

**ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO**

**FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**

**ESCUELA DE INGENIERÍA EN SISTEMAS**

“ESTUDIO COMPARATIVO DE LA PRODUCTIVIDAD ENTRE LOS FRAMEWORKS DE PERSISTENCIA EN JAVA HIBERNATE Y MYBATIS, APLICADO AL SISTEMA DE EVALUACIÓN DOCENTE DEL IPEC.”

**TESIS DE GRADO**

**Previa a la obtención del título de:**

INGENIERO EN SISTEMAS INFORMÁTICOS

**Presentado por:**

JOSÉ LUIS GAVILANES BLACIO

**RIOBAMBA - ECUADOR**

**2016**

©2016, José Luis Gavilanes Blacio

Se autoriza la reproducción total o parcial, con fines académicos, por cualquier medio o procedimiento, incluyendo la cita bibliográfica del documento, siempre y cuando se reconozca el Derecho de Autor.

**FIRMAS DE RESPONSABILIDAD Y NOTA**

|  |  |  |
| --- | --- | --- |
| **NOMBRE** | **FIRMA** | **FECHA** |
| Ing. Gonzalo Samaniego Erazo Ph.D  **DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA** | ……………………..... | …………………….... |
| Dr. Julio Santillán  **DIRECTOR DE LA ESCUELA DE INGENIERÍA EN SISTEMAS** | ……………………..... | …………………….... |
| Dr. Julio Santillán  **DIRECTOR DE TESIS** | ……………………..... | …………………….... |
| Ing. Jorge Menéndez  **MIEMBRO DEL TRIBUNAL** | ……………………..... | …………………….... |
|  |  |  |
| **NOTA DE LA TESIS** | ……………………..... |  |

Yo, José Luis Gavilanes Blacio soy responsable de las ideas, doctrinas y resultados expuestos en esta Tesis y el patrimonio intelectual de la Tesis de Grado pertenece a la Escuela Superior Politécnica De Chimborazo

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

JOSÉ LUIS GAVILANES BLACIO

**AGRADECIMIENTO**

Agradezco a Dios por no permitirme desfallecer y siempre darme fuerzas necesarias para cumplir esta meta de mi vida; a mis padres por su sacrificio y constancia a lo largo de estos años y creer en mí; a mi esposa Vanessa por su apoyo y afecto incondicional; a la familia y amigos por sus palabras de aliento y superación; a los docentes por su tiempo compartido tanto fuera y dentro de las aulas de clase a lo largo de la vida estudiantil; al personal del IPEC por darme la oportunidad de realizar mi estudio y análisis dentro de sus instalaciones

José Luis

**DEDICATORIA**

A mis hijos Dylan y David por ser un fuerte fuente de inspiración y motivación para ser mejor cada día; a mi padres, hermanos, docentes y compañeros estudiantes que fueron parte a lo largo de esta vida estudiantil; a mi esposa Vanessa por apoyarme día a día a pesar de las dificultades y retos que se pusieron en el camino; a todos aquellos que con su apoyo incondicional me ayudaron a seguir adelante, haciendo de mí una persona de bien

**INDICE DE ABREVIATURAS**

**AJAX** (Asynchronous JavaScript And XML), JavaScript asíncrono y XML.

**API** (Application Programming Interface), Interfaz de programación de aplicaciones.

**CES** Consejo Educación Superior

**CORBA** (Common Object Broker Architecture), Arquitectura estándar de objetos gestionados en red.

**CSS** (Cascade Style Sheets), Hojas de Estilo en Cascada.

**DAO** (Data Access Object), Objeto de Acceso a Datos.

**DESITEL** Departamento de Sistemas y Telemática.

**DOM** (Document Object Model), Modelo en Objetos para la Presentación de Documentos.

**ESPOCH** Escuela Superior Politécnica de Chimborazo.

**FTP** (File Transfer Protocol), Protocolo de transferencia de archivos.

**HTML** (HyperText Markup Language), Lenguaje de marcado de hipertexto.

**HTTP** (Hypertext Transfer Protocol), Protocolo de transferencia de hipertexto.

**IDE** (Integrated Development Environment), Entorno integrado de desarrollo.

**IPEC** Instituto de Posgrado y Educación Continua

**IU** Interfaz de Usuario.

**JCP** (Java Community Process), Proceso de la Comunidad Java.

**JDK** (Java Development Kit), Herramientas de desarrollo para Java.

**JEE** (Java Plataform, Enterprise Edition), Plataforma Java, Edición Empresa.

**JDBC** (Java Database Connectivity), Conectividad de Base de Datos Java.

**JNDI** (Java Naming and Directory Interface), Interfaz de Nombrado y Directorio Java.

**JPA** Java Persistence API.

**JSR** (Java Specification Request), Peticiones de Especificación para Java.

**JTA** (Java Transaction API), API para transacciones en Java.

**JVM** (Java Virtual Machine), Máquina Virtual Java.

**MSF** Microsoft Solutions Framework.

**MVC** (Model View Controler), Modelo Vista Controlador.

**ORM** (Object-Relational Mapping), Mapeo Relacional de Objetos.

**POJO** (Plain Old Java Object), Objeto Java Plano a la Antigua.

**PPR** Renderizado Parcial de Página.

**RIA** (Rich Internet Aplicattions), Aplicaciones de Internet enriquecidas.

**RMI** (Java Remote Method Invocation), Invocación remota de métodos en Java.

**RRHH** Recursos Humanos.

**SO** Sistema Operativo.

**SRS** Especificación de Requerimientos de Software.

**SSL** (Secure Sockets Layer), Capa de conexión segura.

**SQL** (Structured Query Language), Lenguaje de consulta estructurado.

**TCP/IP** (Transmission Control Protocol/Internet Protocol), Protocolo de Control de Transmisiones/Protocolo de Internet.

**UIX** User Interface and Executive.

**WEB** (World Wide Web), Red informática mundial.

**XML** (eXtensible Markup Language), Lenguaje de marcas extensible.

**XUL** (XML-based User-interface Language), Lenguaje basado en XML para la interfaz de usuario

INDICE DE CONTENIDO

INDICE DE ABREVIATURAS

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

CAPITULO I

[**1.** **MARCO REFERENCIAL** 19](#_Toc443420109)

[**1.1.** **ANTECEDENTES** 19](#_Toc443420110)

[**1.2.** **JUSTIFICACIÓN** 21](#_Toc443420111)

[**1.2.1.** **TEÓRICA** 21](#_Toc443420112)

[**1.2.2.** **METODOLÓGICA** 22](#_Toc443420113)

[**1.2.3.** **PRÁCTICA** 23](#_Toc443420114)

[**1.2.4.** **ALINEACIÓN** 24](#_Toc443420115)

[**1.3.** **OBJETIVOS** 25](#_Toc443420116)

[**1.3.1.** **OBJETIVO GENERAL** 25](#_Toc443420117)

[**1.3.2.** **OBJETIVOS ESPECIFICOS** 25](#_Toc443420118)

[**1.4.** **MARCO HIPÓTETICO** 25](#_Toc443420119)

[**1.4.1.** **HIPOTESIS** 25](#_Toc443420120)

[**1.5.** **MÉTODOS Y TÉCNICAS** 26](#_Toc443420121)

[**1.5.1.** **MÉTODOS** 26](#_Toc443420122)

[**1.5.2.** **TÉCNICAS** 26](#_Toc443420123)

[**2.** **PERSISTENCIA** 28](#_Toc443420124)

[**2.1.** **Introducción** 28](#_Toc443420125)

[**2.2.** **Bases de Datos Relacionales** 28](#_Toc443420126)

[**2.2.1.** **Definición** 28](#_Toc443420127)

[**2.2.2.** **Historia** 28](#_Toc443420128)

[**2.2.3.** **Sistema Gestor de Base de Datos (SGBD, en ingles DBMS: Database Management System).** 29](#_Toc443420129)

[**2.2.4.** **Elementos de las Bases de Datos relacionales** 30](#_Toc443420130)

[**2.2.5.** **Normalización del esquema relacional** 30](#_Toc443420131)

[**2.2.6.** **Ventajas** 31](#_Toc443420132)

[**2.2.7.** **Desventajas** 31](#_Toc443420133)

[**2.3.** **Persistencia de objetos** 31](#_Toc443420134)

[**2.4.** **Métodos de persistencia** 32](#_Toc443420135)

[**2.4.1.** **Serialización** 32](#_Toc443420136)

[**2.4.2.** **Bases de datos orientadas a objetos (ODBMS)** 33](#_Toc443420137)

[**2.4.3.** **Bases de datos relacionales** 33](#_Toc443420138)

[**2.5.** **Capa de Persistencia** 33](#_Toc443420139)

[**2.5.1.** **Requerimientos de la Capa de Persistencia** 34](#_Toc443420140)

[**2.6.** **Mapeo objeto relacional (ORM)** 35](#_Toc443420141)

[**2.6.1.** **Componentes de un ORM** 36](#_Toc443420142)

[**2.6.2.** **Diferentes técnicas de mapeo** 36](#_Toc443420143)

[**2.6.2.1.** **Mapeo directo sin identidad**. 37](#_Toc443420144)

[**2.6.2.2.** **Mapeo directo con identidad**. 37](#_Toc443420145)

[**2.6.2.3.** **Generación del mapeo a partir de XML**. 37](#_Toc443420146)

[**2.6.3.** **Mapeo de asociaciones** 37](#_Toc443420147)

[**2.6.4.** **Maneras de usar los ORM** 38](#_Toc443420148)

[**2.6.4.1.** **Relacional puro.** 38](#_Toc443420149)

[**2.6.4.2.** **Mapeo objeto ligero**. 38](#_Toc443420150)

[**2.6.4.3.** **Mapeo objeto medio**. 38](#_Toc443420151)

[**2.6.4.4.** **Mapeo objeto completo**. 39](#_Toc443420152)

[**2.6.5.** **Herramientas ORM** 39](#_Toc443420153)

[**2.6.6.** **Ventajas** 40](#_Toc443420154)

[**2.6.7.** **Desventajas** 40](#_Toc443420155)

[**2.7.** **Java Database Conectivity (JDBC)** 40](#_Toc443420156)

[**2.7.1.** **Características de JDBC** 41](#_Toc443420157)

[**2.7.2.** **JDBC como componente necesario para la persistencia** 42](#_Toc443420158)

[**2.8.** **¿Qué es un Framework?** 42](#_Toc443420159)

[**2.8.1.** **Arquitectura del Framework** 43](#_Toc443420160)

[**2.8.2.** **Tipos de framework** 44](#_Toc443420161)

[**2.8.3.** **Importancia de un Framework** 44](#_Toc443420162)

[**2.8.4.** **Ventajas** 45](#_Toc443420163)

[**2.8.5.** **Desventajas** 45](#_Toc443420164)

[**2.9.** **Framework Hibernate** 46](#_Toc443420165)

[**2.9.1.** **Introducción** 46](#_Toc443420166)

[**2.9.2.** **Historia** 46](#_Toc443420167)

[**2.9.3.** **Arquitectura de Hibernate** 46](#_Toc443420168)

[**2.9.3.1.** **Arquitectura Mínima** 47](#_Toc443420169)

[**2.9.3.2.** **Arquitectura Completa** 48](#_Toc443420170)

[**2.9.3.3.** **Ciclo de Vida de los Objetos** 48](#_Toc443420171)

[**2.9.3.4.** **APIs básicas** 50](#_Toc443420172)

[**2.9.4.** **Archivo de Configuración** 50](#_Toc443420173)

[**2.9.5.** **Archivos de Mapeo** 51](#_Toc443420174)

[**2.9.6.** **Mapeo Básico de O/R** 51](#_Toc443420175)

[**2.9.6.1.** **Mapeo de Hibernate** 51](#_Toc443420176)

[**2.9.6.2.** **Entidades y Valores** 52](#_Toc443420177)

[**2.9.6.3.** **Mapeo de una clase más de una vez** 53](#_Toc443420178)

[**2.9.6.4.** **Objetos de bases de datos auxiliares** 53](#_Toc443420179)

[**2.9.7.** **HQL( Hibernate Query Languaje)** 53](#_Toc443420180)

[**2.9.7.1.** **Sensibilidad a las letras mayúsculas** 53](#_Toc443420181)

[**2.9.7.2.** **Cláusula SELECT** 53](#_Toc443420182)

[**2.9.7.3.** **Cláusula WHERE** 54](#_Toc443420183)

[**2.9.7.4.** **Clausula FROM** 54](#_Toc443420184)

[**2.9.7.5.** **Ejecución de Consultas** 54](#_Toc443420185)

[**2.9.8.** **Ventajas** 55](#_Toc443420186)

[**2.9.9.** **Desventajas** 55](#_Toc443420187)

[**2.10.** **Framework MyBatis** 55](#_Toc443420188)

[**2.11.** **Introducción** 55](#_Toc443420189)

[**2.12.** **Historia** 55](#_Toc443420190)

[**2.13.** **Arquitectura** 56](#_Toc443420191)

[**2.14.** **Ámbito y Ciclo de Vida** 56](#_Toc443420192)

[**2.14.1.** **SqlSessionFactoryBuilder** 56](#_Toc443420193)

[**2.14.2.** **SqlSessionFactory** 57](#_Toc443420194)

[**2.14.2.1.** **SqlSessionFactory sin creación de archivo XML** 57](#_Toc443420195)

[**2.14.2.2.** **SqlSessionFactory creado a partir de un archivo XML** 57](#_Toc443420196)

[**2.14.3.** **SqlSession** 58](#_Toc443420197)

[**2.14.4.** **Instancias de mapper** 58](#_Toc443420198)

[**2.14.5.** **Funcionamiento de los Mapped Statament** 58](#_Toc443420199)

[**2.14.5.1.** **Namespace:** 58](#_Toc443420200)

[**2.14.5.2.** **Resolución de nombres** 59](#_Toc443420201)

[**2.15.** **Ficheros de mapeo XML** 59](#_Toc443420202)

[**2.15.1.** **Select** 59](#_Toc443420203)

[**2.15.2.** **Insert, Update y Delete** 60](#_Toc443420204)

[**2.15.3.** **Result Maps** 60](#_Toc443420205)

[**2.15.4.** **Asociaciones** 61](#_Toc443420206)

[**2.15.5.** **Colecciones** 61](#_Toc443420207)

[**2.15.6.** **Discriminadores** 62](#_Toc443420208)

[**2.15.7.** **Auto Mapeo** 62](#_Toc443420209)

[**2.15.8.** **Cache** 63](#_Toc443420210)

[**2.16.** **SQL dinámico** 63](#_Toc443420211)

[**2.17.** **Logging** 63](#_Toc443420212)

[**2.18.** **Ventajas** 63](#_Toc443420213)

[**2.19.** **Desventajas** 64](#_Toc443420214)

[**CAPITULO III** 65](#_Toc443420215)

[**3.** **ESTUDIO COMPARATIVO DE LOS FRAMEWORKS DE PERSISTENCIA HIBERNATE Y MYBATIS** 65](#_Toc443420216)

[**3.1.** **Introducción** 65](#_Toc443420217)

[**3.2.** **Definición de los parámetros de comparación** 66](#_Toc443420218)

[**3.2.1.** **Variable Independiente** 66](#_Toc443420219)

[**3.2.2.** **Variable Dependiente** 67](#_Toc443420220)

[**3.3.** **Criterios de Evaluación** 69](#_Toc443420221)

[**3.4.** **Estudio comparativo de los Frameworks de Persistencia Hibernate y Mybatis** 70](#_Toc443420222)

[**3.4.1.** **Mapeo Objeto / Relacional** 70](#_Toc443420223)

[**3.4.1.1.** **Construcción de métodos CRUD** 70](#_Toc443420224)

[**3.4.1.2.** **Creación automática de esquema** 71](#_Toc443420225)

[**3.4.1.3.** **Dialectos utilizados por la herramienta** 72](#_Toc443420226)

[**3.4.2.** **Producto** 75](#_Toc443420227)

[**3.4.2.1.** **Ultimo versionamiento** 75](#_Toc443420228)

[**3.4.2.2.** **Modo de Licenciamiento** 77](#_Toc443420229)

[**3.4.2.3.** **Disponibilidad de Información.** 78](#_Toc443420230)

[**3.4.3.** **Rendimiento** 80](#_Toc443420231)

[**3.4.3.1.** **Tiempo de respuesta de consultas SQL.** 81](#_Toc443420232)

[**3.4.3.2.** **Uso del CPU.** 81](#_Toc443420233)

[**3.4.3.3.** **Uso de la Memoria RAM.** 82](#_Toc443420234)

[**3.4.4.** **Desarrollo.** 83](#_Toc443420235)

[**3.4.4.1.** **Líneas de código generadas** 84](#_Toc443420236)

[**3.4.4.2.** **Tiempo desarrollo operaciones CRUD** 84](#_Toc443420237)

[**3.4.4.3.** **Integración con otros frameworks** 85](#_Toc443420238)

[**3.5.** **Comprobación de la Hipótesis** 88](#_Toc443420239)

[**3.5.1.** **Planteamiento de la hipótesis** 88](#_Toc443420240)

[**3.5.2.** **Establecimiento del nivel de significancia.** 89](#_Toc443420241)

[**3.5.3.** **Calculo del estadístico.** 89](#_Toc443420242)

[**3.5.4.** **Grados de libertad** 90](#_Toc443420243)

[**3.5.5.** **Criterio de decisión.** 90](#_Toc443420244)

[**CAPITULO IV** 91](#_Toc443420245)

[**4.** **IMPLEMENTACION DEL SISTEMA DE EVALUACION AL DOCENTE EN EL IPEC** 91](#_Toc443420246)

[**4.1.** **Introducción.** 91](#_Toc443420247)

[**4.2.** **Fase I: Visión y Alcances** 91](#_Toc443420248)

[**4.2.1.** **Definición del Problema** 91](#_Toc443420249)

[**4.2.2.** **Visión del Proyecto** 92](#_Toc443420250)

[**4.2.3.** **Metas** 92](#_Toc443420251)

[**4.2.4.** **Perfiles de Usuario** 92](#_Toc443420252)

[**4.2.5.** **Ámbito del Proyecto** 93](#_Toc443420253)

[**4.2.6.** **Concepto de la Solución** 93](#_Toc443420254)

[**4.2.6.1.** **Software a utilizar** 93](#_Toc443420255)

[**4.2.6.2.** **Arquitectura** 94](#_Toc443420256)

[**4.2.7.** **Objetivos del Proyecto** 95](#_Toc443420257)

[**4.2.7.1.** **Objetivos del Negocio** 95](#_Toc443420258)

[**4.2.7.2.** **Objetivos de Diseño** 95](#_Toc443420259)

