarrollada por los mismos creadores, en principio posee la mismas características de MySQL, sin embargo agrega características que lo hacen diferente, tal como menciona Wals, L & Labayén, F. (2016), La compatibilidad entre ambas tecnologías es tal que permite a los usuarios migrar de *software* de un modo transparente. MariaDB incorpora nuevas características respecto de MySQL.

Las características que posee MariaDB según Wals, L & Labayén, F. (2016), son:

- *Relativo a la seguridad. Los desarrolladores de MariaDB producen sus propias actualizaciones de seguridad, que incluyen un análisis exhaustivo de las actualizaciones de MySQL a los efectos de mejorarlas en caso de ser necesario. Cuando se descubre una vulnerabilidad crítica, los desarrolladores modifican el código fuente y lo distribuyen inmediatamente para eliminarla.*
- *Compatibilidad. MariaDB contiene los comandos, interfaces, bibliotecas y APIs utilizados por MySQL; a la vez que incorpora nuevas características no contempladas en su antecesor MySQL.*

Así mismo Wals, L. & Labayén, F. (2016), agrega que; *MariaDB ofrece tres versiones distintas; una estable, ideal para sistemas de producción, una de prueba (testing) y otra para desarrolladores. Cada uno de las versiones es supervisado por una comunidad autorizada que brinda distintos niveles de soporte.*

### 2.3.2.2.Arquitectura

La arquitectura de MariaDB tiene como núcleo base a MySQL, esto se debe a que los desarrolladores que dieron vida a este sistema de base de datos fueron los mismos que crearon MySQL.

Según Razzoli (2014) la arquitectura de MariaDB es la siguiente:

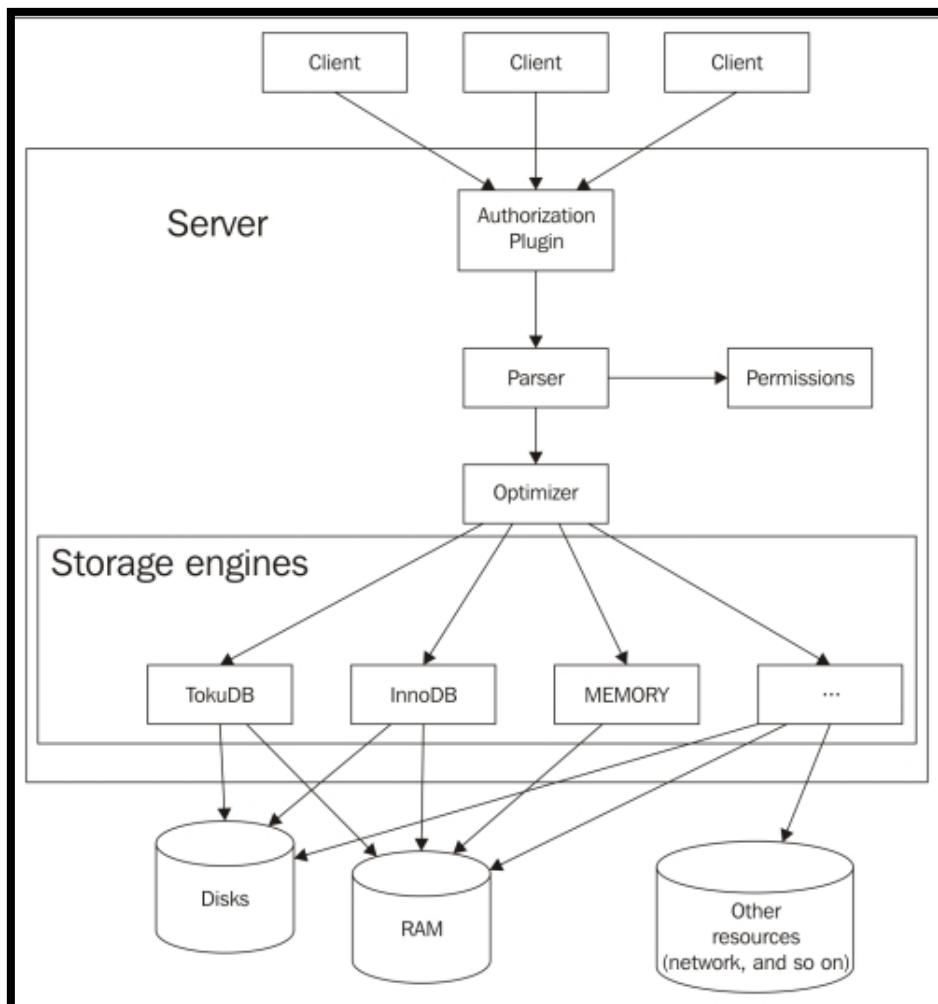
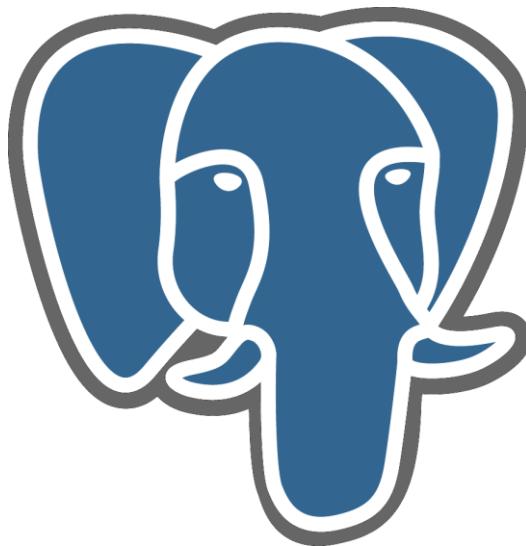


Figura 2. Arquitectura de MariaDB  
Fuente: Razzoli (2014).

### 2.3.3. PostgreSQL



*Figura 3. Logo de PostgreSQL  
Fuente: Página Oficial de PostgreSQL*

PostgreSQL al igual que MySQL es un sistema manejador de base de datos relacional de código abierto, muy utilizado actualmente debido a su robustez y por el almacenamiento de datos, además es uno de los que tiene mayor aceptación en el mercado; hay muchos conceptos que lo definen, entre los más destacados se proporcionan los siguientes:

Matthew, N. & Stones, R. (2005) “*Es un DBMS que incorpora el modelo relacional para sus bases de datos y es compatible con el lenguaje de consulta SQL estándar*”. [It is a DBMS that incorporates the relational model for its databases and supports the SQL standard query language].

Aguilera & Tineo & Cadenas (2009), definen que *PostgreSQL es un manejador de bases de datos relacionales, de tipo cliente / servidor*.

Smith, G. (2010) por su parte agrega “*se caracteriza por tener un rico conjunto de características y versiones de software muy estables*”. [It's known for having a rich feature set and very stable software releases].

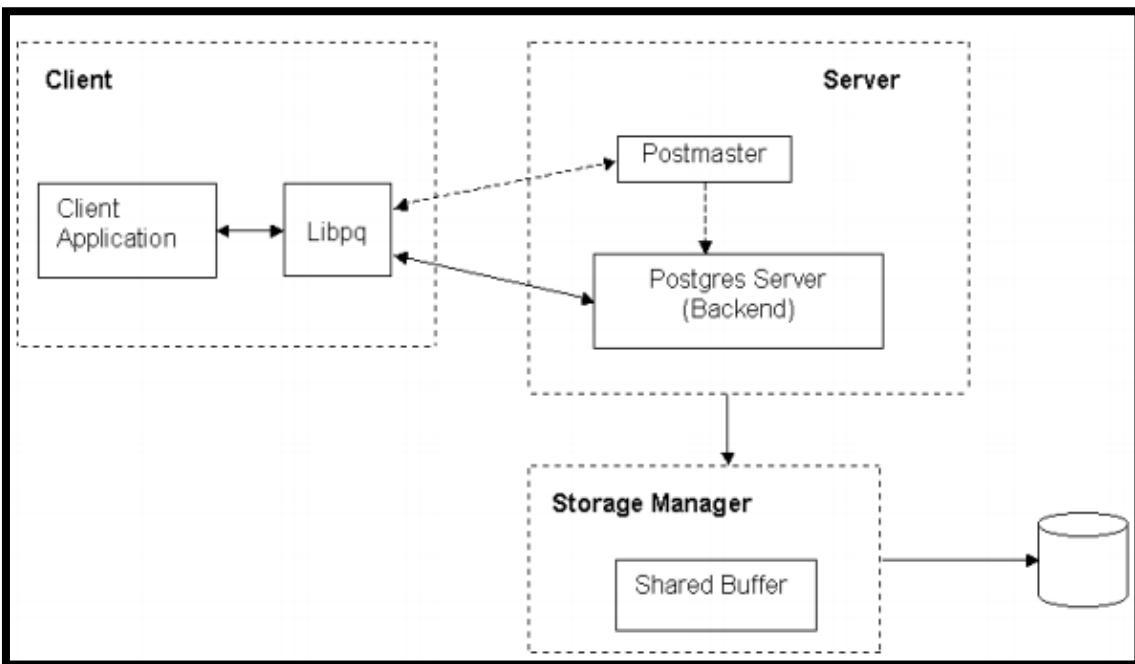
### **2.3.3.1.Características**

*PostgreSQL posee un sin número de características entre las que más se destacan son las define Cáliz, D. (2007).*

- *Contiene las normas de sintaxis de SQL92 y SQL99, en caso de que exista diferencias de sintaxis estas son relacionadas a rasgos únicos de PostgreSQL.*
- *Protege los datos con las transacciones que procesa y coordina a varios usuarios al mismo tiempo, ya que el modelo de transacción está basado en el control de concurrencia de multiversión (MVCC) que da a PostgreSQL un mejor funcionamiento.*
- *Soporta diferentes tipos de datos como fecha, monetarios, elementos gráficos, datos para la comunicación de redes, cadenas de bits, datos booleanos, y datos explícitamente creados para conexiones de red.*

### **2.3.3.2.Arquitectura**

*PostgreSQL implementa una arquitectura cliente-servidor para proveer acceso multiusuario, en este se pueden diferenciar tres grandes subsistemas: el cliente, el servidor y el gestor de almacenamiento los cuales en conjunto permiten el funcionamiento del sistema (Geschwwinde & Shonig, 2001 citado por Aguilera & Tineo & Cadenas (2009)).*



*Figura 4. Arquitectura de PostgreSQL*  
Fuente: Luna & Aguayo & Rossodivita (2007).

#### 2.3.4. MongoDB



*Figura 5. Logo de MongoDB*  
Fuente: Página Oficial de MongoDB

Las bases de datos NoSQL han sido creadas como una alternativa a las bases de datos relacionales o SQL, debido a su mayor capacidad para manejar grandes cantidades de datos, estas bases de datos están orientadas a: documentos, objetos, grafos, columnas y clave-valor; entre las bases de datos orientadas a documentos se encuentra MongoDB; es una de las bases de datos que actualmente es muy utilizada debido a que puede almacenar gran cantidad de datos.

Se puede decir que MongoDB es un tema nuevo debido a que no se encuentra muy distribuido la concepción de lo que son las bases NoSQL.

Chorodow, K. (2013) sostiene que “MongoDB es una base de datos documental, no una relacional” [MongoDB is a document-oriented database, not a relational one]; también define que “Una base de datos documental reemplaza el concepto de una "fila" con un modelo más flexible, el "documento". Al permitir que los documentos y las matrices que integran, el enfoque orientado al documento hace posible la representación de las relaciones jerárquicas complejas con un único registro. Esto encaja de forma natural en la forma en que los desarrolladores de lenguas modernas orientadas a objetos piensan acerca de sus datos” [A document-oriented database replaces the concept of a “row” with a more flexible model, the “document.” By allowing embedded documents and arrays, the document-oriented approach makes it possible to represent complex hierarchical relationships with a single record. This fits naturally into the way developers in modern object-oriented languages think about their data].

Un singular concepto que recibe MongoDB es el que aporta Plugge, E. & Membrey, P. and Hawkins, T. (2013) “MongoDB (derivado de la palabra gigantescos) es una raza relativamente nueva de la base de datos que no tiene concepto de tablas, esquemas, SQL, o filas. No tiene transacciones, compatibilidad ACID, joins, claves foráneas, o muchas de las otras características que poseen las bases de datos relacionales”[ MongoDB (derived from the word humongous) is a relatively new breed of database that has no concept of tables, schemas, SQL, or rows. It doesn’t have transactions, ACID compliance, joins, foreign keys, or many of the other features].

#### **2.3.4.1.Características**

Chorodow, K. (2013) sostiene que “MongoDB pretende ser una base de datos de uso general, por lo que aparte de crear, leer, actualizar y borrar datos, proporciona una lista cada vez mayor de características únicas:” [MongoDB is intended to be a general-purpose database, so aside from creating, reading, updating, and deleting data, it provides an ever-growing list of unique features]

Gaona, P. & Sandoval, E. (2013) definen que las características principales que posee MongoDB son:

- **Consultas Ad hoc:** MongoDB soporta la búsqueda por campos, consultas de rangos y expresiones regulares.
- **Indexación:** Cualquier campo en un documento de MongoDB puede ser indexado, al igual que es posible hacer índices secundarios. El concepto de índices en MongoDB es similar a los encontrados en base de datos relacionales.
- **Replicación:** MongoDB soporta el tipo de replicación maestro-esclavo. El maestro puede ejecutar comandos de lectura y escritura. El esclavo puede copiar los datos del maestro y sólo se puede usar para lectura o para copia de seguridad, pero no se pueden realizar escrituras. El esclavo tiene la habilidad de poder elegir un nuevo maestro en caso del que se caiga el servicio con el maestro actual.
- **Balanceo de carga:** MongoDB se puede escalar de forma horizontal usando el concepto de “shard”. El desarrollador elige una llave shard, la cual determina cómo serán distribuidos los datos en una colección. Los datos son divididos en rangos y distribuidos a través de múltiples shard. Un shard es un maestro con uno o más esclavos. MongoDB tiene la capacidad de ejecutarse en múltiple servidores, balanceando la carga y duplicando los datos para poder mantener el sistema funcionando en caso que exista un fallo de hardware.
- **Almacenamiento de archivos:** MongoDB puede ser utilizado con un sistema de archivos, tomando la ventaja de la capacidad que tiene MongoDB para el balanceo de carga y la replicación de datos utilizando múltiples servidores para el almacenamiento de archivos
- **Agregación:** La función MapReduce se utiliza en MongoDB para el procesamiento por lotes de datos y operaciones de agregación. Esta función permite que los usuarios puedan obtener el tipo de

resultado que se obtiene cuando se utiliza el comando SQL “group-by”.

#### 2.3.4.2.Arquitectura

Según Gaona, P. & Sandoval, E. (2013) la terminología básica de MongoDB es la siguiente:

*En MongoDB, cada registro o conjunto de datos se denomina documento. Los documentos se pueden agrupar en colecciones, las cuales se podría decir que son el equivalente a las tablas en una base de datos relacional (sólo que las colecciones pueden almacenar documentos con diferentes formatos, en lugar de estar sometidos a un esquema fijo). Se pueden crear índices para algunos atributos de los documentos, de modo que MongoDB mantendrá una estructura interna eficiente para el acceso a la información por los contenidos de estos atributos.*

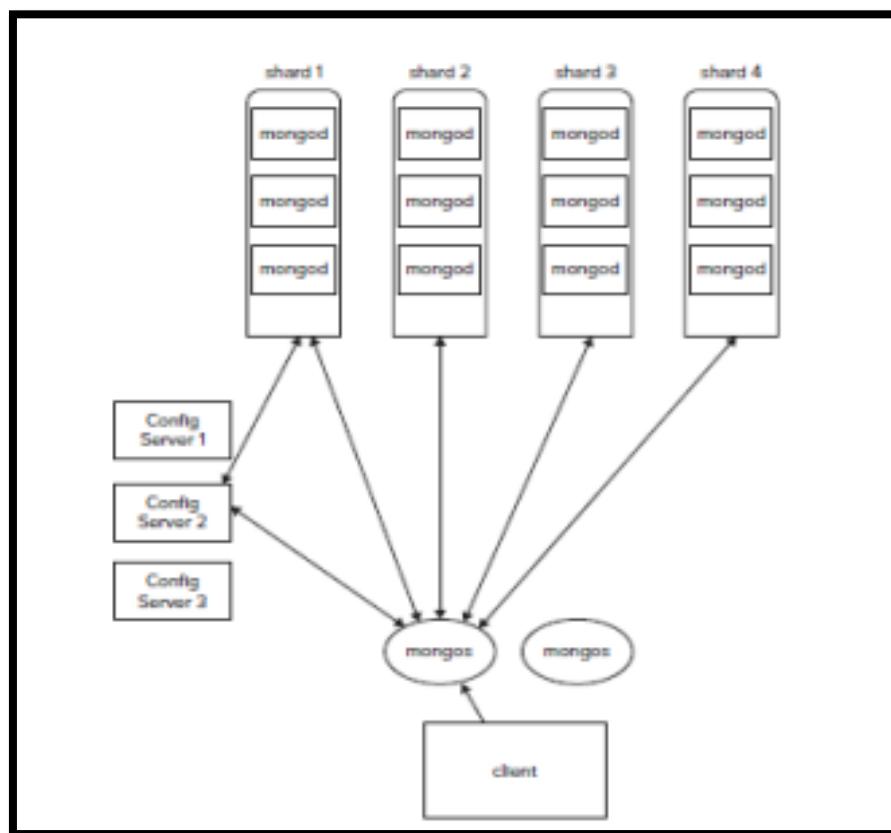


Figura 6. Arquitectura de MongoDB  
Fuente: Gaona, P. & Sandoval, E. (2013)

### 2.3.5. CouchDB

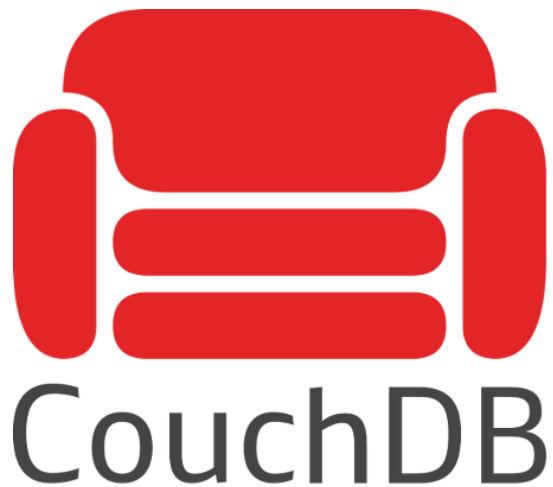


Figura 7. Logo de CouchDB  
Fuente: Página Oficial de CouchDB

CouchDB es una de las bases de datos orientadas a documentos que han tomado un gran relevancia en la actualidad, “CouchDB server es una base de datos distribuida, orientada a documentos que forma parte de las bases de datos NoSQL. CouchDB server es una base de datos persistente que aprovecha una capa de almacenamiento en caché de memoria RAM integrada, lo que le permite soporte rápido para crear, almacenar, actualizar además de las operaciones de recuperación” [*Couchbase Server is a distributed, document-based database that is part of the NoSQL database movement. Couchbase Server is a persistent database that leverages an integrated RAM caching layer, enabling it to support very fast create, store, update, and retrieval operations*] (Brown, MC., 2012).

#### 2.3.5.1. Características

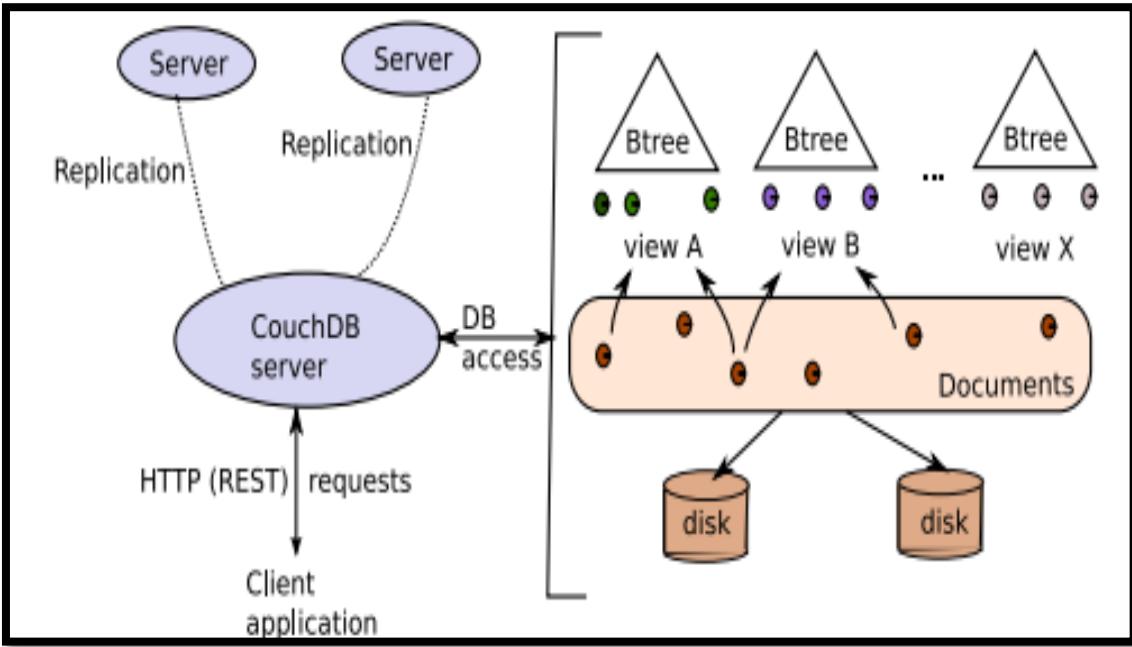
Gaona, P. & Sandoval, E. (2013) sostiene que apache CouchDB está escrita en Erlang, un lenguaje específicamente creado por Ericsson para programar sistemas robustos, distribuidos y multihilo. Couch, es un acrónimo para Clúster Of Unreliable Commodity Hardware, o “clúster no fiable de hardware común”.

Así mismo Gaona, P. & Sandoval, E. (2013) menciona que las características principales de CouchDB son:

- *CouchDB cumple completamente con las propiedades ACID, esto permite reconstruir un documento en cualquier momento de su vida. Al nunca sobrescribir ningún dato, el sistema completo mantiene un control de versiones perfecto, parecido a un control de versiones de software como subversión o CVS.*
- *CouchDB utiliza un modelo de presentación de datos llamado “Views”.*
- *CouchDB utiliza replicación maestro-maestro hecha de forma peer to peer entre todos los servidores. Es una funcionalidad extremadamente avanzada que permite tener todas las instancias de CouchDB en sincronismo, para leer y escribir desde cualquiera de ellas.*
- *El sistema de replicación es incremental. Debido a la forma en que CouchDB guarda todas las versiones de cada uno de sus documentos, solamente se envían los cambios hechos a los documentos cuando se hace la replicación*
- *Los datos son totalmente consistentes debido a la adherencia a los estándares ACID y a la facilidad de crear réplicas de la base de datos que automáticamente pueden ser escritos y leídos.*

### **2.3.5.2.Arquitectura**

La arquitectura de CouchDB está basada en una instancia cliente-servidor, *una instancia COUCHDB se basa en una arquitectura Cliente / Servidor, donde el servidor COUCHDB gestiona las peticiones enviadas por el cliente, procesa las solicitudes en su base de datos y envía una respuesta.* [A COUCHDB instance is based on a Client/Server architecture, where the COUCHDB server handles requests sent by the client, processes the requests on its database(s), and sends an answer]. (Abiteboul, Manolescu, Rigaux, Rousset & Senellart, 2011)



*Figura 8. Arquitectura de CouchDB  
Fuente: Abiteboul, et al., 2011.*

### 2.3.6. Persistencia de objetos

Hay un amplio campo de conceptos que definen lo que es la persistencia, a continuación se presentan los más importantes:

Granados, (2015), *hace mención que la persistencia de objetos se refiere a que, una vez concluida la ejecución del programa que los creo, el estado de estos objetos siga estando disponible para ser usado en el futuro.*

Freire, (2008) menciona en definitiva la persistencia de objetos *es la capacidad que tienen los objetos de sobrevivir al proceso que los creó; permitiendo al programador almacenar, transferir, y recuperar el estado de los objetos.*

Para García, (2015) la persistencia de objetos se refiere *a la capacidad de mantener la información de un objeto, de forma que pueda ser recuperada y modificada cuando sea necesario. Agregando que la persistencia no es ni una capacidad ni una propiedad de la POO, no tiene nada que ver con el paradigma en sí, solo es el mecanismo que se usa para mantener la información.*

Aporta también que según el punto de vista de persistencia, los objetos se clasifican:

- *Transitorios: son almacenados en memoria y su tiempo de vida viene determinado por el tiempo que dure la ejecución del programa.*
- *Persistentes: pueden almacenarse en un medio secundario para su posterior reconstrucción y utilización. Su existencia es independiente del proceso que los creo.*

### **Serialización**

García, (2015) define a la serialización de objetos como la *técnica utilizada habitualmente para almacenar, transferir y recuperar el estado de los objetos. Permite convertir un objeto a una secuencia de bits que pueda ser posteriormente restaurada para reconstruir el objeto original, agregando que no solo es utilizada para implementar la persistencia, sino también para enviar objetos a través de una red, para almacenarlos en una base de datos, etc.*

UNAM, (2007) define que serialización *permite escribir objetos a archivos con una sola instrucción con lo cual quedan grabados hasta que se decida eliminarlos o modificarlos. Agregando que también permite recuperar los objetos grabados en archivos.*

Fernández, (2013) aporta que la serialización *es un proceso mediante el cual una estructura de datos es codificada de un modo específico para su almacenamiento, que puede ser físicamente en un fichero, una base de datos o un buffer de memoria. Adiciona que el propósito de este proceso, además del almacenamiento, suele ser el transporte de datos a través de una red o, simplemente para crear una copia exacta de un objeto determinado.*

#### **2.3.7. Framework (marco de trabajo)**

Un marco de trabajo o en inglés *framework*, es indispensable para la creación de un producto *software*, ya que permite minimizar líneas de código, así mismo tiempo de programación, debido a que integra librerías que poseen

la característica de ser configurables, un *framework* forma parte del esqueleto de una aplicación.

García (2010), define a un *framework Web* como una extensión de un lenguaje mediante una o más jerarquías de clases que implementan una funcionalidad y que (opcionalmente) pueden ser extendidas, agrega también que es una estructura *software* compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación.

Según Perales (2014), un *framework* facilita la programación debido a que muchas funciones que debería ser escritas a mano en muchas líneas ya vienen implementadas en sus librerías.

Ruiz (2011), define a un *framework* como conjunto de componentes de *software* que los programadores pueden usar, extender, o personalizar para una determinada aplicación; así mismo hace mención que con los *frameworks*, los desarrolladores no tienen que empezar desde cero cada vez que construyen una aplicación. La flexibilidad inherente permite la creación rápida y el desarrollo de soluciones de *software* en un entorno de negocio que evoluciona constantemente.

Orellana (2013) menciona que un *framework* provee de:

- Librerías
- Componentes
- Documentación
- Metodología de uso

También menciona que un *framework Web* difiere de un *framework* de escritorio (*desktop*) ya que se centran en el ambiente para lo cual están dirigidos.

### **2.3.7.1.Arquitectura de un *Framework***

Según Robson (2003) referenciado por Orellana (2013), que un *framework* se compone de:

- Controladores
- Configuración
- Vistas

Ganesan (2013), difiere con lo anterior en que un *framework Web* posee solamente dos componentes principales; [A *Web framework* has two main components. One is the actual concept of the *Web framework* where the application coding resides and the other is the navigation code. The conceptual model of the *Web* application contains annotations, which are placed as placeholders for the developers to fill in the *Web* application code, and the navigational model is the control hand offs that happen within a *Web* application. For example, MVC *framework* defines navigational model within each controller where the controller is the manager handing off control to the next controller, essentially the next URL and hence allowing navigation within a site. These two components are the backbone of a WAF]. Los *frameworks Web* tienen dos componentes principales. Uno es el concepto real del *framework Web* en el que reside la codificación de la aplicación y el otro es el código de navegación. El modelo conceptual de la aplicación Web contiene anotaciones, que se colocan como marcadores de posición para que los desarrolladores puedan llenar el código de la aplicación Web y el modelo de navegación es el control que ocurre en una aplicación Web. Por ejemplo, el *framework* MVC define el modelo de navegación dentro de cada controlador en el que el controlador es el administrador que entrega el control al controlador siguiente, esencialmente la siguiente URL y, por lo tanto, permite la navegación dentro de un sitio. Estos dos componentes son la columna vertebral de un WAF.

Para Carreño (2013), otra forma en que se compone un *framework* es; que posee dos principales elementos y son el kernel y los puntos flexibles, *el primero se refiere a los puntos inmutables, aquellas características constantes para cada instancia del framework, y el segundo se refiere a las características flexibles que se pueden extender o adaptar para dar particularidad a las aplicaciones.*

### 2.3.7.2. Tipos de *frameworks*

Hay varios criterios desde el punto de vista de autores; que definen la calificación de los *frameworks*, entre ellos se destacan:

García (2008) menciona que los *frameworks* se clasifican en: *orientados a la interfaz de usuario, orientados a aplicaciones de publicación de documentos, orientados a la parte de control de eventos.*

Parsons, Rashid, Speck, Telea (1999) citado por Carreño (2013), hace mención que los frameworks se clasifican según en:

- *Frameworks de aplicación: Proveen la funcionalidad básica de una aplicación tratándose generalmente de interfaces gráficas y lógica de administración del sistema. A partir de este, se crea cada aplicación especializada. Sin embargo, estos frameworks no contienen elementos del dominio mismo de las aplicaciones.*
- *Frameworks de dominio específico: Modelan los objetos de un dominio específico y proveen la lógica genérica de una aplicación en este dominio. De este modo, una aplicación puede ser creada configurando los objetos del dominio y extendiendo la lógica de aplicación genérica. Por ser muy específicos y requerir un conocimiento muy preciso de los dominios son los más costosos de desarrollar.*

Así mismo Mnkandla (2009), citado por Carreño (2013), define que los *frameworks*, según las características de personalización se clasifican en tres categorías:

- *Caja blanca: los puntos de entrada se presentan como clases y métodos abstractos que deben ser implementados para la extensión del framework, se tiene acceso al código fuente y se permite reutilizar la funcionalidad encapsulada en sus clases mediante herencia y reescritura de métodos.*
- *Caja negra: Ocultan su estructura interna, los usuarios solo conocen una descripción general del framework y sus puntos flexibles. Se extienden mediante composición y delegación. El framework tiene definido una serie de interfaces que deben implementar los componentes que extienden el framework.*
- *Caja gris: Define una forma intermedia de reutilización, el framework presenta características tanto de caja blanca como de caja negra.*

#### **2.3.7.3. Framework de persistencia**

Los frameworks de persistencia según Suarez (2011), *actúan como capa intermedia entre la capa de negocios y la capa de datos, facilitando el almacenamiento de la información proveniente de la aplicación en base de datos, especialmente relacionales.*

#### **2.3.8. ORM**

El mapeo objeto-relacional, es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos. (Yanes, O. & Gracia, H., 2011).

#### **Componentes**

Una solución ORM está formada por cuatro partes:

- Una API que posibilite la realización de operaciones de creación, actualización y borrado sobre objetos de clases persistentes.
- Un lenguaje o API para poder especificar consultas sobre dichas clases.
- Una opción para especificar mapeo de metadatos.
- Una técnica para que la implementación del ORM pueda llevar a cabo búsquedas, asociaciones u otras funciones de optimización. [2].

### **2.3.9. Java Database Connectivity (JDBC)**

Para la conexión a base de datos desde una aplicación java se hace uso de JDBC, con esta API es posible conectar la aplicación con los distintos sistemas gestores de base de datos.

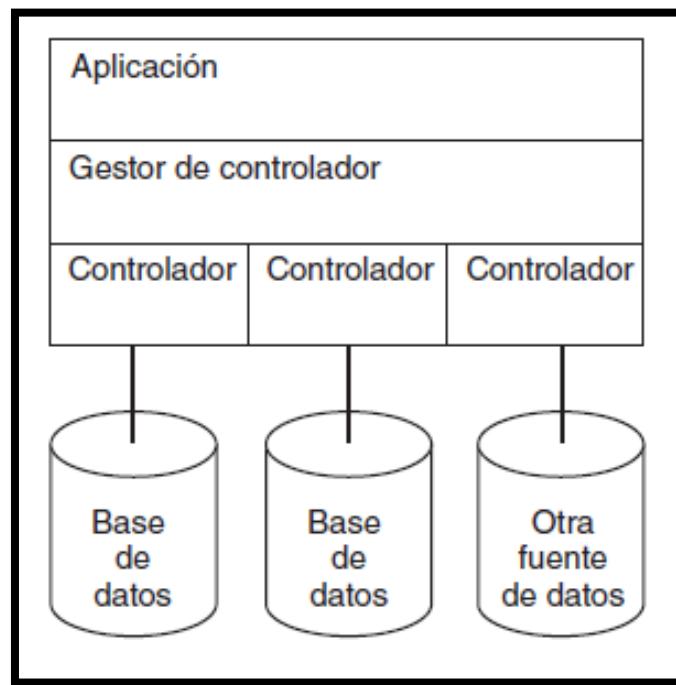
Según Oppel (2010), JDBC (Java Database Connectivity) *es una API, modelada a partir de ODBC, para conectar aplicaciones de Java a una amplia variedad de productos de DBMS relacionales.*

Ricardo (2004) menciona que JDBC al igual que ODBC (*conectividad de base de datos abierta*), proporcionan formas estándar de integrar código SQL y lenguajes de propósito general al proporcionar una interfaz común, y según esta estandarización permite que las aplicaciones accedan a múltiples bases de datos usando diferentes DBMS.

Para complementar la definición de lo que es JDBC, Coronel (2011) define a JDBC como una interfaz de programación de aplicación que permite a un programa de java interactuar con una amplia variedad de fuentes de datos. Asimismo que envía el código SQL al servidor de la base de datos y procesar el conjunto resultado.

### 2.3.9.1.Arquitectura de JDBC

Según Ricardo (2004), requiere de cuatro componentes: la aplicación, gestor de controlador, controlador y fuente de datos (normalmente una base de datos), tal como se muestra en la Figura 9.



*Figura 9. Arquitectura de JDBC, según Ricardo  
Fuente: Ricardo (2004).*

En la Figura 10, proporcionada por Coronel (2011) se muestra la arquitectura de JDBC.

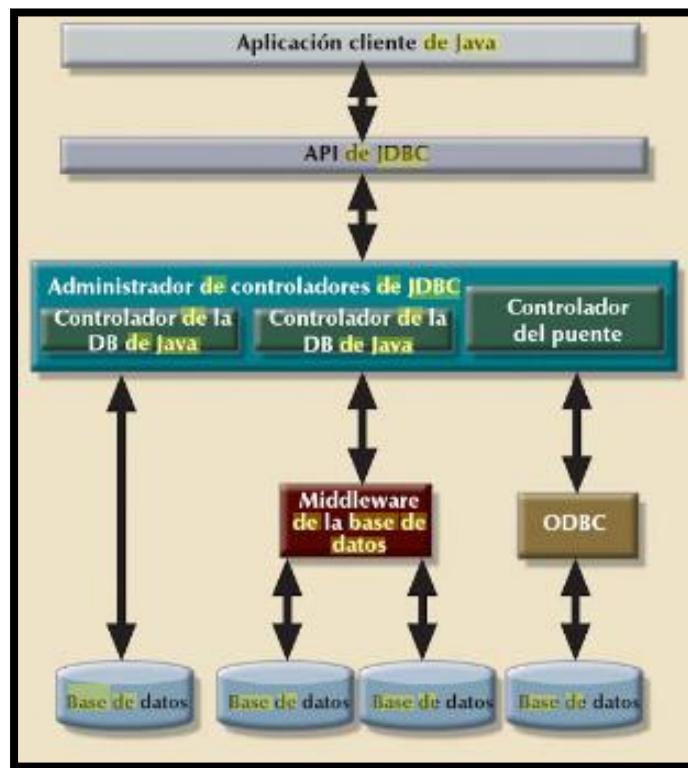


Figura 10. Arquitectura de JDBC, según Coronel  
Fuente: Coronel (2011).

### 2.3.9.2. Características de JDBC

Según OPPEL (2010) JDBC presenta las siguientes características:

- *SQL incrustado para Java: El programador de Java codifica instrucciones de SQL como variables de cadena; las cadenas son transferidas a los métodos de Java, y un procesador de SQL incrustado traduce el SQL de Java a llamadas de JDBC.*
- *Asignación directa de tablas de RDBMS a clases de Java: Los resultados de las llamadas a SQL se ubican automáticamente en variables de clases de Java. Después, el programador en Java puede operar sobre los datos devueltos como si fueran objetos nativos de Java.*

### 2.3.9.3. Ventajas de JDBC

Según Coronel (2011) las ventajas que posee son:

- *Permite el acceso directo al servidor de base de datos*
- *Proporciona una vía para conectarse a bases de datos por medio de un controlador ODBC.*

### 2.3.10. Hibernate



Figura 11. Logo del framework Hibernate  
Fuente: Página Oficial de Hibernate

Hibernate ORM permite a los desarrolladores escribir más fácilmente aplicaciones cuyos datos sobreviven al proceso de solicitud. Como un marco de objeto/Relational Mapping (ORM), Hibernate tiene que ver con la persistencia de datos que se aplica a las bases de datos relacionales (a través de JDBC).

Hibernate es una herramienta Java que facilita el mapeo de atributos entre cualquier tipo de Base de Datos relacional (modelo relacional) y el modelo de objetos (orientación a objetos) de una aplicación usando archivos XML para declarar las relaciones.

Las razones que hacen que el uso de Hibernate para el desarrollo de aplicaciones sea muy importante son:

- ❖ **Simplicidad y flexibilidad:** necesita un único fichero de configuración en tiempo de ejecución y un documento de mapeo para cada aplicación. Este fichero puede ser el estándar de Java (extensión properties) o un fichero XML. También se tiene la alternativa de realizar la configuración de forma programática. El uso de *frameworks* de persistencia, tales como EJBs hace que la aplicación dependa del *framework*, Hibernate no crea esa

dependencia adicional. Los objetos persistentes en la aplicación no tienen que heredar de una clase de Hibernate u obedecer a una semántica específica, tampoco necesita un contenedor para funcionar.

- ❖ **Completo:** ofrece todas las características de orientación a objetos, incluyendo la herencia, tipos de usuario y las colecciones. Además, también proporciona una capa de abstracción SQL llamada HQL. Las sentencias HQL son compiladas por el *framework* de Hibernate y cacheadas para su posible reutilización.
- ❖ **Prestaciones:** uno de las grandes confusiones que aparecen al utilizar este tipo de *frameworks* es creer que las prestaciones de la aplicación se ven muy mermadas. Este no es el caso de Hibernate. La clave en este tipo de situaciones es si se realizan el número mínimo de consultas a la base de datos. Muchos *frameworks* de persistencia actualizan los datos de los objetos incluso cuando no ha cambiado su estado. Hibernate sólo lo hace si el estado de los objetos ha cambiado. El cacheado de objetos juega un papel importante en la mejora de las prestaciones de la aplicación. Hibernate acepta distintos productos de cacheado, tanto de código libre como comercial. (Rivas Greciano, 2013).