[**4.2.8.** **Factores críticos** 95](#_Toc443420260)

[**4.2.9.** **Planificación Inicial** 96](#_Toc443420261)

[**4.2.9.1.** **Equipo de trabajo** 96](#_Toc443420262)

[**4.2.9.2.** **Perfiles de usuario** 96](#_Toc443420263)

[**PERFIL** 96](#_Toc443420264)

[**TIPO DE ACCESO** 96](#_Toc443420265)

[**DESCRIPCION** 96](#_Toc443420266)

[**4.2.9.3.** **Cronograma de trabajo.** 97](#_Toc443420267)

[**4.3.** **FASE II. Planificación** 97](#_Toc443420268)

[**4.3.1.** **Especificación Funcional** 97](#_Toc443420269)

[**4.3.1.1.** **Diseño Conceptual** 97](#_Toc443420270)

[**4.3.1.2.** **Diseño Lógico** 102](#_Toc443420271)

[**4.3.1.2.1.** **Tecnología a usar en el proyecto** 103](#_Toc443420272)

[**4.3.1.2.2.** **Diagramas de Secuencia** 103](#_Toc443420273)

[**4.3.1.2.3.** **Diagramas de Clase** 105](#_Toc443420274)

[**4.3.1.2.4.** **Modelo Lógico de la Base de datos** 105](#_Toc443420275)

[**4.3.1.2.5.** **Diseño de Interfaces de Usuario** 106](#_Toc443420276)

[**4.3.1.3.** **Diseño Físico** 108](#_Toc443420277)

[**4.3.1.3.1.** **Diagrama de Actividades** 108](#_Toc443420278)

[**4.3.1.3.2.** **Diagrama de Componentes** 108](#_Toc443420279)

[**4.3.1.4.** **Modelo físico de la Base de Datos** 111](#_Toc443420280)

[**4.4.** **FASE III. Desarrollo** 111](#_Toc443420281)

[**4.4.1.** **Nomenclaturas y Estándares** 111](#_Toc443420282)

[**4.4.2.** **Capa de Presentación** 112](#_Toc443420283)

[**4.4.2.1.** **Implementación de interfaces de usuario con XTMHL y PrimeFaces**: 113](#_Toc443420284)

[**4.4.2.2.** **Implementación de Beans** 114](#_Toc443420285)

[**4.4.3.** **Capa lógica de Negocios** 114](#_Toc443420286)

[**4.4.4.** **Capa de Persistencia** 114](#_Toc443420287)

[**4.4.5.** **Capa de Base de Datos** 115](#_Toc443420288)

[**4.5.** **Fase IV. Estabilización** 115](#_Toc443420289)

[**4.5.1.** **Revisión general del sistema’** 115](#_Toc443420290)

[**4.5.1.1.** **Código fuente** 115](#_Toc443420291)

[**4.5.1.2.** **Script de Base de datos** 116](#_Toc443420292)

[**4.5.1.3.** **Documentación de la instalación** 116](#_Toc443420293)

[**4.5.2.** **Pruebas** 116](#_Toc443420294)

[**4.6.** **FASE V. Fase de Implementación** 117](#_Toc443420295)

[**4.6.1.** **Estrategia de implantación** 117](#_Toc443420296)

[**4.6.2.** **Soporte** 117](#_Toc443420297)

[**4.6.3.** **Coordinación del entrenamiento** 117](#_Toc443420298)

[**4.6.4.** **Transferencias a operaciones** 118](#_Toc443420299)

[**CONCLUSIONES** 119](#_Toc443420300)

[**RECOMENDACIONES** 120](#_Toc443420301)

[**RESUMEN** 121](#_Toc443420302)

[ABSTRACT 122](#_Toc443420303)

[BIBLIOGRAFÍA 123](#_Toc443420304)

**INDICE DE FIGURAS**

[**Figura II. 1 Procesos de Serialización** 32](#_Toc441007913)

[Figura II. 2 Esquema gráficos de la programación N-Capas 34](#_Toc441007914)

[Figura II. 3 Esquema de la abstracción del ORM 35](#_Toc441007915)

[Figura II. 4 Esquema de Acceso del controlador JDBC 41](#_Toc441007916)

[Figura II. 5 Esquema usando frameworks de persistencia 42](#_Toc441007917)

[Figura II. 6 Esquema del Modelo-Vista-Controlador 43](#_Toc441007918)

[Figura II. 7 Arquitectura de Hibernate 47](#_Toc441007919)

[Figura II. 8 Arquitectura de Hibernate mínima 47](#_Toc441007920)

[Figura II. 9 Arquitectura de Hibernate completa 48](#_Toc441007921)

[Figura II. 10 Estados del Ciclo de Vida 49](#_Toc441007922)

[Figura II. 11 Archivo de configuración de Hibernate 51](#_Toc441007923)

[Figura II. 12 Ejemplo del archivo nombre-de-la-clase.hbm.xml 52](#_Toc441007924)

[Figura II. 13 Arquitectura de Mybatis 56](#_Toc441007925)

[Figura II. 14 SqlSessionFactory creado sin XML 57](#_Toc441007926)

[Figura II. 15 SqlSessionFactoryBuilder creado a través de archivo XML 58](#_Toc441007927)

[Figura II. 16 Ejemplo de select en Mybatis 59](#_Toc441007928)

[Figura II. 17 Ejemplo de Insert, Update y Delete en Mybatis 60](#_Toc441007929)

[Figura II. 18 Ejemplo de Asociación en Mybatis 61](#_Toc441007930)

[Figura II. 19 Ejemplo de Colecciones en Mybatis 61](#_Toc441007931)

[Figura II. 20 Ejemplo de Discriminador en Mybatis 62](#_Toc441007932)

[Figura III. 1 Análisis del parámetro Mapeo Objeto/Relacional 75](#_Toc441007933)

[Figura III. 2 Análisis del Parámetro Producto 80](#_Toc441007934)

[Figura III. 3 Análisis del Parámetro Rendimiento 83](#_Toc441007935)

[Figura III. 4 Análisis del parámetro Desarrollo 87](#_Toc441007936)

[Figura III. 5 Resultado final del Análisis de frameworks 88](#_Toc441007937)

**INDICE DE TABLAS**

[Tabla II. I Correspondencia de elementos 36](#_Toc441007962)

[Tabla II. II Comparación de los framework de persistencia 39](#_Toc441007963)

[Tabla III. I Resumen de parámetros de la variable independiente 66](#_Toc441007938)

[Tabla III. II Resumen de parámetros de la variable dependiente 68](#_Toc441007939)

[Tabla III. III Pesos de Valoración Cualitativa y Cuantitativa 69](#_Toc441007940)

[Tabla III. IV Construcción de métodos CRUD. 71](#_Toc441007941)

[Tabla III. V Resultados para la Creación automática de esquemas 72](#_Toc441007942)

[Tabla III. VI Librerías para DBMS soportados por Hibernate 72](#_Toc441007943)

[Tabla III. VII Resultados de la Evaluación de librerías para soporte a DBMS 74](#_Toc441007944)

[Tabla III. VIII Resultados del Parámetro Mapeo Objeto/Relacional 74](#_Toc441007945)

[Tabla III. IX Versionamiento de Hibernate 76](#_Toc441007946)

[Tabla III. X Versionamiento de Mybatis 76](#_Toc441007947)

[Tabla III. XI Resultados de la Evaluación de versionamientos 77](#_Toc441007948)

[Tabla III. XII Resultados del Modo de Licenciamiento 77](#_Toc441007949)

[Tabla III. XIII Resultados de búsqueda en disponibilidad de información 78](#_Toc441007950)

[Tabla III. XIV Análisis del criterio de Disponibilidad de Información 79](#_Toc441007951)

[Tabla III. XV Resultados del parámetro Producto 79](#_Toc441007952)

[Tabla III. XVI Características del equipo para las pruebas del prototipo. 80](#_Toc441007953)

[Tabla III. XVII Resultados del indicador Tiempo de respuestas en consultas SQL 81](#_Toc441007954)

[Tabla III. XVIII Resultados del indicador Uso del CPU 82](#_Toc441007955)

[Tabla III. XIX Resultados del indicador Uso de la memoria RAM 82](#_Toc441007956)

[Tabla III. XX Resultados del Análisis del Parámetro Rendimiento 83](#_Toc441007957)

[Tabla III. XXI Resultados del indicador Líneas de código generadas 84](#_Toc441007958)

[Tabla III. XXII Resultado del indicador Tiempo desarrollo operaciones CRUD 85](#_Toc441007959)

[Tabla III. XXIII Resultado del indicador Integración con otros frameworks 86](#_Toc441007960)

[Tabla III. XXIV Resultados del Análisis del parámetro Desarrollo 86](#_Toc441007961)

**INTRODUCCIÓN**

El Instituto de Postgrado y Educación Continua con el objetivo de alcanzar uno de sus objetivos “Alcanzar la excelencia académica”, ve la necesidad de implementar un sistema informático que le ayude a evaluar a los docentes en el ámbito académico para con ello tener información más cercana a la realidad para la mejor toma de decisiones, es por ello que dada la complejidad en el desarrollo del sistema se busca alternativas de frameworks orientados a mantener los datos siempre disponibles mediante la persistencia de los objetos con ello los desarrolladores se concentran en resolver problemas de la lógica de negocios, dejando aparte los detalles de la programación de acceso a los datos.

En la actualidad existe variedad de ORM que ayudan a mantener la persistencia de los objetos a lo largo de vida de la aplicación, entre estas herramientas denominadas frameworks los cuales nos ayudan simplificando el acceso a los datos de una manera más transparente

El presente trabajo investigativo intenta dar una idea más clara por medio de un estudio comparativo entre los framework Hibernate y Mybatis con el propósito de determinar el más eficiente para el desarrollo de la aplicación Sistema de Evaluación Docente IPEC dentro de las instalaciones de la ESPOCH.

**CAPITULO I**

1. **MARCO REFERENCIAL**
   1. **ANTECEDENTES**

En la actualidad uno de los objetivos de Instituto de Postgrado y Educación Continua (IPEC) es el de "Alcanzar la excelencia académica" (ESPOCH, 2010). motivo por el cual se ha visto en la necesidad de ajustarse a las nuevas leyes y reglamentos del Consejo de Educación Superior (CES) y basándose en la "Acta Resolutiva de la Sesión Ordinaria No.37 del 31 de octubre de 2012 - RPC-SO-037-No.265-2012 - Codificada" (CONSEJO DE EDUCACIÓN SUPERIOR, 2012, p. 35) en su Título IV "Evaluación y Perfeccionamiento del Personal Académico", Capítulo I "De la Evaluación Integral de Desempeño del Personal Académico", donde se especifican los ámbitos, objetos, instrumentos, garantía, actores, componentes y ponderaciones de evaluación, para poder así lograr alcanzar el objetivo antes mencionado.

Para lograr esto se requiere un sistema informático que permita manejar la información de una manera adecuada y eficiente de tal manera que pueda ofrecer resultados oportunos para una mejor toma de decisiones ya que actualmente el IPEC no maneja ningún sistema automatizado de evaluación a docente, por la cual no pudiere cumplir con uno de sus objetivos como es el de "Alcanzar la excelencia académica", ya que al no tener un sistema confiable se pudiera enviar información errónea a los diferentes departamentos e instituciones como es la Dirección de Evaluación y Aseguramiento de la Calidad - ESPOCH y al Consejo de Evaluación, Acreditación y Aseguramiento de la Calidad de la Educación Superior (CEAACES).

Si bien es cierto que actualmente existe un sistema informático de evaluación al docente en Pregrado este maneja distintos instrumentos, actores y parámetros de ponderación.

El IPEC requiere de un sistema informático implementado en la web, ya que se desea que la información esté disponible en todo momento y se accesible desde cualquier parte del mundo.

Para el desarrollo de aplicaciones web se utilizan muchos los framework ya estos ayudan al programador a facilitar algunas de las implementaciones como pudieran ser las interfaces de usuario, lógica de negocios, abstracción en la capa de datos, dando lugar a que se mejore la productividad.

La persistencia de objetos se define como "*la acción de preservar la información de un objeto de forma permanente (guardar), pero a su vez también se refiere a poder recuperar la información del mismo (leer) para que pueda ser nuevamente utilizada*" (Gonzaga, 2013), esto nos ayuda a entender que la mayoría de las aplicaciones, necesitan almacenar y recuperar información lo cual implica alguna forma de interacción con una base de datos relacional. Esto representa un problema fundamental para los desarrolladores porque algunas veces el diseño de datos relacionales y los orientados a objetos comparten estructuras de relaciones muy diferentes dentro de sus respectivos entornos.

Las bases de datos relacionales están estructuradas de una manera tabular y los sistemas informáticos trabajan con objetos que normalmente están relacionados en forma de árbol, lo cual se denomina objeto-relación. Esta diferencia ha llevado a los desarrolladores crear tecnologías de persistencia de objetos para intentar construir un puente entre el mundo relacional y orientado a objetos a través de los frameworks y técnicas de mapeos objeto-relación, que "*es una destreza de programación para convertir datos entre el sistema de tipos utilizados en un lenguaje de programación orientada a objetos y el utilizado en la base de datos relacional* " (Israel, 2013)

Como se mencionaba anteriormente los frameworks ayudan al desarrollador a implementar más fácil y rápidamente en este caso la abstracción de la capa de datos conservando una coherencia que faciliten su evolución y mantenimiento posterior, tal es así el que el desarrollador ya no tiene que ocuparse de la persistencia sobre los datos, pudiendo ocuparse en mantener la perspectiva orientada a objetos en la solución del problema y mejorar su productividad.

Se ha propuesto usar para el almacenamiento de la información PostgreSQL[[1]](#footnote-1), esto ya que el DTIC se encuentra en una fase de migración de datos en su infraestructura de software ajustándose a los nuevos lineamientos de la institución.

Tras haber hecho una investigación sobre temas similares se ha encontrado varios como:

1. Análisis y Uso de Frameworks de Persistencia en Java ( Payá, 2006)
2. Persistencia de Objetos JDO, Soluciones Java (Mármol, 2004)
3. Análisis Hibernate como Tecnología de Persistencia de Objetos sobre Bases de Datos Relacionales en Aplicaciones Empresariales, (Cando, 2013)

Así como también en el repositorio de datos de la ESPOCH, sobre investigaciones similares, llegándose a concluir de que no existen estudios de comparación para Hibernate y Mybatis como frameworks de persistencia en java.

El no tener una investigación de este tipo hace suponer que los programadores la mayoría de las veces realizan mapeos personalizados de las bases de datos relacionales, lo cual les llevaría mucho tiempo y esfuerzo disminuyendo su productividad al realizar la abstracción de la capa de datos.

* 1. **JUSTIFICACIÓN**
     1. **TEÓRICA**

A medida que los frameworks, evolucionan, se requiere obtener mejores ventajas y beneficios en el entorno de programación y de esa manera poder disminuir el tiempo de desarrollo y facilidad para la construcción de un sistema informático, lo cual ayuda que la productividad del programador sea más eficiente

La clasificación de frameworks mediante un estudio y análisis con diversas técnicas es vital para una selección correcta de una herramienta que permita asegurar un desarrollo adecuado a nuestras necesidades ya que de lo contrario nuestro sistema tendrá desarrollos muy lentos y una calidad muy pobre.

Las principales ventajas de la utilización de un framework son:

* El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel.
* La reutilización de componentes software al por mayor. Los frameworks son los paradigmas de la reutilización.
* Un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que resulta en bibliotecas más fáciles de aprender a usar.

En esta investigación se ha elegido los frameworks de persistencia Hibernate y Mybatis ambos para java ya que los mismo se basan en software libre y pueden ser usados en servidores Glassfish o Tomcat, IDE como Netbeans o Eclipse, por consecuente no se tendría que pagar por licenciamiento semestral, anual o por volumen, pudiendo así la Dirección de Tecnologías de la Información y Comunicación (DTIC) de la ESPOCH disminuir costos en el software.

El presente trabajo de investigación evaluará Hibernate[[2]](#footnote-2) y Mybatis[[3]](#footnote-3) (antiguamente llamado Ibatis SQL Maps) con el objetivo de determinar el más eficiente para abstraer la capa de datos, ambos frameworks de persistencia en java más usados por los programadores actualmente [8].

Esta investigación nos aportará resultados que ayudarán a los programadores a elegir un framework adecuado a sus necesidades, evitando así que disminuyan la productividad cuando se trabaja con la capa de datos de un sistema informático.

* + 1. **METODOLÓGICA**

El sistema informático propuesto se pretende usar dos frameworks de persistencia en java, como es Hibernate y Mybatis, mismo que se probaran en la base de datos relacional del sistema a desarrollarse, con lo cual nos permitirá saber cuál de ellos se adapta mejor a nuestras necesidades y así mejorar la productividad.

La metodología de desarrollo Microsoft Solution Framework (MSF) es la que se usará para la creación del sistema de Evaluación Docente, esta metodología nos ayuda a centrarnos en:

* Alinear objetivos empresariales y tecnológicos
* Establecer objetivos, roles y responsabilidades claros para el proyecto
* Implementar un proceso iterativo, basado en hitos/puntos de control
* Administrar riesgos de forma proactiva
* Respuestas efectivas a los cambios

Las variables tentativas que se estudiará para analizar la productividad serian:

Simplicidad: En el desarrollo de muchas aplicaciones, el tiempo es un obstáculo importante.

Desarrollo: líneas de código, configuración del framework, uso de componentes.

Producto: documentación, modo de licenciamiento, curva de aprendizaje, madurez

Mapeo de Objetos: DBMS soportados, lenguaje de consulta, creación automática del esquema.

* + 1. **PRÁCTICA**

El desarrollo de la parte aplicativa nace de las necesidades que tiene el IPEC de alcanzar su objetivo antes mencionado, para la cual el sistema informático de Evaluación a los docentes del IPEC servirá para las acreditaciones de módulos (maestrías) y escalafones de los profesores e investigadores, generando información que se puede almacenar, la misma que se usará para una mejor toma de decisiones.

Con ello se pretende mejorar los tiempos de respuesta en la entrega de información a las diferentes autoridades o departamentos, al ser una aplicación implementada en la web facilita el acceso en cualquier momento y lugar, logrando con ello mejorar la entrega de resultados en el menor tiempo posible.

Los módulos que se crearán para la solución:

* Módulo de Administración: creación de usuarios, ingreso, modificaciones, eliminaciones de los diferentes datos del sistema
* Módulo de Evaluación Docente: llenado de la encuesta para la evaluación docente por parte de los estudiantes, directivos y pares académicos, autoevaluación del Docente.
* Módulo de Reportes: emisión de certificados, resultados de la evaluación a partir de los diferentes indicadores propuestos en los estándares del documento del Consejo de Educación Superior - CES

El diseño de la aplicación contemplara una interfaz para el ingreso de Estudiantes y Docentes de IPEC - ESPOCH con sus respectivas opciones de menús de navegación, se debe integrar con la base de datos del Sistema Académico Administrativo del IPEC (SAAI) para consultar los datos personales de docentes, estudiantes y personal administrativo, todo con la finalidad de obtener la información lo más verdadera posible y evitar las ambigüedades.