La arquitectura que presenta Hibernate se basa en inicio a su clase org.hibernate.cfg.Configuration, que utiliza las propiedades y los documentos de asignación (archivos .properties, .cfg.xml y hbm.xml) para crear org.hibernate.SessionFactory, un objeto seguro para subprocessos que se instancia una vez y proporciona una factoría para obtener sesiones (org.hibernate.Session). Las instancias de sesión se usan para ejecutar transacciones (JTA) y / o consultas. (Anghel Leonard, 2013)

En la Figura 12 se muestra la arquitectura del *framework* Hibernate.

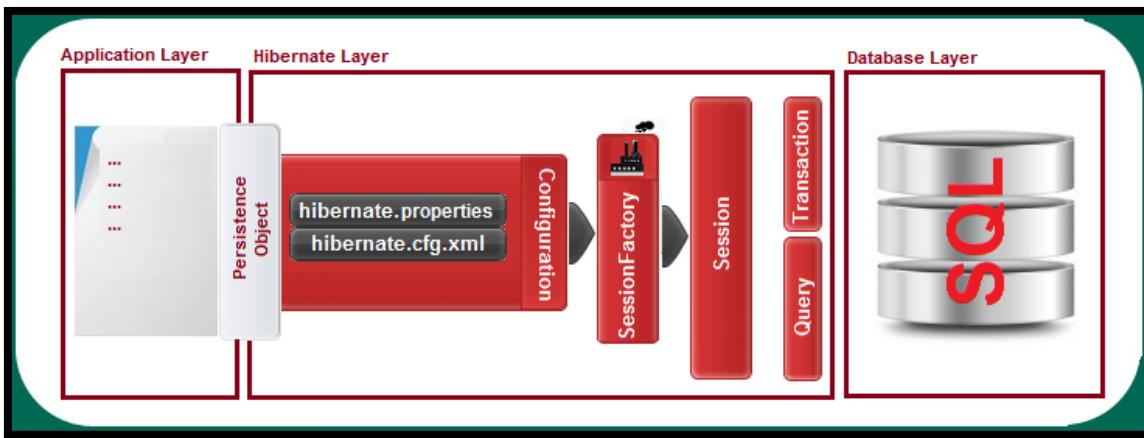


Figura 12. Arquitectura del framework Hibernate  
Fuente: Anghel Leonard (2013)

### 2.3.11. Doctrine



Figura 13. Logo del framework Doctrine  
Fuente: Página Oficial de Doctrine

Doctrine es un mapeador relacional de objetos (ORM) para PHP 5.3.3+ que proporciona persistencia transparente para objetos PHP. Utiliza el patrón Data Mapper en el corazón, con el objetivo de una separación completa de su dominio/lógica de negocios de la persistencia en un sistema de administración de bases de datos relacionales.

El beneficio de Doctrine para el programador es la capacidad de enfocarse en la lógica de negocios orientada a objetos y preocuparse por la persistencia solo como un problema secundario. Esto no significa que Doctrine 2 minimice la persistencia, sin embargo, creemos que existen beneficios considerables para la programación orientada a objetos si la persistencia y las entidades se mantienen separadas.

## Entidades

Las entidades son objetos PHP que se pueden identificar a lo largo de muchas solicitudes mediante un identificador único o clave principal. Estas clases no necesitan extender ninguna clase base o interfaz abstracta. Una clase de entidad no debe ser definitiva ni contener métodos finales. Además, no debe implementar clonar ni despertar ni hacerlo de forma segura.

Una entidad contiene propiedades persistentes. Una propiedad persistente es una variable de instancia de la entidad que se guarda y recupera de la base de datos mediante las capacidades de mapeo de datos de Doctrine.

## Arquitectura

### ❖ Usando un asignador relacional de objetos

Como el término ORM ya insinúa, Doctrine 2 tiene como objetivo simplificar la traducción entre las filas de la base de datos y el modelo de objetos PHP. El caso de uso primario para Doctrine es, por lo tanto, aplicaciones que utilizan el Paradigma de Programación Orientada a Objetos. Para las aplicaciones que no funcionan principalmente con objetos, Doctrine 2 no se adapta muy bien.

### ❖ Paquetes de Doctrine 2

Doctrine 2 se divide en tres paquetes principales.

- Common
- DBAL (incluye Common)
- ORM (incluye DBAL+Common)

La base de código de Doctrine se divide en estos paquetes por varias razones y están destinadas a:

- Hacer las cosas más sostenibles y desacopladas
- Permiten usar el código en Doctrine Common sin ORM o DBAL
- Te permiten usar el DBAL sin el ORM

#### ❖ El paquete Common

El paquete Common contiene componentes altamente reutilizables que no tienen dependencias más allá del paquete en sí (y PHP, por supuesto). El espacio de nombres raíz del paquete Común es Doctrine\Common.

#### ❖ El paquete DBAL

El paquete DBAL contiene una capa de abstracción de base de datos mejorada en la parte superior de PDO, pero no está fuertemente vinculada a PDO. El objetivo de esta capa es proporcionar una única API que vincule la mayoría de las diferencias entre los diferentes proveedores de SGBD. El espacio de nombres raíz del paquete DBAL es Doctrine\DBAL.

#### ❖ El paquete ORM

El paquete ORM contiene el kit de herramientas de asignación relacional de objetos que proporciona persistencia relacional transparente para objetos PHP simples. El espacio de nombres raíz del paquete ORM es Doctrine\ORM.

### 2.3.12. ODM

Herramienta que correlaciona la estructura de las bases de datos NoSQL orientado a documentos con los objetos. Un ODM también abstrae las operaciones habituales y provee de servicios públicos que pueden simplificar mucho las tareas comunes, al igual que un ORM, pero a diferencia de este lo realiza con documentos.

### 2.3.13. Hibernate OGM



*Figura 14. Logo del framework Hibernate OGM  
Fuente: Blog “Java Experience”*

“Proporciona soporte para Java Persistence (JPA) para las soluciones NoSQL. Se vuelve a utilizar el motor de Hibernate ORM, pero persisten entidades en un almacén de datos NoSQL en lugar de una base de datos relacional”. (<http://hibernate.org/ogm/>)

Hibernate OGM (Object Grid Mapper), que ofrece un motor Java Persistence API (JPA) completo para almacenar datos en bases NoSQL. Este proyecto proporciona un impulso real a los desarrolladores de Java que buscan explotar las bases NoSQL, ya que proporciona una interfaz común, el conocido modelo de programación JPA, como interfaz para varios enfoques NoSQL. Hibernate OGM se basa en el motor de Hibernate ORM, reutiliza Java Persistence Query Language (JP-QL) como interfaz para consultar datos almacenados y ofrece soporte para tres bases NoSQL: MongoDB, Ehcache e Infinispan y Apache Cassandra. (Anghel Leonard, 2013)

Las características que Anghel Leonard (2013) define son rápido escalamiento en una base de datos NoSQL e independencia de la tecnología de bases de datos.

Hibernate OGM admite hasta ahora lo siguiente:

- Almacenamiento de datos en base NoSQL (MongoDB)
- Operaciones de creación, lectura, actualización y eliminación (CRUD) para entidades JPA.
- Tipos básicos (como números, cadena, URL, fecha, enumeraciones)
- Asociaciones bidireccionales
- Colecciones (conjunto, lista, mapa, etc.)
- Consultas de texto completo de Hibernate Search.

Hibernate OGM extiende esencialmente la arquitectura de ORM conectando y sacando diferentes componentes. Hibernate ORM convierte y persiste datos entre bases de datos relacionales y lenguajes de programación orientados a objetos utilizando un conjunto de interfaces y clases. Estos incluyen la capa JDBC, utilizada para conectarse a bases de datos y enviar

consultas, y las interfaces Persisters y Loaders, responsables de persistir y cargar entidades y colecciones. Por su parte Hibernate OGM está destinado a lograr los mismos objetivos, pero utilizando bases de datos NoSQL, por ello ya no necesita la capa JDBC en su lugar viene con dos elementos nuevos: un proveedor de datos y un dialecto de almacenamiento de datos. Ambos actúan como adaptadores entre Hibernate OGM Core y la base NoSQL. (Un almacén de datos es un adaptador que conecta el motor de mapeo central con la tecnología específica NoSQL).

El proveedor del almacén de datos es responsable de administrar las conexiones a las bases NoSQL, mientras que el dialecto del almacén de datos gestiona las comunicaciones con los motores de almacenamiento NoSQL.

Se cuenta con dos interfaces:

La interfaz DatastoreProvider es responsable de iniciar, mantener y detener una conexión de tienda, mientras que la interfaz GridDialect se ocupa de la persistencia de los datos en las tiendas NoSQL. Además, las interfaces Persisters y Loaders se reescribieron para admitir las características de la base NoSQL.

En la Figura 15 se muestra la arquitectura que presenta el *framework* Hibernate OGM.

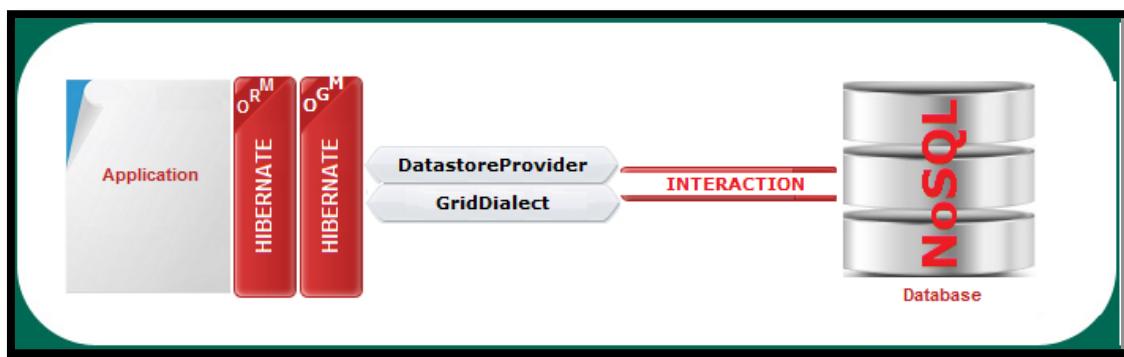


Figura 15. Arquitectura del framework Hibernate OGM.  
Fuente: Anghel Leonard (2013)

### 2.3.14. Doctrine ODM



*Figura 16. Logo del framework Doctrine ODM*

*Fuente: Project MongoDB Object Document Mapper*

“Framework que nos permite el almacenamiento, actualización y eliminación de documentos (NoSQL orientadas o documentos) desde objetos que se manipulan con el lenguaje de programación PHP”. (<http://www.analyticaweb.com/desarrollo-web/doctrine2-la-era-de-los-odm>)

Un ODM nos aporta las ventajas de un ORM pudiendo almacenar los objetos de nuestro modelo de la aplicación en almacenamiento NoSQL (Documentos) con la misma facilidad que existía para persistir el modelo relacional. Doctrine ODM se ha expandido con el objetivo de dar soporte a bases de datos como MongoDB y CouchDB. La unidad básica de persistencia en estos sistemas es el Documento y será fácil transformarlo en objetos de programación. (Fernández, 2013)

Las características que presenta este *framework* descritas por Fernandez (2013) son las siguientes:

- Almacenamiento de nuevos documentos, actualización y eliminación
- Capacidad seguimiento de las acciones en las entidades para posteriormente persistirlos con el método flush() transparentemente.
- Soporte para Colecciones y asociaciones del tipo OneToMany y ManyToOne.

- Soporte para documentos embebidos; tanto para uno como varios campos de documentos. Esto es una diferencia respecto a los ORM ya que no permiten soporte para mapear operaciones de cascada de la persistencia y la eliminación de los objetos
- Especificación de metadatos de mapeo en distintos formatos (XML,YAML,PHP,Annotations)

La arquitectura de Doctrine ODM se basa en el núcleo de Doctrine ORM, como se muestra en la figura 17.

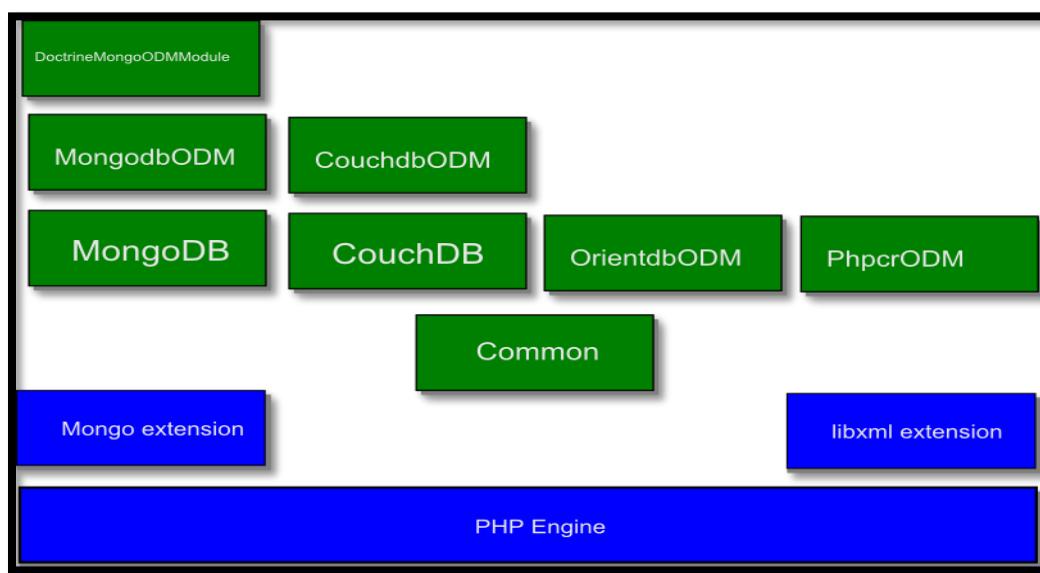


Figura 17. Arquitectura del framework Doctrine y su partición en ODM.  
Fuente: Krivtsov, 2005.

### 2.3.15. Metodología XP

Es una metodología de desarrollo de la ingeniería de *software* formulada por Kent Beck. La programación extrema usa un enfoque orientado a objetos como paradigma preferido de desarrollo, y engloba un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades estructurales: planeación, diseño, codificación y pruebas. (Pressman, R., 2010. Ingeniería del *software*: Un enfoque práctico. 7ma edición).

## **2.4.HIPÓTESIS**

### **2.4.1. Hipótesis general**

- ❖ H<sub>0</sub>: No existen diferencias significativas en el tiempo de ejecución al hacer uso de los *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP.
- ❖ H<sub>1</sub>: Existen diferencias significativas en el tiempo de ejecución al hacer uso de los *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP.

### **2.4.2. Hipótesis específicas**

- ❖ Se puede encontrar el módulo *Web* más usado en la actualidad por las entidades de la ciudad de Piura.
- ❖ Se puede desarrollar el módulo *Web* haciendo uso de los *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP.
- ❖ El tiempo de ejecución del módulo desarrollado depende del *framework* de persistencia ORM u ODM basado en el lenguaje de programación Java y PHP.

### **2.4.3. Identificación y Operacionalización de variables**

#### **2.4.3.1. Variables independientes**

- ❖ Cantidad de registros o documentos en una base de datos relacional y NoSQL, respectivamente.
- ❖ *Framework* de persistencia.
- ❖ Sistema Gestor de Bases de Datos.

#### **2.4.3.2. Variable dependiente**

- ❖ Tiempo de ejecución de las operaciones.

#### 2.4.3.3.Operacionalización de variables

Variable	Definición conceptual	Definición operacional	Indicadores
<b>Dependiente</b> Tiempo de ejecución de las operaciones de los módulos Web desarrollados con los <i>frameworks</i> de persistencia ORM y ODM.	Tiempo de ejecución es definido por el autor a cargo del estudio como la diferencia de tiempo para que se muestre un resultado al realizar una consulta (lectura, actualización, eliminación, búsqueda) a la base de datos, desde la interfaz gráfica del módulo.	Para poder medir el tiempo de ejecución de los <i>frameworks</i> de persistencia se realizarán varias pruebas.	Tiempo de respuesta de operaciones.
<b>Independiente</b> Cantidad de registros o documentos	El registro (también llamado tupla) representa un objeto único de datos implícitamente estructurados en una tabla. Los documentos dentro de una base de datos NoSQL orientada a documentos son similar de algún modo a los registros, pero son menos rígidos.	La cantidad de registros o documentos dentro de una base de datos puede tener gran influencia en el tiempo de ejecución al realizarse operaciones.	Cantidad de registros o documentos en las bases de datos, relacional y NoSQL, respectivamente.
<i>Framework</i> de persistencia	Conjunto de tipos de propósito general, reutilizable y extensible, que proporciona funcionalidad para dar soporte a los objetos persistentes.	El <i>framework</i> de persistencia que se utiliza en el desarrollo de un módulo Web puede tener influencia en el tiempo de ejecución al realizar operaciones.	<i>Framework</i> de persistencia utilizado en el desarrollo del módulo Web.

<b>Variable</b>	<b>Definición conceptual</b>	<b>Definición operacional</b>	<b>Indicadores</b>
Sistema Gestor de Base de Datos	Conjunto de datos interrelacionados y con herramientas computacionales específicas para acceder a dichos datos. Su lugar de almacenamiento se denomina base de datos, pues es aquella que contiene información relevante de una empresa u organización.	El Sistema Gestor de Base de Datos que se utiliza en el desarrollo de un módulo Web puede tener influencia en el tiempo de ejecución al realizar operaciones.	Sistema Gestor de Base de Datos utilizado en el desarrollo del módulo Web.

*Tabla 1. Identificación de variables, indicadores e índices*

*Fuente: Elaboración propia*

## **CAPÍTULO III**

## **METODOLOGÍA**

### **3.1. TIPO DE INVESTIGACIÓN**

El presente trabajo de investigación se considera de tipo aplicada tecnológica, pues se generará conocimiento para elegir los *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP que ofrezcan los menores tiempos de ejecución, esto a través de experimentos (pruebas) con un módulo *Web*.

Según, Lucana Santos (S/A), “la investigación aplicada tecnológica, se entiende como aquella que genera conocimientos o métodos dirigidos al sector productivo de bienes y servicios, ya sea con el fin de mejorarla y hacerla más eficiente”. Es por eso que se realizará un estudio comparativo entre dos ORM y dos ODM para encontrar que tan significativo es su uso dentro de un módulo *Web*, además, cuál de ellos ofrece el menor tiempo de ejecución cuando se realizan operaciones a las bases de datos.

### **3.2. NIVEL DE INVESTIGACIÓN**

El presente estudio de investigación se considera bajo los siguientes niveles: Descriptivo y correlacional.

El estudio será descriptivo porque al iniciar se analizarán las diversas tecnologías que se van a usar, es decir tanto ORM, ODM, SGBDR y SGBD NoSQL, con el fin de conocer a fondo su arquitectura, características, formas de trabajo, entre otras. “Los estudios descriptivos son útiles para analizar cómo es y cómo se manifiesta un fenómeno y sus componentes”. (Hernández Sampieri, Fernández Collado & Baptista Lucio, 2014).

El estudio será correlacional porque se evaluará la relación o no relación que existe entre la variable independiente y la variable dependiente (tiempo de ejecución de las operaciones del módulo *Web*). “Los estudios correlacionales pretenden determinar cómo se relacionan o vinculan diversos conceptos, variables o características entre sí o, también, si no se relacionan”. (Hernández Sampieri et al., 2014)

### **3.3. DISEÑO DE LA INVESTIGACIÓN**

El diseño de la investigación se considera experimental puro, por el motivo que se determinó que *frameworks* de persistencia tanto ORM como ODM, basados en los lenguajes de programación Java y PHP, ofrecen el menor tiempo de ejecución a partir de pruebas experimentales que se realizarán varias veces para lograr el objetivo.

En los experimentos “puros” se manipulan las variables independientes para observar el efecto sobre la o las variables dependientes en una situación de control. Se necesita de tres requisitos para que un experimento sea considerado puro, el primero es la manipulación intencional de una o más variables independientes, el segundo es medir el efecto de la variable independiente sobre la dependiente, esta medición debe ser válida y confiable, como tercer y último requisito tenemos el control o validez interna de la situación experimental, esto quiere decir que sabe que ocurre realmente en la relación entre la variable dependiente e independiente. (Hernández Sampieri et al., 2014).

Para realizar el estudio se utilizó dos tablas comparativas basadas en un modelo factorial 2 x 2 x 4 para recolectar los datos de las variables a utilizar, este proceso se realizó utilizando una cantidad de cuatro réplicas ( $r_1, r_2, r_3, r_4$ ) para de este modo disminuir los errores al momento de procesar los datos obtenidos. Se tomó en cuenta la cantidad de registros ( $c_1, c_2, c_3, c_4$ ), para el caso de ORM, y documentos, para el caso de ODM.

		MariaDB						PostgreSQL									
		C <sub>1</sub>		C <sub>2</sub>		C <sub>3</sub>		C <sub>4</sub>		C <sub>1</sub>		C <sub>2</sub>		C <sub>3</sub>		C <sub>4</sub>	
		I	L	I	L	I	L	I	L	I	L	I	L	I	L	I	L
Hibernate ORM	r <sub>1</sub>																
	r <sub>2</sub>																
	r <sub>3</sub>																
	r <sub>4</sub>																
Doctrine ORM	r <sub>1</sub>																
	r <sub>2</sub>																
	r <sub>3</sub>																
	r <sub>4</sub>																

Tabla 2. Tabla comparativa del tiempo de ejecución para los frameworks de persistencia ORM

Fuente: Elaboración propia

		MongoDB						CouchDB									
		C <sub>1</sub>		C <sub>2</sub>		C <sub>3</sub>		C <sub>4</sub>		C <sub>1</sub>		C <sub>2</sub>		C <sub>3</sub>		C <sub>4</sub>	
		I	L	I	L	I	L	I	L	I	L	I	L	I	L	I	L
Hibernate OGM	r <sub>1</sub>																
	r <sub>2</sub>																
	r <sub>3</sub>																
	r <sub>4</sub>																
Doctrine ODM	r <sub>1</sub>																
	r <sub>2</sub>																
	r <sub>3</sub>																
	r <sub>4</sub>																

Tabla 3. Tabla comparativa del tiempo de ejecución para los frameworks de persistencia ODM

Fuente: Elaboración propia

### 3.3.1. Métodos estadísticos de procesamiento de datos

Se utilizó el **análisis de varianza** para realizar el análisis de datos y determinar si los *frameworks* de persistencia ORM y ODM tienen diferencia significativa en el tiempo de ejecución. El análisis de varianza es un método estadístico que nos permite analizar si la media de una variable - por tanto, métrica – toma valores estadísticamente distintos en los grupos que crea otra variable – por tanto, no métrica; (...) En este análisis se denomina factor a la variable que supuestamente ejerce una influencia sobre la variable estudiada, a la que se le denomina variable dependiente. (Joaquín y Ezequiel, 2017). Para el caso de esta investigación los “factores” fueron: *Framework* de persistencia, Sistema Gestor de Base de Datos y la Cantidad de datos, y la variable dependiente fue el tiempo de ejecución.

Cuando se desea realizar un análisis de varianza se debe tener en cuenta que la variable dependiente tenga una distribución normal (Anderson et al., 2008). Para verificar lo mencionado se realizó el test de **Kolmogorov – Smirnov**, se trata de un método no paramétrico sencillo para probar si existe una diferencia significativa entre una distribución de frecuencias observada y un distribución de frecuencias teórica (Levin y Rubin, 2004). Para este estudio la distribución teórica fue la distribución normal. Luego de aplicar el test a los resultados se llegó a la conclusión que los resultados obtenidos durante la experimentación difieren de una distribución normal. Los resultados del test Kolmogorov – Smirnov se puede revisar en el Anexo 9 del presente estudio.

Para aplicar un análisis paramétrico como el análisis de varianza se tuvo que realizar una transformación de datos para de este modo realizar un ajuste de los datos a una distribución normal. El problema de la varianza no constante suele corregirse al transformar la variable dependiente a otra escala. Por ejemplo, si se trabaja con el logaritmo de la variable dependiente en lugar de la variable dependiente original, los valores de la variable dependiente se comprimirán (estarán más cercanos unos a otros) y con esto disminuirán los efectos de la varianza no constante (Anderson et al., 2008). Para el presente estudio se realizó una transformación logarítmica en base 10.

Por último, para corroborar los resultados que se obtuvieron con el análisis de varianza se realizó la prueba no paramétrica de **Kruskal – Wallis**. La prueba de Kruskal – Wallis se utiliza en situaciones en las que el uso de la curva normal no es apropiado, tal como los resultados obtenidos en esta investigación; es conocida como una prueba sin distribución o, más comúnmente, prueba no paramétrica, se utiliza cuando las poblaciones son más de dos (Levin y Rubin, 2004).

### **3.4. TIPOS Y TÉCNICAS DE MUESTREO**

#### **3.4.1. Universo poblacional**

La población de este estudio comparativo son los *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP, que se utilizan actualmente en el desarrollo de *software*.

#### **3.4.2. Tamaño de muestra**

El tamaño de la muestra del presente estudio son los *frameworks* de persistencia ORM: Hibernate y Doctrine y para el caso de ODM: Hibernate OGM y Doctrine ODM, ambos tipos de *frameworks* basados en los lenguajes de programación Java y PHP.

### **3.5. TÉCNICAS E INSTRUMENTOS PARA LA RECOLECCIÓN DE DATOS**

Las técnicas e instrumentos utilizados serán:

#### **3.5.1. Encuestas**

El objetivo de la encuesta fue determinar si se hace uso de *frameworks* para el desarrollo de *software* (para este caso la prioridad fueron los *frameworks* de persistencia), además de los datos acerca del *software* más utilizado dentro de las empresas de la ciudad de Piura. La encuesta se aplicó específicamente al personal que integran los equipos de desarrollo de *software* de las entidades públicas y privadas, al culminarla se obtuvo información sobre el módulo más utilizado, esto teniendo en cuenta las operaciones que se realizan diariamente. Para la elección de los encuestados se usó lo conocido

como muestreo por conveniencia (muestreo no probabilístico). Creswell, Jhon (2008) define el muestreo por conveniencia como un procedimiento de muestreo cuantitativo en el que el investigador selecciona a los participantes, ya que están dispuestos y disponibles para ser estudiados.

La elección de empresas se realizó de esta manera ya que no se contaba con una lista de empresas de la ciudad de Piura que cuenten con oficinas de desarrollo de *software*.

### **3.5.2. Investigación bibliográfica**

Se realizó como una introducción a la investigación que se plantea, con el fin de conocer a fondo las distintas tecnologías que se usaron en el transcurso del estudio.

### **3.5.3. Guía de observación**

La guía de observación permitió recolectar los tiempos de ejecución que se originaron en cada prueba que se realizó. La elaboración de la guía de observación se hizo mediante el diseño factorial completo de  $2 \times 2 \times 4$ .

### **3.5.4. Programa cronómetro**

Para la toma de tiempos de ejecución durante la experimentación se hizo uso de funciones de los lenguajes de Programación: *currentTimeMillis* de Java y *microtime* de PHP. Estas funciones se encuentran en determinadas partes del código de donde se tomaron los tiempos de ejecución.

### **3.5.5. Validez de la encuesta**

Para realizar la validación de la encuesta se solicitó el juicio de diferentes docentes de la Universidad Nacional de Piura. Los docentes que intervinieron en la validación de la encuesta tienen conocimientos en los campos de la informática y la estadística. El aporte que tuvieron los docentes fue el de evaluar la congruencia, redacción, precisión y amplitud de las preguntas de la encuesta en relación al proyecto de investigación; asimismo aportaron con ideas y sugerencias para mejorar las preguntas de la encuesta.

### **3.5.6. Aplicación de la encuesta**

El objetivo de la encuesta fue determinar el módulo *Web* más utilizado dependiendo de las operaciones que se realizan a diario, así como también identificar el grado de utilización de *frameworks* para el desarrollo de *software*, entre otras preguntas adicionales de importancia para el encuestador; todo esto con la finalidad de implementar el módulo *Web* que resultó como más utilizado haciendo uso de las tecnologías que se seleccionaron en el presente trabajo de investigación.

La encuesta estuvo dirigida al personal que labora en las oficinas de desarrollo de *software* de las entidades públicas y privadas de la ciudad de Piura.

### **3.5.7. Procesamiento de la encuesta**

#### **3.5.7.1. Módulo *Web* más usado**

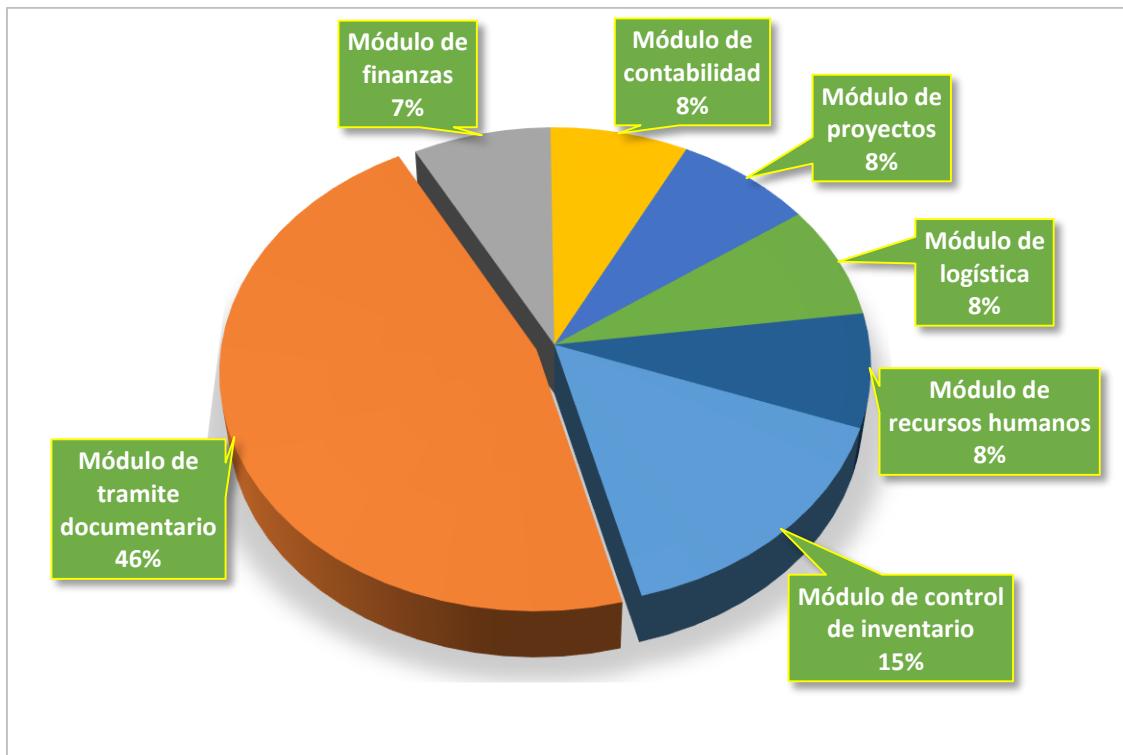
La encuesta fue aplicada a un total de 10 entidades entre empresas e instituciones (tamaño de muestra) que se ubican en la ciudad de Piura. Estas instituciones cuentan con un área que se dedica al desarrollo de *software*. A continuación en la tabla 4 se muestra los nombres de las instituciones en las que se encuestó a los desarrolladores de *software*, en cada uno de las instituciones solo se encuestó a una sola persona.

Nº	Nombre de la Institución
1	Entercomp SAC.
2	RENIEC
3	Caja Paita
4	Municipalidad distrital de Castilla

Nº	Nombre de la Institución
<b>5</b>	Municipalidad provincial de Piura
<b>6</b>	Gobierno regional Piura
<b>7</b>	Universidad Nacional de Piura
<b>8</b>	Team Libera
<b>9</b>	Caja Sullana
<b>10</b>	SAT

*Tabla 4. Instituciones encuestadas de la ciudad de Piura  
Fuente: Elaboración propia.*

Para llegar al objetivo principal de la encuesta que fue determinar el módulo *Web* más utilizado se realizó la pregunta: “En la empresa donde labora actualmente, ¿Cuál es el módulo *Web* más utilizado?”. En el gráfico 4 se muestran los resultados del procesamiento de la respuesta a la pregunta planteada.



*Gráfico 4. Resultados de la encuesta sobre el módulo Web más utilizado*

Fuente: Elaboración propia.

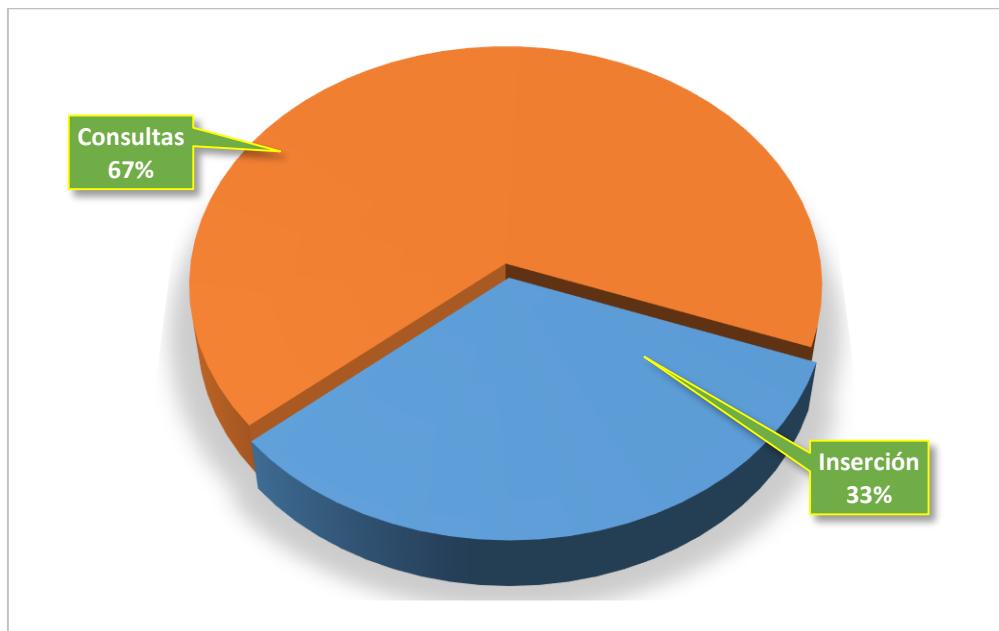
Como se puede observar en la gráfico 4, los encuestados indicaron en un 46% que el módulo de Trámite Documentario es el módulo más usado dentro de sus instituciones, seguido del módulo de Control de Inventory con un porcentaje de 15%. El tercer puesto con un 8% es compartido por los módulos de Contabilidad, Proyectos, Logística y Recursos Humanos; y por último en el cuarto puesto se ubica el módulo de Finanzas con un 7%.

De acuerdo a los resultados que se obtuvieron con esta encuesta el módulo de Trámite Documentario fue el módulo elegido para implementar en el presente trabajo de investigación.

### 3.5.7.2. Operaciones más frecuentes

Del módulo de Trámite Documentario también es necesario conocer cuáles son las operaciones que más se realizan, así que se planteó la pregunta: “Según el módulo seleccionado, ¿Cuál o cuáles son las operaciones más accedidas dentro del módulo, es decir, aquella donde una gran cantidad de usuarios interactúan diariamente?”.

De la pregunta planteada se obtuvieron los resultados que se muestran a continuación en el gráfico 5.



*Gráfico 5. Resultados de la encuesta sobre las operaciones más frecuentes*  
Fuente: Elaboración propia.

Como se puede observar en el gráfico 5, los encuestados indicaron que las operaciones que se realizan con más frecuencia en el módulo de Trámite Documentario son: las consultas y los registros, con un porcentaje de 67% y 33%, respectivamente.

Entonces por ser las únicas operaciones que se indicaron como más utilizadas se eligió la inserción y la consulta de datos como operaciones a usar en el módulo de Trámite Documentario a desarrollar.

### **3.5.8. Generación de datos para las pruebas**

Para generar datos de pruebas que simularon el comportamiento de las aplicaciones desarrolladas, se recurrió a instituciones que brindaron información sobre la data del Sistema de Trámite Documentario. La institución que brindó la información fue la Municipalidad Provincial de Morropón Chulucanas, en la cual se obtuvo que entre los años 2016 y 2017 se registró aproximadamente 100000 (cien mil) expedientes, esta cantidad fue utilizada para definir la cantidad máxima ( $C_4$ ) de registros para las bases de datos con

las que trabajaron los ORM y ODM de las aplicaciones *Web* desarrolladas; las demás cantidades inferiores a C<sub>4</sub>, definidas por el investigador, son: 5000 para C<sub>1</sub>, 10000 para C<sub>2</sub> y 50000 para C<sub>3</sub>.

Para la operación de inserción de datos durante la toma de tiempos de ejecución se hizo uso de estructuras repetitivas para la generación de las cantidades de registros. Además, de funciones de los lenguajes de programación para generar números aleatorios, cuando se requería, tales como: *rand*, para el caso de PHP y la función *floor* de la biblioteca *Math* de Java.

### **3.5.9. Ambiente para desarrollo y simulación**

En la tabla 5 se muestra los recursos del equipo que se utilizaron durante el proceso de desarrollo de los módulos de Trámite Documentario, asimismo como los recursos del equipo que se usaron para la simulación de los módulos.

<b>Proceso</b>	<b>Hardware</b>	<b>Software</b>
<b>Desarrollo de Aplicaciones</b>	1 Laptop lenovo B50-70. Procesador Intel® Core™ i3-4005U CPU @ 1.70 GHz. Memoria RAM 4GB. Sistema Operativo Windows 8.1 Pro 64 bits.	Atom NetBeans IDE 8.0.2 Notepad++ v7.5.4 Hibernate 4.3 Hibernate OGM 5.0.0 Doctrine ORM v.2.4 Doctrine ODM v.1.2 MariaDB 10.1.8 PostgreSQL 10.1 MongoDB 3.4.13 Apache CouchDB v2.1.1 Google Chrome 65.0.3325.181 XAMPP 5.6.14 MAVEN 4.0.0 PgAdmin 4.0 MongoDB Compass Community 1.13.0

		Apache HTTP Server 2.4.17 Apache Tomcat 8.0.3.0 JQuery 2.1.4 GSON 2.2 Bootstrap v.3.3.4
<b>Simulación de operaciones y toma de tiempos</b>	1 Laptop Dell INSPIRON N4110. Procesador Intel® Core™ i5-2450M CPU @ 2.50 GHz. Memoria RAM 4GB. Sistema Operativo Windows 10 Pro 64 bits.	Postman Hibernate 4.3 Hibernate OGM 5.0.0 Doctrine ORM v.2.4 Doctrine ODM v.1.2 MariaDB 10.1.8 PostgreSQL 10.1 MongoDB 3.4.13 Apache CouchDB v2.1.1 XAMPP 5.6.14 Apache HTTP Server 2.4.17 Apache Tomcat 8.0.3.0 GSON 2.2

Tabla 5. Hardware y Software utilizados para el desarrollo y simulación de las aplicaciones desarrolladas

Fuente: Elaboración Propia

### 3.6. CASO DE ESTUDIO

Para poder realizar el estudio comparativo entre *frameworks* de persistencia tanto ORM como ODM se necesitó desarrollar una cierta parte de la aplicación *Web*, que para este caso fue la que más votos obtuvo en la encuesta aplicada a las instituciones públicas y privadas de la ciudad de Piura. Como se dieron los resultados el módulo elegido fue el módulo de Trámite Documentario.

#### 3.6.1. Caso de estudio: Sistema de Trámite Documentario

El Sistema de Trámite Documentario es un aplicativo de uso interno que tiene como fin el seguimiento de la documentación generada en la institución y/o recepcionada en cada una de sus mesas de parte. Los documentos, ya sean de origen interno o externo que necesiten circular por

cualquier área de la institución, son registrados en el sistema, en donde se le aplica los movimientos de envío, recepción y archivo según corresponda. (INGEMMET, Oficina de Sistemas de Información – Manual de usuario. Diciembre 2011).

Del caso de estudio planteado en la presente tesis sólo se desarrolló una parte del módulo, la que luego fue útil para realizar las pruebas de comparación entre los *frameworks* de persistencia.

### **3.6.2. Obtención de los requerimientos**

El desarrollo de un sistema comprende como primera etapa la toma de requerimientos. En esta etapa se muestra los requerimientos del sistema de Trámite Documentario que se obtuvieron con la ayuda de proyectos de investigación realizados anteriormente, además de documentos de carácter público de algunas instituciones (entre ellas algunas de las que se encuestaron) que utilizan este sistema.

#### **3.6.2.1. Requerimientos del usuario**

En la tabla 6 se detallan cada uno de los requerimientos del usuario para el módulo de Trámite Documentario. Dichos requerimientos son expresados en lenguaje natural, de forma que puedan ser comprendidos fácilmente por los clientes o usuarios del sistema.

<b>Requerimientos del usuario</b>	
<b>1</b>	El módulo de Trámite Documentario debe permitir al recepcionista visualizar las bandejas de expedientes: Entrada, Recepcionados, Por Derivar y Derivados.
<b>2</b>	El módulo de Trámite Documentario debe permitir tanto al recepcionista como al solicitante consultar el estado detallado de los trámites que se han registrado, ya sea a través del número de expediente, fecha de registro de

	expediente, nivel de prioridad, tipo de expediente, origen del expediente (interno o externo).
<b>3</b>	El módulo de Trámite Documentario debe permitir al recepcionista registrar nuevos expedientes, ya sean internos o externos a la institución.

*Tabla 6. Requerimientos del usuario para el Trámite Documentario*

*Fuente: Elaboración Propia*

En las siguientes tablas, se identificarán los requerimientos del sistema que corresponden a cada requerimiento de los usuarios.

<b>Requerimientos del usuario</b>	
<b>1</b>	El módulo de Trámite Documentario debe permitir al recepcionista visualizar las bandejas de expedientes: Entrada, Recepcionados, Por Derivar y Derivados.
<b>Requerimientos del sistema</b>	
<b>1.1.</b>	Este permiso sólo está disponible para el usuario recepcionista.
<b>1.2.</b>	La bandeja de expedientes “Entrada” contendrá a todos aquellos expedientes que han sido derivados al área donde el usuario es el encargado, pero que aún no han sido recepcionados.
<b>1.3.</b>	La bandeja de expedientes “Recepcionados” contendrá a todos aquellos expedientes que se encuentran en el área donde el usuario es encargado, pero que aún no ha sido visto para su respectiva evaluación, además de decidir si es archivado, finalizado o derivado a otra área.

<p><b>1.4.</b></p>	La bandeja de expedientes “Por Derivar” contendrá a todos aquellos expedientes que han sido registrados en al área donde el usuario es el encargado, pero que aún no han sido derivados a otra u otras áreas.
<p><b>1.5.</b></p>	La bandeja de expedientes “Derivados” contendrá a todos aquellos expedientes que han sido registrados en al área donde el usuario es el encargado, y que ya han sido derivados a una u otras áreas.

*Tabla 7. Requerimientos del sistema correspondiente al requerimiento de usuario 1*

*Fuente: Elaboración Propia.*

<b>Requerimientos del usuario</b>	
<p><b>2</b></p>	El módulo de Trámite Documentario debe permitir tanto al recepcionista como al solicitante consultar el estado detallado de los trámites que se han registrado, ya sea a través del código, clasificación, prioridad, proveído, fecha de registro, estado, origen (interno o externo) del expediente.
<b>Requerimientos del sistema</b>	
<p><b>2.1</b></p>	El módulo de Trámite Documentario validará aquellos datos que se enviarán para realizar el filtro de la búsqueda, si fuese el caso que sean erróneos se deben mostrar un mensaje para corregirlos.
<p><b>2.2</b></p>	El módulo de Trámite Documentario mostrará todo el flujo (paso a través de las distintas dependencias) del expediente, además de su condición actual (en proceso, archivado, finalizado).

*Tabla 8. Requerimientos del sistema correspondiente al requerimiento de usuario 2*

*Fuente: Elaboración Propia.*

<b>Requerimientos del usuario</b>	
<b>3</b>	El módulo de Trámite Documentario debe permitir al recepcionista registrar nuevos expedientes, ya sean internos o externos a la institución.
<b>Requerimientos del sistema</b>	
<b>3.1</b>	Dependiendo del tipo de trámite (interno o externo) que se vaya a registrar se solicitará en el formulario datos diferentes.
<b>3.2</b>	Se creará un número de identificación único para el expediente. Para crearlo se hará uso del tipo de origen del expediente (interno o externo).
<b>3.3</b>	El módulo de Trámite Documentario debe solicitar los datos de la persona solicitante, ya sea natural (no trabajador) o jurídica. Estos datos incluyen los datos personales, así como también su ubicación. Para el caso de persona natural un dato obligatorio es el DNI y para el caso de persona jurídica el RUC.

*Tabla 9. Requerimientos del sistema correspondiente al requerimiento de usuario 3  
Fuente: Elaboración Propia.*

### **3.6.2.2. Especificación de los requerimientos**

La especificación de requerimientos es la manera de detallar cada uno de los requerimientos para su aclaración y su fácil rastreo en caso de detectar errores o ambigüedades en el diseño o implementación. La continuación se especifica cada uno de los requerimientos del usuario siguiendo una estructura de tablas.

<b>Visualización de bandejas de expedientes</b>	
<b>Versión</b>	1.0
<b>Actor</b>	Recepcionista
<b>Objetivo</b>	Visualizar las bandejas de expedientes: Entrada, Recepcionados, Por Derivar y Derivados.
<b>Resumen</b>	El sistema permitirá visualizar las bandejas de expedientes: Entrada, Recepcionados, Por Derivar y Derivados, a los recepcionistas que han ingresado al sistema.
<b>Precondición</b>	El recepcionista debe estar identificado en el sistema
<b>Entradas</b>	Ninguna
<b>Salidas</b>	Listas de los expedientes.
<b>Post condiciones</b>	Ninguna.
<b>Actividades interiores</b>	
1. Luego de ingresar al sistema, el recepcionista se dirige al submenú “Bandeja”, y luego seleccionará una de las siguientes opciones: Bandeja de entrada, Bandeja de recepcionados o Bandeja por derivar.	

2. El sistema muestra la lista de expedientes de la bandeja seleccionada.	
	3. El usuario elige cual es la siguiente bandeja de expedientes que desea observar.
4. El sistema muestra la lista de expedientes que ha sido elegida por el usuario.	

Tabla 10. Especificación del requerimiento 1

Fuente: Elaboración Propia.

<b>Consulta de expedientes para solicitante</b>	
<b>Versión</b>	1.0
<b>Actor</b>	Recepcionista y solicitante
<b>Objetivo</b>	Permitir consultar expedientes a solicitantes.
<b>Resumen</b>	El sistema permitirá realizar consultas a los usuarios solicitantes, a través del código de expediente.
<b>Precondición</b>	Ingresar a la URL del sistema.
<b>Entradas</b>	Código del expediente.
<b>Salidas</b>	Información del expediente.

<b>Post condiciones</b>	Consultar otro expediente.
<b>Actividades interiores</b>	<b>Actividades exteriores</b>
	1. El solicitante debe ingresar a URL del sistema en un navegador.
2. El sistema muestra el formulario donde se pide ingresar el código del expediente a consultar.	
	3. El solicitante ingresa el código del expediente y da click en el botón “BUSCAR”.
4. El sistema verifica si el código ingresado es válido, de no ser así se muestra un mensaje que el código no es válido. Si el código es válido pero no existe un expediente con tal código se muestra el mensaje que el expediente no existe. Si el sistema encuentra un expediente con el código ingresado lo muestra en pantalla.	
	5. Se muestra la información del expediente, las distintas áreas por las que ha pasado, asimismo el estado actual que tiene el expediente: finalizado, archivado, en proceso.

*Tabla 11. Especificación del requerimiento 2*

*Fuente: Elaboración Propia.*

Consulta de expedientes para recepcionista	
<b>Versión</b>	1.0
<b>Actor</b>	Recepcionista
<b>Objetivo</b>	Permitir consultar expedientes a recepcionista.
<b>Resumen</b>	El sistema permitirá realizar consultas a los recepcionistas, a través de diferentes filtros.
<b>Precondición</b>	El recepcionista debe estar identificado en el sistema.
<b>Entradas</b>	Filtros: Código, clasificación, prioridad, proveído, origen, estado, fecha de registro inicio, fecha de registro final.
<b>Salidas</b>	Información del expediente.
<b>Post condiciones</b>	Consultar otro expediente.
Actividades interiores	
	<b>Actividades exteriores</b>
	1. Luego de ingresar al sistema, el recepcionista se dirige al submenú “Trámite expedientes”, y luego a la opción “Buscar expediente”.
2. El sistema muestra el formulario donde se pueden observar varios filtros.	

	3. El solicitante ingresará aquellos datos que según su criterio le permitirá encontrar el expediente que necesita. Luego de esto da click en el botón “Buscar”.
4. El sistema verifica si existen registros que se adecuan a los datos ingresados por el solicitante. Si el sistema no encuentra ningún expediente entonces se muestra un mensaje que indica que no se encontraron expedientes que coincidan con los datos ingresados. Si se encuentran expedientes entonces se listan en una tabla.	
	5. Se muestra la información de los expedientes, las distintas áreas por las que ha pasado, asimismo el estado actual que tiene el expediente: finalizado, archivado o en proceso.

*Tabla 12. Especificación del requerimiento 3*

*Fuente: Elaboración Propia.*

Registro de expediente	
<b>Versión</b>	1.0
<b>Actor</b>	Repcionista
<b>Objetivo</b>	Ingresar al sistema un nuevo expediente.

<b>Resumen</b>	El sistema permitirá al recepcionista ingresar nuevos expedientes.
<b>Precondición</b>	El recepcionista debe estar identificado en el sistema.
<b>Entradas</b>	Datos para el nuevo registro: Asunto, clasificación, prioridad, proveído, origen, subdocumentos, folios, remitente, representante, responsable, observación.
<b>Salidas</b>	Mensaje de registro exitoso o de error.
<b>Post condiciones</b>	Registrar otro expediente.
Actividades interiores	Actividades exteriores
	1. Luego de ingresar al sistema, el recepcionista se dirige al submenú de “Trámite expediente” y luego a la opción “Registrar Expediente”.
2. El sistema muestra el formulario donde se deben ingresar los datos para registrar el nuevo expediente.	
	3. El recepcionista comienza a ingresar todos los campos que son obligatorios para registrar el nuevo expediente. Luego da click en el botón “Guardar”.

<p>4. El sistema verifica que todos los campos obligatorios hayan sido llenados, de no ser así se muestra un mensaje solicitando el campo que no fue llenado.</p> <p>Si todos los campos obligatorios fueron llenados entonces el sistema verifica que los datos ingresados sean correctos, es decir donde se pide número que se haya ingresado un número, de no ser así entonces se muestra un mensaje comunicando al usuario que lo ingresado es erróneo.</p> <p>Si todo es correcto se procese a registrar el expediente. El sistema es el encargado de asignarle su código de expediente. Si el registro se realiza correctamente se muestra un mensaje de que el registro fue exitoso, si no se muestra el mensaje “Ocurrió un error en el sistema al intentar registrar el expediente”.</p>	
	<p>5. El usuario debe dar click en el mensaje que aparece en el formulario.</p>
<p>6. Se limpia el formulario para un nuevo registro de expediente, si el registro fue realizado correctamente.</p>	

*Tabla 13. Especificación del requerimiento 4  
Fuente: Elaboración Propia.*

### 3.6.3. Diagramas

Los diagramas ayudan a modelar el comportamiento del sistema. El comportamiento del sistema va a estar definido por la interacción con el usuario u otros sistemas.

Después de tener la especificación de cada uno de los requerimientos en lenguaje natural, se necesita traducir los requerimientos a un modelo que permita a los desarrolladores entenderlos. Para una descripción gráfica de cada uno de los requerimientos del sistema utilizamos los diagramas de casos de uso y de secuencia del lenguaje de modelado unificado (UML, por sus siglas en inglés, *Unified Modeling Language*).

#### 3.6.3.1. Diagrama de Casos de Uso del negocio

Los diagramas de Casos de Uso hacen que se muestren las interacciones entre los casos de uso y los actores. Los casos de uso representan la funcionalidad del sistema y los requisitos del sistema desde la perspectiva del usuario. El diagrama de la Figura 18 muestra el Caso de Uso del Negocio para el módulo de Trámite Documentario.

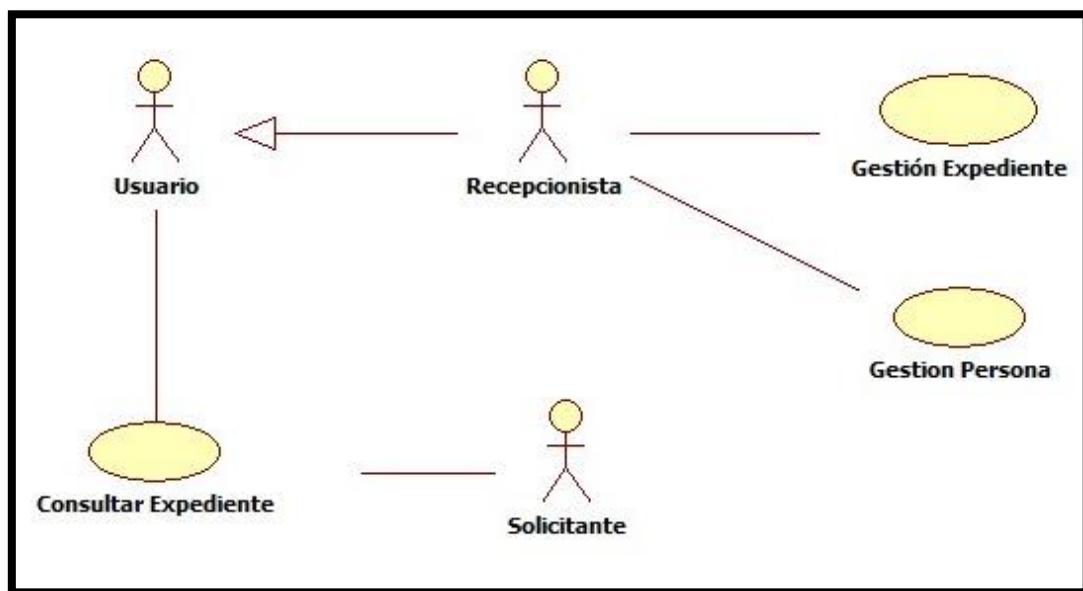


Figura 18. Caso de uso de negocio  
Fuente: Elaboración propia

### ❖ Actores de Negocio

A continuación se describen los actores de negocio encontrados en el diagrama de la Figura 18.

ACTOR	DESCRIPCIÓN
<b>Solicitante</b>	Persona natural o jurídica que se acerca a la entidad a realizar ciertos tipos de trámite, a este tipo se le conoce como trámite externo. Del mismo modo, las dependencias de la institución pueden comportarse como solicitantes al emitir un expediente a otras dependencias de la misma institución, a este tipo de trámite se le conoce como trámite interno.
<b>Repcionista</b>	Usuario encargado de la recepción, registro y derivación de expediente.

*Tabla 14. Actores de Negocio  
Fuente: Elaboración propia*

### ❖ Especificaciones de los Casos de Uso del Negocio

La tabla 15 muestra las especificaciones del Caso de Uso del Negocio: Consultar expediente.

<b>USE CASE 1</b>	<b>Consultar expediente</b>
<b>Actor</b>	Usuario, Solicitante.
<b>Propósito</b>	Consulta de un expediente a través del código de expediente.
<b>Descripción</b>	1. Consultar expediente

*Tabla 15. Especificación del caso de uso “Consultar expediente”  
Fuente: Elaboración propia*

La tabla 16 muestra las especificaciones del Caso de Uso del Negocio: Gestión Expediente.

<b>USE CASE 2</b>	<b>Gestión Expediente</b>
<b>Actor</b>	Recepcionista
<b>Propósito</b>	Realizar el registro, visualización de bandejas y consulta de expediente.
<b>Descripción</b>	<ol style="list-style-type: none"> <li>1. Registrar expediente</li> <li>2. Visualizar bandejas</li> <li>3. Realizar consulta de expediente</li> </ol>

*Tabla 16. Especificación del caso de uso “Gestión Expediente”*

*Fuente: Elaboración propia*

La tabla 17 muestra las especificaciones del Caso de Uso del Negocio: Gestión Persona.

<b>USE CASE 3</b>	<b>Gestión Persona</b>
<b>Actor</b>	Recepcionista
<b>Propósito</b>	Realizar el registro y consulta de personas relacionadas con un expediente (remitente, representante y responsable).
<b>Descripción</b>	<ol style="list-style-type: none"> <li>1. Registrar persona</li> <li>2. Consultar persona</li> </ol>

*Tabla 17. Especificación del caso de uso “Gestión Persona”*

*Fuente: Elaboración propia*

### 3.6.3.2. Diagramas de Objetos del Negocio (DON)

- ❖ Diagrama de Objeto del Negocio de la figura 19 muestra Consultar expediente.



Figura 19. Don Consultar Expediente  
Fuente: Elaboración propia

- ❖ Diagrama de Objeto del Negocio de la figura 20 muestra la Gestión de expediente.



Figura 20. Don Gestión Expediente  
Fuente: Elaboración propia

- ❖ Diagrama de Objeto del Negocio de la figura 21 muestra la Gestión de persona.



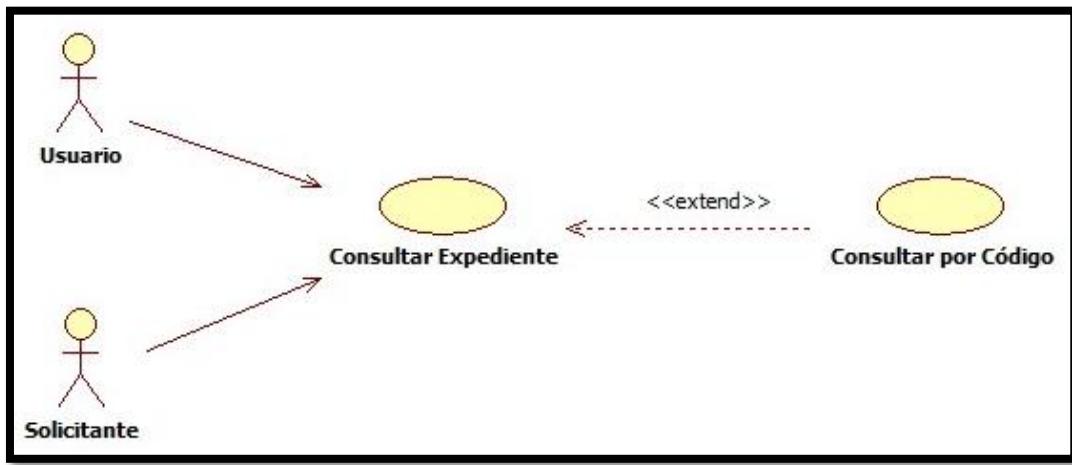
Figura 21. Don Gestión Persona  
Fuente: Elaboración propia

### 3.6.3.3. Diagramas de Casos de Uso de Requerimientos (DUC)

Un modelo de caso de uso divide la funcionalidad de un sistema en comportamientos, conocidos como casos de uso, significativos para los usuarios del sistema, llamados actores.

Se crean diferentes escenarios para cada conjunto diferente de condiciones de un caso.

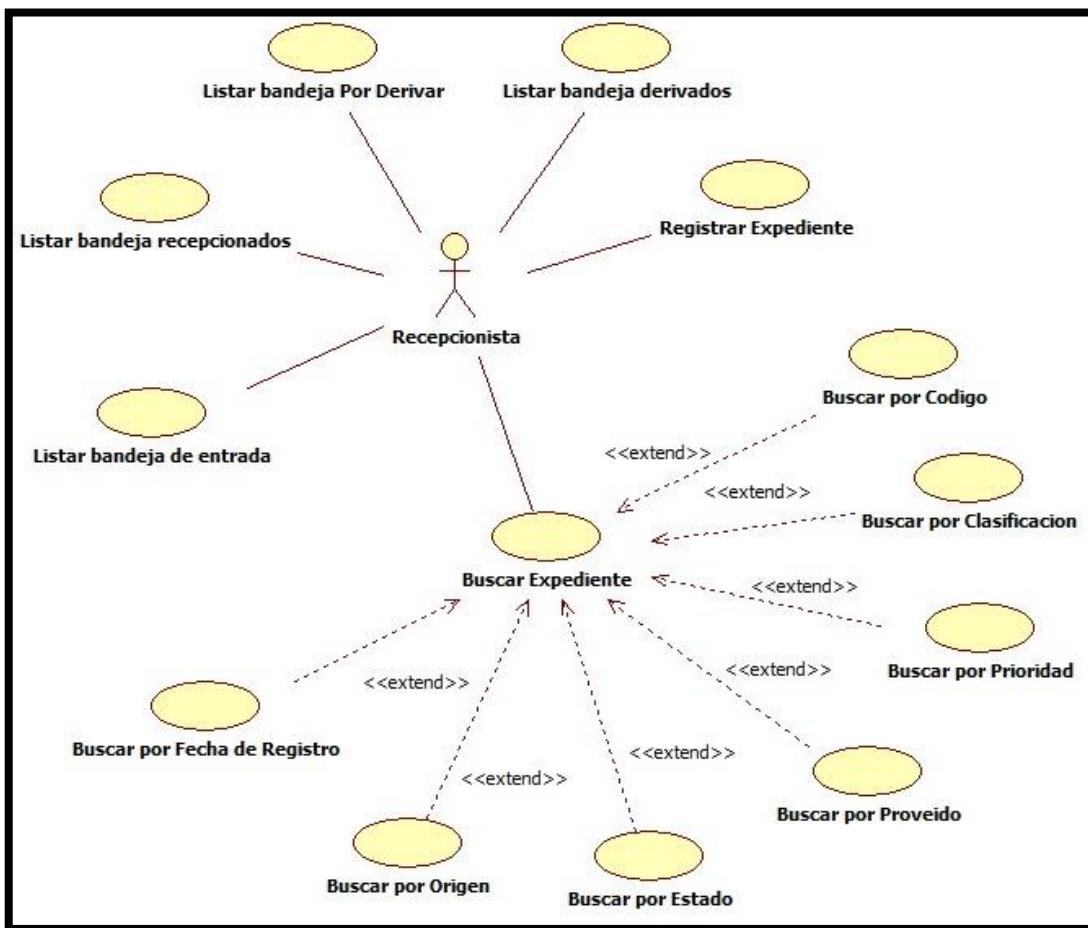
- ❖ Diagrama de caso de uso de la figura 22 muestra la funcionalidad de consultar expediente.



*Figura 22. DUC Consultar Expediente*

Fuente: Elaboración propia

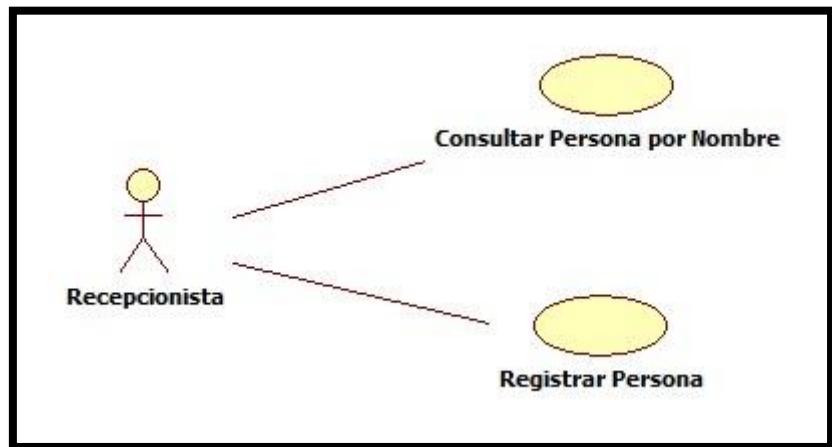
- ❖ Diagrama de caso de uso de la figura 23 muestra la funcionalidad de Gestión Expediente.



*Figura 23. DUC Gestión Expediente*

Fuente: Elaboración propia

- ❖ Diagrama de caso de uso de la figura 24 muestra la funcionalidad de Gestión Persona.



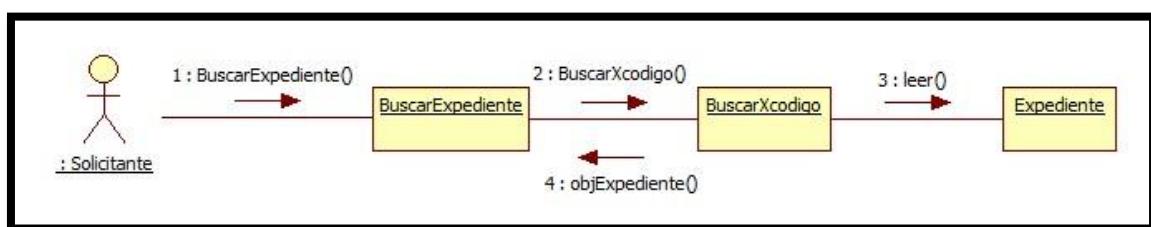
*Figura 24. DUC Gestión Persona  
Fuente: Elaboración propia*

#### 3.6.3.4. Diagramas de Colaboración (DC)

Los diagramas de colaboración presentados en esta fase de elaboración, también llamados diagramas de interacción, muestran la organización estructural de los objetos que envían mensajes.

- ❖ Consultar Expediente

En la figura 25 se muestra el Diagrama de colaboraciones: Consultar Expediente.



*Figura 25. DC Consultar Expediente  
Fuente: Elaboración propia*

## ❖ Gestión de Expediente

En la figura 26 se muestra el Diagrama de colaboraciones:  
Mantenedor Expediente.

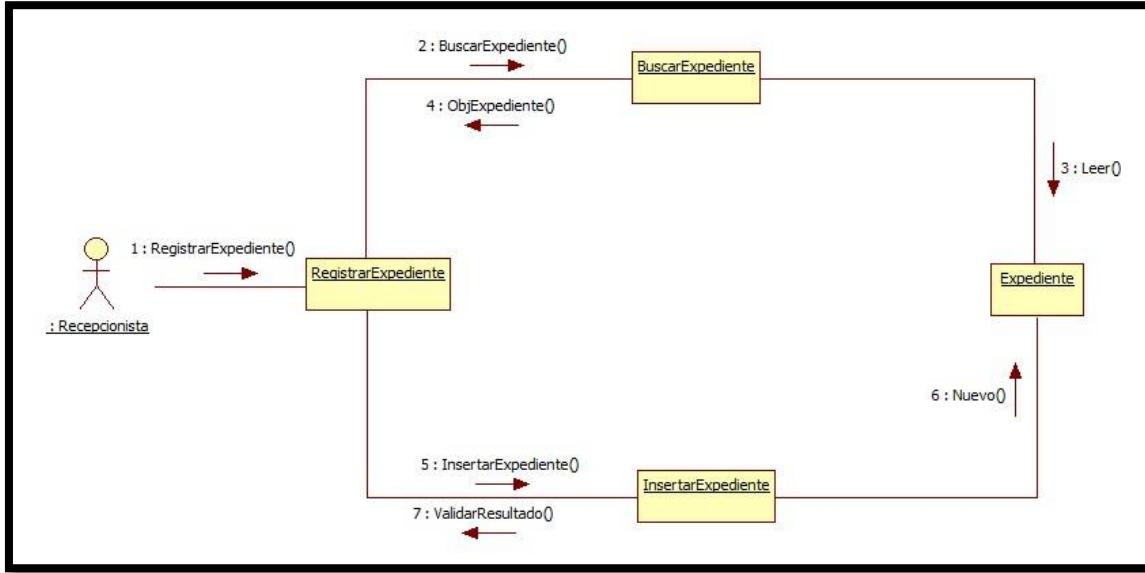


Figura 26. DC Mantenedor Expediente  
Fuente: Elaboración propia

## ❖ Gestión de Persona

En la figura 27 se muestra el Diagrama de colaboraciones:  
Mantenedor Persona.

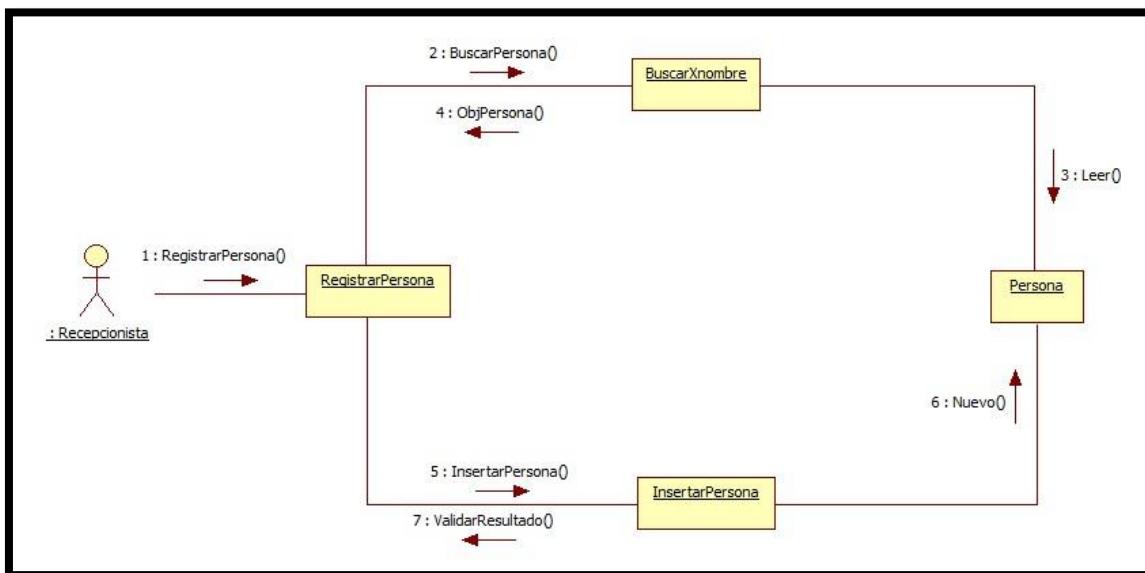


Figura 27. DC Mantenedor Persona  
Fuente: Elaboración propia

### 3.6.3.5. Diagramas de Secuencia (DS)

Cada uno de los diagramas de secuencia presentados en esta fase muestra la interacción de un conjunto de objetos destacando la ordenación temporal de los mensajes entre los objetos. Los diagramas de secuencia muestran los detalles de los casos de uso al nivel de los mensajes intercambiados por los objetos.

- ❖ En la figura 28 muestra el Diagrama de Secuencia: Consultar Expediente.

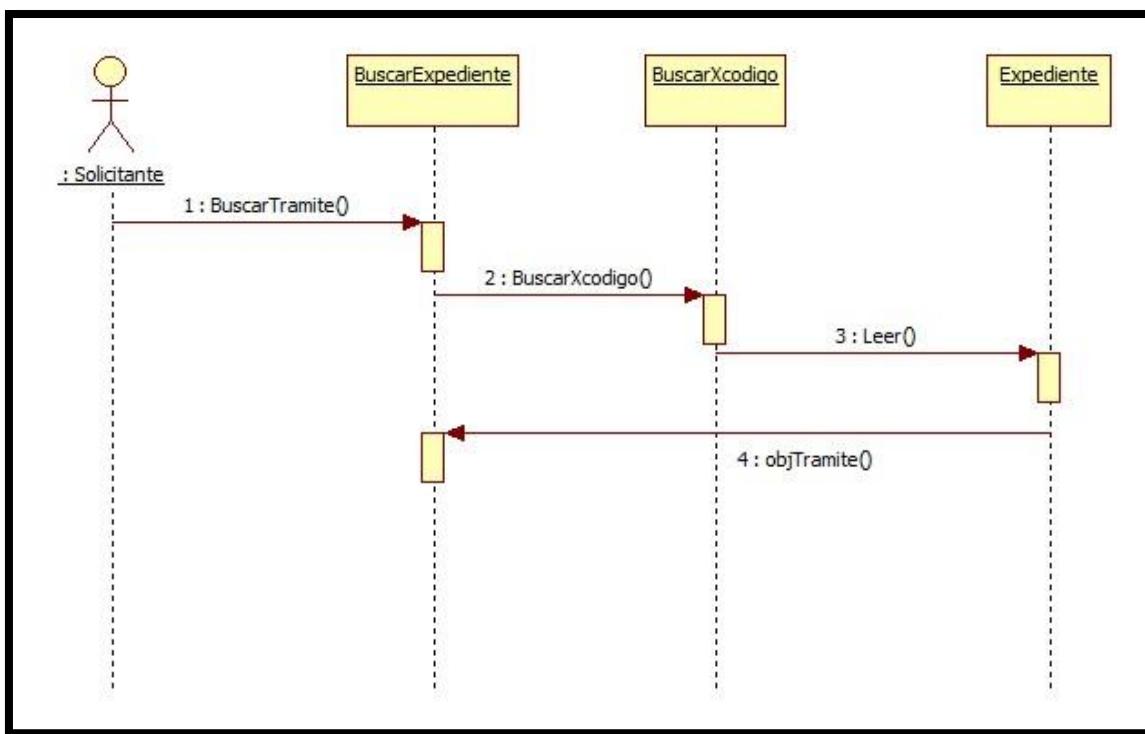


Figura 28. DS Consulta Expediente  
Fuente: Elaboración propia

- ❖ En la figura 29 muestra el Diagrama de Secuencia: Mantenedor Expediente.

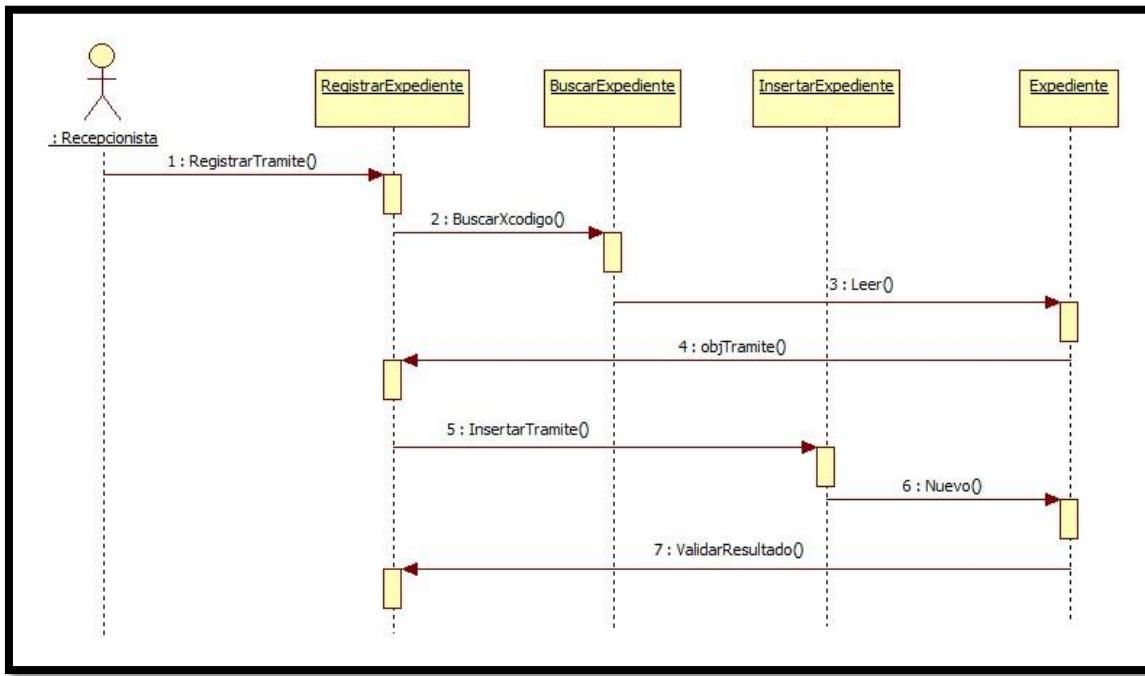


Figura 29. DS Mantenedor Expediente  
Fuente: Elaboración propia

- ❖ En la figura 30 se muestra el Diagrama de Secuencia: Mantenedor Persona.

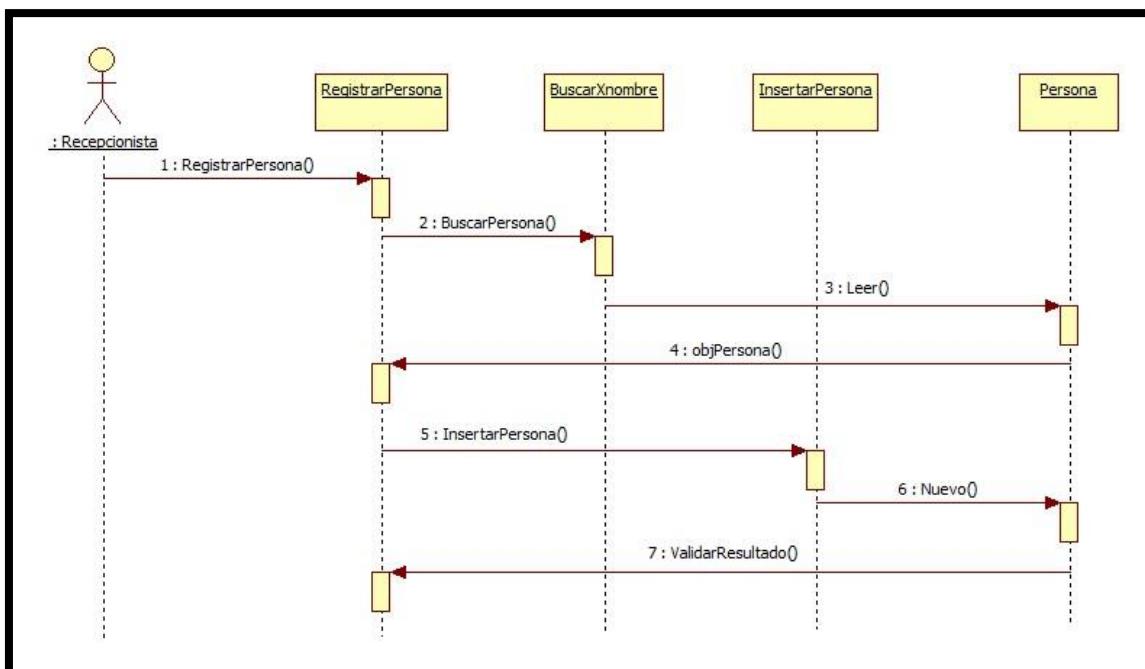


Figura 30. DS Mantenedor Persona  
Fuente: Elaboración propia

### 3.6.3.6. Diagrama de Clases

En la figura 31 se muestra el diagrama de clases para el módulo *Web* de Trámite Documentario, este diagrama muestra las interacciones entre las clases del sistema, para de este modo cumplir con los objetivos del Sistema. El diagrama de clases representa la vista de diseño estática de un sistema.