* + 1. **ALINEACIÓN**

El presente anteproyecto de Tesis está enmarcado

ESPOCH

Línea: “*Tecnología de la Información comunicación y procesos industriales*”.

Programa: “*Para el desarrollo de aplicaciones de software para procesos de gestión y administración pública y privada*” (Escuela Superior Politécnica de Chimborazo, 2012)

Plan Nacional del Buen Vivir (PNBV)

Objetivo 4.5: “*Potenciar el rol de docentes y otros profesionales de la educación como actores clave en la construcción del Buen Vivir*.”

Literal (h): “*Fortalecer los mecanismos de calificación, evaluación y recategorización continua de los docentes y otros profesionales de la educación, en función de su desempeño*”. (SENPLADES, 2013)

* 1. **OBJETIVOS**
     1. **OBJETIVO GENERAL**

Realizar un Estudio comparativo de la productividad entre los Frameworks de persistencia en java Hibernate y Mybatis, Aplicado al sistema de evaluación docente del IPEC.

* + 1. **OBJETIVOS ESPECIFICOS**
    - Estudiar los frameworks de persistencia en java, Hibernate y Mybatis.
    - Definir los parámetros de productividad y herramientas para la evaluación.
    - Comparar los framework de persistencia Hibernate y Mybatis, con la finalidad de elegir el que mejor se adapte a nuestro sistema informático a realizar.
    - Desarrollar el sistema de evaluación de la docencia en el IPEC, empleando el framework seleccionado, una vez determinada la mejor tecnología.
  1. **MARCO HIPÓTETICO**
     1. **HIPOTESIS**

La utilización del framework de persistencia en java Hibernate me permitirá mejorar la productividad que se destina a la abstracción de la capa de datos con respecto al framework Mybatis

**Tipo de Hipótesis:** Hipótesis descriptiva.

* 1. **MÉTODOS Y TÉCNICAS**
     1. **MÉTODOS**

El método a utilizar para la presente investigación experimental es el método científico; en vista que se adapta  a las diferentes variables para la realización de la investigación como son:

* Delimitación y definición del problema motivo del presente trabajo.
* Elaborar los diseños de los diferentes módulos planteados.
* Realizar pruebas a los diferentes módulos planteados.
* Análisis e interpretación de Resultados.
* Proceso de comprobación de la hipótesis.

Para complementar la investigación se aplicará el método deductivo ya que parte de verdades previamente establecidas como principios generales, para luego aplicarlo a casos individuales y comprobar así su validez en el  desarrollo de la aplicación móvil para el Sistema Académico Institucional de la ESPOCH.

La metodología de desarrollo Microsoft Solution Framework (MSF) es la que se usará para la creación del sistema de Evaluación Docente del IPEC, esta metodología nos ayuda a centrarnos en:

* Alinear objetivos empresariales y tecnológicos
* Establecer objetivos, roles y responsabilidades claros para el proyecto
* Implementar un proceso iterativo, basado en hitos/puntos de control
* Administrar riesgos de forma proactiva
* Respuestas efectivas a los cambios

Estos puntos nos ayudan a entregar una solución de calidad. Cada miembro del equipo debe entender y aplicar estos principios en sus interacciones con otros miembros del equipo, con su organización y con las partes interesadas.

* + 1. **TÉCNICAS**

Para la recopilación de la información necesaria que sustente el presente trabajo de investigación, se ha establecido como técnicas las siguientes:

***Técnicas de recopilación de información***

* **Entrevista:**Se la realizara al Ing., William Pilco, Director del IPEC y al Ing. Diego Caisaguano encargado del área de tecnologías del IPEC, para la recopilación de la información a fin de conocer las principales necesidades, características y/o dificultades que se desean resolver.
* **Investigación bibliográfica:** Para lo que tiene que ver a fuentes de información se utilizarán principalmente el internet, libros, revistas, etc.
* **Observación:** En los lugares donde ya mantienen un sistema de similares características al que se plantean en este documento.
* **Pruebas:** En los modulo planteados.

**CAPITULO II**

**MARCO TEÓRICO**

1. **PERSISTENCIA**
   1. **Introducción**

Este capítulo pretender abordar conceptos y términos fundamentales sobre la persistencia, alternativas de solución común para poder ser aplicados a las base de datos relacionales ya que es muy bien sabido que en el desarrollo de una nueva aplicación y en especial las orientada a objetos lo primero a resolver es la integración que tendrán los datos al ser almacenados y recuperados por la aplicación.

* 1. **Bases de Datos Relacionales**
     1. **Definición**

Es un modelo de datos utilizado para modelar problemas reales y administrar datos dinámicamente, su idea fundamental son las relaciones que se dan entre las entidades del diagrama como conjuntos de datos llamados tuplas, la mayoría de veces se conceptualiza de una manera fácil de imaginar, dándose cuenta que cada relación fuese una tabla compuesta por registros representando las tuplas y campos. (Córdova Espinoza & Cuzco Sarango, 2013, pág. 6).

* + 1. **Historia**

Entre los años 60 y 70, Edgar Frank Cood, definió las bases del modelo relacional que con los años se convirtió en modelo de bases más popular, las bases de Cood se definían básicamente los aspectos de independencia física y lógica, flexibilidad, uniformidad y sencillez.

En las bases de Cood se definían los objetivos de este modelo: 

* Independencia física. La forma de almacenar los datos, no debe influir en su manipulación lógica 
* Independencia lógica. Las aplicaciones que utilizan la base de datos no deben ser modificadas por que se modifiquen elementos de la base de datos. 
* Flexibilidad. La base de datos ofrece fácilmente distintas vistas en función de los usuarios y aplicaciones. 
* Uniformidad. Las estructuras lógicas siempre tienen una única forma conceptual (las tablas).
* Sencillez.

En 1978, IBM desarrolla el lenguaje QBE. Que aproximaba la idea relacional a sus ficheros VSAM. En 1979 Oracle se convierte en el primer producto comercial DBMS relacional (RDBMS). En 1980 aparece Ingres que utilizaba el lenguaje Quel que implementaba el cálculo relacional (Sanchéz, 2004)

* + 1. **Sistema Gestor de Base de Datos (SGBD, en ingles DBMS: Database Management System).**

Es una agrupación de programas que sirven para definir, construir y manipular una base de datos, en el que se especificará los tipos de datos, estructuras y restricciones para los datos que se almacenarán e incluye funciones como consulta, actualización, etc. Estos sistemas existen tanto como software propietario y código abierto. (Alegsa, 2010)

Los SGBD están clasificados en

* Relacional (SGBDR): representa a la base de datos como una colección de tablas. Estas bases de datos suelen utilizar SQL como lenguaje de consultas de alto nivel.
* Orientado a objetos: define a la base de datos en términos de objetos, sus propiedades y sus operaciones. Todos los objetos que tienen la misma estructura y comportamiento pertenecen a una clase y las clases de organizan en jerarquías.
* Objeto-relacional o relacional extendido: son los sistemas relacionales con características de los orientados a objetos.
* Jerárquico: representa los datos como estructuras jerárquicas de árbol.
  + 1. **Elementos de las Bases de Datos relacionales**

Las bases de datos relacionales se basan en el uso de tablas (también se las llama relaciones). Las tablas se representan gráficamente como una estructura rectangular formada por filas y columnas (atributos). Cada columna almacena información sobre una propiedad determinada de la tabla donde cada fila (tupla) posee una ocurrencia o relación representada por la tabla. Los elementos de una tabla se detallan a continuación:

* Tupla. Cada fila de la tabla (cada ejemplar que la tabla representa) 
* Atributo. Cada columna de la tabla.
* Campo: donde se van guardando cada uno de los datos necesarios de la tabla
* Cardinalidad. Número de tuplas de una tabla, donde una de las columnas (atributo) hace de clave principal para que esta puede ser referenciada o encontrada desde otras tablas. 
  + 1. **Normalización del esquema relacional**

Una vez obtenido el esquema relacional resultante del modelo entidad relación que representaba la base de datos, normalmente se tendrá una base de datos aceptable para la implementación de la misma. Pero otras veces, debido a fallos en el diseño conceptual o a problemas indetectables en esta fase del diseño, tendremos un esquema que nos puede generar los siguientes problemas: 

* Redundancia. Se llama así a los datos que se repiten continua e innecesariamente por las tablas de las bases de datos. 
* Ambigüedades. Datos que no clarifican suficientemente el registro al que representan.
* Pérdida de restricciones de integridad. 
* Anomalías en operaciones de modificación de datos. El hecho de que al insertar un solo elemento haya que repetir tuplas en una tabla para solventar el inconveniente de unos pocos datos o al momento de eliminar un elemento suponga eliminar varias tuplas.

Para poder solventar estos tipos de inconvenientes no detectados en la fase de diseño, se aplica la técnica de normalización para depurar el modelo.

* + 1. **Ventajas**
* Flexibilidad de adaptación a cada problema, según los requisitos cambien en el futuro inmediato.
* Independencia física y lógica de los datos
* Control de la integridad de los datos.
* Garantía sobre la consistencia de la información.
* Facilidad de acceso concurrente, puesto que está disponible la información para varios usuarios constantemente.
* Protección ante fallos del sistema, sea por factores externos o problemas de hardware
* Seguridad ante accesos no autorizados
  + 1. **Desventajas**
* La integración de los datos es muy deficiente cuando se los desea utilizar en simulación graficas o sistemas de información geográfica.
* No se puede manipulas adecuadamente los bloques de texto grandes como un tipo de dato dentro de los aplicativos.
* El costo de mantenimiento que implica tener la información respalda, segura y disponible en todo momento dentro de la organización.
  1. **Persistencia de objetos**

Se puede encontrar en la web muchas definiciones de persistencia de objetos, según los diferentes autores. Leamos dos de ellas para poder entender el concepto de persistencia de objetos. La primera dice: " *La persistencia es un atributo de los datos que asegura que estarán disponibles incluso más allá de la vida de una aplicación*" (Gonzaga, 2013), otra definición sugiere: " *Se entiende por persistencia la acción de preservar la información de un objeto de forma permanente (guardar), pero a su vez también se refiere a poder recuperar la información del mismo (leer) para que pueda ser nuevamente utilizada*" (Gonzalez, 2011).

Concluyendo se puede decir que la persistencia de objetos es la capacidad que tendrían los objetos de sobrevivir o estar disponibles más allá del tiempo de la aplicación que los creo, pudiendo así poder guardar y recuperar información para ser nuevamente utilizada por otra aplicación.

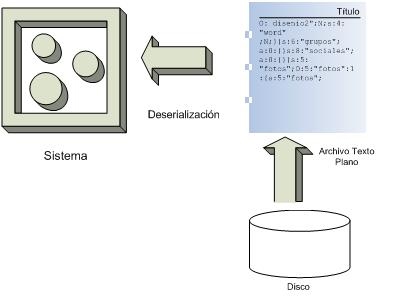
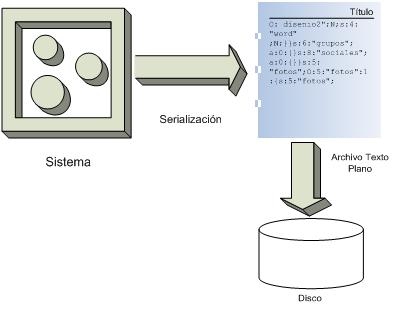
* 1. **Métodos de persistencia**

Se identifica tres formas usuales de poder persistir los objetos:

* Serialización.
* Bases de datos Orientadas a Objetos (ODBMS).
* Bases de datos relacionales.
  + 1. **Serialización**

La “serialización” es el proceso de convertir el estado de un objeto a un formato que se pueda almacenar o transportar. El complemento de la “serialización” es la “deserialización” que convierte una secuencia en un objeto. (Microsoft Corp, 2010) Juntos, estos procesos permiten almacenar y transferir fácilmente datos. Por lo tanto en el sistema los objetos son serializados (guardados) en un archivo de texto plano, luego si se quiere consultar alguna salida del sistema se debe “deserializar” (reconstruir) el archivo de texto plano a objetos.

La serialización permite al programador enviar el objeto de un dominio a otro como una cadena XML o a su vez mantener segura la información entre aplicaciones, la desventaja de la serialización es el que no soporta transacciones, consultas o compartición de datos entre usuarios es por eso que se usa este método para aplicaciones simples o ambientes empotrados que no están en la capacidad de poder gestionar una base de datos de forma eficiente.



**Figura II. 1 Procesos de Serialización**

Fuente: https://sites.google.com/site/jojooa/informatica-tecnologia/definicion-persistencia-de-objetos-que-es-persistencia-de-objetos/definicion-de-serializacion-de-objetos-que-es-serializacion-de-objetos

* + 1. **Bases de datos orientadas a objetos (ODBMS)**

Una base de datos orientada a objetos es aquella que incorpora todo los conceptos importantes del paradigma de programación orientada a objetos, es por ello que el programador puede definir las operaciones sobre los datos como parte de la definición de la base de datos. La información que manejan estas bases de datos es representada mediante objetos lo cual hace que en los lenguajes de programación que lo soporten se representen como objetos, dando como resultado datos persistentes de forma transparente, control de concurrencia, recuperación de datos, consultas asociativas y otras capacidades. (Paszniuk, 2013)

* + 1. **Bases de datos relacionales**

Una breve definición de este tipo de bases de datos es que tienen todos los datos visibles al usuario de una manera organizada estrictamente como tablas, las cuales se perciben como una colección de relaciones normalizadas de diferentes niveles que cumplen con el modelo relacional que describe el lugar y la forma como los datos sean almacenados no tiene relevancia, lo cual hace más fácil de entender y usar para los usuarios esporádicos de la base de datos. La información puede ser recuperada o almacenada a través de lenguaje de consultas que ofrecen una amplia variedad de mecanismos para poder administrar la información de forma eficiente.

De las tres formas de persistencia aquí detalladas, se puede evidenciar que las bases de datos relacionales muestran que son ideales al tener una mayor estandarización, escalabilidad, eficiencia y robustez lo cual la hace ideal para usarlas en aplicaciones empresariales, pero al momento de querer tener una persistencia con los objetos se da problemas de incompatibilidad entre el paradigma orientado a objetos y el modelo relacional, a esto se lo denominada la diferencia o impedancia objeto relacional.

* 1. **Capa de Persistencia**

Esta capa nos brinda servicios para una sincronización a la capa de lógica de dominio con un medio de almacenamiento, para esto debemos identificar los objetos de memoria que deben sobrevivir a la ejecución del programa teniendo así una persistencia de largo plazo.

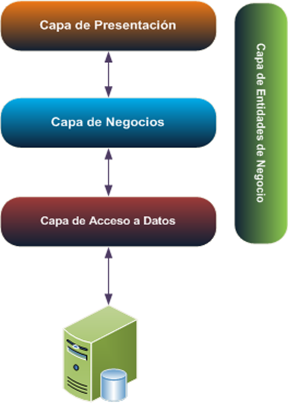


Figura II. 2 Esquema gráficos de la programación N-Capas

Fuente:http://lh4.ggpht.com/diego.rojas/SP6HSUe0cUI/AAAAAAAAAFs/F3JonrZHGy0/image\_thumb%5B6%5D.png?imgmax=800

Dependiendo de la aplicación hay diferentes tipos de requisitos para esta capa, por colocar un ejemplo podremos decir que una aplicación de diseño CAD, al llamar un documento para edición, esta trae un gran conjunto de información a memoria desde el medio de almacenamiento ya que si traería un subconjuntos de información hará que la aplicación tenga tiempos de respuestas muy bajos, es por ello que es importante optimizar la cantidad de información que se trae a memoria para ser usada.

* + 1. **Requerimientos de la Capa de Persistencia**

La capa de persistencia encapsula los comportamientos de los objetos creados para que estos pueden persistir, entre las operaciones están leer, escribir, borrar objetos, un buen framework de persistencia debe proveer de una manera sencilla el siguiente mecanismo:

* Soporte para diversos tipos de mecanismos de persistencia (archivos planos, RDBMS, OODBMS, etc.)
* Encapsulamiento total del mecanismo de persistencia.
* Acciones sobre múltiples objetos (asociaciones, búsquedas, etc.).
* Transacciones: planas o anidadas, locales o distribuidas.
* Extensibilidad: permitir agregar nuevas clases (entidades) de forma sencilla.
* Soportar la manipulación automática de identificadores de objetos (OIDs).
* Cursores que permitan la lectura incremental de objetos desde la base de datos.
* Soporte para proxies que habiliten las técnicas de "lazy loading".
* Registros (Record Sets) para operaciones de bajo nivel.
* Soporte para múltiples arquitecturas (Desktop, Enterprise).
* Soporte para diferentes versiones de una base de datos o diferentes proveedores en forma transparente a la aplicación.
* Permitir ejecutar consultas SQL si es necesario
* Proveer un lenguaje para consultas Orientadas a Objetos, más naturales al modelo de información de la aplicación, y transformación automática de éstas a SQL al momento de ejecutar la consulta.
  1. **Mapeo objeto relacional (ORM)**

El mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) es una técnica de programación para convertir datos, utilizado un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia.(Soy Programador, 2011)Esto nos indica que se crea una base de datos orientada a objetos de manera virtual sobre la base de datos relacional, lo cual nos ayuda al uso de las características propias de la orientación a objetos para su posterior implementación a través de un framework de persistencia.

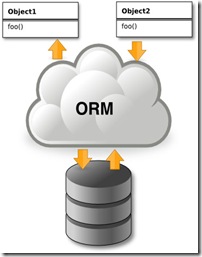


Figura II. 3 Esquema de la abstracción del ORM

**Fuente:** http://soyprogramador.liz.mx/mapeo-objeto-relacional-orm/

En la actualidad, las bases de datos relacionales almacenan datos de un manera primitiva, por lo que no es posible guardar objetos que se van creando por la aplicación, lo que se hace es convertir los datos del objeto en datos primitivos para poder almacenarlos en las tablas, luego debemos recuperar los datos almacenados y volver a construir el objeto, es aquí donde el mapeo objeto-relacional nos ayuda a que este proceso sea transparente.