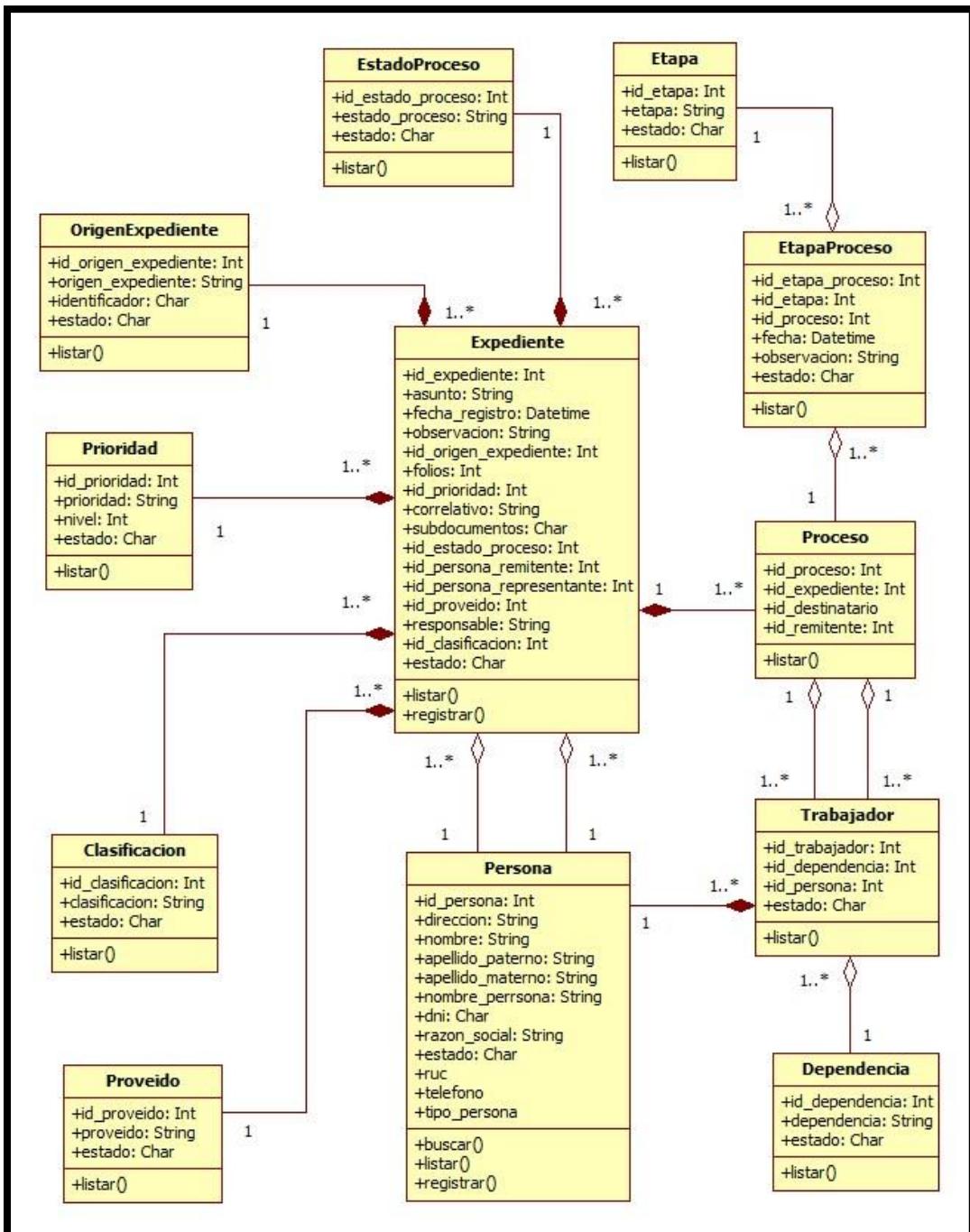


Figura 31. Diagrama de clases de entidades del módulo desarrollado  
Fuente: Elaboración propia

### 3.6.3.7. Diagramas de Componentes

Los diagramas de componentes permiten mostrar una vista física del modelo, como también los componentes de *software* y las relaciones que existen entre ellos. En la figura 32 se muestra el diagrama de componentes del módulo *Web* desarrollado con el *framework* de persistencia Hibernate ORM, utilizando los Sistemas Gestores de Base de Datos MariaDB o PostgreSQL.

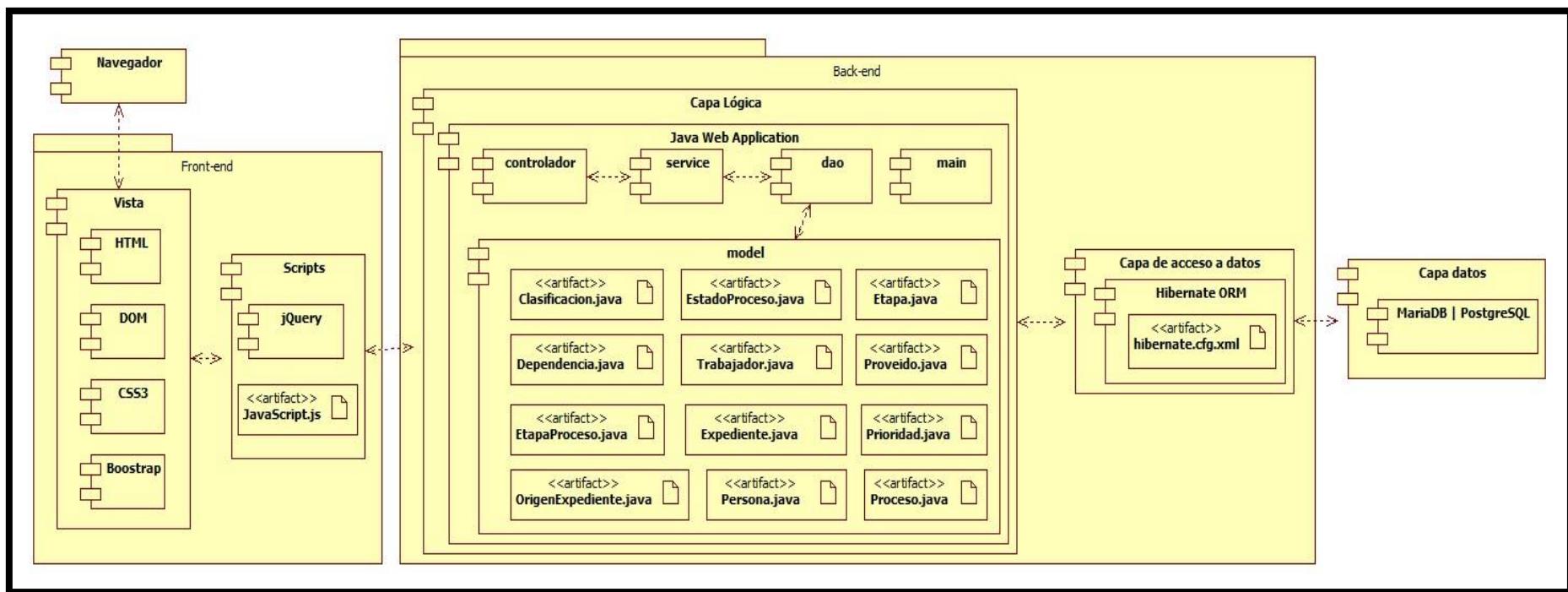


Figura 32. Diagrama de componentes de los módulos de Trámite Documentario desarrollados con Hibernate ORM.

Fuente: Elaboración propia

En la figura 33 se muestra el diagrama de componentes de los módulos *Web* desarrollados con el *framework* de persistencia Doctrine ORM, utilizando como Sistema Gestor de Base de Datos a MariaDB o PostgreSQL.

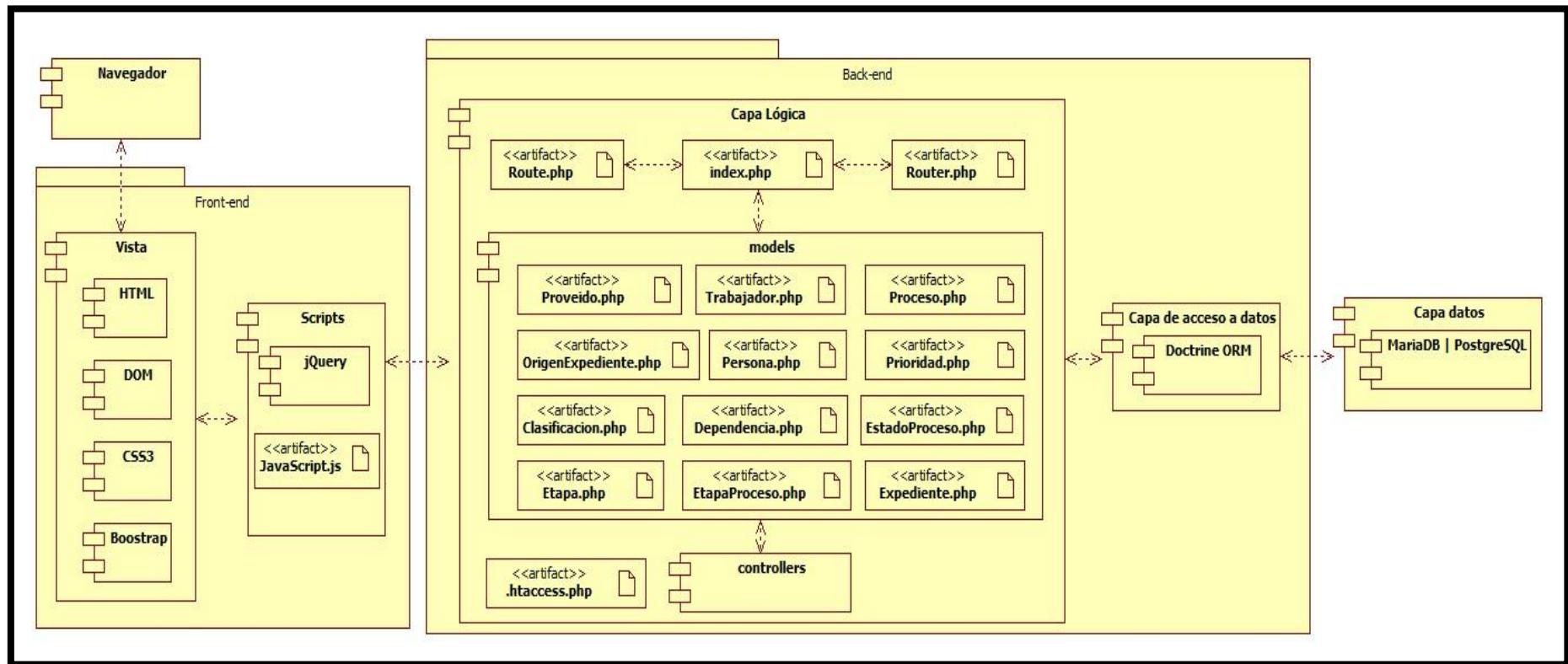


Figura 33. Diagrama de componentes de los módulos de Trámite Documentario desarrollados con Doctrine ORM.

Fuente: Elaboración propia

En la figura 34 se muestra el diagrama de componentes de los módulos *Web* desarrollados con el *framework* de persistencia Hibernate OGM, que utilizan como SGBD NoSQL a MongoDB o CouchDB.

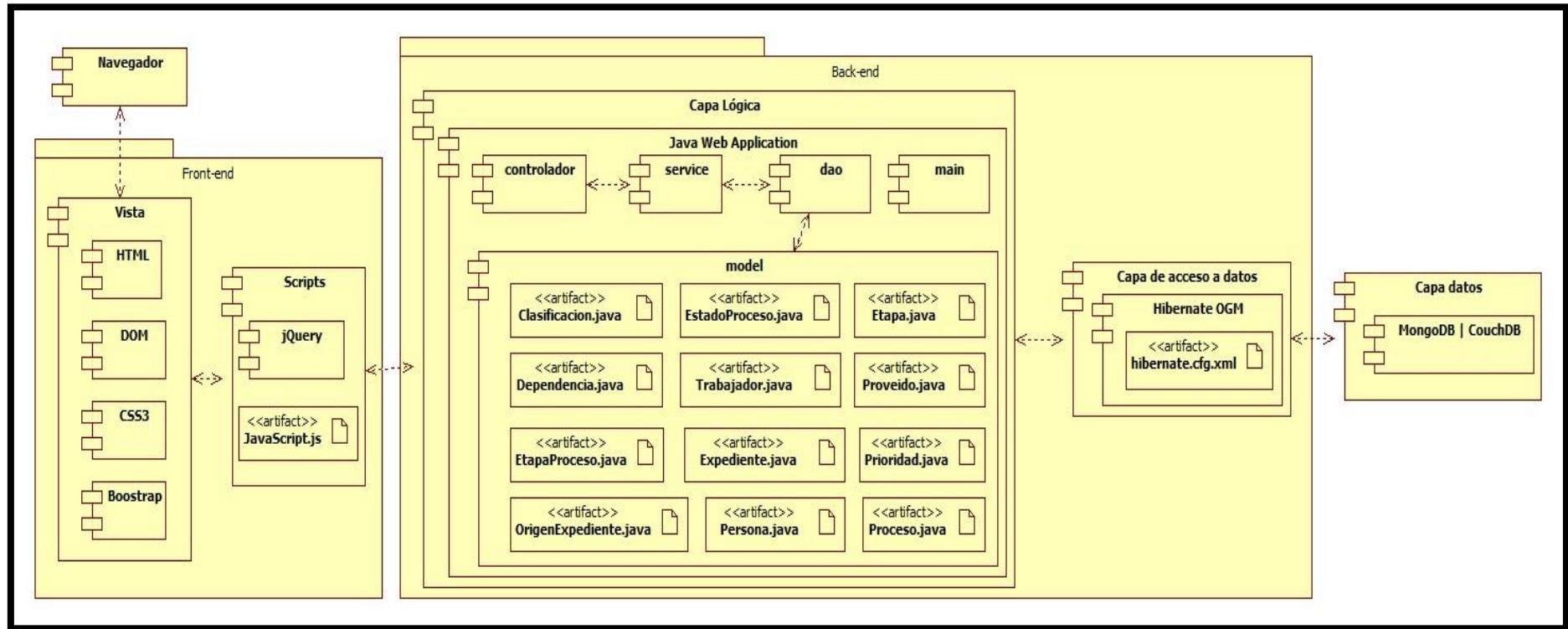


Figura 34. Diagrama de componentes de los módulos de Trámite Documentario desarrollados con Hibernate OGM

Fuente: Elaboración propia

En la figura 35 se muestra el diagrama de componentes de los módulos *Web* desarrollados con el *framework* de persistencia Doctrine ODM, que utilizan como SGBD NoSQL a MongoDB o CouchDB.

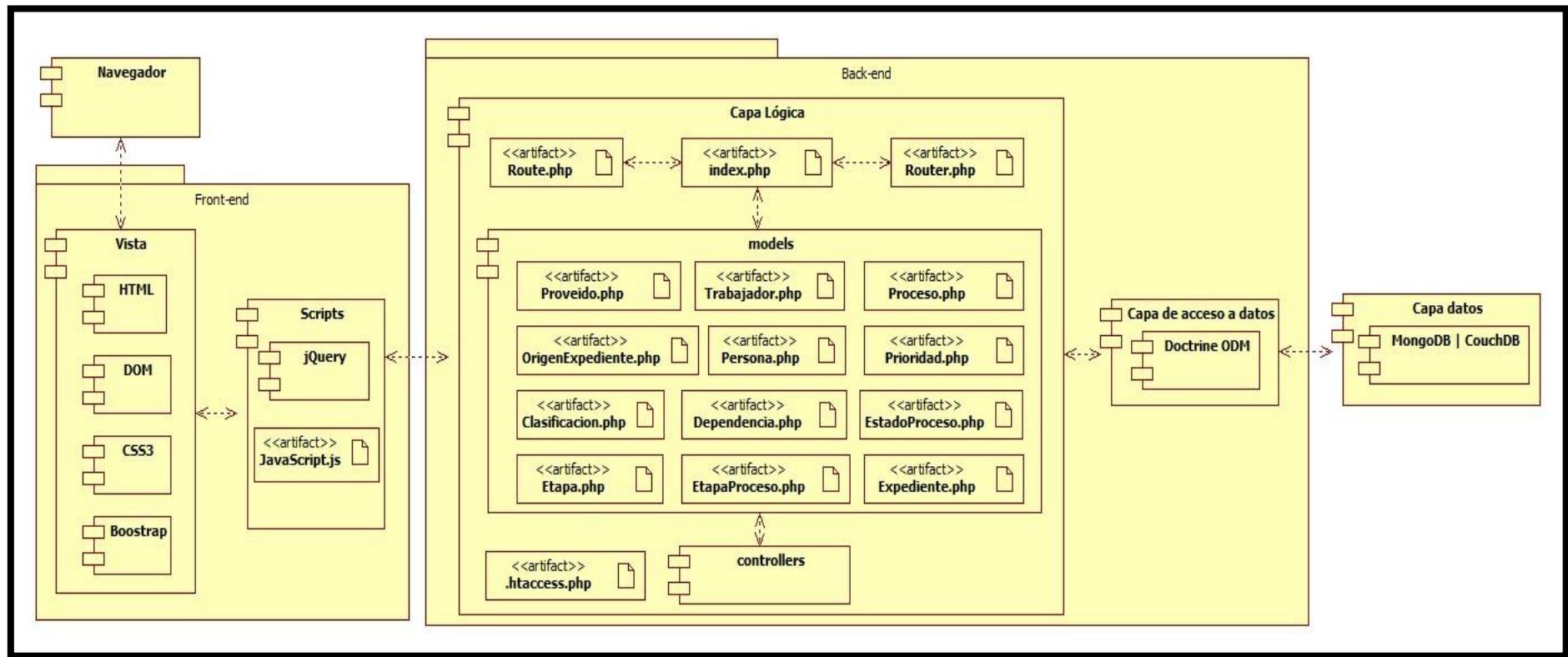


Figura 35. Diagrama de componentes de los módulos de Trámite Documentario desarrollados con Doctrine ODM  
Fuente: Elaboración propia

### **3.6.3.8. Diagramas de Despliegue**

El diagrama de despliegue muestra el diseño físico de la red (*network*) y el lugar donde residirán los componentes. En los diagramas se muestra la “Computadora 1” y “Computadora 2” que representan al recepcionista y el solicitante, respectivamente, desde estos nodos se realizan consultas, pues se comunican con el “Servidor Web” a través del “Navegador” y los equipos de red *Switch* y *Router*, además de hacer uso de una conexión de Internet para el caso del nodo “Computadora 2”, pues desde este nodo el solicitante realizará la consulta de su trámite. El nodo “Servidor Web” incluye los componentes de “Backend” y “Frontend”, es decir la aplicación propiamente dicha del Trámite Documentario, estos nodos varían dependiendo del *framework* de persistencia que se empleó para el desarrollo del módulo, se puede verificar en los diagramas que se mostrarán más adelante, además de esto debemos tener en cuenta que este nodo también se encuentra la capa de acceso a base de datos, que es el *framework* de persistencia. Por último el “Servidor Web” se comunica con el “Servidor de Base de Datos” que puede ser una base de datos relacional o una base de datos no relacional que devuelve la información que el cliente solicitó.

A continuación se muestran los diagramas de despliegue de los distintos módulos desarrollados.

En la figura 36 se muestra el diagrama de despliegue de los módulos Web desarrollados con el *framework* de persistencia Hibernate ORM, que utilizan como Sistema Gestor de Base de Datos a MariaDB o PostgreSQL.

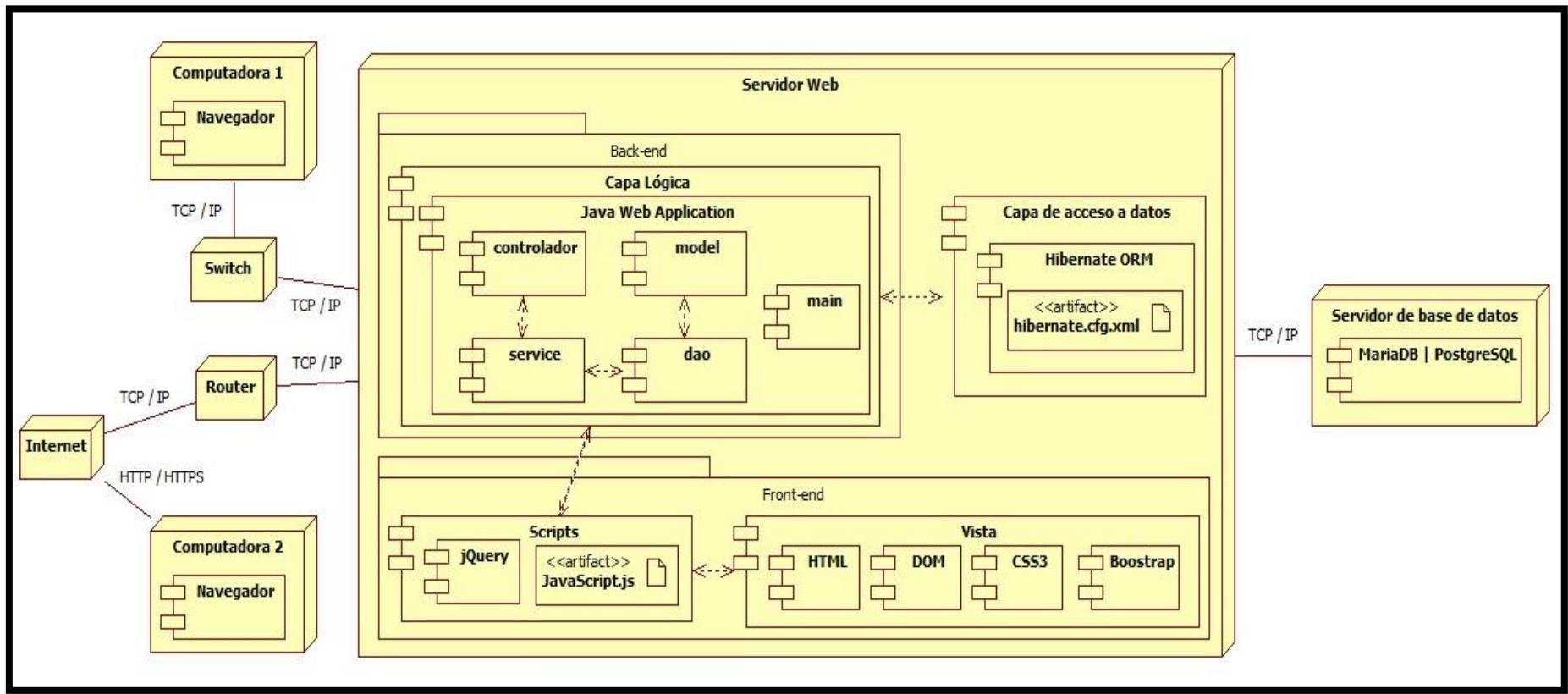


Figura 36. Diagrama de despliegue de los módulos Web desarrollados con el framework de persistencia Hibernate ORM  
Fuente: Elaboración propia

En la figura 37 se muestra el diagrama de despliegue de los módulos Web desarrollados con el *framework* de persistencia Doctrine ORM, que utilizan como Sistema Gestor de Base de Datos a MariaDB o PostgreSQL.

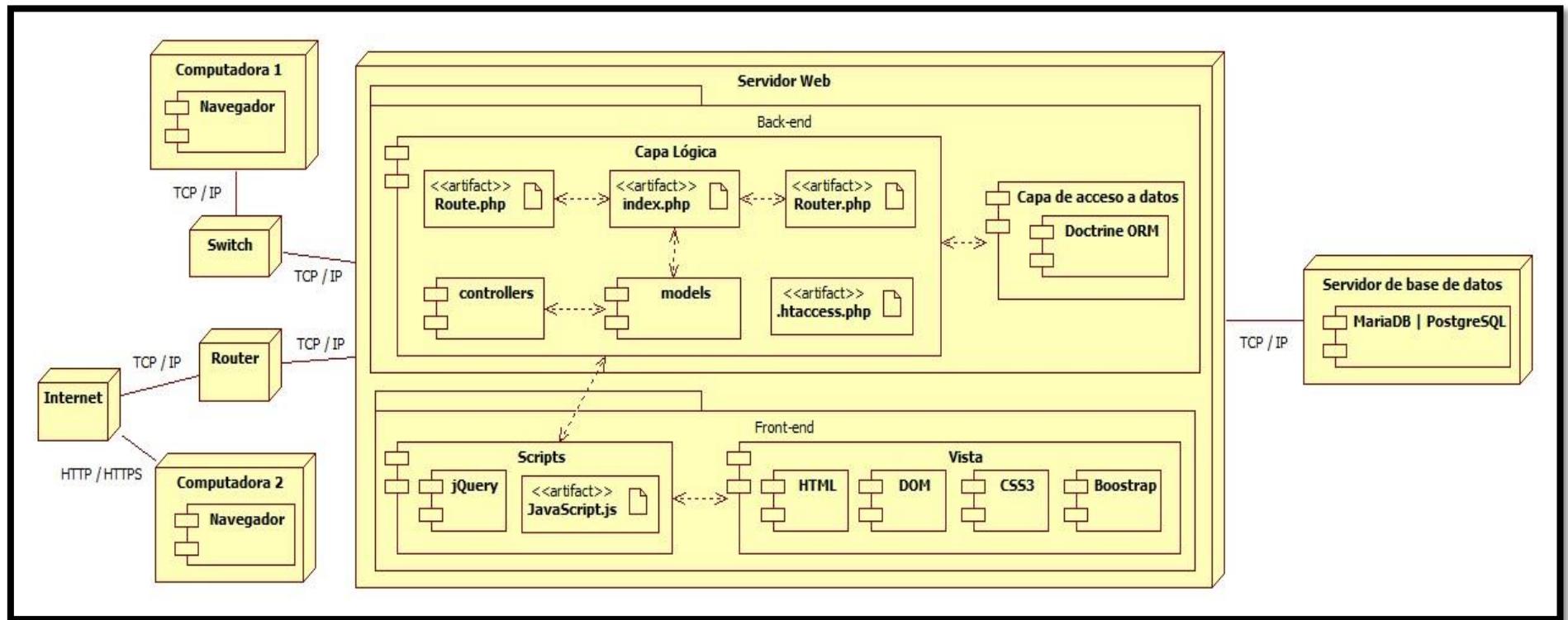


Figura 37. Diagrama de despliegue de los módulos Web desarrollados con el framework de persistencia Doctrine ORM  
Fuente: Elaboración propia

En la figura 38 se muestra el diagrama de despliegue de los módulos Web desarrollados con el *framework* de persistencia Hibernate OGM, que utilizan como SGBD NoSQL a MongoDB o CouchDB.

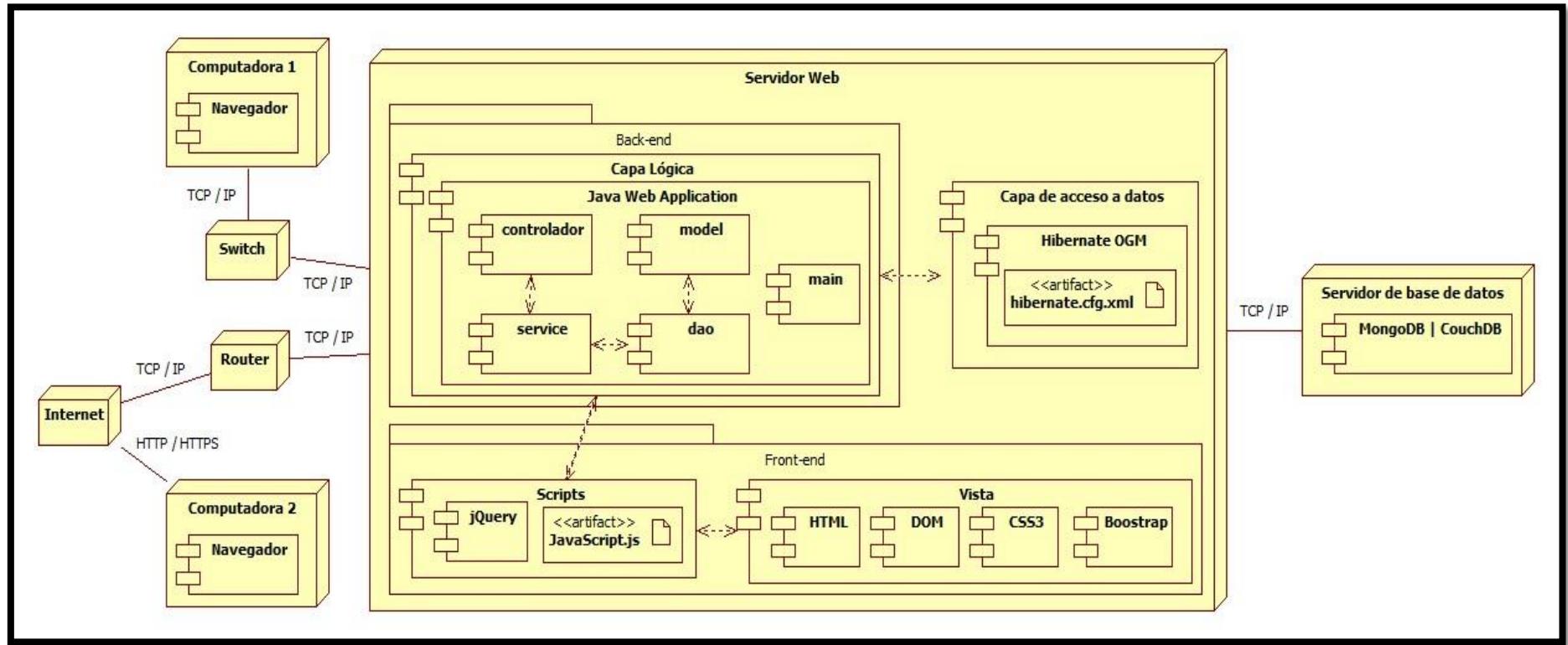


Figura 38. Diagrama de despliegue de los módulos Web desarrollados con el framework de persistencia Hibernate OGM  
Fuente: Elaboración propia

En la figura 39 se muestra el diagrama de despliegue de los módulos Web desarrollados con el *framework* de persistencia Doctrine ODM, que utilizan como SGBD NoSQL a MongoDB o CouchDB.

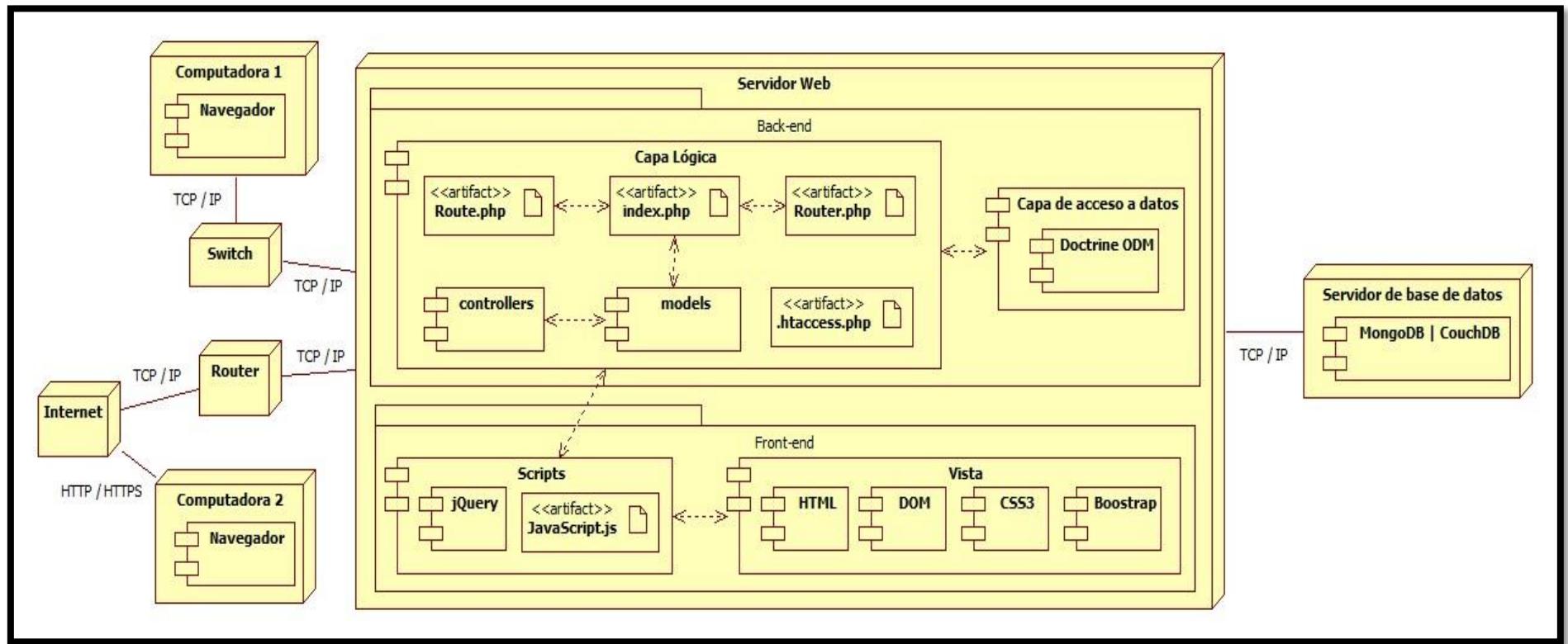


Figura 39. Diagrama de despliegue de los módulos Web desarrollados con el framework de persistencia Doctrine ODM  
Fuente: Elaboración propia

### 3.6.3.9. Diagramas físicos de la base de datos

El diagrama de la figura 40 nos muestra el modelo de la base de datos física relacional, en la que se puede visualizar como se almacenará los datos del módulo *Web* de Trámite Documentario.

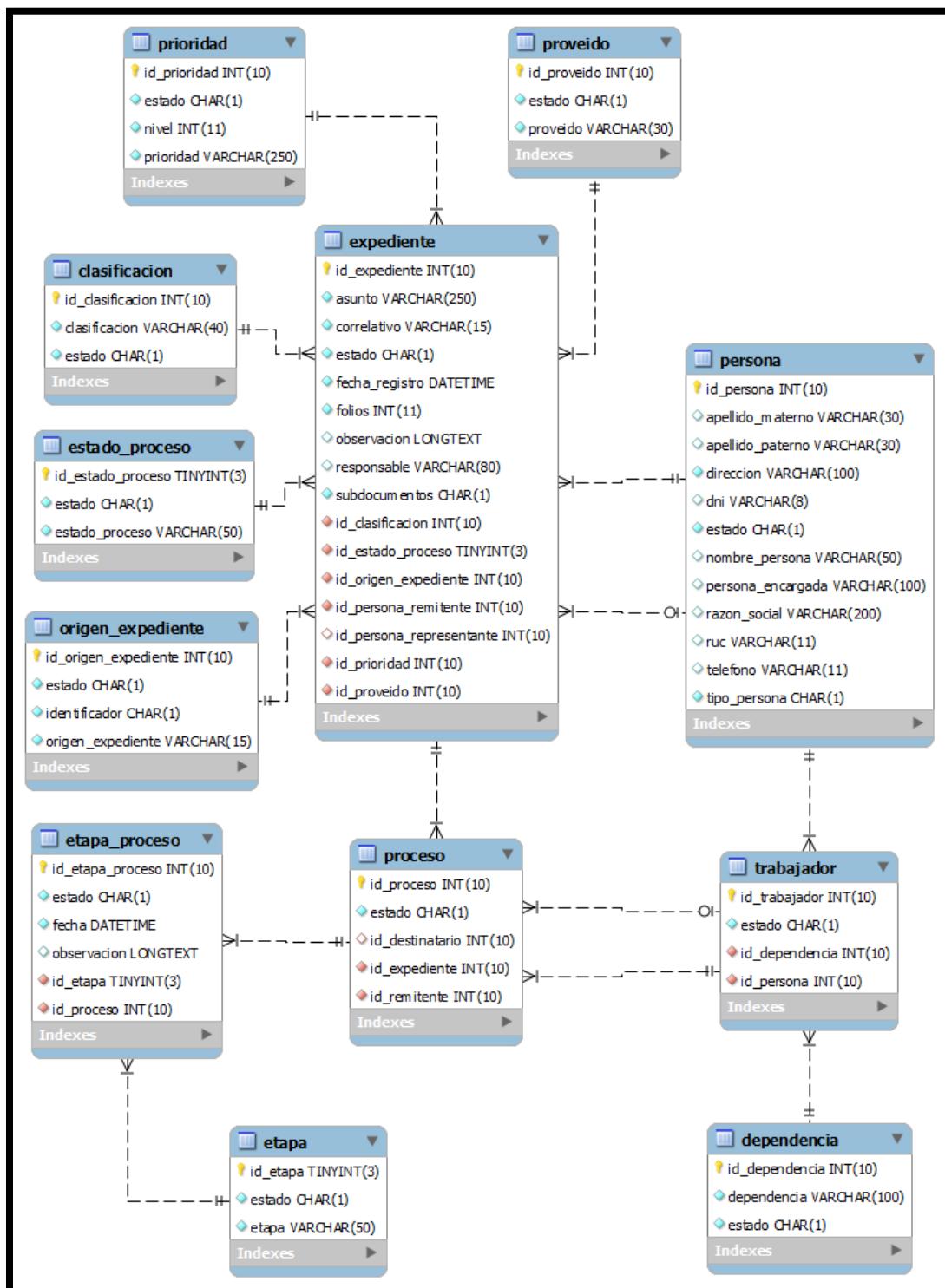


Figura 40. Diagrama físico relacional de la base de datos.  
Fuente: Elaboración propia

El diagrama de la figura 41 nos muestra el modelo de la base de datos física no relacional, en la que se puede visualizar como se almacenará los datos del módulo *Web* de Trámite Documentario.

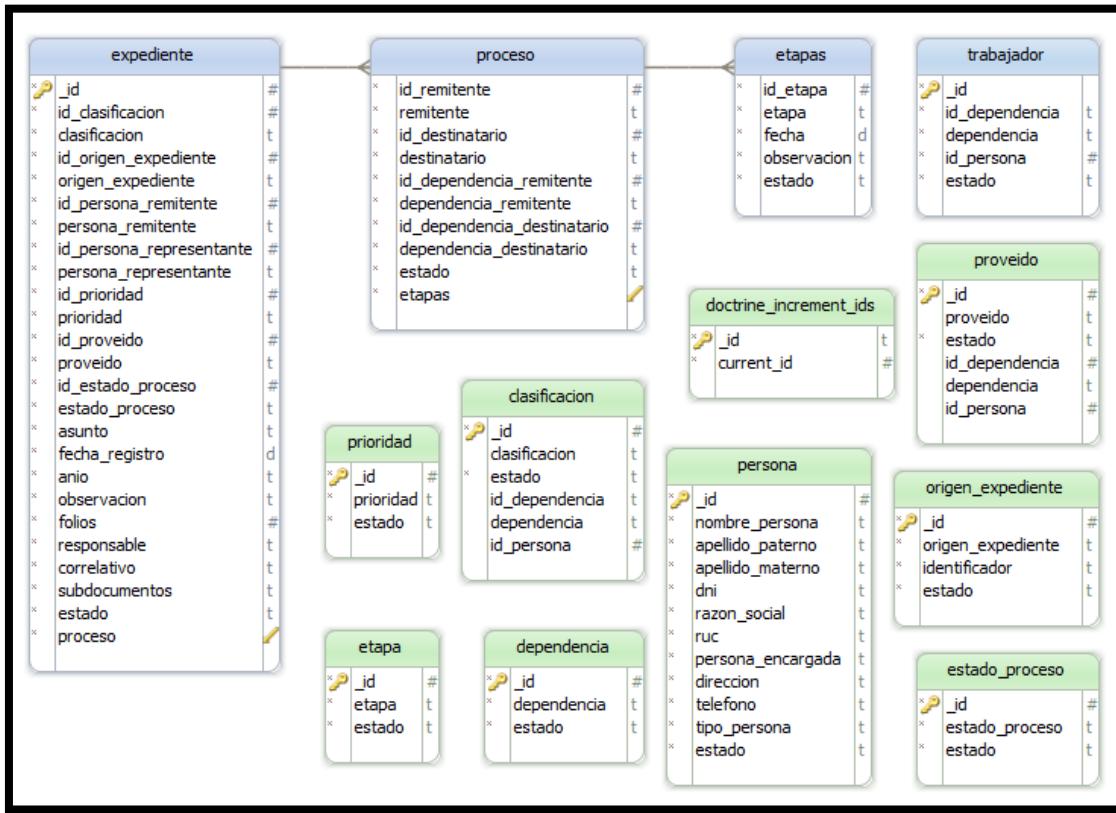


Figura 41. Diagrama físico no relacional de la base de datos.

Fuente: Elaboración propia

## **CAPÍTULO IV**

### **RESULTADOS**

#### 4.1. Presentación de resultados

A continuación se presentan los tiempos de ejecución que se obtuvieron al realizar la experimentación con los módulos desarrollados haciendo uso de los diferentes *frameworks* de persistencia ORM y ODM, en conjunto con los sistemas de base de datos SGBDR y SGBD NoSQL que se seleccionaron en el presente trabajo de investigación.

- Guías de observación: Diseño factorial de 2x2x4 – N° de réplicas: 4
- 64 combinaciones para cada operación.
- Variable de respuesta: Tiempo de ejecución (milisegundos).
- Factores: Base de datos, *Framework* de persistencia y Cantidad de datos.
- C<sub>1</sub>, C<sub>2</sub>, C<sub>3</sub> y C<sub>4</sub> son las representaciones de las cantidades de registros o documentos: 5000, 10000, 50000 y 100000, respectivamente.
- Réplicas de las operaciones: r<sub>1</sub>, r<sub>2</sub>, r<sub>3</sub> y r<sub>4</sub>.

##### ❖ **Tiempos de ejecución de los módulos desarrollados con *frameworks* de persistencia ORM**

**Operación 1:** Resultados de la operación “Registro de expediente” en milisegundos, para la combinación *framework* de persistencia ORM y SGBDR.

		MariaDB				PostgreSQL			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
Hibernate ORM	r <sub>1</sub>	30309	115181	265005	449170	3097	6185	30193	61737
	r <sub>2</sub>	42498	133884	271044	584458	3099	6384	31256	62236
	r <sub>3</sub>	46312	131550	287573	660674	3070	6301	29915	61503
	r <sub>4</sub>	45392	145833	248394	671948	3218	6667	31174	61755
Doctrine ORM	r <sub>1</sub>	61377	243073	1334994	3223714	6086	14259	124730	341561
	r <sub>2</sub>	84716	262506	1489452	3805272	5754	14407	124936	341958
	r <sub>3</sub>	110139	278231	1577601	4139691	5761	14428	124936	341459
	r <sub>4</sub>	134955	238743	1536174	3789469	5917	14352	123903	341480

Tabla 18. Resultados en milisegundos de la operación “Registro de expedientes” tomados de los módulos desarrollados con los frameworks de persistencia ORM.

Fuente: Elaboración propia

**Operación 2:** Resultados de la operación “Consulta de Bandeja de Entrada” en milisegundos, para la combinación *Framework* de persistencia ORM y SGBDR.

		MariaDB				PostgreSQL			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
<b>Hibernate ORM</b>	r <sub>1</sub>	3893	4125	24235	93550	691	464	638	8861
	r <sub>2</sub>	3863	4126	23374	56424	655	275	649	1868
	r <sub>3</sub>	3873	4032	23464	56190	715	339	571	1791
	r <sub>4</sub>	3446	3893	23215	55992	695	472	579	1863
<b>Doctrine ORM</b>	r <sub>1</sub>	5871	7989	26100	236381	174	400	415	998
	r <sub>2</sub>	7229	7713	23617	238205	182	336	421	635
	r <sub>3</sub>	5547	7871	24253	232309	181	227	402	1925
	r <sub>4</sub>	6202	7713	23804	230392	173	211	541	1162

Tabla 19. Resultados en milisegundos para la operación “Consulta de bandeja de entrada” tomados de los módulos desarrollados con los frameworks de persistencia ORM.

Fuente: Elaboración propia

❖ **Tiempos de ejecución de los módulos desarrollados con frameworks de persistencia ODM**

**Operación 1:** Resultados de la operación “Registro de expediente” en milisegundos, para la combinación *framework* de persistencia ODM y SGBD NoSQL.

		MongoDB				CouchDB			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
<b>Hibernate OGM</b>	r <sub>1</sub>	2389	3647	15794	35687	1751638	3509129	11258988	30251475
	r <sub>2</sub>	2168	3623	15480	34677	1760395	4382855	12582358	25893578
	r <sub>3</sub>	2087	3620	15848	37014	1925297	5005157	11025892	38025574
	r <sub>4</sub>	2183	3634	15977	35695	2090014	3968612	12478598	27893657
<b>Doctrine ODM</b>	r <sub>1</sub>	3554	8357	86232	204173	641557	1228194	6650467	12540258
	r <sub>2</sub>	3637	8403	81807	205021	655858	1783118	6778366	15033678
	r <sub>3</sub>	3523	8720	80121	205933	681717	1968410	6559568	17023698
	r <sub>4</sub>	3585	8406	79666	206711	640076	1410309	6484262	18025874

Tabla 20. Resultados en milisegundos para la operación “Registro de Expediente” tomados de los módulos desarrollados con los frameworks de persistencia ODM.

Fuente: Elaboración propia

**Operación 2:** Resultados de la operación “Consulta de Bandeja de Entrada” en milisegundos, para la combinación *framework* de persistencia ODM y SGBD NoSQL.

		MongoDB				CouchDB			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
<b>Hibernate OGM</b>	r <sub>1</sub>	1084	874	3618	10725	21358	32585	45025	100258
	r <sub>2</sub>	784	1241	3777	10733	20589	36895	42589	92358
	r <sub>3</sub>	798	1001	3794	10486	21478	29866	47569	98565
	r <sub>4</sub>	926	971	4001	10238	19785	30478	46325	97025
<b>Doctrine ODM</b>	r <sub>1</sub>	162	253	1188	2135	2587	3587	9875	20258
	r <sub>2</sub>	158	264	1010	2225	2369	3145	10258	26895
	r <sub>3</sub>	158	250	1024	2164	4756	3025	8975	30258
	r <sub>4</sub>	161	254	1041	2130	3258	2978	9900	32587

Tabla 21. Resultados en milisegundos para la operación “Consulta de bandeja de entrada” tomados de los módulos desarrollados con los frameworks de persistencia ODM.

Fuente: Elaboración propia

Los datos que hemos obtenido de la experimentación no se ajustan a un modelo estadístico lineal, y esto se verificó a través del test de Kolmogorov – Smirnov. Se realizó entonces una transformación de datos para de este modo ajustarlos a un modelo lineal.

Al transformar los datos a través del logaritmo en base 10 se obtuvieron los resultados ajustados a una distribución normal, estos se mostraran a continuación:

- ❖ **Tiempos de ejecución transformados de los módulos desarrollados con los frameworks de persistencia ORM**

**Operación 1:** Resultados transformados a través de logaritmo en base 10 de la operación “Registro de expediente”, para la combinación *framework* de persistencia ORM y SGBDR.

		MariaDB				PostgreSQL			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
<b>Hibernate ORM</b>	r <sub>1</sub>	4.482	5.061	5.423	5.652	3.491	3.791	4.480	4.791
	r <sub>2</sub>	4.628	5.127	5.433	5.767	3.491	3.805	4.495	4.794
	r <sub>3</sub>	4.666	5.119	5.459	5.820	3.487	3.799	4.476	4.789
	r <sub>4</sub>	4.657	5.164	5.395	5.827	3.508	3.824	4.494	4.791
<b>Doctrine ORM</b>	r <sub>1</sub>	4.788	5.386	6.125	6.508	3.784	4.154	5.096	5.533
	r <sub>2</sub>	4.928	5.419	6.173	6.580	3.760	4.159	5.097	5.534
	r <sub>3</sub>	5.042	5.444	6.198	6.617	3.760	4.159	5.097	5.533
	r <sub>4</sub>	5.130	5.378	6.186	6.579	3.772	4.157	5.093	5.533

Tabla 22. Resultados transformados de la operación “Registro de expedientes” tomados de los módulos desarrollados con los frameworks de persistencia ORM.

Fuente: Elaboración propia

**Operación 2:** Resultados transformados a través de logaritmo en base 10 de la operación “Consulta de Bandeja de Entrada”, para la combinación *framework* de persistencia ORM y SGBDR.

		MariaDB				PostgreSQL			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
<b>Hibernate ORM</b>	r <sub>1</sub>	3.590	3.615	4.384	4.971	2.839	2.667	2.805	3.947
	r <sub>2</sub>	3.587	3.616	4.369	4.751	2.816	2.439	2.812	3.271
	r <sub>3</sub>	3.588	3.606	4.370	4.750	2.854	2.530	2.757	3.253
	r <sub>4</sub>	3.537	3.590	4.366	4.748	2.842	2.674	2.763	3.270
<b>Doctrine ORM</b>	r <sub>1</sub>	3.769	3.902	4.417	5.374	2.241	2.602	2.618	2.999
	r <sub>2</sub>	3.859	3.887	4.373	5.377	2.260	2.526	2.624	2.803
	r <sub>3</sub>	3.744	3.896	4.385	5.366	2.258	2.356	2.604	3.284
	r <sub>4</sub>	3.793	3.887	4.377	5.362	2.238	2.324	2.733	3.065

Tabla 23. Resultados transformados de la operación “Consulta de bandeja de entrada” tomados de los módulos desarrollados con los frameworks de persistencia ORM.

Fuente: Elaboración propia

❖ **Presentación de tiempos de ejecución transformados de los módulos desarrollados con los frameworks de persistencia ODM**

**Operación 1:** Resultados transformados a través de logaritmo en base 10 de la operación “Registro de expediente”, para la combinación *framework* de persistencia ODM y SGBD NoSQL.

		MongoDB				CouchDB			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
<b>Hibernate OGM</b>	r <sub>1</sub>	3.378	3.562	4.198	4.553	6.243	6.545	7.051	7.481
	r <sub>2</sub>	3.336	3.559	4.190	4.540	6.246	6.642	7.100	7.413
	r <sub>3</sub>	3.320	3.559	4.200	4.568	6.284	6.699	7.042	7.580
	r <sub>4</sub>	3.339	3.560	4.203	4.553	6.320	6.599	7.096	7.446
<b>Doctrine ODM</b>	r <sub>1</sub>	3.551	3.922	4.936	5.310	5.807	6.089	6.823	7.098
	r <sub>2</sub>	3.561	3.924	4.913	5.312	5.817	6.251	6.831	7.177
	r <sub>3</sub>	3.547	3.941	4.904	5.314	5.834	6.294	6.817	7.231
	r <sub>4</sub>	3.554	3.925	4.901	5.315	5.806	6.149	6.812	7.256

Tabla 24. Resultados transformados de la operación “Registro de Expediente” tomados de los módulos desarrollados con los frameworks de persistencia ODM.

Fuente: Elaboración propia

**Operación 2:** Resultados transformados a través de logaritmo en base 10 de la operación “Consulta de Bandeja de Entrada”, para la combinación *framework* de persistencia ODM y SGBD NoSQL.

		MongoDB				CouchDB			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
<b>Hibernate OGM</b>	r <sub>1</sub>	3.035	2.942	3.558	4.030	4.330	4.513	4.653	5.001
	r <sub>2</sub>	2.894	3.094	3.577	4.031	4.314	4.567	4.629	4.965
	r <sub>3</sub>	2.902	3.000	3.579	4.021	4.332	4.475	4.677	4.994
	r <sub>4</sub>	2.967	2.987	3.602	4.010	4.296	4.484	4.666	4.987
<b>Doctrine ODM</b>	r <sub>1</sub>	2.210	2.403	3.075	3.329	3.413	3.555	3.995	4.307
	r <sub>2</sub>	2.199	2.422	3.004	3.347	3.375	3.498	4.011	4.430
	r <sub>3</sub>	2.199	2.398	3.010	3.335	3.677	3.481	3.953	4.481
	r <sub>4</sub>	2.207	2.405	3.017	3.328	3.513	3.474	3.996	4.513

Tabla 25. Resultados transformados de la operación “Consulta de bandeja de entrada” tomados de los módulos desarrollados con los frameworks de persistencia ODM.

Fuente: Elaboración propia

## 4.2. Análisis de resultados

El análisis estadístico de los resultados obtenidos que se utilizó fue el análisis de varianza de tres factores con varias muestras por grupo, para las operaciones de inserción y lectura de datos en los módulos *Web* de Trámite Documentario desarrollados. De esta manera se prueban las hipótesis de que existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por las variables de manera independiente e interactuando unas con otras.

Además del análisis de varianza se realizará un análisis de tipo no paramétrico para corroborar los resultados, este test es llamado Kruskal – Wallis. Este test se utiliza porque en esta investigación el uso de la curva normal no es apropiado, pues tal como se concluyó al hacer la prueba de Kolmogorov – Smirnov (Anexo 9) todos de los tiempos de ejecución difieren de una distribución normal. Se debe tener en cuenta que para el test de Kruskal – Wallis se utilizan los resultados sin realizar la transformación logarítmica.

### Datos para el análisis:

#### ❖ *Frameworks de persistencia ORM:*

A continuación en la tabla 26, se muestran los datos que se utilizaron para llevar a cabo el análisis estadístico de los resultados obtenidos de los tiempos de ejecución en cada operación que se simularon para los módulos *Web* de Trámite Documentario. Estos módulos fueron desarrollados haciendo uso de los *frameworks* de persistencia ORM y utilizando SGBDR.

Ítem	Descripción
<b>Factor A</b>	Sistema Gestor de Base de Datos.
<b>Factor B</b>	<i>Framework</i> de persistencia ORM
<b>Factor C</b>	Cantidad de datos
<b>Variable Dependiente</b>	Tiempo de ejecución en cada operación.

<b>Nº de réplicas o repeticiones de la prueba</b>	n = 4
<b>Nº de niveles del factor A</b>	a = 2
<b>Nº de niveles del factor B</b>	b = 2
<b>Nº de niveles del factor C</b>	c = 4
<b>Nº total de observaciones realizadas</b>	N = 64
<b>Diseño experimental utilizado para el análisis</b>	Diseño factorial 2x2x4
<b>Nivel de Significancia (alfa)</b>	5%

Tabla 26. Datos para el análisis de los resultados obtenidos para los frameworks de persistencia ORM.

Fuente: Elaboración propia

**Prueba de Hipótesis:** En las tablas 27 y 28 se muestran las pruebas de hipótesis que se utilizaron para el análisis estadístico de los resultados obtenidos.

<b>Respecto al efecto de los factores principales</b>	
<b>Factores</b>	<b>Hipótesis</b>
Factor A: Sistema Gestor de Base de Datos.	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por el Sistema Gestor de Base de Datos utilizado, es decir, su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por el Sistema Gestor de Base de Datos utilizado, es decir, su efecto es diferente de cero.</p>
Factor B: Framework de persistencia ORM	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por el framework de persistencia utilizado, es decir, su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por el framework de persistencia utilizado, es decir, su efecto es diferente de cero.</p>

Factor C: Cantidad de datos	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la cantidad de datos que se registran o consultan en una base de datos, es decir, su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la cantidad de datos que se registran o consultan en una base de datos, es decir, su efecto es diferente de cero.</p>
-----------------------------	---

Tabla 27. Prueba de hipótesis respecto al efecto de los factores principales.

Fuente: Elaboración propia

Respecto a la interacción de los factores principales	
Interacción	Hipótesis
Interacción AB	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBDR y <i>framework</i> de persistencia, es decir, cuando los factores actúan de manera conjunta su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBDR y <i>framework</i> de persistencia, es decir, su efecto es diferente de cero.</p>
Interacción AC	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBDR y Cantidad de datos, es decir, cuando los factores actúan de manera conjunta su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBDR y Cantidad de datos, es decir, su efecto es diferente de cero.</p>
Interacción BC	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción <i>framework</i> de Persistencia ORM y Cantidad</p>

	<p>de datos, es decir, cuando los factores actúan de manera conjunta su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción <i>framework</i> de Persistencia ORM y Cantidad de datos, es decir, su efecto es diferente de cero.</p>
Interacción ABC	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBDR, <i>framework</i> de persistencia y Cantidad de datos, es decir, cuando los factores actúan de manera conjunta su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBDR, <i>framework</i> de persistencia y Cantidad de datos, es decir, cuando los factores actúan de manera conjunta su efecto es diferente de cero.</p>

Tabla 28. Prueba de hipótesis respecto a la interacción de los factores principales.

Fuente: Elaboración propia

**Conclusión:** Cuando el valor de “F Calculado” es mayor que el valor de “F de tabla”, se rechaza H<sub>0</sub>.

A continuación, se muestra el análisis varianza (ANOVA) por cada operación simulada en los módulos *Web* de Trámite Documentario desarrollados con los *frameworks* de persistencia ORM.

## Operación 1: Registro de Expediente.

		MariaDB				PostgreSQL			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
<b>Hibernate ORM</b>	r <sub>1</sub>	4.482	5.061	5.423	5.652	3.491	3.791	4.480	4.791
	r <sub>2</sub>	4.628	5.127	5.433	5.767	3.491	3.805	4.495	4.794
	r <sub>3</sub>	4.666	5.119	5.459	5.820	3.487	3.799	4.476	4.789
	r <sub>4</sub>	4.657	5.164	5.395	5.827	3.508	3.824	4.494	4.791
<b>Doctrine ORM</b>	r <sub>1</sub>	4.788	5.386	6.125	6.508	3.784	4.154	5.096	5.533
	r <sub>2</sub>	4.928	5.419	6.173	6.580	3.760	4.159	5.097	5.534
	r <sub>3</sub>	5.042	5.444	6.198	6.617	3.760	4.159	5.097	5.533
	r <sub>4</sub>	5.130	5.378	6.186	6.579	3.772	4.157	5.093	5.533

Tabla 29. Tiempos de ejecución transformados de la operación “Registro de operación” para los frameworks de persistencia ORM.

Fuente: Elaboración propia

Fuente de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F Calculado	F Tabla Alfa=5%	Conclusión
Factor A: Framework de Persistencia ORM	4.366	1	4.366	1630.293	4.0427	Rechazo H <sub>0</sub>
Factor B: Sistema Gestor de Base de Datos	19.840	1	19.840	7408.498	4.0427	Rechazo H <sub>0</sub>
Factor C: Cantidad de datos	20.561	3	6.854	2559.251	2.7981	Rechazo H <sub>0</sub>
Interacción AB	0.12	1	0.12	4.580	4.0427	Rechazo H <sub>0</sub>
Interacción AC	0.674	3	0.225	83.906	2.7981	Rechazo H <sub>0</sub>
Interacción BC	0.211	3	0.070	26.260	2.7981	Rechazo H <sub>0</sub>
Interacción ABC	0.21	3	0.007	2.666	2.7981	Acepto H <sub>0</sub>
Error	0.129	48	0.003			
Total	1612.866	64				

Tabla 30. Resultados del análisis de varianza para la operación 1

Fuente: IBM SPSS Statistics 21

Hipótesis nula (H <sub>0</sub> ) / Nivel de significancia de 5%	Sig.	Conclusión
La distribución de tiempo de ejecución es la misma entre las categorías de Framework de persistencia	0.025	Rechazar H <sub>0</sub>
La distribución de tiempo de ejecución es la misma entre las categorías de Sistema Gestor de Base de Datos	0.000	Rechazar H <sub>0</sub>
La distribución de tiempo de ejecución es la misma entre las categorías de Cantidad de datos	0.000	Rechazar H <sub>0</sub>

Tabla 31. Resultados del Test de Kruskal – Wallis para la operación 1

Fuente: IBM SPSS Statistics 21

En la tabla 30 se muestra el análisis de varianza para la operación 1, donde se busca demostrar si los módulos *Web* de Trámite Documentario desarrollados con los *frameworks* de persistencia ORM y usando SGBDR, afectan significativamente sobre el tiempo de ejecución de la operación que se está evaluando. En dicha tabla se observa que el factor “Cantidad de datos” es el que produce una mayor variabilidad en el tiempo de ejecución ( $\sigma^2 = 20.561$ ).

Además, se comprueba que al 5% de nivel de significancia, los factores: *Framework* de persistencia ( $F_c = 1630.293$ ), Sistema Gestor de Base de Datos ( $F_c = 7408.498$ ), los y la Cantidad de datos ( $F_c = 2559.251$ ), actuando de manera independiente, afectan significativamente al tiempo de ejecución de la operación, esto se puede corroborar con los resultados emitidos por el test de Kruskal – Wallis. Para las interacciones de dos factores, AB ( $F_c = 4.580$ ), BC ( $F_c = 26.260$ ) y AC ( $F_c = 83.906$ ), afectan significativamente al tiempo de ejecución de la operación. Pero, la interacción de los tres factores ABC ( $F_c = 164.369$ ) no afecta de manera significativa al tiempo de ejecución de la operación. Entonces, se concluye que se aceptan las hipótesis  $H_1$ , formuladas en las tablas 27 y 28, es decir, existen diferencias significativas en el tiempo de ejecución para los factores y las interacciones entre dos factores pero, para la interacción de los tres factores se acepta la hipótesis  $H_0$ , es decir, no existe diferencia significativa en el tiempo de ejecución.

## Operación 2: Consulta de la Bandeja de Entrada

		MariaDB				PostgreSQL			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
<b>Hibernate ORM</b>	r <sub>1</sub>	3.590	3.615	4.384	4.971	2.839	2.667	2.805	3.947
	r <sub>2</sub>	3.587	3.616	4.369	4.751	2.816	2.439	2.812	3.271
	r <sub>3</sub>	3.588	3.606	4.370	4.750	2.854	2.530	2.757	3.253
	r <sub>4</sub>	3.537	3.590	4.366	4.748	2.842	2.674	2.763	3.270
<b>Doctrine ORM</b>	r <sub>1</sub>	3.769	3.902	4.417	5.374	2.241	2.602	2.618	2.999
	r <sub>2</sub>	3.859	3.887	4.373	5.377	2.260	2.526	2.624	2.803
	r <sub>3</sub>	3.744	3.896	4.385	5.366	2.258	2.356	2.604	3.284
	r <sub>4</sub>	3.793	3.887	4.377	5.362	2.238	2.324	2.733	3.065

Tabla 32. Tiempos de ejecución en milisegundos de la operación “Consulta de Bandeja de Entrada” para los frameworks de persistencia ORM  
Fuente: Elaboración propia

Fuente de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F Calculado	F Tabla Alfa=5%	Conclusión
Factor A: Framework de persistencia ORM	0.007	1	0.007	0.548	4.0427	Acepto H <sub>0</sub>
Factor B: Sistema Gestor de Base de Datos	34.170	1	34.710	2678.449	4.0427	Rechazo H <sub>0</sub>
Factor C: Cantidad de datos	11.592	3	3.684	298.181	2.7981	Rechazo H <sub>0</sub>
Interacción AB	1.361	1	1.361	105.048	4.0427	Rechazo H <sub>0</sub>
Interacción AC	0.201	3	0.067	5.169	2.7981	Rechazo H <sub>0</sub>
Interacción BC	1.390	3	0.463	35.749	2.7981	Rechazo H <sub>0</sub>
Interacción ABC	0.405	3	0.135	10.420	2.7981	Rechazo H <sub>0</sub>
Error	0.622	48	0.13			
Total	829.256	64				

Tabla 33. Resultados del análisis de varianza para la operación 2

Fuente: IBM SPSS Statistics 21

Hipótesis nula (H <sub>0</sub> ) / Nivel de significancia de 5%	Sig.	Conclusión
La distribución de tiempo de ejecución es la misma entre las categorías de Framework de persistencia	0.667	Aceptar H <sub>0</sub>
La distribución de tiempo de ejecución es la misma entre las categorías de Sistema Gestor de Base de Datos	0.000	Rechazar H <sub>0</sub>
La distribución de tiempo de ejecución es la misma entre las categorías de Cantidad de datos	0.007	Rechazar H <sub>0</sub>

Tabla 34. Resultados del Test de Kruskal – Wallis para la operación 2

Fuente: IBM SPSS Statistics 21

En la tabla 33 se muestra el análisis de varianza para la operación 2, donde se busca demostrar si los módulos *Web* de Trámite Documentario desarrollados con los *frameworks* de persistencia ORM, y usando SGBDR relacionales afectan significativamente sobre el tiempo de ejecución de la operación que se está evaluando. En dicha tabla se observa que el factor “Sistema Gestor de Base Datos” es el que produce una mayor variabilidad en el tiempo de ejecución ( $\sigma^2 = 34.170$ ).

Además, se comprueba que al 5% de nivel de significancia, los factores: Sistemas Gestores de Base de datos ( $F_c = 2678.449$ ) y Cantidad de Datos ( $F_c = 298.181$ ), actuando de manera independiente, afectan significativamente al tiempo de ejecución de la operación, pero no sucede lo mismo el factor “framework de persistencia” ( $F_c = 0.548$ ), pues de manera independiente no afecta

significativamente al tiempo de ejecución, lo dicho es corroborado por el test de Kruskal – Wallis. Para las interacciones de dos factores, AB ( $F_c = 105.048$ ), BC ( $F_c = 10.420$ ) y AC ( $F_c = 5.169$ ), afectan significativamente al tiempo de ejecución de la operación, lo mismo sucede con la interacción de los tres factores ABC ( $F_c = 10.420$ ); con lo que se concluye que se aceptan las hipótesis  $H_1$ , formuladas en las tablas 27 y 28, es decir, existen diferencias significativas en el tiempo de ejecución para los factores y las interacciones, excepto por el factor “*Framework de persistencia*”, para este factor se acepta la hipótesis  $H_0$ , es decir, no existe diferencia significativa en el tiempo de ejecución.

Después de haber realizado el análisis estadístico de los tiempos de ejecución por cada operación, se realizó un resumen de las conclusiones que se obtuvieron en cada operación por cada factor e interacción entre ellos, las cuales se muestran a continuación en la tabla 35.

	Operaciones		Conclusión de Hipótesis $H_0$
	1	2	
<b>Factor A: Framework de persistencia</b>	Rechazo $H_0$	Acepto $H_0$	
<b>Factor B: Sistema Gestor de Base de Datos</b>	Rechazo $H_0$	Rechazo $H_0$	
<b>Factor C: Cantidad de datos</b>	Rechazo $H_0$	Rechazo $H_0$	
<b>Interacción AB</b>	Rechazo $H_0$	Rechazo $H_0$	
<b>Interacción AC</b>	Rechazo $H_0$	Rechazo $H_0$	
<b>Interacción BC</b>	Rechazo $H_0$	Rechazo $H_0$	
<b>Interacción ABC</b>	Acepto $H_0$	Rechazo $H_0$	

Tabla 35. Resumen del análisis estadístico de los tiempos de ejecución de cada operación para los módulos desarrollados con frameworks de persistencia ORM

Fuente: Elaboración propia

En la tabla 35 se determinó que en el 50% de las operaciones analizadas existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por el factor “*Framework de persistencia*”. Además, que para el factor “Sistema Gestor de Base de Datos”, en el 100% de las operaciones analizadas se concluyó que existen diferencias significativas en el tiempo de ejecución de cada operación, ocasionada por el ORM utilizado. Por último, para el factor “Cantidad de datos”, en el 100% de las operaciones analizadas se concluyó que existen diferencias

significativas en el tiempo de ejecución ocasionada por la cantidad de datos que se desea consultar o registrar.

Para las interacciones de los factores se puede concluir que la interacción Sistema Gestor de Base de Datos – *Framework* de persistencia, en el 100% de las operaciones analizadas se concluyó que existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción de los factores. Además para la interacción Sistema Gestor de Base de Datos – Cantidad de datos, en el 100% de las operaciones analizadas se concluyó que existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción de los factores. Por último, para la interacción *Framework* de persistencia – Cantidad de datos, en el 100% de las operaciones analizadas se concluyó que existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción de los factores.

Para la interacción entre los tres factores: Gestor de Base de Datos, *Framework* de persistencia y Cantidad de datos, en el 50% de las operaciones analizadas se concluyó que existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción de los tres factores.

Para determinar cuál es el *framework* de persistencia ORM que ocasiona la diferencia significativa en el tiempo de ejecución de las operaciones de los módulos *Web* desarrollados, utilizando un Sistema Gestor de Base de Datos y que contiene cierta cantidad de datos, se realizó el análisis de modelo lineal general univariante.

Con el modelo lineal general se pueden contrastar hipótesis nulas sobre los efectos de otras variables en las medias de varias agrupaciones de una única variable dependiente. Se pueden investigar las interacciones entre los factores, así como los efectos de los factores individuales, algunos de los cuales pueden ser aleatorios (MLG Análisis Univariante: IBM Knowledge Center, 2017).

## Resultados del análisis lineal general univariante

### ❖ Medias marginales

Variable dependiente: Tiempo de ejecución

Sistema Gestor de Base de Datos	Media	Intervalo de confianza 95%	
		Límite inferior	Límite superior
<b>MariaDB</b>	4.865	4.843	4.887
<b>PostgreSQL</b>	3.572	3.550	3.594

Tabla 36. Tiempo promedio de ejecución en milisegundos para SGBDR

Fuente: IBM SPSS Statistics 21

En la tabla 36 se puede determinar que el Sistema Gestor de Base de Datos MariaDB tiene una media ( $\bar{x} = 4.865$ ) superior a PostgreSQL ( $\bar{x} = 3.572$ ); concluyendo así, que MariaDB ocasiona tiempos de ejecución mayores.

Variable dependiente: Tiempo de ejecución

Framework de persistencia ORM	Media	Intervalo de confianza 95%	
		Límite inferior	Límite superior
<b>Doctrine ORM</b>	4.344	4.322	4.366
<b>Hibernate ORM</b>	4.093	4.071	4.115

Tabla 37. Tiempo promedio de ejecución en milisegundos para frameworks de persistencia ORM.

Fuente: IBM SPSS Statistics 21

En la tabla 37 se puede determinar que el framework de persistencia ORM Doctrine ORM tiene una media ( $\bar{x} = 4.344$ ) superior a Hibernate ORM ( $\bar{x} = 4.093$ ); concluyendo así, que Doctrine ORM ocasiona tiempos de ejecución mayores.

Variable dependiente: Tiempo de ejecución

Cantidad de datos	Media	Intervalo de confianza 95%	
		Límite inferior	Límite superior
<b>C1 (5000)</b>	3.662	3.631	3.693
<b>C2 (10000)</b>	3.877	3.846	3.908
<b>C3 (50000)</b>	4.421	4.390	4.452
<b>C4 (100000)</b>	4.914	4.883	4.945

Tabla 38. Tiempo promedio de ejecución en milisegundos para Cantidad de datos

Fuente: IBM SPSS Statistics 21

En la tabla 38 se puede determinar que la cantidad de datos C<sub>4</sub> equivalente a 100000 registros tiene una media ( $\bar{x} = 4.914$ ) superior a C<sub>3</sub> ( $\bar{x} = 4.421$ ), C<sub>2</sub> ( $\bar{x} =$

3.877) y C1 ( $\bar{X} = 3.662$ ); concluyendo así, que a mayor cantidad de datos ocasionará mayores tiempos de ejecución mayores.

**Variable dependiente: Tiempo de Ejecución**

<i>Framework</i> de persistencia	Sistema Gestor de Base de Datos	Media	Intervalo de confianza 95%	
			Límite inferior	Límite superior
Doctrine ORM	MariaDB	5.070	5.039	5.101
	PostgreSQL	3.617	3.586	3.648
Hibernate ORM	MariaDB	4.660	4.629	4.691
	PostgreSQL	3.526	3.495	3.557

*Tabla 39. Tiempo promedio de ejecución en milisegundos para la interacción SGBDR y framework de persistencia ORM*

Fuente: IBM SPSS Statistics 21

En la tabla 39 se puede determinar que el módulo *Web* de Trámite Documentario desarrollado con el *framework* de persistencia Doctrine ORM y utilizando el Sistema Gestor de Base de Datos MariaDB, tienen una media ( $\bar{X} = 5.070$ ) superior a las demás interacciones; concluyendo así, que la interacción entre Doctrine ORM y MariaDB, ocasionan tiempos de ejecución mayores.

**Variable dependiente: Tiempo de Ejecución**

Sistema Gestor de Base de Datos	Cantidad de datos	Media	Intervalo de confianza 95%	
			Límite inferior	Límite superior
MariaDB	C <sub>1</sub>	4.237	4.193	4.281
	C <sub>2</sub>	4.506	4.462	4.550
	C <sub>3</sub>	5.090	5.046	5.133
	C <sub>4</sub>	5.628	5.584	5.672
PostgreSQL	C <sub>1</sub>	3.088	3.044	3.131
	C <sub>2</sub>	3.248	3.204	3.292
	C <sub>3</sub>	3.753	3.709	3.797
	C <sub>4</sub>	4.199	4.155	4.243

*Tabla 40. Tiempo promedio de ejecución en milisegundos para la interacción SGBDR y Cantidad de datos.*

Fuente: IBM SPSS Statistics 21

En la tabla 40 se puede determinar que el módulo *Web* de Trámite Documentario que utiliza el Sistema Gestor de Base de Datos MariaDB y tiene como cantidad de datos cien mil registros (C<sub>4</sub> = 100000), tienen una media ( $\bar{X} = 5.628$ )

superior a las demás interacciones; concluyendo así, que la interacción entre MariaDB y mayor cantidad de datos, ocasionan tiempos de ejecución mayores.

Variable dependiente: Tiempo de Ejecución				
Framework de persistencia	Cantidad de datos	Media	Intervalo de confianza 95%	
			Límite inferior	Límite superior
Doctrine ORM	C <sub>1</sub>	3.695	3.651	3.739
	C <sub>2</sub>	3.977	3.933	4.021
	C <sub>3</sub>	4.575	4.531	4.619
	C <sub>4</sub>	5.128	5.084	5.172
Hibernate ORM	C <sub>1</sub>	3.629	3.585	3.673
	C <sub>2</sub>	3.777	3.733	3.821
	C <sub>3</sub>	4.268	4.224	4.311
	C <sub>4</sub>	4.700	4.656	4.743

Tabla 41. Tiempo promedio de ejecución en milisegundos para la interacción framework de persistencia y Cantidad de datos  
Fuente: IBM SPSS Statistics 21

En la tabla 41 se puede determinar que el módulo *Web* de Trámite Documentario desarrollado con el *framework* de Persistencia Doctrine ORM y que tiene como cantidad de datos cien mil registros (C<sub>4</sub> = 100000), tienen una media ( $\bar{x}$  = 5.128) superior a las demás interacciones; concluyendo así, que la interacción entre Doctrine ORM y mayor cantidad de datos, ocasionan tiempos de ejecución mayores.

**Variable dependiente: Tiempo de Ejecución**

Framework de persistencia	Sistema Gestor de Base de Datos	Cantidad de datos	Media	Intervalo de confianza 95%	
				Límite inferior	Límite superior
Doctrine ORM	MariaDB	C <sub>1</sub>	4.382	4.320	4.444
		C <sub>2</sub>	4.650	4.588	4.712
		C <sub>3</sub>	5.279	5.217	5.341
		C <sub>4</sub>	5.970	5.908	6.032
	PostgreSQL	C <sub>1</sub>	3.009	2.947	3.071
		C <sub>2</sub>	3.305	3.243	3.367
		C <sub>3</sub>	3.870	3.808	3.932
		C <sub>4</sub>	4.286	4.223	4.348
Hibernate ORM	MariaDB	C <sub>1</sub>	4.092	4.030	4.154
		C <sub>2</sub>	4.362	4.300	4.424
		C <sub>3</sub>	4.900	4.838	4.962
		C <sub>4</sub>	5.286	5.224	5.348
	PostgreSQL	C <sub>1</sub>	3.166	3.104	3.228
		C <sub>2</sub>	3.191	3.129	3.253
		C <sub>3</sub>	3.635	3.573	3.697
		C <sub>4</sub>	4.113	4.051	4.175

Tabla 42. Tiempo promedio de ejecución en milisegundos para la interacción entre SGBDR, framework de persistencia y Cantidad de datos

Fuente: IBM SPSS Statistics 21

En la tabla 42 se puede determinar que los módulos Web de Trámite Documentario desarrollados con el framework de persistencia Doctrine ORM, que utiliza el Sistema Gestor de Base de Datos MariaDB y que tiene como cantidad de datos cien mil registros (C<sub>4</sub> = 100000), tienen una media ( $\bar{X}$  = 5.970) superior a las demás interacciones; concluyendo así, que la interacción entre Doctrine ORM, MariaDB y mayor cantidad de datos, ocasionan tiempos de ejecución mayores.