El software orientado a objetos, la información es representada como clases y objetos, en las bases de datos relacionales como tablas y restricciones, por ello el mapeo objeto-relacional hace una correspondencia entre los elementos siguientes:

Tabla II. I Correspondencia de elementos

|  |  |
| --- | --- |
| Software Orientado a Objetos | Base de Datos Relacional |
| Clase | Tabla |
| Propiedad | Campo |
| Objeto | Fila |
| Identificador | Clave primaria |
| Puntero a otro objeto | Clave foránea |

* + 1. **Componentes de un ORM**
* Una solución que se base en ORM debe tener por lo menos aspectos siguientes (Freire, 2008, p. 14):
* Un API para realizar las operaciones básicas de CRUD sobre objetos de clases persistente.
* Un lenguaje o API para especificar las consultas que hacen referencias a las clases y sus propiedades
* Facilidad para especificar el mapeo de metadatos
* Una técnica para la implementación del ORM pueda llevar a cabo búsquedas, asociaciones y otras funciones de optimización.
  + 1. **Diferentes técnicas de mapeo**

En el mercado de software existen gran variedad de herramientas y productos ORM los cuales usan diferentes técnicas para comunicarse con las bases de datos relacionales.

* + - 1. **Mapeo directo sin identidad**.

El programador debe crear clases con atributos de cuyo nombre se derivan las columnas de las tablas de la base de datos y viceversa, los frameworks examinan estas clases y hacen los querys necesarios, el problema radica en que no mantiene la identidad basada en llave primaria por consecuente pueden existir en memoria objetos diferentes que pueden corresponder a una misma hilera de la tabla y al ser actualizados uno de ellos se perderán los cambios en el otro, lo cual implica que la lógica de la aplicación este pendiente de esto no ocurra.

* + - 1. **Mapeo directo con identidad**.

Parecido al anterior, pero este mapeo mantiene el índice de todos los registros que residen en memoria, lo cual asegura que no existan objetos duplicados que pertenezcan a un misma hilera de la tabla, el problema que se tiene es que la capa ORM debe estar al pendiente de todo el ciclo de vida de los objetos por lo tanto la lógica de la aplicación no sabrá cuando se debe eliminar un objeto que esta compartido.

* + - 1. **Generación del mapeo a partir de XML**.

Esto trata de generar código a partir de definiciones en XML, luego este código generado lo más aconsejable es no modificarlo a menos que sea estrictamente necesario.

* + 1. **Mapeo de asociaciones**

Al mapear relaciones o colecciones entre entidades se tiene que utilizar asociaciones las cuales están en diferentes categorías:

En base a la asociación entre objetos:

* Asociación.
* Agregación.
* Composición.

En base a la multiplicidad:

* Uno a uno.
* Uno a muchos (muchos a uno según desde donde se parta o lea).
* Muchos a muchos.

En base a la dirección:

* Unidireccionales.
* Bidireccionales.

En cuanto a la direccionalidad, todas esta relaciones en la base de datos son bidireccionales, algo importante es que si los objetos asociados a otros se leen automáticamente todos sus objetos asociados en memoria deben cargarse a memoria lo cual disminuirá el rendimiento del sistema, para ello existe una técnica denominada lazy loading la cual carga los objetos a medida que estos son solicitados, por ejemplo si una asociación nunca es solicitada entonces nunca será leída de la base de datos.

* + 1. **Maneras de usar los ORM**

El programador puede usar las herramientas de ORM de las siguientes maneras:

* + - 1. **Relacional puro.**

Todas las aplicaciones, están de cierta manera diseñadas en base al modelo relacional lo cual es una aproximación válida para aplicaciones sencillas que no utilizaran gran cantidad de código, estas aplicaciones comúnmente suelen usar procedimientos almacenados para dejar parte del trabajo lejos de la lógica de negocio y dentro de la base de datos.

* + - 1. **Mapeo objeto ligero**.

Las entidades se representa como clases que luego serán mapeadas manualmente en las tablas relacionales, se oculta el código a las bases de datos con los patrones de diseño más utilizados, este mapeo es el de mayor aceptación y efectivo cuando se tiene muchas entidades.

* + - 1. **Mapeo objeto medio**.

Las asociaciones entre los objetos son manejadas por los mecanismos de persistencia y se refina el acceso a los datos a través de expresiones que comúnmente se usan en los lenguajes orientado a objetos, aquí existe una cache de objetos en la capa de persistencia, este tipo de mapeo es adecuado para aplicaciones de complejidad media donde se la da más importancia a la portabilidad hacia diferentes RDBMS, por lo general no se usa procedimientos almacenados.

* + - 1. **Mapeo objeto completo**.

Este mapeo consiste en el modelado de objetos con características propias de la programación orienta a objetos como es la herencia y polimorfismo, la capa de persistencia es transparente al manipular los datos almacenados en el RDBMS directamente a través de objetos. Este nivel de funcionalidad es muy complejo ya que se puede requerir meses o años para su desarrollo.

* + 1. **Herramientas ORM**

En el mercado existen múltiples alternativas para los programadores de Java al momento de querer trabajar con un mapeador de objeto-relación, existen muchas comunidades implicadas en este mundo de la persistencia de objetos para Java que se basan en estándares open source y grupos comerciales. En la tabla siguiente se tiene diferentes características de los frameworks de persistencia (Junta de Andalucía, 2012).

Tabla II. II Comparación de los framework de persistencia

|  |  |  |  |
| --- | --- | --- | --- |
| Características | iBatis | Hibernate | JPA |
| Simplicidad | Muy bueno | Bueno | Bueno |
| Solución completa ORM | Mejorable | Muy bueno | Muy bueno |
| Adaptabilidad a cambios en el modelo de datos | Bueno | Mejorable | Mejorable |
| Complejidad | Muy Bueno | Mejorable | Mejorable |
| La dependencia de SQL | Bueno | Mejorable | Mejorable |
| Rendimiento | Bueno | Muy bueno | - |
| Portabilidad a través de diferentes bases de datos relacionales | Mejorable | Muy bueno | - |
| Portabilidad a las plataformas de no Java | Muy bueno | Bueno | No soportado |
| Comunidad de soporte y documentación | Mejorable | Muy bueno | Muy bueno |

* + 1. **Ventajas**
* Entre las ventajas principales que aportan ORM es la de
* Reutilización, ya que permite llamar a los métodos del objeto des de diferentes partes de la aplicación.
* Seguridad, al evitar tipos de ataque como SQL injections.
* Mantenimiento de la codificación, esto gracias a la correcta ordenación de la capa de datos haciendo que la mantenibilidad del código sea más sencillo.
* Lenguaje propio, para realizar las consultas lo que permite al programador utilizar el lenguaje de cada herramienta.
* Abstracción de la base de datos, esto nos separa del motor de base de datos que pudiéramos utilizar, lo cual a futuro si se desearía cambiar de motor de base de datos estos cambios no afectaran a nuestro sistema, haciendo el cambio más sencillo y rápido.
* Rapidez, con los frameworks actuales que permiten la creación de un modelo a través del esquema de la base de datos, al crearnos el modelo adecuado.
  + 1. **Desventajas**
* Tiempo de aprendizaje de la herramienta, esto debido a que muchas de las veces es muy compleja o no existe una documentación adecuada que respalde al ORM.
* Aplicaciones lentas, esto a que las consultas que se hagan sobre la base de datos, el sistema transformara al lenguaje propio de la herramienta, luego leer los registros y crear los objetos, esto en aplicaciones con muchas tablas.
  1. **Java Database Conectivity (JDBC)**

El Java Database Conectivity es el API de java estandarizado para la conectividad hacia la base de datos, independiente entre el lenguaje de programación Java y el motor de base de datos (TutorialsPoint, 2015), utilizando el lenguaje de consulta SQL del modelo de datos que se utilice, esto nos dice que las sentencias SQL se pueden mezclar con Java, dado que una variable de Java puede ser usada en sentencias SQL para recibir o dar valores. El mapeo de las tablas de la base de datos hacia las clases de Java, donde cada fila de una tabla se convertirá en una instancia de la clase y lo mismo que cada columna corresponde a un atributo de esa instancia.

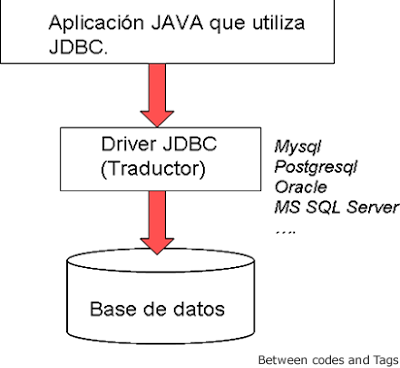


Figura II. 4 Esquema de Acceso del controlador JDBC

**Fuente:** http://dmk-ricteam.blogspot.com/2011/07/conectar-posgresql-con-java-jdbc-1.html

El API de JDBC son librerías donde se incluyen tareas comúnmente asociadas al uso de bases de datos como son:

* Hacer la conexión a la base de datos.
* Creación de procedimientos SQL o MySQL.
* Ejecución de consultas SQL en la base de datos.
* Visualización y modificación de los registros de las tablas.

JDBC logra todas estas tareas de peticiones a través de un software intermediario llamado Driver JDBC, el cual se encarga de traducir las llamadas a los métodos en ordenas nativas para el gestor de bases de datos.

* + 1. **Características de JDBC**

Cuando SUN MicroSystems (propiedad de Oracle), decidió trabajar en este ámbito, siguió diferentes normas para la definición de la interfaz que se usa actualmente, algunas de esas características son (Martínez, 2006):

* JDBC debe ser utilizable sobre cualquier API de acceso a datos.
* JDBC debe proveer un interfaz homogéneo al resto de APIs de Java
* JDBC debe ser un API simple para que desde ahí pueda seguir creciendo.
* JDBC deberá mantener los casos comunes de acceso a base de datos de la manera más sencilla posible
* Crear múltiples métodos para múltiple funcionalidad.
  + 1. **JDBC como componente necesario para la persistencia**

La Figura II. 5 Nos muestra una participación de los diversos mecanismos de mapeo objeto-relacional y como estos se relacionan con la codificación de la aplicación y fuentes de datos relacionales, la eficiencia del controlador JDBC hace una profunda diferencia en el rendimiento, escabilidad y fiabilidad de las aplicaciones. En vista que se tiene múltiples factores a tomar en cuenta al momento de elegir un controlador JDBC, se tendrá que seleccionar aquel que mejor se adapte a nuestra solución aportando escabilidad y fiabilidad que es la clave para obtener grandes beneficios de cualquier aplicación.

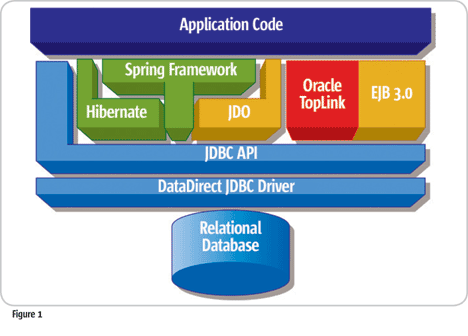


Figura II. 5 Esquema usando frameworks de persistencia

**Fuente:** http://gemsres.com/story/oct05/140123/bruce-fig1.gif

* 1. **¿Qué es un Framework?**

Un Framework es un conjunto estandarizado de conceptos, prácticas y criterios para hacer frente a un tipo común de problema, que puede ser usado para ayudarnos a resolverlo de forma rápida y eficaz.

El objetivo de los Frameworks es proporcionar una estructura común, de modo que los desarrolladores no tienen que hacer el código de cero cada vez y pueden volver a utilizar la gran mayoría que ya está realizado. (Gómez, 2013)

Un framework se basa en términos generales, que es una herramienta que tiene un conjunto estandarizado de conceptos, prácticas y criterios para enfocar o resolver una problemática en particular, en el área de software un framework es una estructura conceptual y tecnológica, en la que normalmente se definen módulos concretos de software, bibliotecas y un lenguaje interpretado, entre otras herramientas para poder así ayudar al buen desarrollo de la solución.

* + 1. **Arquitectura del Framework**

Aquí se basa en el Modelo Vista Controlador (MVC), ya que nos permite fragmentar nuestra programación lo cual es muy importante en cuanto a la implementación de una solución de software.

* **Modelo**: aquí se maneja las operaciones lógicas y manejo de la información, en otras palabras se encarga de la lógica de negocio y el acceso a los datos.
* **Vista**: es la forma en como los datos serán presentas al usuario y como este debe interactuar con los mismos, son las interfaces visuales de la aplicación.
* **Controlador**: este ítem es el que controla el acceso a los datos de nuestra aplicación la cual puede incluir archivos, scripts y partes de software y cualquier otro tipo de información que permita la Vista.

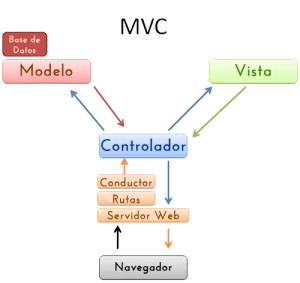


Figura II. 6 Esquema del Modelo-Vista-Controlador

Fuente: http://www.suenyos.com/2011/07/20/crear-un-framework-en-php5-desde-cero-1%C2%AA-parte/

* + 1. **Tipos de framework**

En el mercado de software existen básicamente dos tipos para diferenciar como es los backend y frontend ya que debemos elegir si el framework a usar es para la parte visual o la lógica de la aplicación, luego están los framework de acceso a datos.

* Backend: es la parte en la que se procesa la información recolecta desde el frontend, esto por lo general consta de tres actores: un servidor, una aplicación y una base de datos.
* Frontend: es la parte visual que se muestra al usuario, ya sea de una menara web o de escritorio, también es responsable de la recolección de datos por parte del usuario a través de la implementación de interfaces
* Acceso a datos: este trata del acceso a las bases de datos a través de herramientas de persistencia.
  + 1. **Importancia de un Framework**

Varios aspectos sean tomados en cuenta para definir la importancia de un framework, según algunos autores indican lo siguiente (NAULA & MAIQUIZA, 2012, p. 24):

* Modularidad y reducción de la complejidad: la aplicación está formada por subsistemas especializados en distintos aspectos fundamentales de toda la aplicación (persistencia, presentación, manejo de log, etc.)
* Documentación: el framework promueve el uso correcto del mismo y disminuya el esfuerza necesario para el mantenimiento
* Eficiencia: el desarrollador puede concentrarse en los requisitos funcionales de la aplicación
* Aplicaciones ricas: posibilidades de dar más funcionabilidad a los usuarios de la aplicación.
* Buen manejo de base de datos del que puede crear cada desarrollador: es muy conocido que si se dispone de poco tiempo para desarrollar toda una aplicación el framework permite disminuir tiempos para poder acceder a las bases de datos a través del patrón MVC.
* Seguridad de la aplicación: es una de las razones más importantes, desafortunadamente la seguridad en el proceso de desarrollo en una aplicación aun no es algo que se considere en toda la importancia que debería dársele.
  + 1. **Ventajas**
* Desarrollo rápido de aplicaciones: debido a que el framework tiene varios componentes incluidos, se hace reutilización de componentes de software lo que libera al programador de la escritura de código de bajo nivel.
* Programación de componentes: ya que le framework y sus componentes siguen por lo general una filosofía de orientación a objetos se logra que estos componentes sean clases que pertenecen a una jerarquías de clases, lo cual hace en que genere bibliotecas más fáciles de aprender a usar.
* Rapidez y facilidad: las tareas que normalmente le podrían tomar horas al programador en escribir cientos de líneas de código, ahora con el framework puede hacerlo en cuestión de minutos con clases y funciones pre construidas, los desarrollos se convierte mucho más fácil y eficiente.
* Seguridad: un framework que se vuelve popular para los programadores, siempre será más seguro ya que los mismo se convierten en probadores de mediano o largo plazo, en caso de ellos encontrar vulnerabilidades o inseguridades, pueden aportar esos conocimientos a la comunidad que esta tras el desarrollo del framework y hacérselos saber para qué pueden mejorar en futuras versiones.
* Costos: muchos de los framework son open source, por ende los costos para el cliente final serían menores.
  + 1. **Desventajas**
* Dependencia: se dependerá del código fuente de la aplicación con respecto al framework al momento de desear cambiar de framework, lo que resulta en que la mayor parte del código deberá escribirse nuevamente.
* Demanda de recursos computacionales: esto debido a la reutilización ya que los frameworks tienden a generalizar funcionalidades en sus componentes, lo cual hace que se incluyan características adicionales que talvez no usemos, provocando así una sobrecarga de recursos.
* Aprendizaje: al usar un framework se sabe muy poco en que lenguaje de programación está construido, es por ello que se aprende el framework y no el lenguaje del mismo
* Limitación a la personalización: el comportamiento básico del framework no puede ser modificado a nuestras necesidades, lo cual nos obliga a respetar las limitaciones y trabajar dentro de las condiciones del framework.
* Documentación: muchas de la veces la documentación no está disponible en los idiomas nativos de los programadores los cuales le llevara más tiempo aprender el framework.
  1. **Framework Hibernate**
     1. **Introducción**

La persistencia de los datos se puede ver en cualquier lugar de una aplicación, la gestión de datos persistentes es uno de los pocos desafíos que las tecnologías modernas se enfrentan. Una solución llamada Object-Relational Mapping (ORM) ha ganado gran popularidad en los últimos años. ORM es una pieza de software para la representación y la conversión de datos, entre la base de datos y el lenguaje de programación orientado a objetos.

Hibernate es una de las soluciones ORM y es un proyecto de código abierto, aunque el framework Hibernate no es la única solución de persistencia, se ha hecho muy famosa actualmente, debido a su gran variedad de características en comparación con sus competidores. Se necesita mucho de las bases de datos relacionales con código legible del paradigma orientado a objetos, siendo esto así el framework soluciona esta brecha pudiendo así los desarrolladores concentrarse en la lógica de negocio principal de la aplicación y no con la sintaxis SQL propensa a errores.

* + 1. **Historia**

Hibernate fue una iniciativa de un grupo de desarrolladores dispersos alrededor del mundo conducidos por Gavin King. Tiempo después, JBoss Inc. (empresa comprada por Red Hat) contrató a los principales desarrolladores de Hibernate y trabajó con ellos en brindar soporte al proyecto.

* + 1. **Arquitectura de Hibernate**

La figura II.7 muestra como Hibernate está en medio de la comunicación entre la aplicación y la base de datos con la que debe comunicarse para acceder a los datos, esto es una representación visual de la arquitectura, con la cual proporciona servicios y objetos persistentes a la aplicación que está por encima del él.