#### Variable dependiente: Tiempo de Ejecución

Operación	Frameworks de persistencia	Media	Intervalo de confianza 95%	
			Límite inferior	Límite superior
Consulta	<b>Doctrine ORM</b>	3.478	3.447	3.509
	<b>Hibernate ORM</b>	3.499	3.468	3.530
Inserción	<b>Doctrine ORM</b>	5.209	5.178	5.240
	<b>Hibernate ORM</b>	4.687	4.656	4.718

Tabla 43. Tiempo promedio de ejecución en milisegundos para la interacción entre SGBDR, Framework de Persistencia y Cantidad de datos.

Fuente: IBM SPSS Statistics 21

En la tabla 43 se puede determinar que la Operación “Inserción” realizada en el módulo *Web* de Trámite Documentario desarrollado con el *framework* de persistencia Doctrine ORM, tiene una media ( $\bar{x} = 5.209$ ) superior a las demás interacciones; concluyendo así, que la Operación “Inserción” y el *framework* Doctrine ORM, ocasionan tiempos de ejecución mayores.

#### ❖ **Frameworks de persistencia ODM:**

A continuación se muestran los datos que se utilizaron para llevar a cabo el análisis estadístico de los resultados obtenidos de los tiempos de ejecución en cada operación que se simularon para los módulos *Web* de Trámite Documentario, que se desarrollaron haciendo uso de los *frameworks* de persistencia ODM y utilizando SGBD NoSQL.

Ítem	Descripción
<b>Factor A</b>	Sistema Gestor de Base de Datos NoSQL
<b>Factor B</b>	<i>Framework</i> de Persistencia ODM
<b>Factor C</b>	Cantidad de datos
<b>Variable Dependiente</b>	Tiempo de ejecución en cada operación.
<b>Nº de réplicas o repeticiones de la prueba</b>	n = 4
<b>Nº de niveles del factor A</b>	a = 2

<b>Nº de niveles del factor B</b>	b = 2
<b>Nº de niveles del factor C</b>	c = 4
<b>Nº total de observaciones realizadas</b>	N = 64
<b>Diseño experimental utilizado para el análisis</b>	Diseño factorial 2x2x4
<b>Nivel de Significancia (alfa)</b>	5%

Tabla 44. Datos para el análisis de los resultados obtenidos para los frameworks de persistencia ODM

Fuente: Elaboración propia

**Prueba de Hipótesis:** En las tablas 45 y 46 se muestran las pruebas de hipótesis que se utilizaron para el análisis estadístico de los resultados obtenidos.

<b>Respecto al efecto de los factores principales</b>	
<b>Factores</b>	<b>Hipótesis</b>
<b>Factor A: Sistema Gestor de Base de Datos NoSQL</b>	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por el Sistema Gestor de Base de Datos NoSQL utilizado, es decir, su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por el Sistema Gestor de Base de Datos NoSQL utilizado, es decir, su efecto es diferente de cero.</p>
<b>Factor B: Framework de persistencia ODM</b>	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por el <i>framework</i> de persistencia ODM utilizado, es decir, su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por el <i>framework</i> de persistencia ODM utilizado, es decir, su efecto es diferente de cero.</p>
<b>Factor C: Cantidad de datos</b>	<b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la cantidad

	<p>de datos que se registran o consultan en una base de datos, es decir, su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la cantidad de datos que se registran o consultan en una base de datos, es decir, su efecto es diferente de cero.</p>
--	---

*Tabla 45. Prueba de hipótesis respecto a la interacción de los factores principales*  
*Fuente: Elaboración propia*

<b>Respecto a la interacción de los factores principales</b>	
<b>Interacción</b>	<b>Hipótesis</b>
<b>Interacción AB</b>	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBD NoSQL y <i>framework</i> de persistencia ODM, es decir, cuando los factores actúan de manera conjunta su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBD NoSQL y <i>framework</i> de persistencia ODM, es decir, su efecto es diferente a cero.</p>
<b>Interacción AC</b>	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBD NoSQL y Cantidad de datos, es decir, cuando los factores actúan de manera conjunta su efecto es igual a cero.</p> <p><b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBD NoSQL y Cantidad de datos, es decir, su efecto es diferente de cero.</p>
<b>Interacción BC</b>	<p><b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción <i>framework</i> de persistencia ODM y Cantidad de datos, es decir, cuando los factores actúan de manera conjunta su efecto es igual a cero.</p>

	<b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción <i>framework</i> de Persistencia ODM y Cantidad de datos, es decir, su efecto es diferente de cero.
<b>Interacción ABC</b>	<b>H<sub>0</sub>:</b> No existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBD NoSQL, <i>Framework</i> de persistencia ODM y Cantidad de datos, es decir, cuando los factores actúan de manera conjunta su efecto es igual a cero. <b>H<sub>1</sub>:</b> Existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción SGBD NoSQL, <i>Framework</i> de persistencia ODM y Cantidad de datos, es decir, cuando los factores actúan de manera conjunta su efecto es diferente de cero.

Tabla 46. Prueba de hipótesis respecto a la interacción de los factores principales  
Fuente: Elaboración propia

**Conclusión:** Cuando el valor de “F Calculado” es mayor que el valor de “F de tabla”, se rechaza H<sub>0</sub>.

A continuación, se muestra el análisis varianza (ANOVA) por cada operación simulada en los módulos *Web* de Trámite Documentario desarrollados con los *frameworks* de persistencia ODM.

## Operación 1: Registro de Expediente

		MongoDB				CouchDB			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
<b>Hibernate OGM</b>	r <sub>1</sub>	3.378	3.562	4.198	4.553	6.243	6.545	7.051	7.481
	r <sub>2</sub>	3.336	3.559	4.190	4.540	6.246	6.642	7.100	7.413
	r <sub>3</sub>	3.320	3.559	4.200	4.568	6.284	6.699	7.042	7.580
	r <sub>4</sub>	3.339	3.560	4.203	4.553	6.320	6.599	7.096	7.446
<b>Doctrine ODM</b>	r <sub>1</sub>	3.551	3.922	4.936	5.310	5.807	6.089	6.823	7.098
	r <sub>2</sub>	3.561	3.924	4.913	5.312	5.817	6.251	6.831	7.177
	r <sub>3</sub>	3.547	3.941	4.904	5.314	5.834	6.294	6.817	7.231
	r <sub>4</sub>	3.554	3.925	4.901	5.315	5.806	6.149	6.812	7.256

Tabla 47. Tiempos de ejecución en milisegundos de la operación “Registro de Expediente” para los frameworks de persistencia ODM

Fuente: Elaboración propia

Fuente de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F Calculado	F Tabla Alfa=5%	Conclusión
<b>Factor A: Framework de Persistencia</b>	0.099	1	0.099	59.271	4.0427	Rechazo H <sub>0</sub>
<b>Factor B: Sistema Gestor de Base de Datos NoSQL</b>	101.080	1	101.080	60522.846	4.0427	Rechazo H <sub>0</sub>
<b>Factor C: Cantidad de datos</b>	19.061	3	6.354	3804.347	2.7981	Rechazo H <sub>0</sub>
<b>Interacción AB</b>	3.022	1	3.022	1809.420	4.0427	Rechazo H <sub>0</sub>
<b>Interacción AC</b>	0.402	3	0.134	80.163	2.7981	Rechazo H <sub>0</sub>
<b>Interacción BC</b>	0.228	3	0.076	45.594	2.7981	Rechazo H <sub>0</sub>
<b>Interacción ABC</b>	0.088	3	0.029	17.637	2.7981	Rechazo H <sub>0</sub>
<b>Error</b>	0.080	48	0.002			
<b>Total</b>	2008.999	64				

Tabla 48. Resultados del análisis de varianza para la operación 1

Fuente: IBM SPSS Statistics 21

Hipótesis nula (H <sub>0</sub> ) / Nivel de significancia de 5%	Sig.	Conclusión
La distribución de tiempo de ejecución es la misma entre las categorías de SGBD NoSQL	0.000	Rechazar H <sub>0</sub>
La distribución de tiempo de ejecución es la misma entre las categorías de Cantidad de datos	0.004	Rechazar H <sub>0</sub>

Tabla 49. Resultados del Test de Kruskal – Wallis para la operación 1

Fuente: IBM SPSS Statistics 21

En la tabla 49 se muestra el análisis de varianza para la operación 1, donde se busca demostrar si los módulos Web de Trámite Documentario desarrollados con

los *frameworks* de persistencia ODM, y usando SGBD NoSQL afectan significativamente sobre el tiempo de ejecución de la operación que se está evaluando. En dicha tabla se observa que el factor “Sistema Gestor de Base de Datos NoSQL” es el que produce una mayor variabilidad en el tiempo de ejecución ( $\sigma^2 = 101.080$ ).

Además, se comprueba que al 5% de nivel de significancia que los factores: Sistemas Gestor de Base de Datos NoSQL ( $F_c = 60522.846$ ) y la Cantidad de Datos ( $F_c = 3804.347$ ), actuando de manera independiente, afectan significativamente al tiempo de ejecución de la operación, lo dicho es corroborado por el test de Kruskal Wallis. Según el análisis de varianza, el factor *Frameworks* de persistencia ( $F_c = 59.271$ ), actuando de manera independiente, afectan significativamente al tiempo de ejecución de la operación. Para las interacciones de dos factores, AB ( $F_c = 1809.420$ ), BC ( $F_c = 45.594$ ) y AC ( $F_c = 80.163$ ), afectan significativamente al tiempo de ejecución de la operación, lo mismo sucede con la interacción de los tres factores ABC ( $F_c = 17.637$ ); con lo que se concluye que se aceptan las hipótesis  $H_1$ , formuladas en las tablas 45 y 46, es decir, existen diferencias significativas en el tiempo de ejecución para los factores y las interacciones.

Durante esta etapa de la experimentación, el módulo desarrollado con la tecnología Hibernate OGM y el SGBD NoSQL con MongoDB emitía un error al intentar registrar los 100000 (cien mil) documentos, el mismo problema se obtuvo durante la experimentación con el módulo que se desarrolló con Hibernate OGM y el SGBD NoSQL CouchDB, pero para este caso la memoria se emitía el error desde el momento que se intentaba registrar los 5000 (cinco mil) documentos. Se encontró que durante la transacción se debía limpiar la memoria de Java a la mitad de la operación, pues Hibernate OGM almacena la gran cantidad de objetos que se intentan registrar en la memoria y esta tiende a desbordarse.

Para el módulo que se desarrolló con la tecnología Doctrine ODM y el SGBD NoSQL MongoDB se encontró la misma deficiencia que con el caso de Hibernate OGM, al intentar registrar 100000 (cien mil) documentos se desbordaba la memoria de PHP, el problema fue resuelto de la misma forma que se resolvió los anteriores errores.

## Operación 2: Consulta de Bandeja de Entrada.

		MongoDB				CouchDB			
		C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>
<b>Hibernate OGM</b>	r <sub>1</sub>	3.035	2.942	3.558	4.030	4.330	4.513	4.653	5.001
	r <sub>2</sub>	2.894	3.094	3.577	4.031	4.314	4.567	4.629	4.965
	r <sub>3</sub>	2.902	3.000	3.579	4.021	4.332	4.475	4.677	4.994
	r <sub>4</sub>	2.967	2.987	3.602	4.010	4.296	4.484	4.666	4.987
<b>Doctrine ODM</b>	r <sub>1</sub>	2.210	2.403	3.075	3.329	3.413	3.555	3.995	4.307
	r <sub>2</sub>	2.199	2.422	3.004	3.347	3.375	3.498	4.011	4.430
	r <sub>3</sub>	2.199	2.398	3.010	3.335	3.677	3.481	3.953	4.481
	r <sub>4</sub>	2.207	2.405	3.017	3.328	3.513	3.474	3.996	4.513

Tabla 50. Tiempos de ejecución en milisegundos de la operación “Consulta de Bandeja de Entrada” para los frameworks de persistencia ODM

Fuente: Elaboración propia

Fuente de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F Calculado	F Tabla Alfa=5%	Conclusión
Factor A: Framework de persistencia	7.947	1	7.947	3091.098	4.0427	Rechazo H <sub>0</sub>
Factor B: Sistema Gestor de Base de Datos NoSQL	21.900	1	21.900	8518.593	4.0427	Rechazo H <sub>0</sub>
Factor C: Cantidad de datos	9.216	3	3.072	1194.941	2.7981	Rechazo H <sub>0</sub>
Interacción AB	0.055	1	0.055	21.253	4.0427	Rechazo H <sub>0</sub>
Interacción AC	0.128	3	0.043	16.624	2.7981	Rechazo H <sub>0</sub>
Interacción BC	0.337	3	0.112	43.659	2.7981	Rechazo H <sub>0</sub>
Interacción ABC	0.150	3	0.050	19.440	2.7981	Rechazo H <sub>0</sub>
Error	0.123	48	0.003			
Total	893.021	64				

Tabla 51. Resultados del análisis de varianza para la operación 2

Fuente: IBM SPSS Statistics 21

Hipótesis nula (H <sub>0</sub> ) / Nivel de significancia de 5%	Sig.	Conclusión
La distribución de tiempo de ejecución es la misma entre las categorías de Framework de persistencia	0.001	Rechazar H <sub>0</sub>
La distribución de tiempo de ejecución es la misma entre las categorías de SGBD NoSQL	0.00	Rechazar H <sub>0</sub>
La distribución de tiempo de ejecución es la misma entre las categorías de Cantidad de datos	0.003	Rechazar H <sub>0</sub>

Tabla 52. Resultados del Test de Kruskal – Wallis para la operación 2

Fuente: IBM SPSS Statistics 21

En la tabla 52 se muestra el análisis de varianza para la operación 2, donde se busca demostrar si los módulos *Web* de Trámite Documentario desarrollados con los *frameworks* de persistencia ODM, y usando SGBD NoSQL afectan significativamente sobre el tiempo de ejecución de la operación que se está evaluando. En dicha tabla se observa que el factor “Sistema Gestor de Base de Datos NoSQL” es el que produce una mayor variabilidad en el tiempo de ejecución ( $\sigma^2 = 21.900$ ).

Además, se comprueba que al 5% de nivel de significancia, tanto los *Frameworks* de persistencia ODM ( $F_c = 3091.098$ ), los SGBD NoSQL ( $F_c = 8518.593$ ), y la Cantidad de datos ( $F_c = 1194.941$ ), actuando de manera independiente, afectan significativamente al tiempo de ejecución de la operación, lo dicho es corroborado por el test de Kruskal – Wallis. Para las interacciones de dos factores, AB ( $F_c = 21.253$ ), BC ( $F_c = 43.659$ ) y AC ( $F_c = 16.624$ ), afectan significativamente al tiempo de ejecución de la operación, lo mismo sucede con la interacción de los tres factores ABC ( $F_c = 19.440$ ); con lo que se concluye que se aceptan las hipótesis  $H_1$ , formuladas en las tablas 45 y 46, es decir, existen diferencias significativas en el tiempo de ejecución para los factores y las interacciones.

Después de haber realizado el análisis estadístico de los tiempos de ejecución por cada operación, se realizó un resumen de las conclusiones que se obtuvieron en cada operación por cada factor e interacción entre ellos, las cuales se muestran a continuación en la tabla 53.

	Operaciones		Conclusión de Hipótesis $H_0$
	1	2	
Factor A: <i>Framework</i> de persistencia	Rechazo $H_0$	Rechazo $H_0$	
Factor B: Sistema Gestor de Base de Datos NoSQL	Rechazo $H_0$	Rechazo $H_0$	
Factor C: Cantidad de datos	Rechazo $H_0$	Rechazo $H_0$	
Interacción AB	Rechazo $H_0$	Rechazo $H_0$	
Interacción AC	Rechazo $H_0$	Rechazo $H_0$	
Interacción BC	Rechazo $H_0$	Rechazo $H_0$	
Interacción ABC	Rechazo $H_0$	Rechazo $H_0$	

Tabla 53. Resumen del análisis estadístico de los tiempos de ejecución de cada operación para los módulos desarrollados con frameworks de persistencia ODM

Fuente: Elaboración propia

En la tabla 53 se determinó que en el 100% de las operaciones analizadas existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por los SGBD NoSQL usados. Además, que para el factor “*Framework* de Persistencia ODM”, en el 100% de las operaciones analizadas se concluyó que existen diferencias significativas en el tiempo de ejecución de cada operación ocasionada por el ODM utilizado. Por último para el factor Cantidad de datos, en el 100% de las operaciones analizadas se concluyó que existen diferencias significativas en el tiempo de ejecución ocasionada por la cantidad de datos que se desea consultar o registrar.

Para las interacciones de los factores se puede concluir que la interacción SGBD NoSQL – *Framework* de persistencia ODM, en el 100% de las operaciones analizadas se concluyó que existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción de los factores. Además para la interacción SGBD NoSQL – Cantidad de datos, en el 100% de las operaciones analizadas se concluyó que existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción de los factores. Por último para la interacción *Frameworks* de persistencia – Cantidad de datos, en el 100% de las operaciones analizadas se concluyó que existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción de los factores.

Lo mismo ocurre para la interacción entre los tres factores: Base de datos, *Frameworks* de persistencia y Cantidad de datos, en el 100% de las operaciones analizadas se concluyó que existe diferencia significativa en el tiempo de ejecución de cada operación ocasionada por la interacción de los tres factores.

Para determinar la combinación Base de datos - *Frameworks* de persistencia que ocasionan la diferencia significativa en el tiempo de ejecución en las operaciones (con distintas cantidades de datos) de los módulos *Web* desarrollados se realizó el análisis de modelo lineal general univariante.

Con el modelo lineal general se pueden contrastar hipótesis nulas sobre los efectos de otras variables en las medias de varias agrupaciones de una única variable dependiente. Se pueden investigar las interacciones entre los factores, así como los

efectos de los factores individuales, algunos de los cuales pueden ser aleatorios (MLG Análisis Univariante: IBM Knowledge Center, 2017).

## Resultados del análisis lineal general univariante

### ❖ Medias marginales

Variable dependiente: Tiempo de ejecución			
Sistema Gestor de Base de Datos NoSQL	Media	Intervalo de confianza 95%	
		Límite inferior	Límite superior
CouchDB	5.460	5.448	5.471
MongoDB	3.618	3.607	3.630

Tabla 54. Tiempo promedio de ejecución en milisegundos para SGBD NoSQL

Fuente: IBM SPSS Statistics 21

En la tabla 54 se puede determinar que el SGBD NoSQL CouchDB tiene una media ( $\bar{x} = 5.460$ ) superior a MongoDB ( $\bar{x} = 3.618$ ); concluyendo así, que CouchDB ocasiona tiempos de ejecución mayores.

Variable dependiente: Tiempo de ejecución			
Framework de Persistencia ODM	Media	Intervalo de confianza 95%	
		Límite inferior	Límite superior
Doctrine ODM	4.383	4.371	4.394
Hibernate OGM	4.696	4.684	4.707

Tabla 55. Tiempo promedio de ejecución en milisegundos para frameworks de persistencia ODM.

Fuente: IBM SPSS Statistics 21

En la tabla 55 se puede determinar que el framework de persistencia Hibernate OGM tiene una media ( $\bar{x} = 4.696$ ) superior a Doctrine ODM ( $\bar{x} = 4.383$ ); concluyendo así, que Hibernate OGM ocasiona tiempos de ejecución mayores.

#### Variable dependiente: Tiempo de ejecución

Cantidad de datos	Media	Intervalo de confianza 95%	
		Límite inferior	Límite superior
C <sub>1</sub> (5000)	3.994	3.978	4.010
C <sub>2</sub> (10000)	4.216	4.200	4.232
C <sub>3</sub> (50000)	4.782	4.766	4.798
C <sub>4</sub> (100000)	5.164	5.148	5.180

Tabla 56. Tiempo promedio de ejecución en milisegundos para Cantidad de datos

Fuente: IBM SPSS Statistics 21

En la tabla 56 se puede determinar que la cantidad de datos C<sub>4</sub> equivalente a 100000 documentos tiene una media ( $\bar{x} = 5.164$ ) superior a C<sub>3</sub> ( $\bar{x} = 4.782$ ), C<sub>2</sub> ( $\bar{x} = 4.216$ ) y C<sub>1</sub> ( $\bar{x} = 3.994$ ); concluyendo así, que a mayor cantidad de datos occasionará mayores tiempos de ejecución mayores.

#### Variable dependiente: Tiempo de Ejecución

Framework de persistencia	Sistema Gestor de Base de Datos NoSQL	Media	Intervalo de confianza 95%	
			Límite inferior	Límite superior
Doctrine ODM	CouchDB	5.180	5.164	5.196
	MongoDB	3.585	3.569	3.601
Hibernate OGM	CouchDB	5.740	5.724	5.756
	MongoDB	3.651	3.635	3.668

Tabla 57. Tiempo promedio de ejecución en milisegundos para la interacción SGBD NoSQL y

Framework de persistencia ODM

Fuente: IBM SPSS Statistics 21

En la tabla 57 se puede determinar que los módulos Web de Trámite Documentario desarrollado con el framework de persistencia Hibernate OGM y utilizando el SGBD NoSQL CouchDB, tienen una media ( $\bar{x} = 5.740$ ) superior a las demás interacciones; concluyendo así, que la interacción entre Hibernate OGM y CouchDB, occasionan tiempos de ejecución mayores.

#### Variable dependiente: Tiempo de Ejecución

Sistema Gestor de Base de Datos NoSQL	Cantidad de datos	Media	Intervalo de confianza 95%	
			Límite inferior	Límite superior
<b>CouchDB</b>	C <sub>1</sub>	4.975	4.953	4.998
	C <sub>2</sub>	5.207	5.184	5.230
	C <sub>3</sub>	5.635	5.612	5.657
	C <sub>4</sub>	6.023	6.000	6.045
<b>MongoDB</b>	C <sub>1</sub>	3.012	2.990	3.035
	C <sub>2</sub>	3.225	3.202	3.248
	C <sub>3</sub>	3.929	3.906	3.952
	C <sub>4</sub>	4.306	4.283	4.329

Tabla 58. Tiempo promedio de ejecución en milisegundos para la interacción SGBD NoSQL y Cantidad de datos.

Fuente: IBM SPSS Statistics 21

En la tabla 58 se puede determinar que el módulo *Web* de Trámite Documentario que utiliza el SGBD NoSQL CouchDB y tiene como cantidad de datos cien mil documentos (C<sub>4</sub> = 100000), tienen una media ( $\bar{x}$  = 6.023) superior a las demás interacciones; concluyendo así, que la interacción entre CouchDB y mayor cantidad de datos, ocasionan tiempos de ejecución mayores.

#### Variable dependiente: Tiempo de Ejecución

Framework de persistencia ODM		Media	Intervalo de confianza 95%	
			Límite inferior	Límite superior
<b>Doctrine ODM</b>	C <sub>1</sub>	3.767	3.744	3.790
	C <sub>2</sub>	4.008	3.985	4.031
	C <sub>3</sub>	4.687	4.665	4.710
	C <sub>4</sub>	5.068	5.045	5.091
<b>Hibernate OGM</b>	C <sub>1</sub>	4.221	4.198	4.244
	C <sub>2</sub>	4.424	4.401	4.447
	C <sub>3</sub>	4.876	4.853	4.899
	C <sub>4</sub>	5.261	5.238	5.284

Tabla 59. Tiempo promedio de ejecución en milisegundos para la interacción Framework de Persistencia ODM y Cantidad de datos.

Fuente: IBM SPSS Statistics 21

En la tabla 59 se puede determinar que los módulos *Web* de Trámite Documentario desarrollados con el *framework* de persistencia Hibernate OGM y que tiene como cantidad de datos cien mil registros (C<sub>4</sub> = 100000), tienen una media ( $\bar{x}$

= 5.261) superior a las demás interacciones; concluyendo así, que la interacción entre Hibernate OGM y mayor cantidad de datos, ocasionan tiempos de ejecución mayores.

**Variable dependiente: Tiempo de Ejecución**

Sistema Gestor de Base de Datos NoSQL	Framework de persistencia	Cantidad de datos	Media	Intervalo de confianza 95%	
				Límite inferior	Límite superior
Doctrine ODM	CouchDB	C <sub>1</sub>	4.655	4.623	4.688
		C <sub>2</sub>	4.849	4.817	4.881
		C <sub>3</sub>	5.405	5.372	5.437
		C <sub>4</sub>	5.812	5.779	5.844
	MongoDB	C <sub>1</sub>	2.879	2.846	2.911
		C <sub>2</sub>	3.168	3.135	3.200
		C <sub>3</sub>	3.970	3.938	4.002
		C <sub>4</sub>	4.324	4.291	4.356
Hibernate OGM	CouchDB	C <sub>1</sub>	5.296	5.263	5.328
		C <sub>2</sub>	5.566	5.533	5.598
		C <sub>3</sub>	5.864	5.832	5.897
		C <sub>4</sub>	6.233	6.201	6.266
	MongoDB	C <sub>1</sub>	3.146	3.114	3.179
		C <sub>2</sub>	3.283	3.251	3.315
		C <sub>3</sub>	3.888	3.856	3.921
		C <sub>4</sub>	4.288	4.256	4.321

*Tabla 60. Tiempo promedio de ejecución en milisegundos para la interacción entre SGBD NoSQL, framework de persistencia ODM y Cantidad de datos*  
*Fuente: IBM SPSS Statistics 21*

En la tabla 60 se puede determinar que los módulos Web de Trámite Documentario desarrollados con el *framework* de persistencia Hibernate OGM, que utiliza el SGBD NoSQL CouchDB y que tiene como cantidad de datos cien mil documentos (C<sub>4</sub> = 100000), tienen una media ( $\bar{x}$  = 6.233) superior a las demás interacciones; concluyendo así, que la interacción entre Hibernate OGM, CouchDB y mayor cantidad de datos, ocasionan tiempos de ejecución mayores.

#### Variable dependiente: Tiempo de Ejecución

Operación	Framework de persistencia	Media	Intervalo de confianza 95%	
			Límite inferior	Límite superior
Consulta	<b>Doctrine ODM</b>	3.299	3.283	3.315
	<b>Hibernate OGM</b>	4.004	3.987	4.020
Inserción	<b>Doctrine ODM</b>	5.466	5.450	5.482
	<b>Hibernate OGM</b>	5.388	5.371	5.404

Tabla 61. Tiempo promedio de ejecución en milisegundos para la interacción entre SGBD NoSQL, Framework de persistencia ODM y Cantidad de datos

Fuente: IBM SPSS Statistics 21

En la tabla 61 se puede determinar que la operación “Inserción” realizada en el módulo *Web* de Trámite Documentario desarrollado con el *framework* de persistencia Doctrine ODM, tiene una media ( $\bar{x} = 5.466$ ) superior a las demás interacciones; concluyendo así, que la Operación “Inserción” y el *framework* Doctrine ODM, ocasionan tiempos de ejecución mayores.

A continuación en la tabla 62 se muestra un cuadro de resumen de lo obtenido en el análisis de resultados.

Resumen de análisis de resultados			
Módulos desarrollados con <i>framework</i> de persistencia ORM		Módulos desarrollados con <i>framework</i> de persistencia ODM	
Registro de expediente	Los 3 factores: <i>Framework</i> de persistencia, SGBDR y Cantidad de datos tienen efecto significativo en el tiempo de ejecución.	Registro de expediente	Los 3 factores: <i>Framework</i> de persistencia, SGBD NoSQL y Cantidad de datos tienen efecto significativo en el tiempo de ejecución.
	Orden de efecto sobre el tiempo de ejecución de mayor a menor: 1. Cantidad de datos 2. SGBDR 3. <i>Framework</i> de persistencia		Orden de efecto sobre el tiempo de ejecución mayor a menor: 1. SGBD NoSQL 2. Cantidad de datos 3. <i>Framework</i> de persistencia
Consulta de bandeja de expediente	Los factores: SGBDR y Cantidad de datos tienen efecto significativo en el tiempo de ejecución.	Consulta de bandeja de expediente	Los 3 factores: <i>Framework</i> de persistencia, SGBD NoSQL y Cantidad de datos tienen efecto significativo en el tiempo de ejecución.

Resumen de análisis de resultados			
	Orden de efecto sobre el tiempo de ejecución de mayor a menor: 1. SGBDR 2. Cantidad de datos		Orden de efecto sobre el tiempo de ejecución de mayor a menor: 1. SGBD NoSQL 2. Cantidad de datos 3. <i>Framework</i> de persistencia
<b>Factores que ocasionan menores tiempos de ejecución</b>		<b>Factores que ocasionan menores tiempos de ejecución</b>	
SGBDR	PostgreSQL	SGBD NoSQL	MongoDB
<i>Framework</i> de persistencia	Hibernate ORM	<i>Framework</i> de persistencia	Doctrine ODM
Cantidad	C <sub>1</sub>	Cantidad	C <sub>1</sub>
Interacción <i>framework</i> de persistencia y SGBDR	Hibernate ORM – PostgreSQL	Interacción <i>framework</i> de persistencia y SGBD NoSQL	Doctrine ODM - MongoDB
<i>Framework</i> de persistencia en la operación “Inserción”	Hibernate ORM	<i>Framework</i> de persistencia en la operación “Inserción”	Hibernate OGM
<i>Framework</i> de persistencia en la operación “Consulta”	Doctrine ORM	<i>Framework</i> de persistencia en la operación “Consulta”	Doctrine ODM

Tabla 62. Resumen de análisis de resultados

Fuente: Elaboración propia

#### 4.3. Discusión de resultados

Luego de realizar el análisis estadístico de los tiempos de ejecución obtenidos de los módulos *Web* de Trámite Documentario desarrollados y el análisis de modelo lineal general univariante, se obtuvieron los siguientes resultados:

El módulo *Web* desarrollado con el *framework* de persistencia ORM: Hibernate, obtuvo menores tiempos de ejecución en la mayoría de las operaciones realizadas. Según la documentación oficial de Hibernate ORM, su alto rendimiento se debe a que no requiere tablas ni campos en la base de datos y genera la mayor parte del código SQL en la iniciación del sistema en lugar de en tiempo de ejecución. Además, una de las características de Hibernate es que cuenta con varios niveles de memoria caché, la primera de ellas es la que corresponde con el objeto de sesión (*net.sf.hibernate.Session*) que se obtiene de una factoría de sesiones (*net.sf.hibernate.SessionFactory*), esta caché guarda todos los objetos que se recuperan de la base de datos, de modo que si se vuelven a pedir, se ahorra el acceso

a la base de datos. En general todas las operaciones ofrecidas por la interfaz “Session” interactúan de manera transparente con la caché de primer nivel, ya sea realizando consultas, actualizando objetos de la caché, eliminando objetos de la caché, etc.

El módulo *Web* desarrollado con el *framework* de persistencia ODM: Doctrine ODM, obtuvo menores tiempos de ejecución en la mayoría de las operaciones realizadas. Según la documentación oficial de Doctrine ODM, su alto rendimiento se debe a que cuenta con la estrategia llamada “escritura retrospectiva transaccional” que retrasa la ejecución de las instrucciones de consulta para ejecutarlas de la manera más eficiente y ejecutarlas al final de una transacción para que todos los bloqueos de escritura se liberen rápidamente. Se debería ver a Doctrine como una herramienta para sincronizar sus objetos en memoria con la base de datos en unidades de trabajo bien definidas.

Los módulos *Web* desarrollados que usan como Sistema Gestor de Base de Datos a PostgreSQL obtuvieron los mejores tiempos de ejecución tanto en inserción como en la lectura de datos, independientemente del *framework* de persistencia con el que trabajó. Según la documentación oficial de PostgreSQL, su alto rendimiento se debe a que cuenta con Procesamiento Paralelo Masivo (MPP), esto implica consultas paralelas distribuidas en varios nodos, con la cual las consultas SQL pueden ser ejecutadas hasta N veces más rápido en N nodos, distribuyendo la utilización de I/O y CPU de forma uniforme a través de un clúster. También, la redistribución dinámica de datos para consultas SQL complejas, por lo que se reorganiza datos automáticamente para permitir consultas SQL complejas, y no solo consultas sencillas de Esquema en Estrella o tareas del estilo Map/Reduce. Y por último los escaneos cooperativos, por lo que las consultas trabajan en conjunto para evitar el escaneo repetido de los mismos datos.

Los módulos *Web* desarrollados que usan como SGBD NoSQL a MongoDB obtuvieron los mejores tiempos de ejecución tanto en inserción como en la lectura de datos, independientemente del *framework* de persistencia con el que trabajó. Según la documentación oficial de MongoDB su alto rendimiento se debe a que trabaja con la información en memoria, si bien la guarda en los discos duros de las máquinas en las que está funcionando, para llevar a cabo una operación, la información afectada en cada caso tiene que estar en memoria (mucho más rápida que un disco duro). Si

no está en memoria se copia del disco duro (o SSD). De este modo, al trabajar directamente con la memoria las operaciones se llevan a cabo más rápido.

#### 4.4. Validación de hipótesis

De la hipótesis general planteada en la presente investigación:

- ❖ H<sub>0</sub>: No existen diferencias significativas en el tiempo de ejecución al hacer uso de los *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP.
- ❖ H<sub>1</sub>: Existen diferencias significativas en el tiempo de ejecución al hacer uso de los *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP.

Según los resultados obtenidos, se rechazó la hipótesis H<sub>0</sub> con un nivel de significancia del 95%, pues según el uso de los *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP sí hubo diferencias significativas en el tiempo de ejecución.

De las hipótesis específicas planteadas en la presente investigación:

- ❖ Se identificó al módulo *Web* de Trámite Documentario como el más usado por las entidades de la ciudad de Piura en la actualidad.
- ❖ Se logró desarrollar el módulo *Web* de Trámite Documentario haciendo uso de los *frameworks* de persistencia ORM y ODM basados en los lenguajes de programación Java y PHP.
- ❖ Se determinó a los *frameworks* de persistencia basados en los lenguajes de programación Java y PHP: Hibernate y Doctrine ODM, como los *frameworks* que ofrecen los menores tiempos de ejecución para las tecnologías ORM y ODM, respectivamente.

## **CAPÍTULO V**

### **CONCLUSIONES Y RECOMENDACIONES**

## 5.1. Conclusiones

1. La encuesta aplicada a las entidades de la ciudad de Piura emitió como resultado que el 46% de los encuestados eligió al módulo de Trámite Documentario como el módulo *Web* más usado.
2. Se determinó que existen diferencias significativas en los tiempos de ejecución al hacer uso de los *frameworks* de persistencia tanto para ORM como para ODM.
3. Al realizar el análisis de los datos que se obtuvieron de la experimentación se encontró que, para los módulos *Web* desarrollados con los *frameworks* de persistencia ORM y ODM el factor “Base de datos” es el que más influye al momento de realizar inserciones de datos, con un nivel de significancia del 95%. Además, para la operación que implica lectura de datos, para el caso de los módulos desarrollados con ORM el factor “Cantidad de datos” es el más influyente y para el caso de los ODM es el factor “Sistema Gestor de Base de Datos NoSQL”, esto con un nivel de significancia del 95%.
4. Al realizar el análisis de los datos que se obtuvieron de la experimentación, se encontró que con un nivel de significancia del 95%, el Sistema Gestor de Base de Datos PostgreSQL emite los mejores tiempos de ejecución, pero no con el mismo *framework* de persistencia, al parecer para la lectura de datos trabaja mejor con Doctrine ORM y para la inserción de datos trabaja mejor con Hibernate ORM, entonces es decisión de los desarrolladores y de los requerimientos de las aplicaciones que desarrollarán hacer uso de uno u otro. Lo mismo sucede con el SGBD NoSQL MongoDB, con un nivel de significancia del 95% emite los mejores tiempos de ejecución, pero no con el mismo *framework* de persistencia, si se trata de lectura de datos trabaja mejor con Doctrine ODM y para la inserción de datos trabaja mejor con Hibernate OGM.

## 5.2. Recomendaciones

1. Realizar un nuevo estudio comparativo con el SGBDR MySQL, y el *framework* de persistencia eloquent del *framework* Laravel, uno de los *framework* PHP más conocidos.
2. Realizar un estudio comparativo pero donde esta vez se incluya las otras dos operaciones que se realizan en las bases de datos, es decir, “*Update*” y “*Delete*”, de esta manera se podría determinar cómo se comportan las tecnologías frente a esas operaciones.
3. Realizar el mismo estudio comparativo, pero en este caso tomar como módulo *Web*, de referencias para las operaciones, el módulo del segundo puesto de la encuesta realizada, o sea, el módulo de Inventario.
4. Realizar un estudio comparativo donde se incluya el factor relación de tablas para el caso de los *frameworks* de persistencia ORM, de esta manera, determinar que tanto influye ese factor en el tiempo de ejecución.
5. En el desarrollo de la aplicación el “Sistema Gestor de Base de Datos” utilizado, tiene más efecto significativo en el tiempo de ejecución que el *framework* de persistencia, como se comprobó en la presente investigación; entonces también se debe elegir la mejor opción en cuanto al “Sistema Gestor de Base de Datos” para las aplicaciones donde se desea que el tiempo de ejecución sea un requerimiento imprescindible.

## REFERENCIAS BIBLIOGRÁFICAS

- Amador. P. & Johana, L. (enero, 2013). Estudio comparativo de sistemas de mapeo objeto relacional desarrollados en plataformas Open Source. Revista CITECSA (Ciencia, Tecnología, Sociedad y Ambiente), 3, 74-83. Recuperado el 14 de julio de 2016: [http://s3.amazonaws.com/academia.edu.documents/35816957/34-68-1-SM\\_POLO.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1468560973&Signature=PkJLs1IzDCt5gUTFZIVrv9lCC5Y%3D&response-content-disposition=attachment%3B%20filename%3DEstudio\\_comparativo\\_de\\_sistemas\\_de\\_mapeo.pdf](http://s3.amazonaws.com/academia.edu.documents/35816957/34-68-1-SM_POLO.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1468560973&Signature=PkJLs1IzDCt5gUTFZIVrv9lCC5Y%3D&response-content-disposition=attachment%3B%20filename%3DEstudio_comparativo_de_sistemas_de_mapeo.pdf)
- Anderson D., Sweeney D. & Williams T. (2008). Estadística para Administración y Economía. Décima edición. Cengage Learning Editores, S.A. México. Pág. 494, 702.
- Anghel Leonard. (2013). Pro Hibernate and MongoDB. España.
- Bauer, C., Gavin, K. (2004). *Hibernate in Action. Practical Object/Relational Mapping*. Manning Publications.
- Carmona, G. (2014). *Aplicaciones informáticas de bases de datos relacionales*. IC Editorial.
- Carneiro, J. (2008). Desarrollo de un *framework* de Persistencia para tecnología Java: WayPersistence ORM. Pág. 7, 10.
- Constanzo, M. (2014, abril). *Comparación de modelos de calidad, factores y métricas en el ámbito de la Ingeniería de Software*. Dpto. Ciencias Exactas y Naturales, Universidad Nacional de la Patagonia Austral, Río Gallegos, Argentina.
- Coronel, C. (2011). *Bases de Datos, Diseño, Implementación y Administración*. Cengage Learning Editores.
- Date, C. J. (2001). Introducción a los sistemas de bases de datos. Séptima edición. Pearson Educación. México. Pág. 78.

Eric, J., Cervera, R. Evans, I., Haase, K. & Markito, W. (2014). The Java EE 7 Tutorial. Volumen 1. Primera edición. Addison – Wesley. Pág. 61.