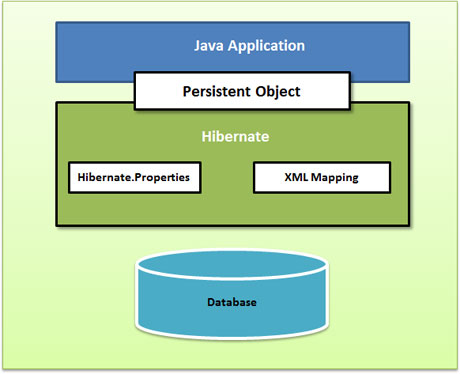


Figura II. 7 Arquitectura de Hibernate

Fuente: http://www.tutorialspoint.com/hibernate/images/hibernate\_high\_level.jpg

* + - 1. **Arquitectura Mínima**

La arquitectura minimalista tiende a gestionar sus propias conexiones JDBC y proporcionar estas conexiones a Hibernate, además que gestiona transacciones por sí mismo, por lo cual este enfoque utiliza un subconjunto mínimo de las API de Hibernate.

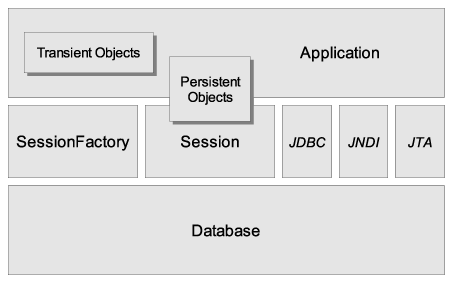


Figura II. 8 Arquitectura de Hibernate mínima

Fuente: http://docs.jboss.org/hibernate/orm/4.3/manual/en-US/html/images/lite.png

* + - 1. **Arquitectura Completa**

La arquitectura completa abstrae de las API de JDBC / JTA subyacentes datos que permiten a Hibernate pueda manejar los detalles para la aplicación.

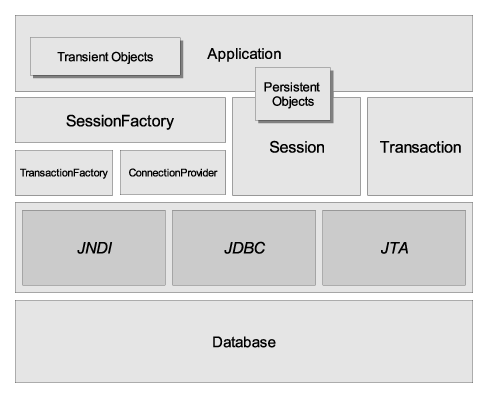


Figura II. 9 Arquitectura de Hibernate completa

Fuente: http://docs.jboss.org/hibernate/orm/4.3/manual/en-US/html/images/full\_cream.png

* + - 1. **Ciclo de Vida de los Objetos**

El ciclo de vida de un objeto de Hibernate, muestra como este logro persistir en sus estados más básicos dado que Hibernate puede guardar, actualizar y eliminar registros en una tabla de la base de datos y como esos valores se asocia con sus clases en Java y la tabla de la base de datos.

Hibernate por lo general su ciclo de vida tiene cuatro estados:

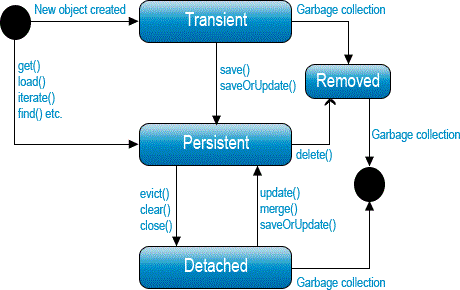


Figura II. 10 Estados del Ciclo de Vida

Fuente: http://www.roseindia.net/tutorialfiles/44253.HibernateLifeCycle.gif

* **Transitorio (Transient)**: cuando se crea un objeto en nuestra clase POJO, se puede decir que este se encuentra en estado transitorio pero no es aun persistente, en este estado no puede hacer modificaciones a la bases de datos, si por largo tiempo no es referenciado este va director al administrador de procesos muertos "basurero"
* **Persistente (Persistent)**: al momento de guardar el objeto transitorio pasa a estar en estado persistente es aquí cuando ya tiene instancias de la base de datos con la tabla previamente asignada y es posible hacer modificaciones en la base de datos y todo cambio se guardara automáticamente
* **Eliminado (Removed)**: cuando este objeto se elimina de la base de datos, pasa a estado independiente, los cambios invocados por el objeto no se guardan en la base de datos y es ignorado por Hibernate y si no es usado por largo tiempo va al administrador de procesos muestros "basurero
* **Independiente (Detached)**: en este estado el objeto aún persiste a pesar de que la sesión haya finalizado, lo cual hace que el objeto sea independiente por consecuente no está instanciado dando así que si deseara hacer algún cambio a la base de datos no se podrá guardar ya que no hay objeto persistente explicito, es por ello que hay métodos de sesión para separar al objeto de la cache de sesión. Por el contrario si se desea regresar al objeto al estado persistente y volverlo a conectar para hacer modificaciones en la base de datos se usan otros métodos como
  + update()
  + mergo()
  + saveOrUpdate()
  + lock()
    - 1. **APIs básicas**

Hibernate utiliza muchas API existentes, como JDBC, Java Transaction API (JTA) y JNDI. Se sabe que JDBC proporciona niveles rudimentarios de abstracción a las bases de datos relacionales, lo cual permite a casi cualquier base de datos con un driver JDBC puede apoyarse para integrarse con Hibernate y los servidores de aplicaciones J2EE.

* + 1. **Archivo de Configuración**

La aplicación que utilizará el API de Hibernate tendrá un solo archivo de configuración, en el que esta información como, la dirección URL del servidor de base de datos para que la aplicación pueda conectarse, el nombre de usuario / contraseña de la base de datos, nombres de las clases del controlador y con otro conjunto de preferencias.

Un archivo de configuración Hibernate puede ser basado en XML (*hibernate.cfg.xml)* o puede ser un ordinario archivo de propiedades Java (*hibernate.properties*) con las combinaciones de clave y valor. Este archivo de configuración debe ser colocado en la ruta de clases (classpath) en tiempo de ejecución de la aplicación. Si tanto el archivo de configuración basado en XML y el archivo de configuración basada en propiedades se encuentran en la ruta de clases, entonces el archivo de configuración basado en XML tendrá preferencia sobre el otro.

La siguiente es la estructura de un archivo de configuración Hibernate:

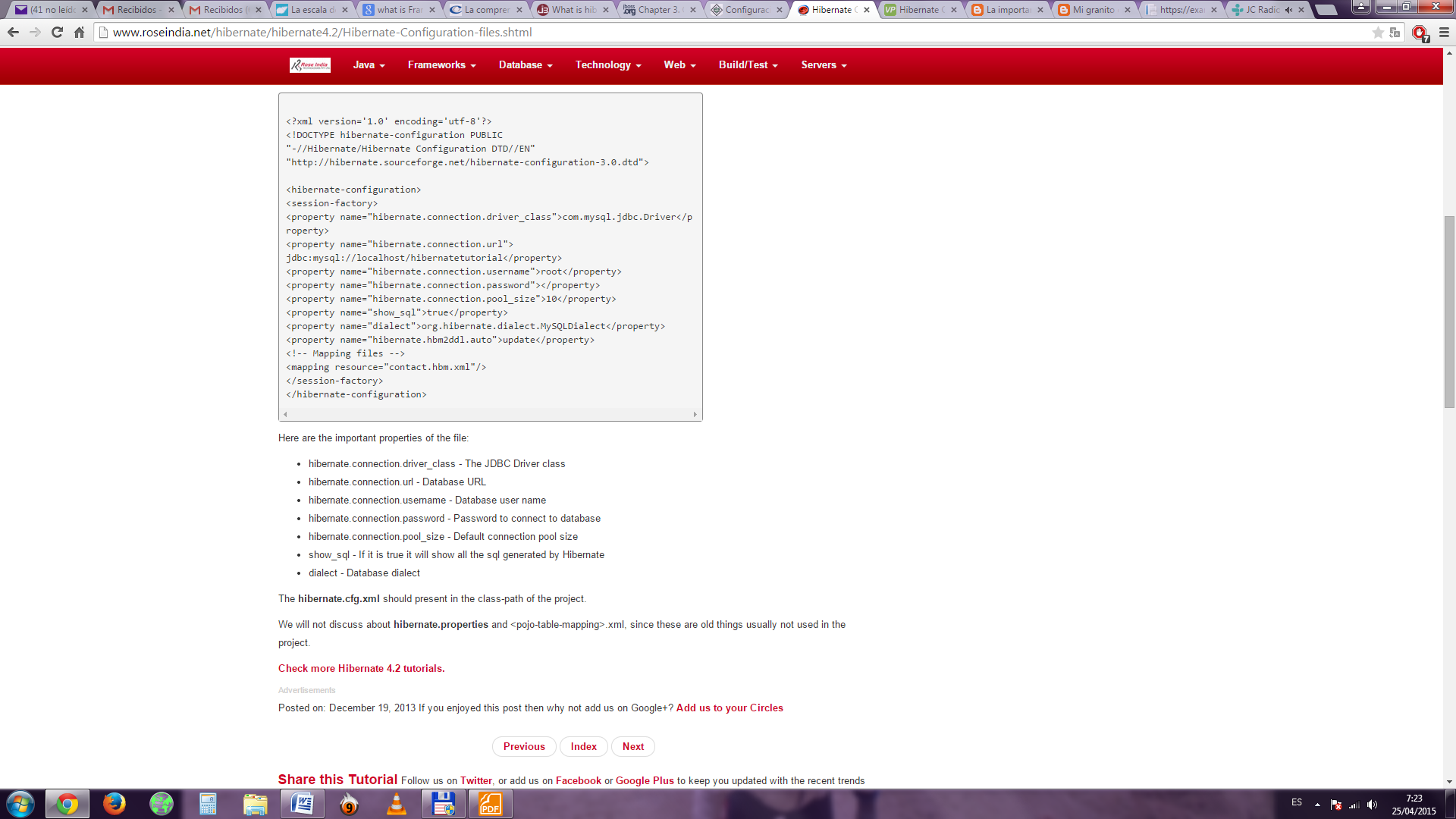


Figura II. 11 Archivo de configuración de Hibernate

Fuente: http://www.roseindia.net/hibernate/hibernate4.2/Hibernate-Configuration-files.shtml

* + 1. **Archivos de Mapeo**

Este archivo proporciona información en como una clase Java está asociada a una tabla de la base de datos relacional, adicional a esto también esta como cada propiedad o atributo de la clase Java está asignada a la columna de la tabla, las relaciones entre las entidades también están aquí definidas.

* + 1. **Mapeo Básico de O/R**

Los mapeos objeto/relacional por lo general se definen en un documento XML. El documento de mapeo se crea de manera que se pueda leer y editar a mano. El lenguaje de mapeo está centrado en lenguaje Java, lo que significa que los mapeos se construyen alrededor de declaraciones de clases persistentes y no alrededor de declaraciones de tablas.

* + - 1. **Mapeo de Hibernate**

Este archivo o elemento tiene varios atributos opcionales. Los atributos schema y catalog especifican que las tablas a las que se refieren en este mapeo, pertenecen al esquema y/o catálogo mencionado. De especificarse, los nombres de tablas serán calificados por el nombre del esquema y del catálogo. De omitirse, los nombres de las tablas no serán calificados. El atributo default-cascade especifica qué estilo de cascada se debe asumir para las propiedades y colecciones que no especifican un atributo cascade. Por defecto, el atributo auto-import nos permite utilizar nombres de clase sin calificar en el lenguaje de consulta.

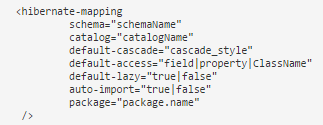
****

Figura II. 12 Ejemplo del archivo nombre-de-la-clase.hbm.xml

Fuente: https://docs.jboss.org/hibernate/orm/3.5/reference/es-ES/html/mapping.html#mapping-declaration-mapping

Para un mapeo en Hibernate, no se considera diferente una vista y tabla base, esto es transparente a nivel de bases de datos, aunque algunos DMBS no logran soportar correctamente las vistas de actualización

* + - 1. **Entidades y Valores**

El servicio de persistencia, los objetos a nivel de lenguaje Java se clasifican en dos grupos:

Una **entidad** existe independientemente del otro objeto que referencie a la entidad, lo que comparado con el modelo habitual de Java donde un objeto no referenciado es considerado para gargabe(basura), por lo contrario entidades grabadas y borradas pueden ser tratados en cascada desde la entidad padre a sus descendientes.

La persistencia de una entidad consta de referencias a otras entidades e instancias del tipo **valor**, estos pueden ser primitivos (colecciones, componentes y algunos objeto es inmutables. Al contrario de las entidades, los valores en especial las colecciones y los componentes, son persistidos y borrados por alcance.

* + - 1. **Mapeo de una clase más de una vez**

Existe la posibilidad de proveer más de un mapeo para una clase persistente en particular, si fuera el caso debemos proporcionar un nombre de entidad para precisar entre las instancias de las dos entidades mapeadas. Hibernate deja especificar el nombre de entidad al momento de trabajar con objetos persistentes, al escribir consultas, o al mapear asociaciones a la entidad mencionada

* + - 1. **Objetos de bases de datos auxiliares**

Estos objetos de las bases de datos nos permiten la creación”CREATE” y eliminación “DROP” de objetos de bases de datos arbitrarios. Junto con las demás herramientas que han evolucionado para el esquema de Hibernate, nos dan la habilidad de definir de una manera completa el esquema para un usuario dentro de los archivos de mapeo de Hibernate. Debido a que están diseñados particularmente para crear y eliminar cosas como disparadores “triggers” o procedimientos almacenados.

* + 1. **HQL( Hibernate Query Languaje)**

El Hibernate Query Languaje (HQL) es el lenguaje de consultas usado por Hibernate este es muy parecido al SQL estándar con la diferencia de que el HQL se orienta a objetos donde se puede usar los nombres de clase y sus atributos en lugar de las tablas y columnas, lo cual nos permite usar herencias polimorfismo y asociaciones, el desarrollador escribe la consulta en HQL e Hibernate se encarga de convertirla en SQL estándar para que sea entendido por el motor de base de datos y proceda a ejecutar la operación indicada. A continuación podemos ver los objetos Java creados a partir de tablas de la base de datos.

* + - 1. **Sensibilidad a las letras mayúsculas**

Las consultas que se efectúen con HQL no son sensibles a las mayúsculas esto se conoce en programación case-insensitive, lo que nos indica que una instrucción “*SeLect*” es lo mismo que “*sELEct*” o “*SELECT*”, pero esto no es así cuando se trata de los nombres de clases y sus propiedades.

* + - 1. **Cláusula SELECT**

Esta cláusula escoge que propiedades y objetos se devolverán en el resultado de la consulta**,** observemos el siguiente ejemplo:

“***SELECT*** *c* ***FROM*** *Ciclo c* ***ORDER BY*** *nombre*”

Podemos ver que es necesario definir un alias “*c*” de la clase Java “*Ciclo*” , tras la escritura de la palabra “*SELECT*” se usa el alias en lugar de “*\**” y por ultimo para ordenar los objetos usamos la propiedad “*nombre*” de la clase *“Ciclo”*.

Las consultas tipo select también pueden retornar múltiples objetos o propiedades como un Array tipo Objects[], ejemplo: “***SELECT*** mother, offspr, mate.name ***FROM*** DomesticCat as mother ***INNER JOIN*** mother.mate as mate ***LEFT OUTER JOIN*** mother.kittens as offspr”.

* + - 1. **Cláusula WHERE**

Muy similar como es en SQL en HQL es posible filtrar los resultados de las consultas mediante esta cláusula, ejemplo: “***SELECT*** *p* ***FROM*** Profesor p ***WHERE*** nombre = ‘ÍSABEL’***AND*** ape1 <> ‘ÓRELLANA’”, podemos observar como nombre de la clase y los nombres de los campos del apartado “***WHERE***” hace referencia a las propiedades de Java mas no a los nombres de columnas de las tablas de la base de datos.

* + - 1. **Clausula FROM**

Una de las consultas de lo más simple posible en Hibernate podría ser “***FROM*** *eg.Cat*”, lo cual me retornaría todas las instancias de la clase “*eg.Cat*”. Con la finalidad de referirse al “*Cat*”en otras partes de la consulta es necesario colocar un alias por ejemplo: “***FROM*** *Cat as cat*”, con esto la consulta me asigna el alias “*cat*” a las instancias de “*Cat*” por consiguiente podremos usar ese alias en la consulta.

Otra caso a seguir es cuando se dan múltiples clases, lo que causa que se tenga un producto cartesiano o unión cruzada para esto se pudiera usar, “***FROM*** *Formula**as form, Parameter as param*”.

* + - 1. **Ejecución de Consultas**

HQL y SQL nativos, se representa instanciando el *org.hibernate.Query*, esta interfaz nos proporciona métodos para ligar parámetros, para el manejo de resultados y ejecución de las consultas. Dado que la consulta se realiza invocando al método *list()* lo cual será carga completamente dentro de una colección de la memoria donde todas las instancias a la entidad recuperadas por alguna consulta se encontraran en estado persistente.

* + 1. **Ventajas**
* La portabilidad de Hibernate, debido a que usa su propio lenguaje de consulta (HQL), esto hará que cambiarse de base de datos sea más fácil, ya que solo deberíamos de cambiar las sentencias SQL.
* La documentación es muy amplia ya que este bien estructurado y concisa en diferentes idiomas, de las diferentes versiones de Hibernate.
* El rendimiento de Hibernate ya que este almacena en cache los datos, por consiguiente si volvieran a ser llamados se recuperarían más rápido de la base de datos.
  + 1. **Desventajas**
* Al momento de existir cambios en las tablas (creación de una nueva columna), esto requerirá de cambiar los objetos ya previamente mapeados y los archivos XML asociados a estos.
* Debido a que las clases son generadas en tiempo de ejecución para poder acceder a los objetos de las bases de datos, esto nos genera demasiada sobrecarga más de lo que lo haría la propia consulta SQL.
* La curva de aprendizaje es mucho mayor, que lo que se requiere para aprender SQL nativo.
* Para un correcto uso de la herramienta es necesario una adecuada configuración de los muchos ficheros que esta maneja para el proceso de las transacciones SQL y creación de objetos Java.
  1. **Framework MyBatis**
  2. **Introducción**

MyBatis es un framework de persistencia con soporte SQL, procedimientos almacenados y mapeos avanzados. Este framework elimina casi todo el código JDBC, la corporación manual de los parámetros y la preparación de resultados. Se lo puede configurar con XML o anotaciones y nos permite mapear mapas de las bases de datos y POJOs (Plain Old Java Objects), esta herramienta es software libre y se desarrolla bajo la licencia de Apache 2.0.  Mybatis asocia los objetos del modelo (JavaBeans) con sentencias SQL o procedimientos almacenados mediante ficheros descriptores XML, simplificando la utilización de bases de datos.

* 1. **Historia**

MyBatis es una evolución del proyecto iBATIS y mantenido por una comunidad de desarrolladores entre los que se encuentran los primeros desarrolladores de iBATIS originalmente. El proyecto nació el 19 de mayo de 2010, justamente después de la publicación de iBATIS 3.0 la comunidad de desarrollo de iBATIS anunció que continuarían el proyecto bajo un nuevo nombre y un nuevo hogar. Luego el 10 de noviembre de 2013 el proyecto fue anunciado y movido a Github.

* 1. **Arquitectura**

La arquitectura del framework Mybatis está compuesta por varios ficheros basados en XML más un mapa de archivos SQL (sqlMap), donde hay un fichero por cada tabla de la base de datos relacional, cada fichero XML tiene plantillas de SQL que ejecutarán querys como fueron definidos en un principio y sus mapas de resultados para las clases Java dominantes. La aplicación (solución) se encuentra objetos de acceso a Mybatis que los denominados DAO, dado que estos como el API que ejecuta las plantillas de mapas basados en SQL y asignan los resultados a las clases Java dominantes.

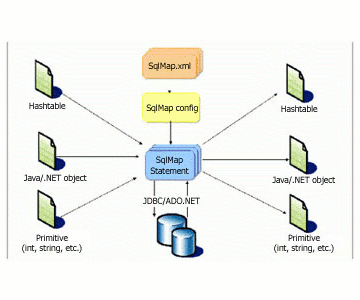


Figura II. 13 Arquitectura de Mybatis

Fuente: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/98

* 1. **Ámbito y Ciclo de Vida**

Es muy importante entender los distintos ámbitos y ciclos de vida de las clases que se mencionaran más adelante. Usarlas de forma incorrecta puede traer serias complicaciones en el desarrollo de una aplicación.

* + 1. **SqlSessionFactoryBuilder**

Esta clase puede instanciarse, usarse y desecharse. No es necesario seguirla manteniendo a lo largo del uso de la aplicación, puede ser desechada claro esto una vez creada la SqlSessionFactory. Por ejemplo un ámbito ideal para el SqlSessionFactoryBuilder será un método (ej. una variable local de método). Pues aquí se puede reusar el SqlSessionFactoryBuilder para construir más de una instancia de SqlSessionFactory.

* + 1. **SqlSessionFactory**

Cuando esta creado el SqlSessionFactory deberá estar disponible durante toda la ejecución de la aplicación, este a diferencia del SqlSessionFactoryBuilder no debe eliminarse o recrearlo por ningún motivo. Se considera una buena práctica no recrear el SqlSessionFactory en la aplicación más de una vez, por lo contrario si esto sucede debe considerarse código sospechoso o malicioso. En palabras resumidas el ámbito ideal para el SqlSessionFactory es el de la aplicación.

* + - 1. **SqlSessionFactory sin creación de archivo XML**

En la configuración siguiente se puede apreciar, que se añade una clase tipo mapper, estas clases Java contienen anotaciones de mapeo SQL que nos permiten evitar el uso de XML.



Figura II. 14 SqlSessionFactory creado sin XML

Fuente: https://mybatis.github.io/mybatis-3/es/getting-started.html

* + - 1. **SqlSessionFactory creado a partir de un archivo XML**

Las aplicaciones que usan MyBatis deben instanciar un SqlSessionFactory, esta se obtiene mediante SqlSessionFactoryBuilder. EL SqlSessionFactoryBuilder puede crear una instancia SqlSessionFactory a través de un fichero de configuración XML.

Crear una instancia SqlSessionFactory desde un fichero XML es relativamente sencillo, se aconseja usar una dirección del recurso, por ejemplo “org/Mybatis/ejemplo/Mybatis-config.xml” el framework MyBatis nos facilita una clase de utilidad, que se llama Resources, la cual contiene métodos que aliviaran la carga de recursos desde la dirección del recurso u otras carpetas y ubicaciones.



Figura II. 15 SqlSessionFactoryBuilder creado a través de archivo XML

Fuente: https://mybatis.github.io/mybatis-3/es/getting-started.html

* + 1. **SqlSession**

Cada thread (hilo de ejecución) deberá tener su propia instancia del SqlSession. Estas instancias thread seguras y no deben ser compartidas. Por ende el ámbito adecuado es el de petición (request), no es aconsejable guardar instancias de SqlSession en un campo estático y menos en una propiedad que es instancia de una clase, tampoco se debe guardar referencias a una SqlSession en ninguno de los ámbitos gestionado s como un HttpSession.

* + 1. **Instancias de mapper**

Los mappers son interfaces que se crean como enlaces a los mapped statements, estas instancias de mappers se obtienen de un SqlSession, por ello es que una instancia de Mapper es lo mismo que el de un SqlSession de la cual fueron creados. Pero a pesar de estos lo recomendado para una instancia mapper es el ámbito de métodos, en otras palabras se puede decir que deberían ser obtenidos en el método que vaya a usarlos y posteriormente descartarlos, dado que no es un problema propagar estos objetos por varias clases dentro de una misma llamada.

* + 1. **Funcionamiento de los Mapped Statament**

Se puede definir tantos mapped statements en un solo fichero XML como sean necesarios lo cual nos ayuda ahorrándonos líneas XML extra que corresponden a la cabecera XML y a la declaración de doctype.

* + - 1. **Namespace:**

Estos espacios de nombres eran opcionales en versiones antiguas de MyBatis, pero esto muchas de las veces generaba confusión y no ayudaba a los desarrolladores, por ello ahora son obligatorios, estos namespace permiten enlazar interfaces. Usar un namespace y colocarlo en el paquete java que corresponde con el namespace hará el código más legible y mejorará la usabilidad de MyBatis a largo plazo.

* + - 1. **Resolución de nombres**

Para reducir la cantidad de texto a escribir MyBatis usa las siguientes normas de resolución de nombres para todos los elementos de configuración, incluidos statements, result maps, cachés, etc.

* Primeramente se buscan directamente los nombres completamente cualificados (fully qualified names) (ej. “com.mypackage.MyMapper.selectAllThings”).
* Pueden usarse los nombres cortos (ej. “selectAllThings”) siempre que no haya ambigüedad. Sin embargo, si hubiera dos o más elementos (ej. “com.foo.selectAllThings" y "com.bar.selectAllThings”), entonces se obtendrá un error indicando que el nombre es ambiguo y que debe ser “fully qualified”
  1. **Ficheros de mapeo XML**

El gran desempeño de MyBatis puede decirse que reside en los Mapped Statements, ya que aquí es donde se encuentra toda la magia, dado lo potentes que son los ficheros XML de mapeo son relativamente simples, los ficheros XML de mapeos SQL solo tienen unos pocos elementos de alto nivel.

* + 1. **Select**

La sentencia select es de la palabras de SQL más usadas y en Mybatis también lo será, puesto que no es útil almacenar datos en las bases de datos si después no pueden ser leídos, por lo general las aplicaciones leen más datos de los que modifican, ya que en cada insert, update o delete siempre podrá haber implícito un select. La sentencia select es bastante simple para consultas simples. Por ejemplo

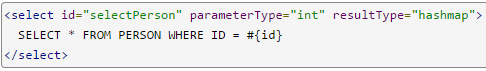


Figura II. 16 Ejemplo de select en Mybatis

*Fuente: https://mybatis.github.io/mybatis-3/es/sqlmap-xml.html*

La sentencia anterior se llama “selectPerson”, la cual recibe un parámetro de tipo int (o Integer), y devuelve una HashMap usando los nombres de columna como clave y los valores del registro como valores.

* + 1. **Insert, Update y Delete**

Estas sentencias son muy similares en su implementación, podemos ver algunos ejemplos:

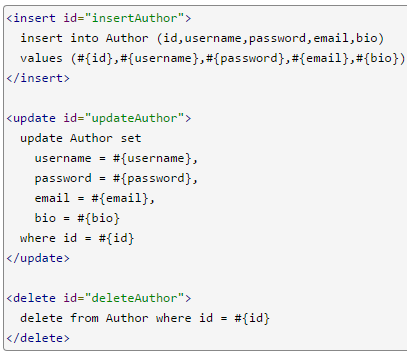


Figura II. 17 Ejemplo de Insert, Update y Delete en Mybatis

*Fuente: https://mybatis.github.io/mybatis-3/es/getting-started.html*

La sentencia insert es algo más compleja dado que posee algunos atributos extra para la gestión de claves de distintas maneras, esto en vista que algunas bases de datos no soportan auto generación de claves primarias. Para el caso en la bases de datos soporte auto generación de claves se puede usar el atributo useGeneratedKeys=”true” y comunicar también en keyProperty el nombre de la propiedad donde se guardara el valor.

* + 1. **Result Maps**

Este elemento es uno de los más importantes y potentes dentro de Mybatis ya que nos facilita la obtención de datos a través del ResultSets que se implementaría con JDBC, muchas de las veces permite hacer cosas que JDBC no soporta. El diseño de los ResultMaps es bastante amplio dado que las sentencias simples no necesitan de un ResultMap explícito, y las sentencias más complejas requieren sólo la información importante para describir relaciones.

* + 1. **Asociaciones**

El elemento association sirve para las relaciones de tipo “tiene-un”, en ello se debe especificar la propiedad destino, el tipo de dato javaType de la propiedad (que por lo general MyBatis sabe el tipo de dato que es), el jdbcType si fuera necesario y un typeHandler si se desea re-escribir el tratamiento de los valores de retorno.

Se debe indicar a Mybatis como es la carga de la asociación, el framework lo podrá hacer de las siguientes maneras

* *Nested Select*: Ejecutando otra select que devuelve el tipo complejo deseado.
* *Nested Results*: Usando un ResultMap anidado que trata con los datos repetidos de resultsets provenientes de joins.



Figura II. 18 Ejemplo de Asociación en Mybatis

*Fuente: https://mybatis.github.io/mybatis-3/es/getting-started.html*

* + 1. **Colecciones**

Este elemento hace referencia a las relaciones de “Tiene Muchos” este elemento collection funciona de forma casi idéntica al association. Tan similar, que documentar todas las similitudes sería redundante.

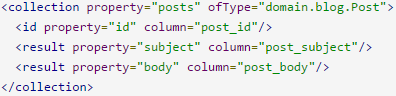


Figura II. 19 Ejemplo de Colecciones en Mybatis

*Fuente: https://mybatis.github.io/mybatis-3/es/getting-started.html*

* + 1. **Discriminadores**

Algunas de las veces las bases de datos nos podrían devolver muchos resultados posibles y distintos, el elemento discriminator está diseñado para tratar estas situaciones, y otras como la jerarquías de herencia de clases, este elemento discriminador es bastante fácil de comprender, dado que funciona muy parecido la sentencia switch de Java.

El elemento discriminator contiene los atributos column y javaType, la propiedad column nos señala donde debe MyBatis obtener el valor con el que comparar. El javaType es necesario para asegurar que se utiliza el tipo de comparación adecuada. Por ejemplo:



Figura II. 20 Ejemplo de Discriminador en Mybatis

*Fuente: https://mybatis.github.io/mybatis-3/es/getting-started.html*

* + 1. **Auto Mapeo**

MyBatis puede auto-mapear los resultados automáticamente, al auto-mapeador da resultado en MyBatis obteniendo el nombre de columna .El auto-mapeo funciona incluso cuando hay un result maps específico. Cuando esto sucede, para cada result map, todas las columnas que están presentes en el ResultSet y que no tienen un mapeo manual se auto-mapearán.

* + 1. **Cache**

MyBatis incluye una funcionalidad de caché transaccional de segundo nivel muy potente que es ampliamente configurable y personalizable. Se han realizado muchos cambios en la caché de MyBatis 3 para hacer la a la vez más potente y más sencilla de configurar.

* 1. **SQL dinámico**

Entre las características más sobresalientes de MyBatis ha sido siempre sus capacidades de SQL dinámico. Si el desarrollador tiene habilidades o experiencias con JDBC o algún framework similar, sabrá que muchas de las veces es complicado concatenar strings de SQL. MyBatis de alguna forma mejora la situación con un lenguaje de SQL dinámico potente que puede usarse en cualquier mapped statement.

* 1. **Logging**

Mybatis proporciona información de logging mediante el uso de una librería a la cual se le transmite información de logging la cual se basa en la introspección en tiempo de ejecución de la propia librería de Mybatis, algunos de los entornos vienen con Commons Logging incluido como parte del framework o el classpath del servidor como es el caso de Tomcat y WebSphere.

* 1. **Ventajas**
* Con Mybatis es posible mapear fácilmente las consultas que sean necesarias para proyectos tipo empresarial, dado que la sintaxis está aislada en la lógica de negocio, lo cual es ventajoso en el sentido que facilita modificaciones futuras de la aplicación.
* Se ha observado una serie de técnicas que proporciona Mybatis para resolver problemas cotidianos cuando se trabaja en librerías de acceso a datos.
* MyBatis está más centrado en las consultas SQL y es apropiado en bases de datos que requieran consultas optimizadas o donde el modelo de tablas de la base de datos es fijo.
* Este framework se basa en open source, lo cual hasta cierto punto es ventajoso ya que deja la puerta abierta a futuras modificaciones por parte de los desarrolladores a nivel mundial.
  1. **Desventajas**
* Las bases de datos que se van a mapear con Mybatis deben ser exclusivamente relacionales.
* No es del todo transparente, puesto que aún hay que programar o escribir código SQL directamente en los archivos de mapeo(XML)
* Se pierde funcionalidad cuando la mayoría de las sentencias SQL son construidas dinámicamente.
* No tiene versionamientos frecuentemente rápidos como otras herramientas.

**CAPITULO III**

1. **ESTUDIO COMPARATIVO DE LOS FRAMEWORKS DE PERSISTENCIA HIBERNATE Y MYBATIS**
   1. **Introducción**

En la actualidad es posible encontrar una variedad de frameworks ORM en el mercado, algunos gratuitos y otros de pago, todos con muy buenas características, es por ello que la elección de un framework que cumpla con las necesidades y requerimientos que un desarrollador de software desea es de vital importancia y por tanto su selección una tarea difícil.

Entre los frameworks ORM usados por la comunidad de programadores en Java, según las evaluaciones de encuestas realizadas por el sitio web zeroturnaround.com[[4]](#footnote-4), en el año 2014 está liderando Hibernate seguido de los objetos planos de JDBC.

Es por ello que se ha elegido el framework Hibernate como primer candidato dado por la cantidad de características que presenta a su favor y la gran documentación del mismo. Luego ponemos a consideración el segundo candidato para la comparación, basándonos en su baja calificación según la encuesta de la web antes mencionada.

El objetivo de este análisis comparativo es determinar que parámetros nos servirán para comparar los dos frameworks de persistencia para el desarrollo de aplicaciones y establecer indicadores que faciliten la elección al momento de implementar un sistema informático.

* 1. **Definición de los parámetros de comparación**

La elección de las variable, parámetros, indicadores y porcentajes nos ayudara a tener una más clara y mejor idea de cómo se procederá hacer la comparación de los frameworks.

* + 1. **Variable Independiente**

A continuación se muestran los parámetros elegidos en la variable independiente y como cada uno de ellos aportara para el total acumulado del framework.

**Hipótesis**: “*La utilización del framework de persistencia en java Hibernate me permitirá mejorar la productividad que se destina a la abstracción de la capa de datos con respecto al framework Mybatis*”.

Variable independiente: ***framework de persistencia***

Tabla III. I Resumen de parámetros de la variable independiente

|  |  |  |
| --- | --- | --- |
| PARÁMETROS | INDICADORES | PORCENTAJE (%) |
| MAPEO OBJETO/RELACIONAL | Construcción de métodos CRUD | 55 |
| Creación automática de esquema | 15 |
| Dialectos usados por la herramienta | 30 |
| TOTAL | | 100 |
| PRODUCTO | Ultimo versionamiento | 25 |
| Modo de Licenciamiento | 25 |
| Disponibilidad de información | 50 |
| TOTAL | | 100 |

* **Mapeo Objeto/Relacional**:
  + *Construcción de métodos CRUD*: Se evalúa si la herramienta proporciona facilidad para generar/crear sus propios métodos para acceder a los datos. Para lo ponderación del 55% es debido a la gran importancia ya que este indicador es la base para el manejo de las operaciones hacia las bases de datos
  + *Creación automática de esquema*: Si el framework provee la funcionabilidad de generar automáticamente el esquema de la base de datos de la cual se mapean los objetos, la ponderación es este punto es bajo ya que se puede usar herramientas de terceros para proveer una mejor solución a esta fase.
  + *Dialectos utilizados por la herramienta*: Se refiere si el framework da soporte para los diferentes DBMS del mercado, es de un valor medio bajo a pesar de que importante saber las librerías usadas al final de la solución, esto es pasado de manera transparente para el desarrollador.
* **Producto**:
  + *Ultimo Versionamiento*: Este criterio es interesante debido a que al tener diferentes versiones indicaría la madurez y estabilidad del framework, su puntaje es acertado ya que nos indicara el desarrollo de la herramienta a lo largo del tiempo.
  + *Modo de Licenciamiento*: Nos permite apreciar de qué manera es distribuido y liberado el framework para poder ser utilizado, los resultados de este puntaje nos arrojaría como el uso de la herramienta afectaría en el financiamiento del proyecto de software
  + *Disponibilidad de Información*: Es la facilidad que tendrá el desarrollador de poder acceder a la información de las herramientas en estudio, el puntaje obtenido nos permitirá saber cuál de las herramientas tiene una mayor disponibilidad ya sea en medios digitales o escritos.
    1. **Variable Dependiente**

Los parámetros elegidos en la variable dependiente y como estos aportaran para el total acumulado del framework.

**Hipótesis**: “*La utilización del framework de persistencia en java Hibernate me permitirá mejorar la productividad que se destina a la abstracción de la capa de datos con respecto al framework Mybatis*”.