Estudio de la Técnica ORM (Mapeo Objeto – Relacional). Tesis Previa a la Obtención del Título de Ingeniera en Sistemas Computacionales.

Fernandez, A. (2013). Doctrine2, La era de los ODM. Madrid.

Fernández, A. (2013). *Python 3 al descubierto. RC Libros*

Freire, T. (2008). *Estudio de la Técnica ORM (Mapeo Objeto – Relacional)*: Tesis. Universidad Técnica del Norte: Ecuador.

Freire, T. (2008). TESIS: “*Sistema de Gestión de Información Odontológica utilizando ORM para el Departamento de Bienestar Universitario de la UTN*”. Ecuador: Universidad Técnica del Norte.

Gabillaud, J. (2009). *SQL Server 2008 - SQL, Transact SQL: Diseño y creación de una base de datos*. Barcelona: Eni.

Gavilanes, J. (2016). *Estudio comparativo de la productividad entre los frameworks de persistencia en Java Hibernate y MyBatis, Aplicado al sistema de evaluación docente del IPEC*. Tesis previa a la obtención del título de Ingeniero en Sistemas Informáticos, Escuela de Ingeniería en Sistemas, Facultad de Informática y Electrónica, Escuela Superior Politécnica de Chimborazo, Chimborazo, Ecuador.

Gesvin, R. (2004). UML con Rational Rose. Lima: Grupo Editorial MEGABYTE S.A.C.

González, V. (2009). *Frameworks para mapeo objeto relacional: Un análisis comparativo*. Tesis para obtener el título de Ingeniero en Ciencias y Sistemas, Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería, Universidad de San Carlos de Guatemala, Ciudad de Guatemala, Guatemala.

Granados, R. (2015). *Desarrollo de aplicaciones web en el entorno servidor*. IC Editorial.

Hernández, C., Crespo, Y., Romay, P. & Laguna M. *Una herramienta para el aprendizaje del álgebra relacional*. Departamento de Informática, Universidad de Valladolid, Valladolid, España. Recuperado el 22 de julio de 2016 de:

[https://www.academia.edu/578543/Una\\_Herramienta\\_para\\_el\\_Aprendizaje\\_del\\_%C3%81lgebra\\_Relacional](https://www.academia.edu/578543/Una_Herramienta_para_el_Aprendizaje_del_%C3%81lgebra_Relacional)

Hernández, R., Fernández C. & Baptista, P. (2014). Metodología de la investigación científica. 6<sup>a</sup>. Ed. México: McGRAW-HILL.

Java Tools and Technologies Landscape for 2014. A global survey of 2164 Java professionals. RebelLabs, company ZeroTurnaround. Recuperado el 27 de julio de 2016: <http://www.yunsobi.com/blog/attachment/1549755630.pdf>

Java Tools and Technologies Landscape Report 2016. Survey results from 2040 geeks. RebelLabs, company ZeroTurnaround. Recuperado el 02 agosto de 2016: [http://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016/?utm\\_source=hashnode.com](http://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016/?utm_source=hashnode.com)

Joaquín Aldás & Ezequiel Uriel (2017). Análisis Multivariante aplicado con R. Segunda edición. Ediciones Paraninfo, S.A. España. Pág. 189.

Leisalu, O. (2009). *Comparative evaluation of two PHP persistence frameworks*. Tesis de licenciatura. Especialidad de Tecnología de la Información, Instituto de Ciencias de la Computación, Facultad de Matemáticas y Ciencias de la Computación, Universidad de Tartu, Tartu, Estonia.

Levin, R. & Rubin, D. (2004). Estadística para Administración y Economía. Séptima edición. Pearson Educación de México, S. A. de C.V. México. Pág. 622, 630, 655.

Lucana, S. Métodos de investigación científica y técnica aplicados a la Ingeniería de Telecomunicación. Ingeniería de Telecomunicación Avanzada, Escuela Técnica Superior de Ingenieros de Telecomunicación.

Moniruzzaman, A. & Hossain, S., 2013. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison

Navathe, S. & Elmasri, R. (2000). *Fundamentals of database systems*. 3<sup>a</sup>. Ed. Estados Unidos: Addison-Wesley

Nevado, M. (2010). *Introducción a las Bases de Datos Relacionales*. Madrid: Visión Libros.

- Oleg Kristov. (2005). *Using Zend framework 3*. Madrid.
- Oppel, A. (2009). *Fundamentos de Bases de Datos*. México: The McGraw-Hill Companies, Inc.
- Oppel, A. (2010). *Fundamentos de Bases de Datos*. México: McGraw-Hill Companies.
- Paszniuk, R. (15 de Julio de 2013). Investigación sobre estado del arte y aplicación de SBDoo. Obtenido de Programación & Bases de Datos: <http://www.programacion.com.py/varios/db/investigacion-sobre-estado-del-arte-y-aplicacion-de-sbdo>.
- Payá, M. (2010). *Análisis y uso de frameworks de persistencia en Java*. Proyecto de fin de carrera. Ingeniería Técnica en Informática de Gestión, Escuela Técnica Superior de Ingeniería (ICAI), Universidad Pontificia Comillas, Madrid, España.
- Revista Telem@tica. Vol. 10. No. 3, septiembre-diciembre, 2011, p. 1-7 ISSN 1729-3804. *Mapeo Objeto / Relacional (ORM)*. Osmel Yanes Enriquez, Hansel Gracia del Busto
- Ricardo, C. (2004). *Bases de Datos*. México: McGraw-Hill Companies.
- Rivas Graciano, R. (2013). Desarrollo De Sistemas De Información Con *Framework Spring-Hibernate*. Madrid.
- Silberschatz, A. (2002). *Fundamentos de bases de datos: cuarta edición*. McGraw-Hill/Interamericana: España.
- Talledo J. (2015). Acceso a datos en aplicaciones *Web* del entorno servidor UF1845. Ediciones Paraninfo, S. A. España. Pág. 12.
- Trends for ORM during 20/01/2012 – 02/08/2016. Recuperado el 03 de agosto de 2016: <http://trendyskills.com/#trends>
- Tumabrell A., Pupo, J., Álvarez, M. & Ricart, O. 2016, 14 de Marzo. *Aplicación informática para realizar el mapeo relacional de objetos para marcos de trabajo*. Informática 2016, XVI Convención y feria internacional.
- UNAM, (2007). *Introducción al Desarrollo de Programas con Java*. UNAM.

Valencia, S. & Aguilera, P. & Campaña, R. (2014). Aplicaciones informáticas para el comercio. Primera edición. Editorial Editex. España. Pág. 265.

Yanes, O. & Gracia, H. (2011, Septiembre). *Mapeo Objeto / Relacional (ORM)*. Revista digital de las tecnologías de la Información y las comunicaciones. Revista Telem@tica. Vol. 10. No. 3, septiembre-diciembre, 2011, p. 1-7. Recuperado el 14 de julio de 2016:

<http://revistatelematica.cujae.edu.cu/index.php/tele/article/view/23/21>

## **ANEXO**

## Anexo 1: Matriz de consistencia

MATRIZ DE CONSISTENCIA						
ESTUDIO COMPARATIVO DEL TIEMPO DE EJECUCIÓN ENTRE LOS FRAMEWORKS DE PERSISTENCIA ORM Y ODM APLICADO A UN MÓDULO WEB						
PROBLEMA	OBJETIVOS	HIPÓTESIS	VARIABLES	INDICADORES	METODOLOGÍA	INSTRUMENTOS DE RECOLECCIÓN DE DATOS
PROBLEMA GENERAL	OBJETIVO GENERAL	HIPÓTESIS GENERAL	X=VARIABLES INDEPENDIENTES			
¿Existen diferencias significativas en el tiempo de ejecución al hacer uso de los frameworks de persistencia ORM y ODM escritos con los lenguajes de programación PHP y Java?	Estudiar comparativamente el tiempo de ejecución entre frameworks de persistencia ORM y ODM haciendo uso de un módulo Web	Existen diferencias significativas en el tiempo de ejecución al hacer uso de las frameworks de persistencia ORM y ODM que operan con los lenguajes de programación PHP y Java	Framework de persistencia Sistema Gestor de Base de Datos Cantidad de registros o documentos	Desarrollo de software	1. Tipo de Investigación: Aplicada Tecnológica	Investigación Bibliográfica Encuesta Guía de encuesta Programa cronómetro Guía de observación Microsoft Excel 2013 SPSS Statistics 21
		HIPÓTESIS NULA	Variable Dependiente	Cronómetro	2. Nivel de Investigación: Descriptivo y correlacional	
		No existen diferencias significativas en el tiempo de ejecución al hacer uso de las frameworks de persistencia ORM y ODM que operan con los lenguajes de programación PHP y Java	Diferencias significativas en el tiempo de ejecución		3. Métodos de Investigación: Experimental y de medición.	
PROBLEMAS ESPECÍFICOS	OBJETIVOS ESPECÍFICOS	HIPÓTESIS ESPECÍFICAS	Y= VARIABLE DEPENDIENTE			
¿Es posible desarrollar un módulo Web haciendo uso de las tecnologías seleccionadas	Desarrollar el módulo Web haciendo uso de las tecnologías seleccionadas	Se puede desarrollar el módulo Web haciendo uso de las tecnologías seleccionadas	Variable Dependiente Desarrollo del módulo Web Variable Independiente Tecnologías seleccionadas	Desarrollo de aplicaciones Uso de tecnologías informáticas		
¿Qué framework de persistencia ORM y ODM ofrecen el menor tiempo de ejecución?	Determinar el framework de persistencia ORM y ODM que ofrecen el menor tiempo de ejecución	El tiempo de ejecución depende del framework de persistencia ORM u ODM usado	Variable Dependiente Tiempo de ejecución Variable Independiente Framework de persistencia	Cronómetro Software		
¿Qué módulo Web es el más usado por las entidades de la ciudad de Piura?	Identificar el módulo Web más usado por las entidades de la ciudad de Piura	El módulo Web más usado por las entidades de la ciudad de Piura se encontrará mediante encuestas	Variable Dependiente Módulo Web más usado Variable Independiente Encuestas realizadas a entidades de la ciudad de Piura	Porcentaje de uso Resultados de encuestas	4. Diseño de la Investigación: Experimental puro	

**Anexo 2: Constancia de validación de instrumento**

**CONSTANCIA DE VALIDACIÓN**  
**DE INSTRUMENTO**

Yo, \_\_\_\_\_, identificado con DNI N° \_\_\_\_\_, de profesión \_\_\_\_\_, con grado de \_\_\_\_\_ ejerciendo actualmente como \_\_\_\_\_, en la Institución \_\_\_\_\_.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento (encuesta), a los efectos de su aplicación al personal que labora en los centros de desarrollo de *software* de las distintas empresas de la ciudad de Piura.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Items	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Congruencia de preguntas				
Amplitud de contenido				
Redacción de preguntas				
Claridad y precisión				

En Piura, a los \_\_\_\_\_ días del mes de \_\_\_\_\_ de \_\_\_\_\_.

\_\_\_\_\_  
Firma

### **Anexo 3: Encuesta para desarrolladores para obtener el módulo Web más utilizado**

## **ENCUESTA PARA DESARROLLADORES DE EMPRESAS PÚBLICAS Y PRIVADAS DE LA CIUDAD DE PIURA**

Cargo del encuestado: \_\_\_\_\_

Nombre de la empresa donde labora: \_\_\_\_\_

### **OBJETIVO DE LA ENCUESTA**

La presente encuesta está dirigida al personal que labora en las oficinas de desarrollo de *software* de las entidades públicas y privadas de la ciudad de Piura. El objetivo principal de la misma es determinar el módulo Web más utilizado dependiendo de las operaciones que se realizan a diario, así como también identificar el grado de utilización de *frameworks* para el desarrollo de *software*; entre otras preguntas adicionales de importancia para el encuestador.

Agradezco dar sus respuestas con la mayor transparencia y veracidad.

### **INSTRUCCIONES**

La mayor parte de las preguntas de la presente encuesta han sido formuladas para que seleccione varias de las alternativas que se brindan, en caso contrario se especificará en el enunciado de la misma. Existe otra clases de preguntas que se contestarán o no dependiendo de la respuesta de la pregunta anterior, para estos casos se específica a que pregunta se debe dirigir en caso de que su respuesta sea negativa.

### **PREGUNTAS**

- |  |   |
|--|---|
| <p>1. ¿En la empresa donde labora qué tipo de <i>software</i> utilizan?</p> <p>( ) Móvil<br/>( ) Escritorio<br/>( ) Web</p> <p>2. ¿De qué forma es obtenido el <i>software</i> utilizado en la empresa?</p> <p>( ) Compra de <i>software</i> genérico.<br/>( ) Desarrollado por una empresa de <i>software</i>.<br/>( ) Contratación de personal para el desarrollo del <i>software</i>.<br/>( ) Desarrollado dentro de la misma sede de la empresa. (En caso sea sucursal o sede)</p> | <p>( ) Desarrollado en la sede principal de la empresa. (Se selecciona esta alternativa en caso sea una sola sucursal o sede)</p> <p>3. ¿Cuál o cuáles son los lenguajes que utilizan para desarrollar el lado servidor de los sistemas web?</p> <p>( ) Java<br/>( ) PHP<br/>( ) .NET<br/>( ) Visual FoxPro<br/>( ) Otros. Especifique:<br/>_____<br/>_____</p> |
|--|---|

\_\_\_\_\_

\_\_\_\_\_

4. ¿Qué editores de texto o IDE's utilizan para realizar la programación?

- ( ) Microsoft Visual Studio  
( ) NetBeans IDE  
( ) Eclipse  
( ) Sublime Text  
( ) Notepad++  
( ) Otros. Especifique:
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_

5. ¿Ha utilizado algún *framework* para el desarrollo de sistemas web?

- ( ) Sí.  
( ) No. (Continuar a partir de la pregunta 7)

6. ¿Cuál o cuáles son los *frameworks* que ha utilizado para el desarrollo de sistemas dentro de la empresa donde labora?

- ( ) Spring *framework*  
( ) Laravel  
( ) CodeIgniter  
( ) Openxava  
( ) Symfony  
( ) CakePHP  
( ) Otros. Especifique:
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_

7. ¿Ha utilizado algún ORM (Mapeo Relacional de Objetos) para el desarrollo de sistemas web?

- ( ) Sí.  
( ) No. (Si marcó "Sí" en la pregunta 5 continuar con la pregunta 9, sino continuar con la pregunta 10)

8. ¿Cuál o cuáles son los ORM (*framework* de persistencia) que ha utilizado para el desarrollo de sistemas web dentro de la empresa donde labora?

- ( ) Hibernate  
( ) Eloquent de Laravel  
( ) LINQ  
( ) Doctrine  
( ) Otros. Especifique:
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_

9. De las siguientes características dadas, enumere el orden de importancia que se tuvo en cuenta para poder tomar la elección para usar un framework. (Se incluye los ORM)

- ( ) Alta popularidad.  
( ) Aumento en velocidad de desarrollo.  
( ) Seguridad. (Ataques SQL Injections)  
( ) Por el lenguaje de programación en el que está basado.  
( ) Buena documentación.  
( ) Rapidez para aprender a usarlo.  
( ) Abstracción de la base de datos.  
( ) Mantenimiento del código.  
( ) Otros. Especifique:
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_

---

---

10. ¿Conoce el término ODM?

- ( ) Sí.  
( ) No. (Continuar a partir de la pregunta 13)

11. ¿Cuál o cuáles de los siguientes ODM conoce?

- ( ) Hibernate OGM  
( ) Doctrine ODM  
( ) Mandango  
( ) Kundera  
( ) Morphia  
( ) Otros. Especifique:
- 
- 
- 

12. ¿Ha utilizado alguno de los *frameworks* de persistencia ODM en un proyecto dentro o fuera de la empresa?

- ( ) Sí.  
( ) No.

13. ¿Cuál o cuáles sistemas gestores de base de datos utilizan para la administración de datos? (Subraye aquel que es más usado o especifíquelo)

- ( ) SQL Server  
( ) MySQL  
( ) Oracle  
( ) PostgreSQL  
( ) Access  
( ) Otros. Especifique:
- 
- 

---

---

14. De las siguientes características dadas, enumere el orden de importancia que se tuvo en cuenta para poder tomar la elección del o de los gestores de base de datos elegidos en la pregunta anterior.

- ( ) Alta disponibilidad.  
( ) Cantidad y naturaleza de los tipos de datos.  
( ) Volumen de datos que soporta.  
( ) Seguridad.  
( ) Integración con otras tecnologías.  
( ) Costos de licencia.  
( ) Asistencia técnica.  
( ) Otros. Especifique:
- 
- 
- 
- 
- 
- 

15. ¿En algún momento ha utilizado un sistema manejador de base de datos NoSQL?

- ( ) Sí.  
( ) No. (Continuar a partir de la pregunta 20)

16. ¿Cuál o cuáles sistemas manejadores de base de datos NoSQL ha usado? ( Se incluyen pruebas con el *Software*)

- ( ) MongoDB  
( ) Apache Cassandra  
( ) CouchDB  
( ) Redis  
( ) Neo4j  
( ) Otros. Especifique:
- 
-

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

17. ¿Ha utilizado algunos de los sistemas gestores de base de datos anteriores en un proyecto dentro o fuera de la empresa?

- ( ) Sí.  
( ) No. (Continuar a partir de la pregunta 20)

18. ¿Con qué objetivo se utilizó el o los SGBD marcados en la pregunta 16?

- ( ) La escalabilidad era un factor importante.  
( ) Se decidió probar el rendimiento de este tipo de SGBD.  
( ) Para hacer uso de su esquema dinámico o flexible  
( ) Otros. Especifique:

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

19. ¿Cómo calificaría los resultados obtenidos tras usar el o los SGBD elegidos en la pregunta 16?

- ( ) Se obtuvieron mejores resultados de los esperados.  
( ) Aceptable, lo volvería a usar.  
( ) Regular, no lo volvería a usar.  
( ) Deficiente

Por favor describa dichos resultados:

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

20. En la empresa donde labora actualmente, ¿Cuál es el módulo Web más utilizado? (Sólo puede elegir una alternativa)

- ( ) Módulo de Finanzas.  
( ) Módulo de Facturación.  
( ) Módulo de Contabilidad.  
( ) Módulo de Ventas.  
( ) Módulo de Control de inventario.  
( ) Módulo de Trámite documentario.  
( ) Otros: Especifique:

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

21. Según la respuesta de la pregunta anterior, ¿Qué tan complejo es el desarrollo del módulo en relación a las funciones o interacciones que realiza?

Por ejemplo la complejidad del módulo puede ser debido a varias causas, como las que se mencionan a continuación: se requiere de muchos componentes para su elaboración, interactúa frecuentemente con la base de datos, realiza muchos cálculos, usa una web services, etc. (Sólo puede elegir una alternativa)

- ( ) Muy complejo  
( ) Complejo  
( ) No precisa  
( ) Sencillo  
( ) Muy sencillo

Por favor, explique según lo respondido:

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

---

---

---

---

---

---

---

22. Seg n el m dulo seleccionado en la pregunta 21, ¿Cu l o cu les son las operaciones m s accedidas dentro del m dulo, es decir aquella donde una gran cantidad de usuarios interact an frecuentemente diariamente?

---

---

---

---

---

---

---

---

---

---

---

Además, cuál es la cantidad aproximada de usuarios que acceden usualmente a utilizar esta operación.

23. Aproximadamente, ¿Cuál es la cantidad de tablas con las que cuenta la base de datos del módulo elegido en la pregunta 20?

(Sólo puede elegir una alternativa)

- ( ) De 10 a 20.
  - ( ) Entre 20 y 40.
  - ( ) Entre 40 y 60.

- ( ) Entre 60 y 80.  
( ) Entre 80 y 100.  
( ) Mas de 100. Por favor indique la cantidad aproximada:

24. Del módulo elegido en la pregunta 20,  
¿Cuál es la máxima cantidad de datos  
almacenados que almacena una tabla de  
la base de datos del módulo?  
(Sólo puede elegir una alternativa)

- ( ) Menos de 100 mil.
  - ( ) Entre 100 mil y 400 mil
  - ( ) Entre 400 mil y 800 mil
  - ( ) Entre 800 mil y 1 millón 200 mil
  - ( ) Más de 1 millón 200 mil.

Nota: Si eligió la primera o la última alternativa, por favor especifique un aproximado de datos:

25. Mencione algunos de los problemas que se han detectado al hacer uso del módulo elegido en la pregunta 20.

- ( ) Funciones poco entendibles para el usuario.
  - ( ) Interfaz gráfica de usuario poco amigable para el usuario.
  - ( ) La información que se muestra al usuario no es la que espera.
  - ( ) Demora al realizar consultas al servidor.
  - ( ) Otros. Especifique:

---

---

---

---

#### Anexo 4: Constancias de validación de encuesta

<u>CONSTANCIA DE VALIDACIÓN DE INSTRUMENTO</u>				
Items	DEFICIENTE	ACCEPTABLE	BUENO	EXCELENTE
Congruencia de preguntas			✓	
Amplitud de contenido			✓	
Redacción de preguntas			✓	
Claridad y precisión			✓	

Yo, Arturo Sandoval Rivero,  
identificado con DNI N° 02887733, de profesión  
INGENIERO INDUSTRIAL, con grado de  
INGENIERO, ejerciendo actualmente como  
DOCENTE, en la Institución  
UNIVERSIDAD NACIONAL DE PIURA.

Por medio de la presente hago constar que he revisado con fines de Validación del instrumento (encuesta), a los efectos de su aplicación al personal que labora en los centros de desarrollo de software de las distintas empresas de la ciudad de Piura.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

En Piura, a los 31 días del mes de Octubre de 2016.

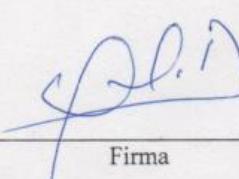
  
Firma

Figura 42. Constancia de validación de instrumento

## Anexo 5: Constancia de validación de guía de observación de encuesta

### CONSTANCIA DE VALIDACIÓN DE GUÍA DE OBSERVACIÓN

Yo, \_\_\_\_\_, identificado con  
DNI N° \_\_\_\_\_, de profesión \_\_\_\_\_,  
con grado de \_\_\_\_\_ ejerciendo  
actualmente como \_\_\_\_\_ en la Institución  
\_\_\_\_\_.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento (guía de observación), a los efectos de su aplicación a la toma de tiempos de ejecución del módulo *Web* de Trámite Documentario desarrollado con *frameworks* de persistencia ORM y ODM, en un entorno simulado.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Ítems	Criterios	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Claridad	Está formulado en un lenguaje apropiado				
Objetividad	Está expresado en conductas observables				
Coherencia	Coherencia entre los indicadores				
Metodología	Responde al propósito				

En Piura, a los \_\_\_\_\_ días del mes de \_\_\_\_\_ de \_\_\_\_\_.

\_\_\_\_\_  
Firma

**Anexo 6: Guía de observación para la toma de tiempos de ejecución del módulo Web de Trámite Documentario**

**Guía de observación N° 01**

**TIEMPOS DE EJECUCIÓN DEL MÓDULO WEB DE TRÁMITE DOCUMENTARIO DESARROLLADO CON *FRAMEWORKS* DE PERSISTENCIA ORM Y ODM**

La presente guía de observación es utilizada con la intención de medir los tiempos de ejecución de las de las aplicaciones desarrolladas con los *frameworks* de persistencia ORM y ODM basados los lenguajes de programación Java y PHP, estos *frameworks* de persistencia trabajan en conjunto con las bases de datos relacionales y no relacionales (NoSQL), respectivamente.

La guía proporcionará la información necesaria para definir las mejores combinaciones:

- ❖ *Framework* de persistencia ORM y sistema gestor de base de datos relacional (SGBDR)
- ❖ *Framework* de persistencia ODM y sistema gestor de base de datos NoSQL (SGBD NoSQL)

En base al tiempo de ejecución que se obtenga por cada combinación.

También se obtendrá los tiempos tanto de inserción como de lectura de datos en las bases de datos.

El observador deberá registrar el tiempo respuesta, expresado en milisegundos de cada prueba para cada combinación.

		MariaDB								PostgreSQL							
		C <sub>1</sub>		C <sub>2</sub>		C <sub>3</sub>		C <sub>4</sub>		C <sub>1</sub>		C <sub>2</sub>		C <sub>3</sub>		C <sub>4</sub>	
		I	L	I	L	I	L	I	L	I	L	I	L	I	L	I	L
Hibernate ORM	r <sub>1</sub>																
	r <sub>2</sub>																
	r <sub>3</sub>																
	r <sub>4</sub>																
Doctrine ORM	r <sub>1</sub>																
	r <sub>2</sub>																
	r <sub>3</sub>																
	r <sub>4</sub>																

	MongoDB								CouchDB							
	C <sub>1</sub>		C <sub>2</sub>		C <sub>3</sub>		C <sub>4</sub>		C <sub>1</sub>		C <sub>2</sub>		C <sub>3</sub>		C <sub>4</sub>	
	I	L	I	L	I	L	I	L	I	L	I	L	I	L	I	L
Hibernate OGM	r <sub>1</sub>															
	r <sub>2</sub>															
	r <sub>3</sub>															
	r <sub>4</sub>															
Doctrine ODM	r <sub>1</sub>															
	r <sub>2</sub>															
	r <sub>3</sub>															
	r <sub>4</sub>															

Fecha de inicio de la observación: \_\_\_\_/\_\_\_\_/\_\_\_\_ (dd/mm/aaaa)

Fecha de finalización de la observación: \_\_\_\_/\_\_\_\_/\_\_\_\_ (dd/mm/aaaa)

### Observaciones:

---



---



---

Investigador: \_\_\_\_\_

Fecha de entrega: \_\_\_\_/\_\_\_\_/\_\_\_\_ (dd/mm/aaaa)

(   ) Verificado    (   ) Archivado    (   ) Entregado

## Anexo 7: Constancia de Validación de Guía de Observación

**CONSTANCIA DE VALIDACIÓN  
DE GUÍA DE OBSERVACIÓN**

Yo, Carmen Z. Quito Rodríguez, identificado con DNI N° \_\_\_\_\_, de profesión Ingr. Industrial, con grado de Magíster ejerciendo actualmente como Docente en la Institución UNP.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento (guía de observación), a los efectos de su aplicación a la toma de tiempos de ejecución del módulo web de trámite documentario desarrollado con frameworks de persistencia ORM y ODM, en un entorno simulado.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Ítems	Criterios	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Claridad	Está formulado en un lenguaje apropiado			X	
Objetividad	Está expresado en conductas observables			X	
Coherencia	Coherencia entre los indicadores			X	
Metodología	Responde al propósito			X	

En Piura, a los 16 días del mes de abril de 2018.

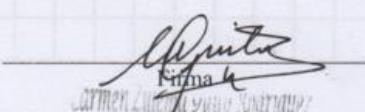
  
Firma  
CARMEN ZULIA QUITO RODRIGUEZ  
Ingeniera Industrial  
CIP. N° 81312

Figura 43. Primera constancia de validación de guía de observación

**CONSTANCIA DE VALIDACIÓN**  
**DE GUÍA DE OBSERVACIÓN**

Yo, Luis Armando Sánchez Yarlequé, identificado con DNI N° 02849179, de profesión INGENIERO INFORMATICO, con grado de MAGISTER ejerciendo actualmente como DOCENTE, en la Institución UNIVERSIDAD NACIONAL DE PIURA.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento (guía de observación), a los efectos de su aplicación a la toma de tiempos de ejecución del módulo web de trámite documentario desarrollado con frameworks de persistencia ORM y ODM, en un entorno simulado.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Ítems	Criterios	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Claridad	Está formulado en un lenguaje apropiado				✓
Objetividad	Está expresado en conductas observables			✓	
Coherencia	Coherencia entre los indicadores				✓
Metodología	Responde al propósito				✓

En Piura, a los 16 días del mes de ABRIL de 2018.

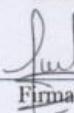
  
Firma

Figura 44. Segunda constancia de validación de guía de observación

**CONSTANCIA DE VALIDACIÓN**  
**DE GUÍA DE OBSERVACIÓN**

Yo, Saul Arnaldo Sullon Iachipa, identificado con DNI N° 4239 0203, de profesión INGENIERO DE SISTEMAS e INF, con grado de INGENIERO DE SISTEMAS ejerciendo actualmente como jefe de tecnología de la INF. en la Institución Municipalidad Provincial Nazca - Chulucanas

Por medio de la presente hago constar que he revisado con fines de validación del instrumento (guía de observación), a los efectos de su aplicación a la toma de tiempos de ejecución del módulo web de trámite documentario desarrollado con frameworks de persistencia ORM y ODM, en un entorno simulado.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

Ítems	Criterios	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Claridad	Está formulado en un lenguaje apropiado		X		
Objetividad	Está expresado en conductas observables		X		
Coherencia	Coherencia entre los indicadores		X		
Metodología	Responde al propósito		X		

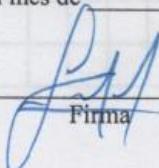
En Piura, a los \_\_\_\_\_ días del mes de \_\_\_\_\_ de \_\_\_\_\_.  
  
\_\_\_\_\_  
Firma

Figura 45. Tercera constancia de validación de guía de observación

## Anexo 8: Capturas de pantalla del módulo de Trámite Documentario desarrollado

Información Expediente

**Asunto:** SOLICITUD PARA PRACTICAS PRE-PROFESIONALES

**Clasificar:** GENERAL    **Prioridad:** NORMAL    **Proveido:** ACCION NECESARIA

**Origen:** INTERNO    **Subdocumentos:** No    **Código de exp.:** 000002-2018-I

**Folios:** 8    **Fecha de registro:** 13/04/2018

**Remitente:** 61152516    **Representante:** 03323212    **Responsable:** JONATHAN PAUL AREVALO GARAVITO, GUILLERMO MORALES PULACHE

**Observación:** ESTA ES LA OBSERVACION

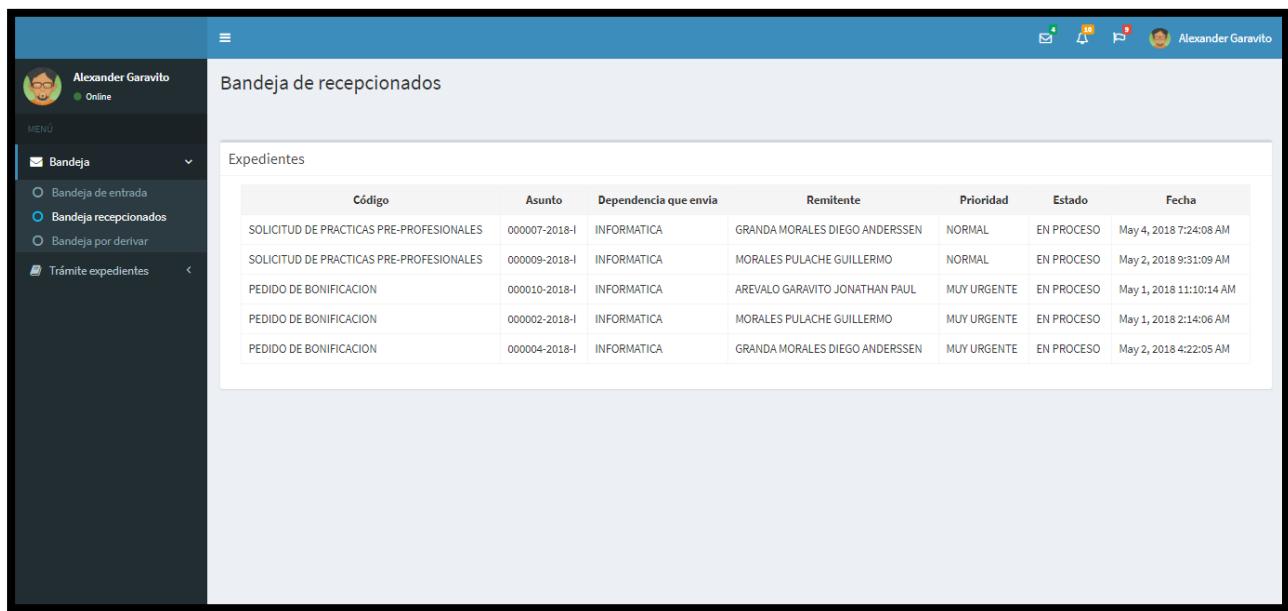
**Guardar**

Figura 46. Registro de expediente en el Módulo de Trámite Documentario  
Fuente: Módulo desarrollado

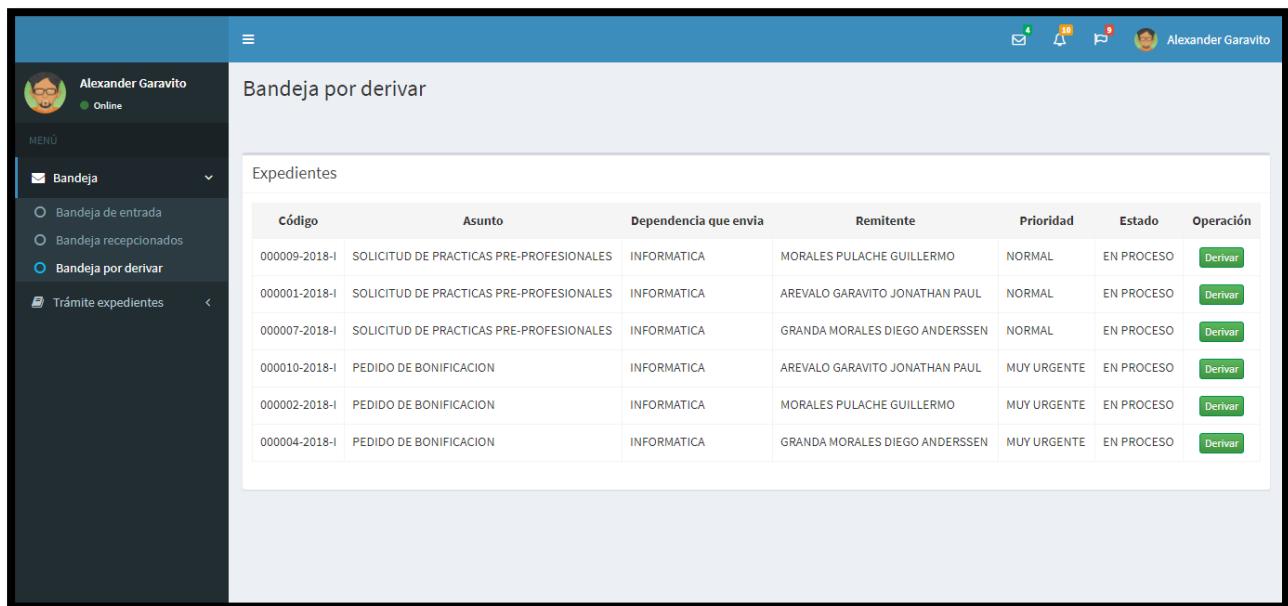
Expedientes

Correlativo	Asunto	Dependencia que envía	Remitente	Prioridad	Estado	Fecha
SOLICITUD DE PRACTICAS PRE-PROFESIONALES	000009-2018-I	INFORMATICA	MORALES PULACHE GUILLERMO	NORMAL	EN PROCESO	May 2, 2018 9:31:09 AM
SOLICITUD DE PRACTICAS PRE-PROFESIONALES	000001-2018-I	INFORMATICA	AREVALO GARAVITO JONATHAN PAUL	NORMAL	EN PROCESO	May 13, 2018 8:28:09 AM
SOLICITUD DE PRACTICAS PRE-PROFESIONALES	000007-2018-I	INFORMATICA	GRANDA MORALES DIEGO ANDERSSEN	NORMAL	EN PROCESO	May 4, 2018 7:24:08 AM
PEDIDO DE BONIFICACION	000002-2018-I	INFORMATICA	MORALES PULACHE GUILLERMO	MUY URGENTE	EN PROCESO	May 1, 2018 2:14:06 AM
PEDIDO DE BONIFICACION	000004-2018-I	INFORMATICA	GRANDA MORALES DIEGO ANDERSSEN	MUY URGENTE	EN PROCESO	May 2, 2018 4:22:05 AM

Figura 47. Bandeja de Entrada en el Módulo de Trámite Documentario  
Fuente: Módulo desarrollado



*Figura 48. Bandeja de Recepcionados en el Módulo de Trámite Documentario  
Fuente: Módulo desarrollado*



*Figura 49. Bandeja de por Derivar en el Módulo de Trámite Documentario  
Fuente: Módulo desarrollado*

Búsqueda del expediente

Información Expediente

Código expediente:	Clasificar:	Prioridad:
	GENERAL	NORMAL
Proveido:	Origin:	Estado del exp.:
ACCION NECESARIA	INTERNO	-- Seleccionar --
Desde:	Hasta:	
dd/mm/aaaa	dd/mm/aaaa	

Expedientes encontrados

Código	Asunto	Remitente	Prioridad	Estado
000001-2018-I	ESTE ES EL ASUNTO	AREVALO GARAVITO JONATHAN PAUL	NORMAL	EN PROCESO

Figura 50. Búsqueda de expedientes en el Módulo de Trámite Documentario  
Fuente: Módulo desarrollado

## Anexo 9: Test de Kolmogorov-Smirnov para los tiempos de ejecución

### Hipótesis

$H_0$ : La distribucion de la variable dependiente no difiere de una distribucion normal.

$H_1$ : La distribucion de la variable dependiente difiere de una distribución normal.

Para este test se utiliza un nivel de significancia del 5%.

Operación	Valor calculado	Valor de P	Conclusión
Registro de expediente en módulo desarrollado con ORM	2.868	0.00	Rechazo $H_0$
Consulta de bandeja de entrada en módulo desarrollado con ORM	2.859	0.00	Rechazo $H_0$
Registro de expediente en módulo desarrollado con ODM	2.449	0.00	Rechazo $H_0$
Consulta de bandeja de entrada en módulo desarrollado con ODM	2.194	0.00	Rechazo $H_0$

Tabla 63. Resultados del test de Kolmogorov – Smirnov

Fuente: IBM SPSS Statistics 21

Con los datos obtenidos podemos llegar a la conclusión que ninguna de las variables dependientes (tiempos de ejecución de las operaciones de los módulos) se ajusta a una distribución normal.