Variable dependiente: ***mejorar la productividad***

Tabla III. II Resumen de parámetros de la variable dependiente

|  |  |  |
| --- | --- | --- |
| PARÁMETROS | INDICADORES | PORCENTAJE (%) |
| RENDIMIENTO | Tiempo de respuesta de consultas SQL | 45 |
| Uso del CPU | 25 |
| Uso de la Memoria RAM | 30 |
| TOTAL | | 100 |
| DESARROLLO | Líneas de código generadas | 45 |
| Tiempo desarrollo operaciones CRUD | 35 |
| Integración con otros Frameworks | 20 |
| TOTAL | | 100 |

* **Rendimiento**
  + *Tiempo de respuesta de consultas SQL*: Es el lapso de tiempo que transcurrirá desde que el usuario realiza la petición hasta que esta es procesada en el servidor por el prototipo (aplicación), El puntaje dado al indicador es alto debido a que nos puede mostrar la capacidad de procesar la información de la herramienta
  + *Uso del CPU*: El porcentaje de uso del procesador de la maquina en donde se ejecuta el prototipo (aplicación), los valores resultantes nos ayudara indicando si en algún momento los procesos se quedarían redundando en el CPU.
  + *Uso de la Memoria RAM*: La cantidad de memoria usada para realizar las consultas enviadas por prototipo (aplicación), el puntaje dado es medio bajo y nos ayudaría a revisar el consumo de memoria por parte de la herramienta.
* **Desarrollo**
  + *Líneas de código generadas*: Número de líneas de código que se genera al momento de aplicar ORM a la capa de abstracción de datos del prototipo y evaluar cual genera menos líneas de código, un indicador alto ya que nos permite el esfuerzo de programación por parte del desarrollador.
  + *Tiempo desarrollo operaciones CRUD*: Estas operaciones nos permite mostrar el tiempo necesario para el prototipo de realizar las tareas de creación, lectura, actualización y eliminación, Este indicador medio alto nos daría indicios de que herramienta nos mejoraría los tiempos de desarrollo
  + *Integración con otros frameworks*: Esto nos indica si el framework es compatible con otras herramientas y ampliar sus capacidades aún más, El puntaje aquí dado es bajo ya que hasta cierto punto es irrelevante saber cómo se integraría la herramienta con otras, ya que el desarrollador podría estar solo enfocado al backend y no al frontend de la solución.
  1. **Criterios de Evaluación**

Aquí estableceremos la escala de medición para poder determinar los porcentajes, que nos llevara a seleccionar el framework adecuado para el desarrollo de la aplicación. A continuación se muestra la tabla con las interpretaciones que se le otorgara a los frameworks.

Tabla III. III Pesos de Valoración Cualitativa y Cuantitativa

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cuantitativa | 0 | 1 | 2 | 3 |
| Porcentaje (%) | < 25 | >= 25 y < 50 | >= 50 y < 75 | >= 75 |
| Cualitativa | Malo | Regular | Bueno | Muy Bueno |
| Muy difícil | Difícil | Fácil | Muy Fácil |
| Muy poco | Algo | Bastante | Mucho |
| SI |  |  | No |

* 1. **Estudio comparativo de los Frameworks de Persistencia Hibernate y Mybatis**

El estudio comparativo de los frameworks de persistencia Hibernate y Mybatis, que se desarrollara en adelante, será a través de la observación, experimentación, recopilación de información que se pudiera obtener de revisiones bibliográficas de textos, documentos y publicaciones tanto online como offline.

* + 1. **Mapeo Objeto / Relacional**

Con este parámetro se intenta analizar los frameworks de estudio en cómo aprovechar las capacidades de la técnica de mapeo de las tablas que se convierten en objetos para poder aprovechar el paradigma de la programación de objetos.

* + - 1. **Construcción de métodos CRUD**

La construcción de métodos CRUD es importante en el desarrollo de la solución ya que ello nos permite acceder a las operaciones básicas que se realizan en las bases de datos, como los son la creación, lectura, actualización y eliminación de datos.

Con respecto al framework Hibernate tenemos que la construcción manual del código para generar CRUD, es aceptable siendo no muy invasivo puesto que usa su propio lenguaje de consulta HQL para el manejo de operaciones CRUD, lo que lleva al desarrollador a liberarse de cierta manera de escribir sentencias SQL directamente.

Por esta gran ayuda que ofrece el framework al desarrollador se le ha considerado dar una puntuación alta de tres, equivalente a Muy bueno, según la Tabla III. III Pesos de Valoración Cualitativa y Cuantitativa**.**

La valoración del framework Mybatis para este literal se ha considerado el valor de dos, equivalente a Bueno según la Tabla III.III Pesos de Valoración Cualitativa y Cuantitativa, esto en vista que para generar CRUD el framework tiene que usar librerías de terceros, además de que se usa lenguaje SQL directamente en todo tipo de operación con las tablas de las bases de datos.

El valor máximo es 3 (ver Anexo A – Sección 2), el cual correspondería al 55% de un total del 100%, que nos indicaría cuál de los frameworks ofrece más facilidades para la construcción de métodos CRUD sin que esto ocupe tiempo innecesario al desarrollador, para tener una mejor apreciación de los valores se aplicó una regla de tres simple (ver Anexo A – Sección 2), el resultado final para este indicador está presente en la tabla:

Tabla III. IV Construcción de métodos CRUD.

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Construcción de métodos de CRUD | 60% | 40% |
| Valor Cualitativo | Bueno | Regular |

* + - 1. **Creación automática de esquema**

Determinamos si las herramientas ofrecen su propio generador de mapeos del esquema de la base de datos o si se usan herramientas de terceros para poder lograr este objetivo.

El framework Hibernate ofrece su propia herramienta denominada **hbm2ddl** para generación de esquemas desde la base de datos que vayamos a usar, se lo puede configurar para seguir ciertas normas que en algunos casos el cliente pudiera requerir para la solución. Es por esta facilidad de tener herramientas propias que se le da una puntuación alta de tres, equivalente a Muy bueno, según la Tabla III.III Pesos de Valoración Cualitativa y Cuantitativa.

El framework Mybatis utiliza herramientas de terceros como es el Mybatis Generator para poder proporcionar una generación automática del esquema, Por ello se la da una puntuación de dos, equivalente a Bueno, basándonos en la Tabla III.III Pesos de Valoración Cualitativa y Cuantitativa.

El valor máximo es 3 (ver Anexo A – Sección 2), el cual correspondería al 15% de un total del 100%, que nos indicaría cuál de los frameworks ofrece más facilidades para una configuración y creación rápida del esquema sin que esto ocupe tiempo innecesario al desarrollador, para tener una mejor apreciación de los valores se aplicó una regla de tres simple (ver Anexo A – Sección 2), el resultado final para este indicador está presente en la tabla:

Tabla III. V Resultados para la Creación automática de esquemas

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Creación automáticas del esquema de Bases de datos | 60% | 40% |
| Valor Cualitativo | Bueno | Regular |

* + - 1. **Dialectos utilizados por la herramienta**

Se refiere si el framework da soporte para los diferentes DBMS del mercado, es de un valor medio bajo a pesar de que es importante saber las librerías para el soporte, al final de la solución esto es pasado de manera transparente para el desarrollador.

Tabla III. VI Librerías para DBMS soportados por Hibernate

|  |  |
| --- | --- |
| RDBMS | Dialecto |
| DB2 | org.Hibernate.dialect.DB2Dialect |
| DB2 AS/400 | org.Hibernate.dialect.DB2400Dialect |
| DB2 OS390 | org.Hibernate.dialect.DB23900Dialect |
| postgreSQL | org.hibernate.dialect.PostgreSQLDialect |
| MySQL5 | org.Hibernate.dialect.MySQL5Dialect |
| MySQl5 con InnoDB | org.Hibernate.dialect.MySQL5InnoDBDialect |
| MySQL con MyISAM | org.Hibernate.dialect.MySQLMyISAMDialect |
| Oracle (cualquier versión) | org.Hibernate.dialect.OracleDialect |
| Oracle 9i | org.Hibernate.dialect.Oracle9iDialect |
| Oracle 10g | org.Hibernate.dialect.Oracle10gDialect |
| Oracle 11g | org.Hibernate.dialect.Oracle11Dialect |
| Sysbase | org.Hibernate.dialect.SybaseASE15Dialect |
| Sysbase Anywhere | org.Hibernate.dialect.SybaseAnywhereDialect |
| Microsoft SQL Server 2000 | org.Hibernate.dialect.SQLServerDialect |
| Microsoft SQL Server 2005 | org.Hibernate.dialect.SQLServer2005Dialect |
| Microsoft SQL Server 2008 | org.Hibernate.dialect.SQLServer2008Dialect |
| SAP DB | org.Hibernate.dialect.SAPDBDialect |
| Informix | org.Hibernate.dialect.InformixDialect |
| HypersonicSQL | org.Hibernate.dialect.HSQLDialect |
| Ingres | org.Hibernate.dialect.IngresDialect |
| Progress | org.Hibernate.dialect.ProgressDialect |
| Mckoi SQL | org.Hibernate.dialect.MckoiDialect |
| Interbase | org.Hibernate.dialect.InterbaseDialect |
| Pointbase | org.Hibernate.dialect.PointbaseDialect |
| FrontBase | org.Hibernate.dialect.FrontbaseDialect |
| Firebird | org.Hibernate.dialect.FirebirdDialect |
| Total de librerías DMBS Soportadas | 26 |

*Fuente: https://docs.jboss.org/hibernate/orm/3.5/reference/es-ES/html/session-configuration.html#configuration-optional-dialects* Tabla Mybatis

El framework MyBatis tiene entre sus características la simplicidad y su infinita flexibilidad que permite usar cualquier base de datos por mal diseñada que esté usar el framework sin demasiados problemas.

A diferencia de JPA o Hibernate, MyBatis no mapea un modelo relacional a un modelo de objetos. No es obligatorio describir las relaciones de las tablas en ficheros XML ni utilizar dialectos del SQL para solucionar muchas de las veces relaciones imposibles. Con mención a que Mybatis no utiliza librerías para dialectos con las bases de datos, este framework tendría cero librerías.

El valor máximo es 26 (ver Anexo A – Sección 3), el cual correspondería al 30% de un total del 100%, que son la cantidad de librerías que dan soporte a los diferentes DBMS que se encuentran en el mercado, para tener una mejor apreciación de los valores se aplicó una regla de tres simple (ver Anexo A – Sección 3), el resultado final para este indicador está presente en la siguiente tabla:

Tabla III. VII Resultados de la Evaluación de librerías para soporte a DBMS

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Librerías para soporte de los DBMS | 100,00% | 0,00% |
| Valor Cualitativo | Muy Bueno | Malo |

Como resultado final para el parámetro lo que es el Mapeo Objeto/Relacional, tenemos el detalle en la siguiente tabla de resultados obtenidos en cada uno de sus indicadores para los framework de estudio.

Tabla III. VIII Resultados del Parámetro Mapeo Objeto/Relacional

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Construcción de métodos de CRUD | 33,00% | 22,00% |
| Creación automáticas del esquema de Bases de datos | 9,00% | 6,00% |
| Librerías para soporte de los DBMS | 30,00% | 0,00% |
| TOTAL | 72,00 % | 28,00% |
| Valor Cualitativo | Bueno | Regular |

Los resultados nos muestran que el framework Hibernate tiene una gran ventaja con las librerías de soporte para los diferentes DBMS, al igual que la facilidad de creación de los esquemas de las bases de datos y su posterior creación de los métodos para acceder a los datos de las tablas, liberando hasta cierto punto al desarrollador de la lógica de datos.

El framework Mybatis por lo contrario muestra muy pocas librerías para soportar los diferentes DBMS dado que al tener que escribir código del lenguaje SQL directamente para poder acceder a los datos presentes en las tablas, además de que al poder generar CRUD de manera automática y los esquemas de las bases de datos pero con herramientas de terceros ello implica dedicarle más tiempo en configuraciones y esfuerzo para el desarrollador.

Como podremos apreciar en la figura a continuación el framework Hibernate supera ampliamente a Mybatis, en los resultados finales para el estudio del parámetro de Mapeo Objeto/Relacional.

Figura III. 1 Análisis del parámetro Mapeo Objeto/Relacional

* + 1. **Producto**

Estudiaremos como la herramienta ha progresado a lo largo del tiempo y bajo qué condiciones o licenciamientos esta es liberada para la comunidad de desarrolladores, a su vez también si la información para aprender o solventar inquietudes es bastante amplia o nula.

* + - 1. **Ultimo versionamiento**

Se apreciaría como las herramientas en estudio, tiene su madurez, dado que siempre se encuentra bugs o errores y son solventados con nuevas versiones dando una mayor seguridad y estabilidad a la herramienta. Para la evaluación se ha tomado un rango de fechas determinado en las que se lanzaron los versionamientos en las respectivas webs de las herramientas, como es el caso desde 01/01/2014 hasta 31/05/2015. Las siguientes tablas muestran los versionamientos de los frameworks Hibernate y Mybatis.

Tabla III. IX Versionamiento de Hibernate

|  |  |  |
| --- | --- | --- |
| Versión | Fecha | Estado |
| 5.0.0.CR1 | 2015/05/27 | Desarrollo |
| 4.3.10.Final | 2015/05/14 | Estable |
| 4.3.9.Final | 2015/04/15 | Estable |
| 4.3.8.Final | 2015/01/06 | Estable |
| 4.3.7.Final | 2014/10/30 | Estable |
| 4.3.6.Final | 2014/07/17 | Estable |
| 4.3.5.Final | 2014/04/02 | Estable |
| 4.3.4.Final | 2014/03/03 | Estable |
| 4.3.3.Final | 2014/02/28 | Estable |
| 4.3.2.Final | 2014/02/27 | Estable |
| 4.3.1.Final | 2014/01/22 | Estable |
| TOTAL Versionamientos |  | 11 |

Tabla III. X Versionamiento de Mybatis

|  |  |  |
| --- | --- | --- |
| Versión | Fecha | Estado |
| 3.3.0 | 2015/05/23 | Estable |
| 3.2.8 | 2014/10/10 | Estable |
| 3.2.7 | 2014/04/06 | Estable |
| 3.2.6 | 2014/03/21 | Estable |
| 3.2.5 | 2014/02/14 | Estable |
| 3.2.4 | 2014/01/15 | Estable |
| TOTAL Versionamientos |  | 6 |

El valor máximo es 17 (ver Anexo A – Sección 4), el cual correspondería al 25% de un total del 100%, que son los versionamientos en las fechas indicadas anteriormente, para tener una mejor apreciación de los valores se aplicó una regla de tres simple (ver Anexo A – Sección 4), el resultado final para este indicador está presente en la siguiente tabla:

Tabla III. XI Resultados de la Evaluación de versionamientos

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Últimos versionamientos | 64,71% | 35,29% |
| Valor Cualitativo | Bastante | Algo |

* + - 1. **Modo de Licenciamiento**

Se determinara que tipos de licencia aplican los frameworks de persistencia en estudio para su posterior distribución y liberación a la comunidad de desarrolladores.

En Hibernate tenemos dos tipos de licenciamiento:

* LGPL v 2.1 (Lesser General Public License)[[5]](#footnote-5), Licencia Pública General Reducida
* ASL 2.0 (Apache Software License), Licencia de Software Apache

Para Mybatis tenemos una sola licencia para su distribución que es la ASL 2.0[[6]](#footnote-6)

Para el cálculo de este indicador (ver Anexo A – Sección 5), se aplicó una regla de tres simple, dando como resultados para este indicador, detallados en la siguiente tabla:

Tabla III. XII Resultados del Modo de Licenciamiento

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Licenciamiento | 66,67% | 33,33% |
| Valor Cualitativo | Bueno | Regular |

* + - 1. **Disponibilidad de Información.**

Para poder determinar la complejidad o simplicidad de la herramienta, acerca del uso del framework y sus componentes, se trata de demostrar si las herramientas que son parte de la solución del proyecto abarcaran la información necesaria para poder solventar inquietudes durante su uso y estudio.

Para la evaluación se ha tomado la cantidad de fuentes de información donde se ha obtenido un promedio, que corresponde a la búsqueda en español, la web en general y libros de venta en Amazon.com[[7]](#footnote-7) sección de libros y Google books, determinando así la valoración máxima como el resultado más alto. La tabla a continuación muestra los resultados de las búsquedas.

Tabla III. XIII Resultados de búsqueda en disponibilidad de información

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Búsqueda general en la Web | Búsqueda en Español | Libros en Amazon.com | Google books | Promedio |
| Hibernate | 20’500.000 en 0.18 segundos | 321.000 en 0.17 segundos | 314 en 0.16 segundos | 3.130 en 0.47 segundos | 5’206.111 |
| Mybatis | 856.000 en 0.16 segundos | 10.400 en 0.20 segundos | 4 en 0.17 segundos | 45 en 0.35 segundos | 216.612,25 |
| TOTAL | 21’356.000 | 331.400 | 318 | 3175 | 5’422.723,35 |

El valor máximo es 5’422.723,55 (ver Anexo A – Sección 6), el cual correspondería al 50% de un total del 100%, que es la información disponible, no obstante para tener una mejor apreciación de los valores para Hibernate y Mybatis se aplicó una regla de tres simple (ver Anexo A – Sección 6), el resultado final para este indicador está presente en la siguiente tabla:

Tabla III. XIV Análisis del criterio de Disponibilidad de Información

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Disponibilidad de Información | 96,01% | 3,99% |
| Valor Cualitativo | Mucho | Muy poco |

Como resultado final para el parámetro Producto, en la tabla siguiente se detalla los resultados obtenidos en cada uno de sus indicadores para los framework de estudio.

Tabla III. XV Resultados del parámetro Producto

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Ultimo Versionamiento | 16,67% | 8,33% |
| Modo de Licenciamiento | 16,18% | 8,82% |
| Disponibilidad de información | 48,00 % | 2,00% |
| TOTAL | 80,85% | 19,15% |
| Valor Cualitativo | Muy Bueno | Malo |

Los resultados nos indican que el framework Hibernate tiene una gran disponibilidad de documentación tanto en su página web oficial como otros sitios de internet, facilitando a los desarrolladores el aprendizaje de la herramienta, los diferentes versionamientos en un periodo de tiempo determinado nos indica la evolución y madurez que tiene Hibernate por parte de la comunidad, además sus dos tipos de licenciamiento.

El framework Mybatis presenta en sus resultados de evaluación la muy poca disponibilidad de información, complicando de alguna manera el aprendizaje de la herramienta, cuenta con muy pocos versionamientos demostrando la poca evolución y soporte por parte de la comunidad.

Como podemos ver en la figura a continuación el framework Hibernate supera ampliamente a Mybatis, en los resultados finales para el estudio del parámetro producto.

Figura III. 2 Análisis del Parámetro Producto

* + 1. **Rendimiento**

Para la interpretación de cada indicador de este parámetro, se hizo pruebas en los dos prototipos uno creado con la herramienta Hibernate y el otro con Mybatis, estas pruebas de carga, afectación de registros, uso de memoria y procesador al momento en que se realizan peticiones o consultas a la base de datos. Las dos aplicaciones se probaron en una misma computadora con las características mostradas en la siguiente tabla.