**Anexo 10: Gráficos obtenidos del análisis lineal univariante para los frameworks de persistencia ORM**

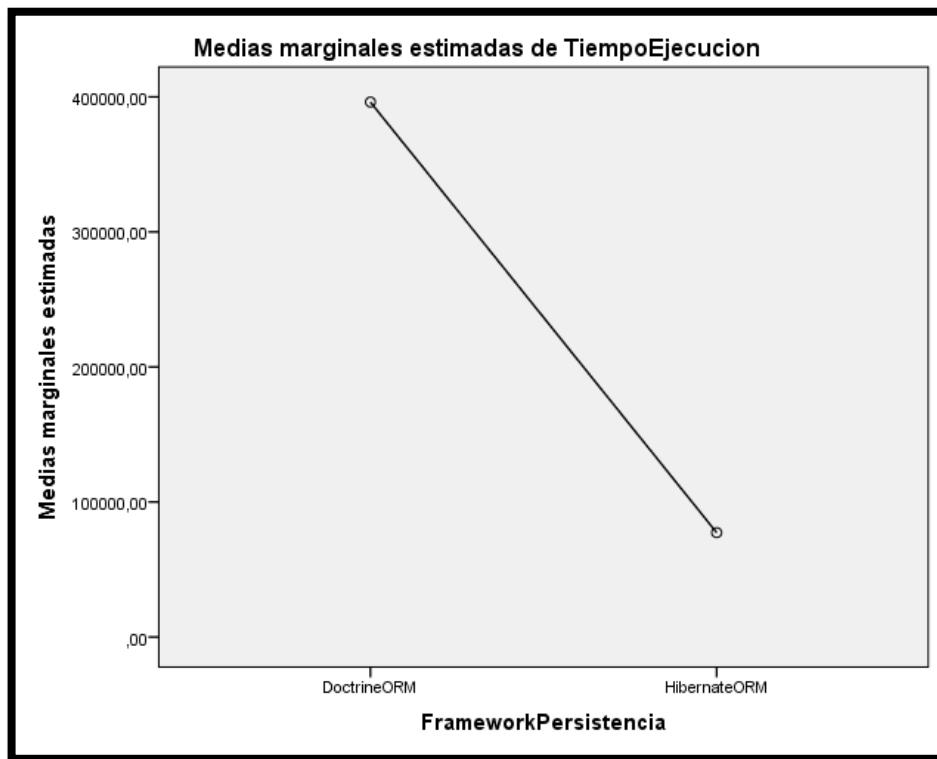


Gráfico 6. Medias marginales estimadas en tiempo de ejecución del factor ORM

Fuente: IBM SPSS Statistics 21

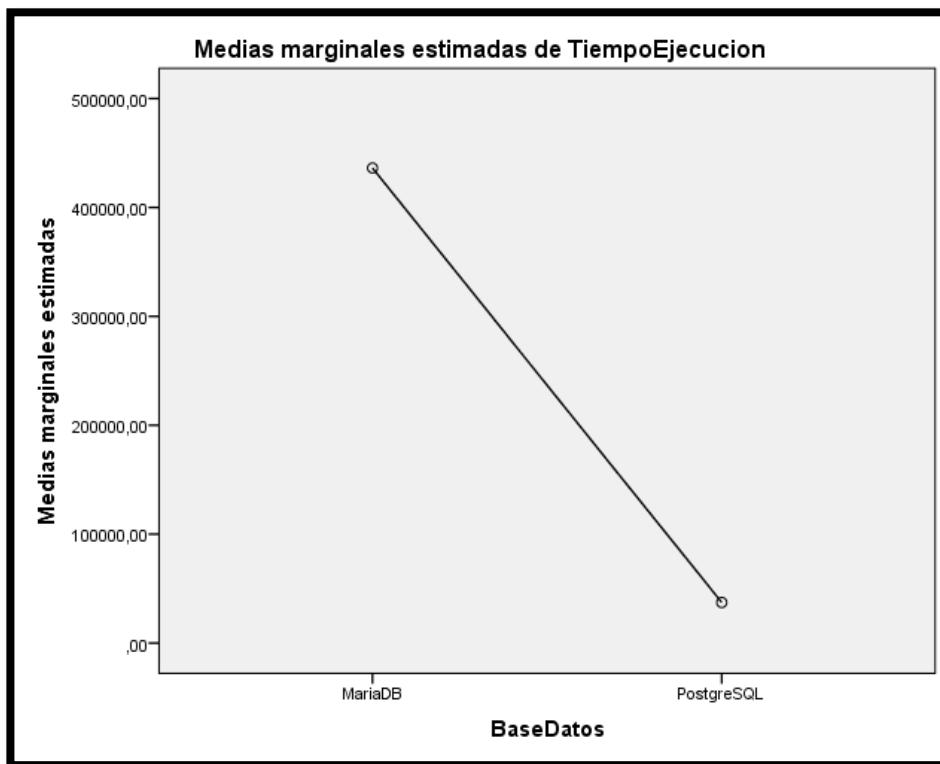


Gráfico 7. Medias marginales estimadas en tiempo de ejecución del factor SGBDR.

Fuente: IBM SPSS Statistics 21

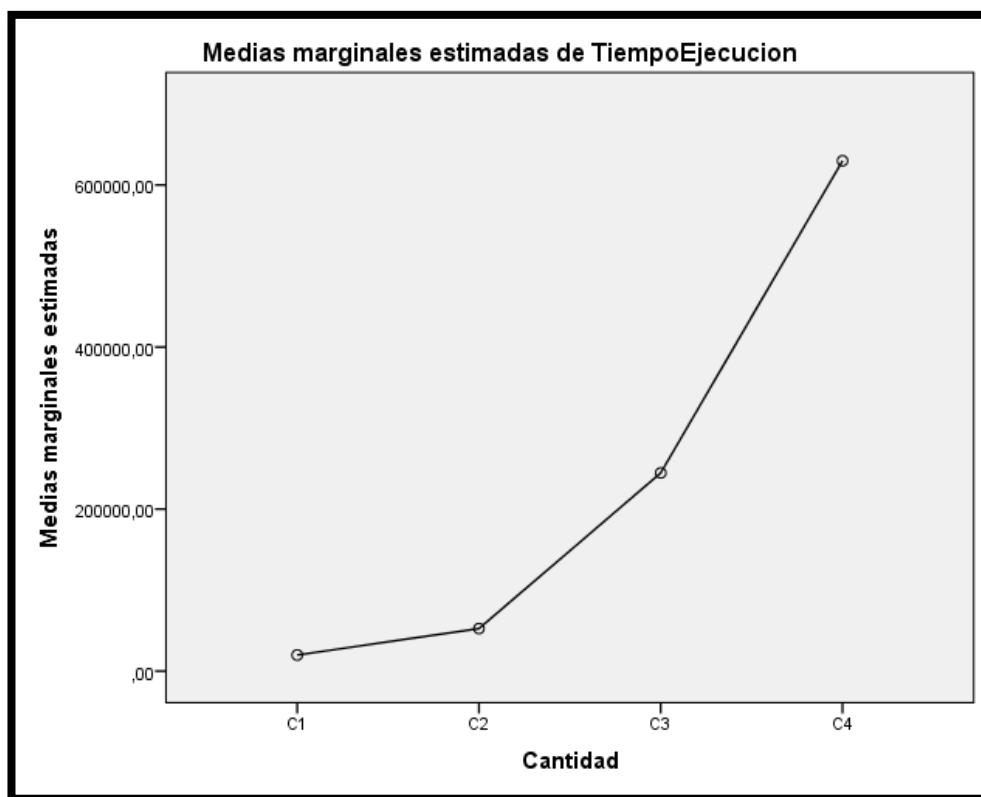


Gráfico 8. Medias marginales estimadas en tiempo de ejecución del factor Cantidad de datos  
Fuente: IBM SPSS Statistics 21

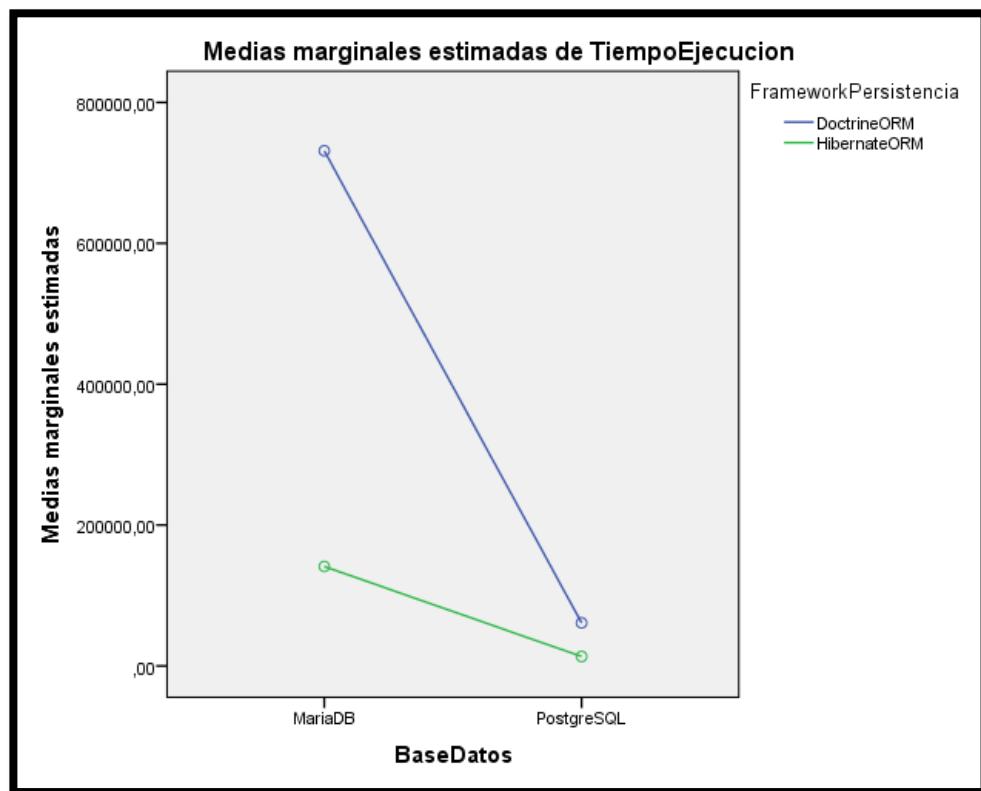


Gráfico 9. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ORM y SGBDR.

Fuente: IBM SPSS Statistics 21

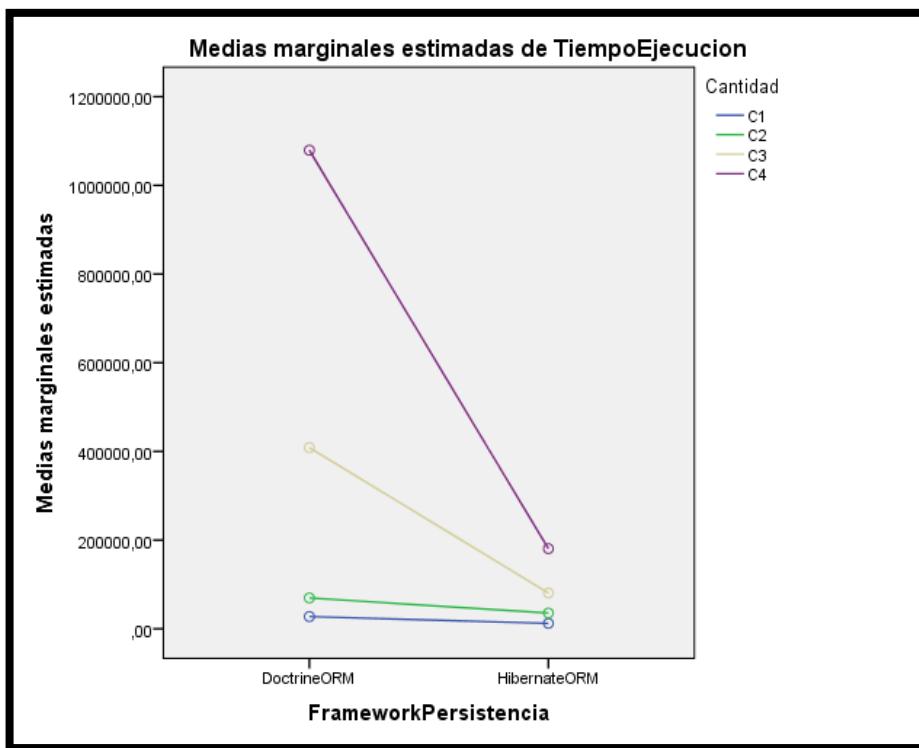


Gráfico 10. *Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ORM y Cantidad de datos.*

Fuente: IBM SPSS Statistics 21

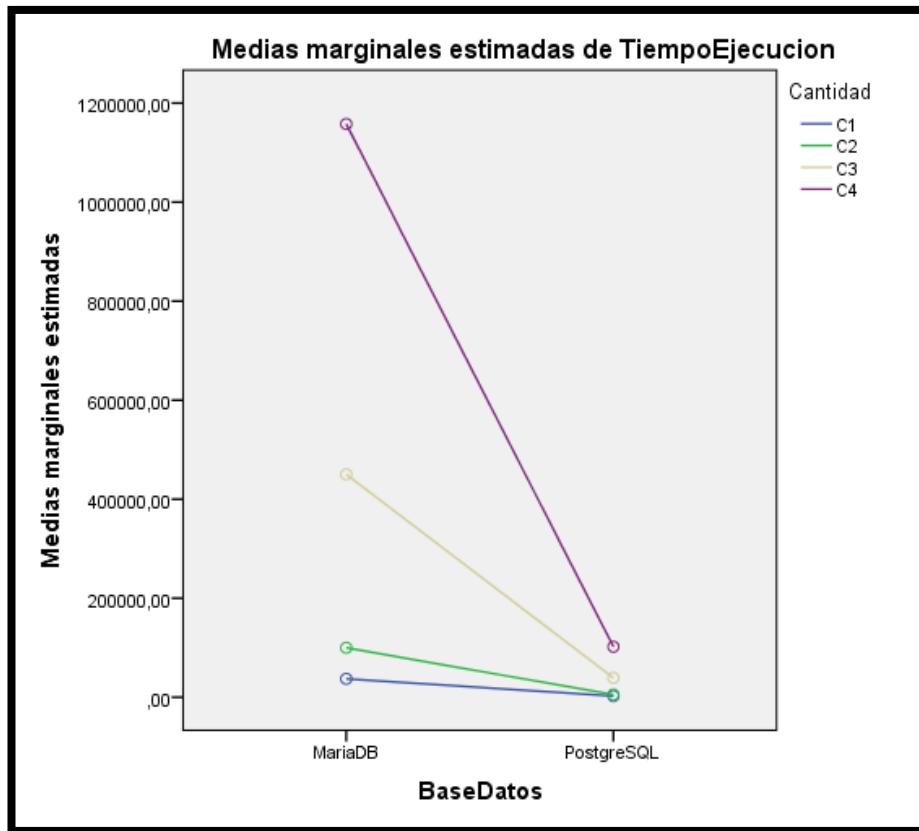


Gráfico 11. *Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: SGBDR y la Cantidad de datos.*

Fuente: IBM SPSS Statistics 21

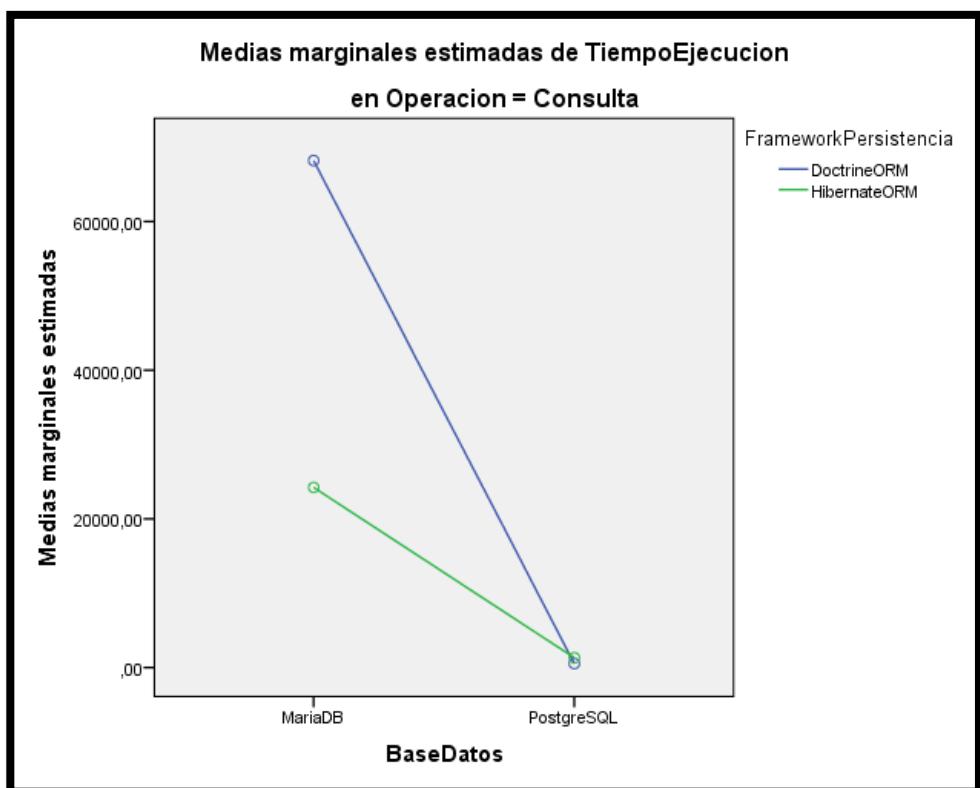


Gráfico 12. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ORM, SGBDR para la operación consulta de datos.

Fuente: IBM SPSS Statistics 21

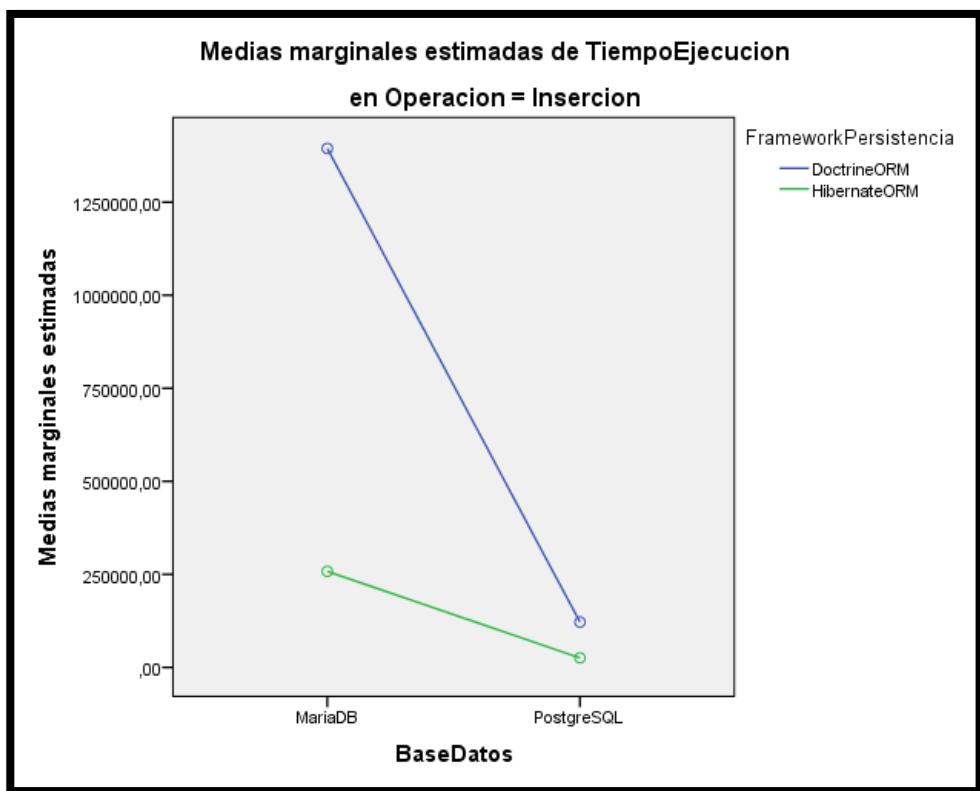
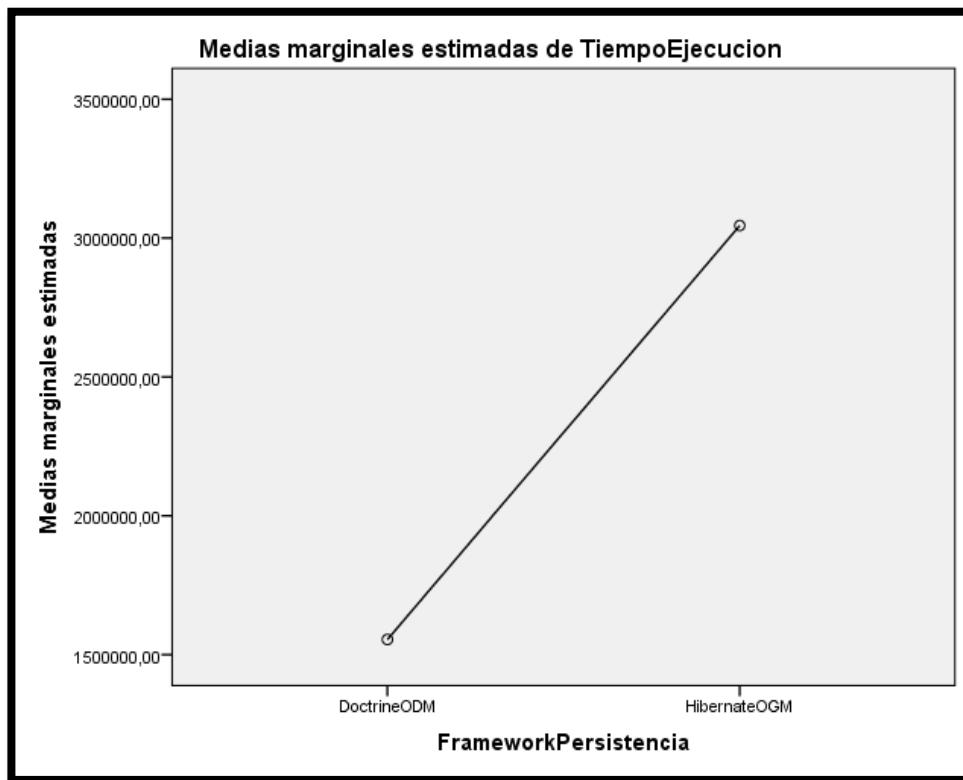


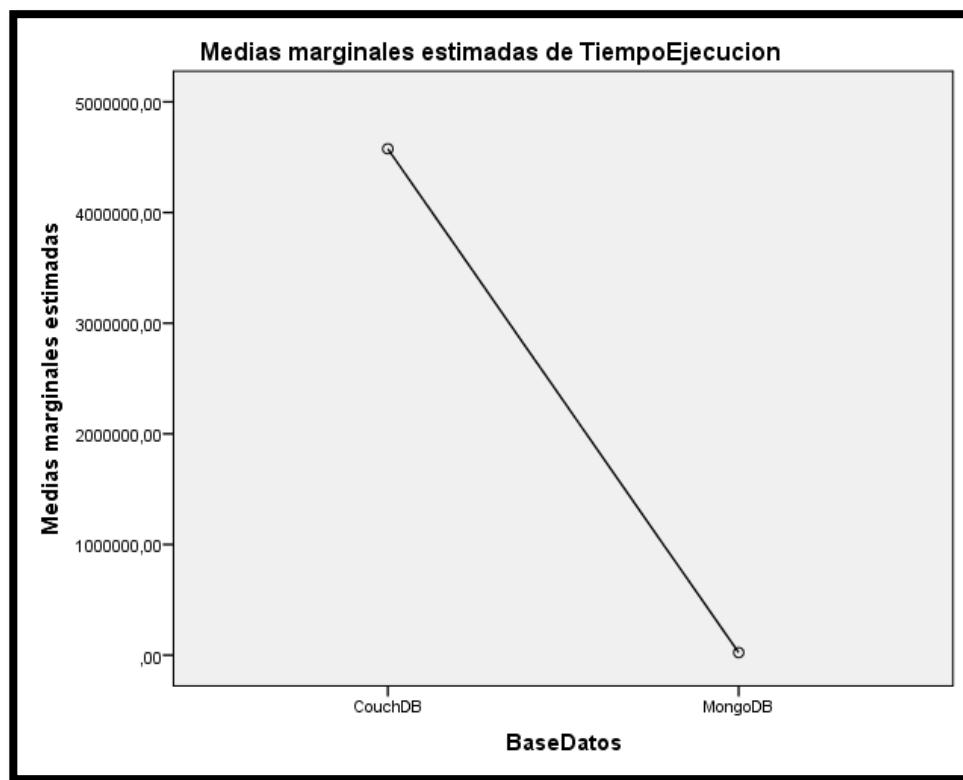
Gráfico 13. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ORM, SGBDR para la operación inserción de datos.

Fuente: IBM SPSS Statistics 21

**Anexo 11: Gráficos obtenidos del análisis lineal univariante para los frameworks de persistencia ODM**



*Gráfico 14. Medias marginales estimadas en tiempo de ejecución del factor ODM.  
Fuente: IBM SPSS Statistics 21*



*Gráfico 15. Medias marginales estimadas en tiempo de ejecución del factor SGBD NoSQL.  
Fuente: IBM SPSS Statistics 21*

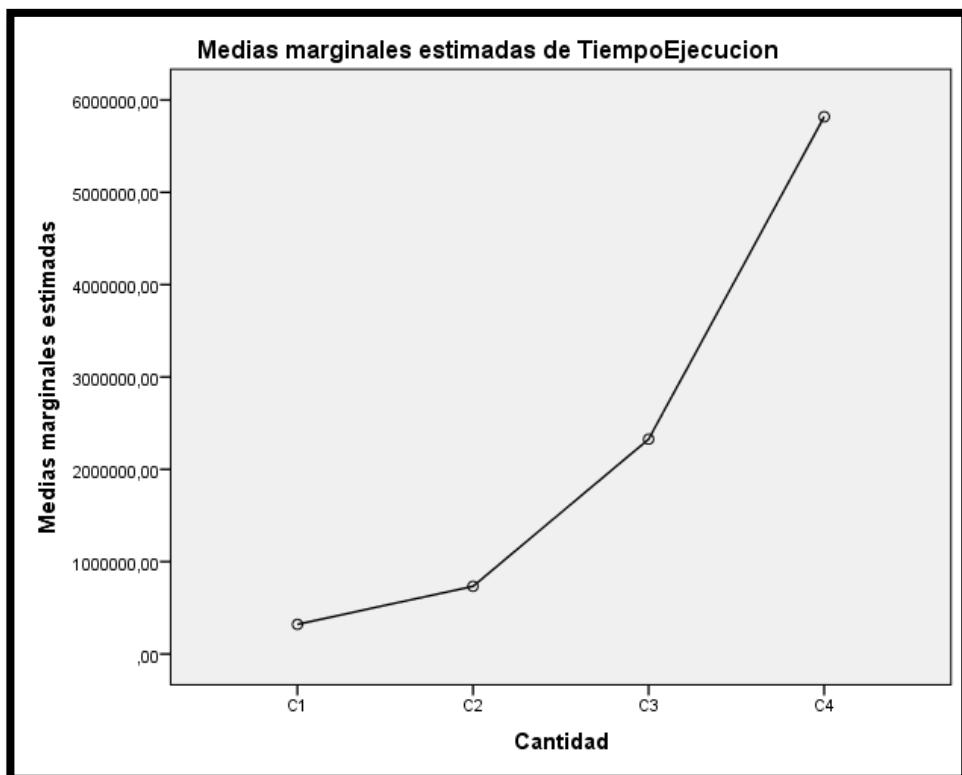


Gráfico 16. Medias marginales estimadas en tiempo de ejecución del factor cantidad de datos.

Fuente: IBM SPSS Statistics 21

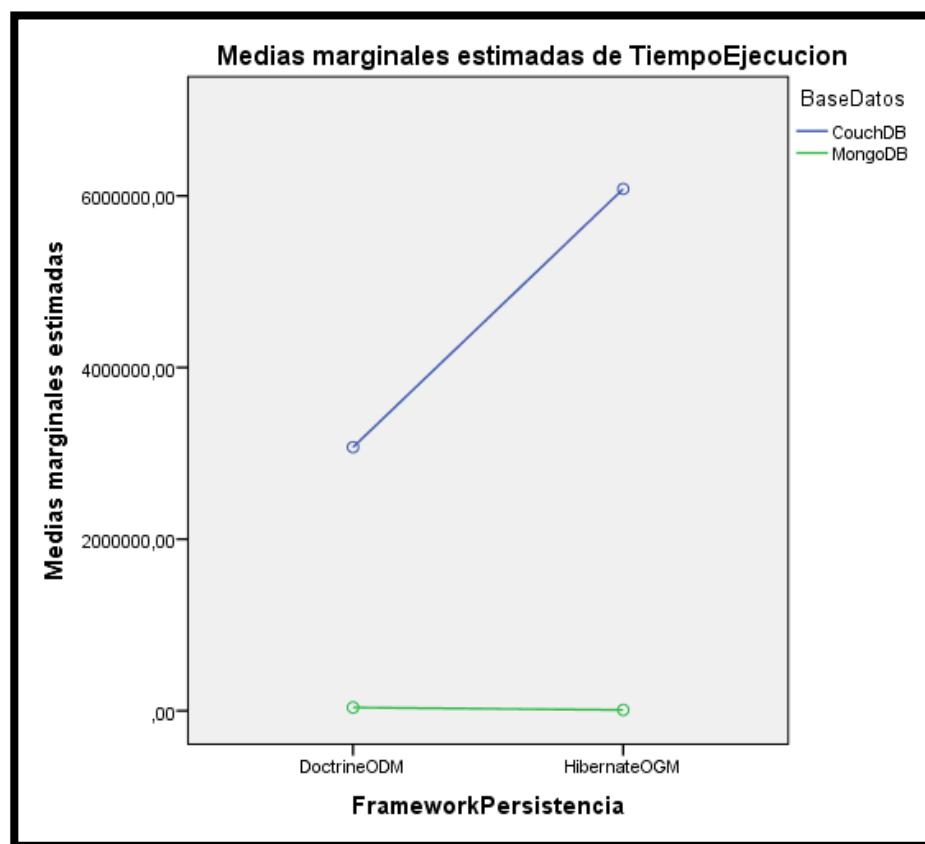


Gráfico 17. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores:

ODM y SGBD NoSQL.

Fuente: IBM SPSS Statistics 21

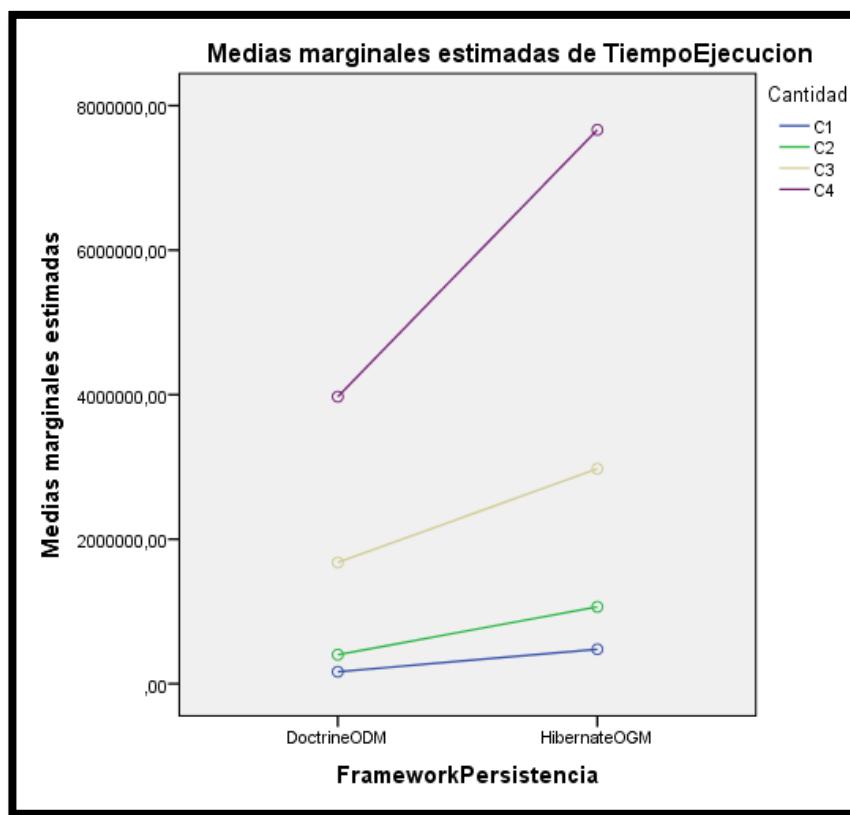


Gráfico 18. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ODM y cantidad de datos.

Fuente: IBM SPSS Statistics 21

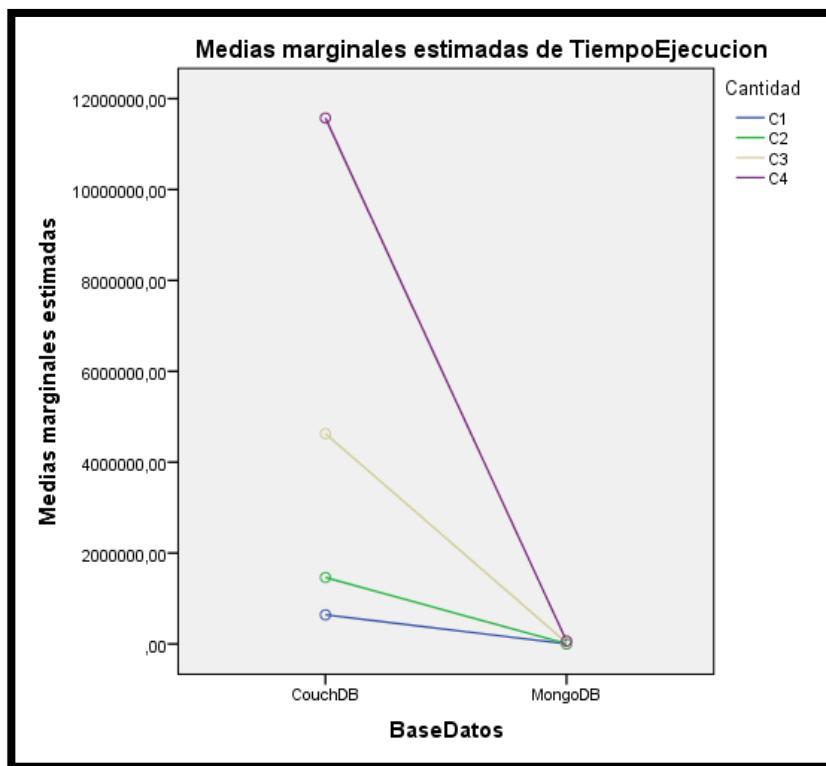
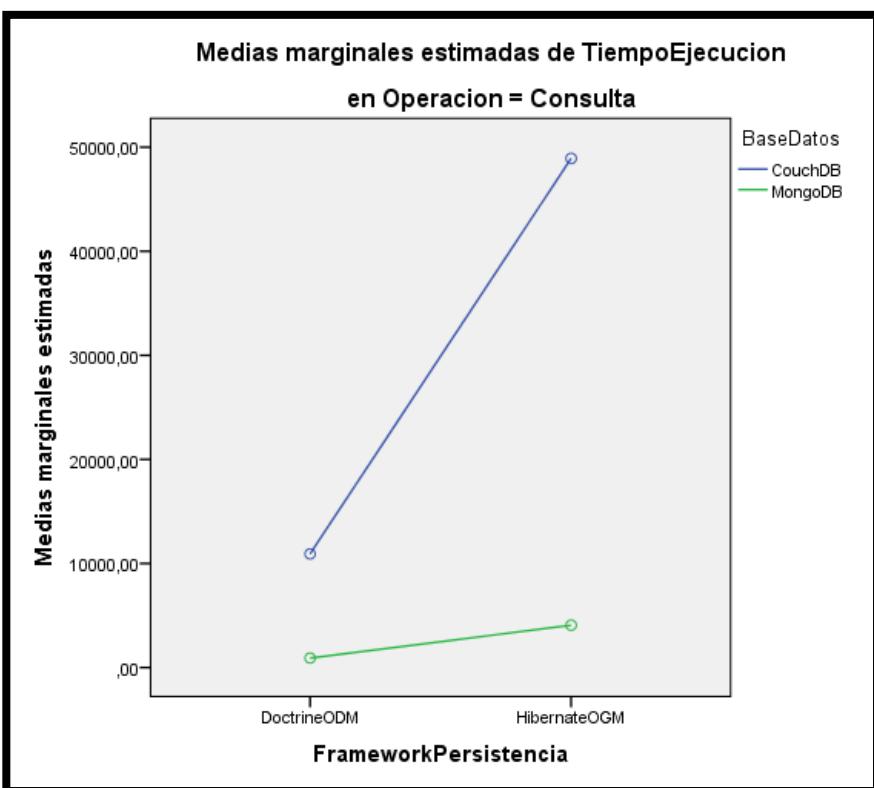


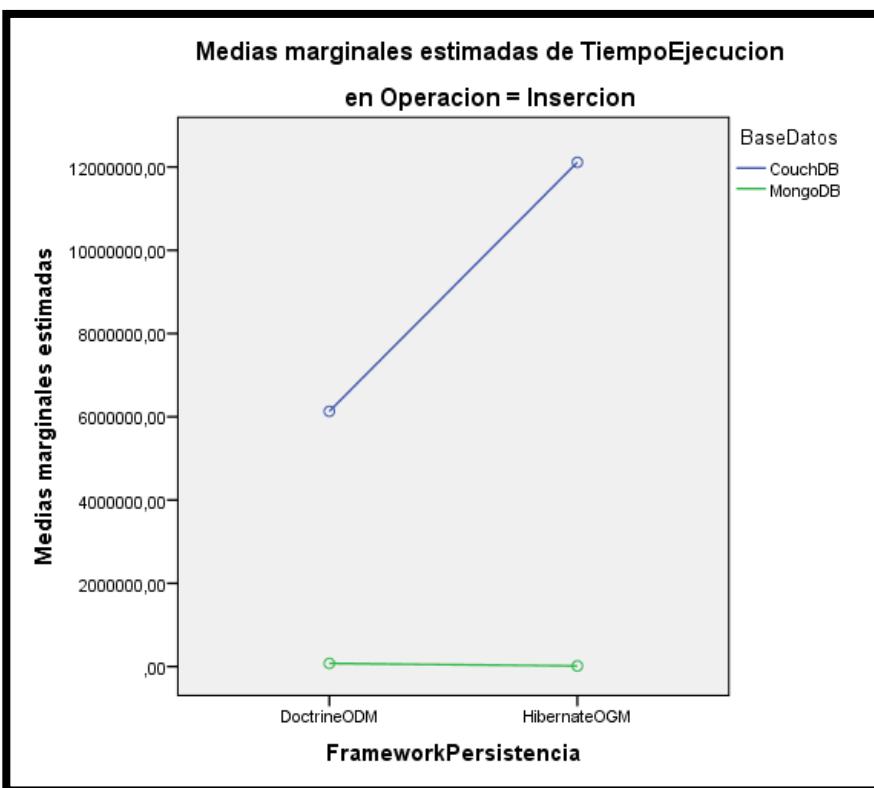
Gráfico 19. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: SGBD NoSQL y cantidad de datos.

Fuente: IBM SPSS Statistics 21



*Gráfico 20. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ODM y SGBD NoSQL para la operación consulta de datos.*

*Fuente: IBM SPSS Statistics 21*



*Gráfico 21. Medias marginales estimadas en tiempo de ejecución de la interacción entre los factores: ODM y SGBD NoSQL para la operación inserción de datos.*

*Fuente: IBM SPSS Statistics 21*