Tabla III. XVI Características del equipo para las pruebas del prototipo.

|  |  |  |
| --- | --- | --- |
| Framework de Persistencia | Hibernate | Mybatis |
| Microprocesador | Core i3 – 2.13GHz | Core i3 – 2.13GHz |
| Memoria RAM | 4 GB | 4 GB |
| Sistema Operativo | Windows 7 Professional 64 bits | Windows 7 Professional 64 bits |
| IDE | Netbeans 8.0 | Netbeans 8.0 |
| Servidor de aplicaciones | Apache Tomcat 8.0.3 | Apache Tomcat 8.0.3 |
| Motor de Bases de datos | PostgreSQL 9.1 | PostgreSQL 9.1 |
| Versión del framework | 4.3.5.Final | 3.2.7 |

* + - 1. **Tiempo de respuesta de consultas SQL.**

Este indicador podremos medir el tiempo de respuesta (milisegundos) que tomo procesar la petición o peticiones de consultas (query), para ello se ha enviado diferentes cantidades de peticiones (ver Anexo A – Sección 7) y se hizo una auditoria en la base de datos (ver Anexo B – Sección 1) del prototipo empleado para el estudio de los frameworks.

Los resultados de este indicador muestran pequeñas diferencias con respecto al manejo de tiempo para proceder con las peticiones de consulta, puesto que el framework Hibernate presenta un porcentaje ligeramente menor con respecto a Mybatis.

El valor máximo es 2.540792911 (ver Anexo A – Sección 7), el cual correspondería al 45% de un total del 100%, no obstante para tener una mejor apreciación de los valores para Hibernate y Mybatis se aplicó una regla de tres simple (ver Anexo A – Sección 7), el resultado final para este indicador está presente en la siguiente tabla.

Tabla III. XVII Resultados del indicador Tiempo de respuestas en consultas SQL

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Tiempo de respuesta de consultas SQL | 49,44% | 50,56% |
| Valor Cualitativo | Poco | Mucho |

* + - 1. **Uso del CPU.**

Se podrá ver la capacidad del CPU (en porcentaje) del servidor para soportar las diferentes cargar de peticiones que le son enviadas y estas deben ser procesadas, para ello se ha enviado diferentes cantidades de peticiones (ver Anexo B – Sección 2).

El valor máximo es 96.33% (ver Anexo A – Sección 8), el cual correspondería al 25% de un total del 100%, no obstante para tener una mejor apreciación de los valores para Hibernate y Mybatis se aplicó una regla de tres simple (ver Anexo A – Sección 8), el resultado final para este indicador está presente en la siguiente tabla.

Tabla III. XVIII Resultados del indicador Uso del CPU

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Uso del CPU | 49,10% | 50,90% |
| Valor Cualitativo | Poco | Mucho |

* + - 1. **Uso de la Memoria RAM.**

Se podrá ver la cantidad de memoria RAM que es ocupada por el servidor al momento de cargar a memoria las peticiones para que estas sean atendidas por el administrador de bases de datos (ver Anexo B – Sección 3).

El valor máximo del promedio de los dos frameworks obtenidos es 124.50 (ver Anexo A – Sección 9), el cual correspondería al 30% de un total del 100%, no obstante para tener una mejor apreciación de los valores para Hibernate y Mybatis se aplicó una regla de tres simple (ver Anexo A – Sección 9), el resultado final para este indicador está presente en la siguiente tabla.

Tabla III. XIX Resultados del indicador Uso de la memoria RAM

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Uso de la memoria RAM | 46,05% | 53,95% |
| Valor Cualitativo | Poco | Mucho |

Como resultado final para el parámetro Desarrollo se detalla en la siguiente tabla los resultados obtenidos en cada uno de sus indicadores para los framework de estudio.

Tabla III. XX Resultados del Análisis del Parámetro Rendimiento

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Tiempo de respuesta de consultas SQL | 22,25 % | 22,75 % |
| Uso del CPU | 12,28 % | 12,72 % |
| Uso de la memoria RAM | 13,82 % | 16,18 % |
| TOTAL | 48,35 % | 51,65 % |
| Valor Cualitativo | Poco | Mucho |

Después de los datos obtenidos con los indicadores del parámetro Rendimiento, tenemos que Hibernate ayuda de cierta manera aprovechar los recursos hardware evaluados para un mejor desempeño del servidor en comparación con Mybatis a pesar de tener una pequeña diferencia en los resultados, esto podría verse afectado en el futuro a medida que pudieran aumentar los usuarios o nuevos requerimientos para el prototipo. En la figura a continuación podemos apreciar los resultados de los framework Hibernate y Mybatis.

Figura III. 3 Análisis del Parámetro Rendimiento

* + 1. **Desarrollo.**

Con este parámetro se pretende llegar a saber cómo podríamos mejorar la productividad al momento de desarrollar, donde los diferentes indicadores nos mostraran cuál de los frameworks de estudio mejora el desempeño y reduciría los tiempos para tener una solución viable y efectiva.

* + - 1. **Líneas de código generadas**

La cantidad de líneas de código generada por una aplicación varían según el lenguaje y herramientas adicionales que se pudieran integrar para facilitar la solución por parte del desarrollador, pero cuando se está dentro de un mismo lenguaje este indicador nos permitirá medir las líneas de código dentro de un determinado proceso, con Hibernate o Mybatis con el fin de validar cuál de ellos genera menos líneas de código, al momento de mapear el esquema de la base de datos y su posterior creación de los métodos CRUD necesarios para acceder a los datos.

La cantidad de líneas de código que se genera con Hibernate es menor con relación a Mybatis, dado que este último necesita de configuraciones y librerías de terceros para mapear y realizar los CRUD automáticamente, además de las sentencias SQL para poder trabajar con los datos.

El valor máximo es 18050 (ver Anexo A – Sección 10), el cual correspondería al 45% de un total del 100%, no obstante para tener una mejor apreciación de los valores para Hibernate y Mybatis se aplicó una regla de tres simple (ver Anexo A – Sección 10), el resultado final para este indicador está presente en la siguiente tabla.

Tabla III. XXI Resultados del indicador Líneas de código generadas

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Líneas de código generadas | 47,36% | 52,63% |
| Valor Cualitativo | Poco | Mucho |

.

* + - 1. **Tiempo desarrollo operaciones CRUD**

Con este indicador demostraremos los tiempos que le tomara al desarrollador implementar los diferentes métodos CRUD para el acceso de los datos en las bases de datos, esto se lo realizara a través de técnicas de Observación y Experimentación.

Usando el framework Hibernate y aplicando la técnica antes descrita, la realización de todos los métodos CRUD necesarios para que la aplicación funcione correctamente, se invirtió un tiempo de 24 horas, no es el caso para el framework Mybatis que se invirtió un tiempo de 40 horas.

El valor máximo de horas es 22 (ver Anexo A – Sección 11), el cual correspondería al 35% de un total del 100%, no obstante para tener una mejor apreciación de los valores para Hibernate y Mybatis se aplicó una regla de tres simple (ver Anexo A – Sección 11), el resultado final para este indicador está presente en la siguiente tabla.

Tabla III. XXII Resultado del indicador Tiempo desarrollo operaciones CRUD

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Tiempo desarrollo operaciones CRUD | 37,50% | 62,50% |
| Valor Cualitativo | Poco | Mucho |

* + - 1. **Integración con otros frameworks**

Esta variable lo que pretende demostrarnos es como los frameworks que estamos evaluado se integran o no fácilmente con otros framework de interfaces, modelos vista controlador, etc., También se puede decir que el porcentaje para esta variable es bajo en vista que el desarrollador pudiera estar más interesado en el backend y no el frontend de la solución.

El framework Hibernate es bien conocido por su buena integración con los framework Struts, Spring, y las variantes de JSF (Java Server Faces) como los es Primefaces, Richfaces, etc., es por ello que se le otorga un punta de tres que es el más alto puntaje, según la Tabla III.III Pesos de Valoración Cualitativa y Cuantitativa.

Respecto al framework Mybatis se puede mencionar que la integración con otros frameworks es muy baja, según lo analizado y consultado en las páginas web de la Internet, dado que no se hallan muchos datos en este aspecto, pero si hace integración con esta herramienta como lo es Spring, por ello el puntaje otorgado es de uno que equivaldría a Regular, según la Tabla III.III Pesos de Valoración Cualitativa y Cuantitativa.

El valor máximo es 4 (ver Anexo A – Sección 12), el cual correspondería al 20% de un total del 100%, que es la capacidad de integración del framework, no obstante para tener una mejor apreciación de los valores para Hibernate y Mybatis se aplicó una regla de tres simple (ver Anexo A – Sección 12), el resultado final para este indicador está presente en la siguiente tabla.

Tabla III. XXIII Resultado del indicador Integración con otros frameworks

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Integración con otros frameworks | 75,00% | 25,00% |
| Valor Cualitativo | Muy bueno | Regular |

Como resultado final para el parámetro Desarrollo se detalla en la siguiente tabla los resultados obtenidos en cada uno de sus indicadores para los framework de estudio.

Tabla III. XXIV Resultados del Análisis del parámetro Desarrollo

|  |  |  |
| --- | --- | --- |
|  | Hibernate | Mybatis |
| Líneas de Código generadas | 21,32 % | 23,68 % |
| Tiempo desarrollo operaciones CRUD | 13,12 % | 21,88 % |
| Integración con otros frameworks | 15,00 % | 5,00 % |
| TOTAL | 49,44 % | 50,56 % |
| Valor Cualitativo | Poco | Mucho |

Después de las estimaciones y con los resultados obtenidos, podemos apreciar que el framework Hibernate ayuda al desarrollador reduciéndole el esfuerzo al momento de desarrollar las diferentes tareas que tenga que solventarse con el aplicativo, puesto que le genera menos líneas de código y se integra fácilmente con otros frameworks, no así el framework Mybatis que aumenta el esfuerzo, al codificar más líneas de código para que la aplicación pueda llegar a ser factible, además de la poca integración con otras herramientas para un mejor desempeño.

Como podemos ver en la figura a continuación el framework Hibernate tiene un porcentaje menor que el framework Mybatis, que nos indica que la herramienta en realidad reduce en tiempo y esfuerzo, para que el desarrollador tenga un mejor desempeño al dar soluciones cuando se implementa aplicaciones web.

Figura III. 4 Análisis del parámetro Desarrollo

En la Figura III. 5 podemos ver como el framework Hibernate supera ampliamente con un valor de 62,39% a Mybatis que solo obtuvo 37,61%, lo cual nos indica que el desarrollo de la aplicación web para el Sistema de Evaluación Docente del IPEC se lo debe realizar en con la ayuda de este framework.

Figura III. 5 Resultado final del Análisis de frameworks

* 1. **Comprobación de la Hipótesis**

La hipótesis de la tesis de investigación está formada por variables correlacionales, este trabajo investigativo corresponde a distribuciones no paramétricas, por lo que se ha seleccionado chi cuadrado como método de investigación para poder llegar a su demostración, se consideraron los siguientes pasos:

* Planteamiento de la hipótesis
* Establecimiento del nivel de significancia
* Calculo del estadístico
* Grados de libertad
* Criterio de decisión.
  + 1. **Planteamiento de la hipótesis**

Para poder demostrar la hipótesis se establece, la hipótesis nula (H0) y alterna (H1), encontrándose H0 dentro del área de aceptación y H1 en el área de rechazo. Por lo tanto tenemos que:

**H0**: “*La utilización del framework de persistencia en java Hibernate no me permitirá mejorar la productividad que se destina a la abstracción de la capa de datos con respecto al framework Mybatis*”

**H1**: “*La utilización del framework de persistencia en java Hibernate me permitirá mejorar la productividad que se destina a la abstracción de la capa de datos con respecto al framework Mybatis*”

* + 1. **Establecimiento del nivel de significancia.**

Una vez establecido la hipótesis nula y alternativa, se procede a determinar el nivel de significancia, para el presente estudio comparativo se utilizara un nivel de significancia estadística de , que nos ayudara a rechazar la hipótesis del estudio comparativo que no permitirá la utilización del framework Hibernate para mejorar la productividad con respecto al framework Mybatis.

* + 1. **Calculo del estadístico.**

En el presente trabajo de investigación como se mencionó anteriormente hemos seleccionado el estadístico **chi** **cuadrado** para probar la hipótesis, la fórmula de este estadístico es como sigue:

Pero antes de poder aplicar la ecuación anterior hay que determinar las frecuencias observadas y esperadas (ver Anexo A – Sección 13), donde para calcular el valor de las frecuencias esperadas usamos la siguiente fórmula:

Luego de aplicar la ecuación del estadístico chi cuadrado (ver Anexo A – Sección 14) tenemos el valor de x2 = 18,61.

* + 1. **Grados de libertad**

Para determinar los grados de libertad (**gl**), el cual está en función del número de filas(**r**) y el número de columnas (**k**), tenemos la siguiente ecuación:

Por lo tanto:

Según la tabla estadística de la distribución chi cuadrado (ver Anexo A – Sección 15 Comprobación de la Hipótesis.- Tabla de distribución de chi cuadrado), donde al tener un nivel de significancia de 0,05 a 23 *gl*, obtenemos un valor de 35,17.

* + 1. **Criterio de decisión.**

Determinando el criterio de decisión, donde se aceptara H0 cuando x2 calculado es menor que x2 de la tabla, caso contrario se rechazará H0.

Donde el valor de x2 de la tabla viene representado por el valor obtenido de la tabla de distribución chi cuadrado (ver Anexo A – Sección 15 Comprobación de la Hipótesis.- Tabla de distribución de chi cuadrado) y x2 calculado viene representado por las frecuencias observadas y esperadas.

Por lo tanto volviendo al criterio de decisión tenemos que se rechazará H0 y se aceptará H1 , si el valor de x2 calculado es menor que x2 de la tabla, por lo contrario si el valor x2 calculado es igual o mayor al valor x2 de la tabla, se aceptará H1 y se rechazará H0 , entonces se tiene x2 calculado = 18,61 < x2 de la tabla = 35,17; por lo tanto se rechaza H0 y se acepta H1 que es “***La utilización del framework de persistencia en java Hibernate no me permitirá mejorar la productividad que se destina a la abstracción de la capa de datos con respecto al framework Mybatis***”

**CAPITULO IV**

1. **IMPLEMENTACION DEL SISTEMA DE EVALUACION AL DOCENTE EN EL IPEC**
   1. **Introducción.**

El desarrollo de una aplicación va de la mano con una serie de etapas y procesos que se deben seguir, por ello se hace necesario seguir una metodología adecuada a cada tipo solución, es por ello que para el análisis, diseño y desarrollo del Sistema de Evaluación Docente del IPEC, se ha tomado la metodología Microsoft Solution Framework (MSF), en la cual se encuentra los procesos de visión, planificación, desarrollo, estabilización y despliegue.

* 1. **Fase I: Visión y Alcances**

En esta fase se define los requerimientos del negocio, donde se tiene la intervención del cliente con el equipo de trabajo o analista encargado de recolectar los requisitos para el aplicativo, es de vital importancia puesto que también se definen los objetivos generales del proyecto.

* + 1. **Definición del Problema**

El Instituto de Postgrado y Educación Continua (IPEC), siguiendo sus objetivos de Alcanzar la excelencia académica y de ajustar a los nuevos lineamientos y reglamentos del Consejo de Educación Superior (CONSEJO DE EDUCACIÓN SUPERIOR, 2012) , donde se especifican los ámbitos, objetos, instrumentos, garantías, actores, componentes y ponderaciones de evaluación, para poder así lograr alcanzar el objetivo antes mencionado requiere de un sistema informático para cubrir dichos lineamientos.

* + 1. **Visión del Proyecto**

El sistema de evaluación al docente pretende ser una aplicación web estable y robusta que ofrezca servicios de calidad, donde a través de la configuración de ciertos parámetros se pretende evaluar al docente emitiendo de esta manera un resultado de desempeño académico mediante el cual el evaluado le permitirá mejorar las áreas académicas donde tuviera problemas.

* + 1. **Metas**

El sistema de evaluación al Docente para el Instituto de Postgrado y Educación Continua de la ESPOCH, se consolidara la información dentro de una base de datos sólida y robusta, para tener un mejor control de acceso de los usuarios del sistema basándose en mecanismos de validación de credenciales y posterior habilitar las opciones y permisos que dicho usuario le estén permitidas, de esta forma se manejara la información de manera eficiente y automatizada donde se pueda ofrecer resultados oportunos para una mejor toma de decisiones.

* + 1. **Perfiles de Usuario**

Para determinar los potenciales usuarios del aplicativo se deben identificar adecuadamente hacia quien o quienes va orientada la aplicación para asi alcanzar las metas propuestas y la funcionalidad del aplicativo satisfaga a los mismos. Los usuarios identificados del Sistema de Evaluación al Docente están definidos en la Tabla IV.I Usuarios del Sistema.

Tabla IV. 1 Usuarios identificados del Sistema

|  |  |
| --- | --- |
| **USUARIO** | **RESPONSABLE/ES** |
| Administrador | Delegado por el Director del IPEC, tiene control total del sistema. |
| Docente | Es aquel que realiza la encuesta de autoevaluación |
| Estudiante | Es aquel que realiza la encuesta al Docente |
| Coordinador | Es aquel que realiza la encuesta al Docente |
| Director | Tiene control total del sistema |

* + 1. **Ámbito del Proyecto**

El sistema de Evaluación al Docente del IPEC, esta enfocado en el almacenamiento de información personal, preguntas y respuestas, además de enfocarse en los resultados de la evaluación para una toma de decisiones por parte de los entes encargados del IPEC

* + 1. **Concepto de la Solución**

La solución que se presenta en la Figura IV.1 nos indica como el Sistema de Evaluación al Docente es una aplicación web la cual se desarrolla con el framework de persistencia Hibernate y java server faces (JSF).



Figura IV. 1 Concepto de Solución

La solución está alojada en un servidor de aplicaciones Apache Tomcat conjuntamente con el motor de bases de datos PostgreSQL, adicional trabaja con la base de datos del Sistema Académico del Instituto de Postgrado y Educación Continua SISEPEC por medio de servicios WEB para la extracción de información de los docentes y estudiantes registrados en este sistema.

* + - 1. **Software a utilizar**

El software que se usará para realizar el desarrollo del Sistema de Evaluación al Docente, de lista a continuación:

* ArgoUML: herramienta para modelado de objetos
* Microsoft Project: herramienta de planificación de procesos
* Netbeans 8..0: IDE para desarrollo de aplicaciones
* JDK 1.7: kit de desarrollo
* Tomcat: servidor de aplicaciones web basadas en java server pages (JSP)
* PostgreSQL 9.3: motor de base de datos
* Ireport/JasperReport: herramienta para realizar reportes
* Toad Data Modeler: herramienta modeladora del diseño lógico y físico de la base de datos.
  + - 1. **Arquitectura**

El Sistema de Evaluación Docente del IPEC se implementa bajo cuatro capas, misma que se pueden apreciar en la Figura IV.2 y a continuación se describe cada una de ellas

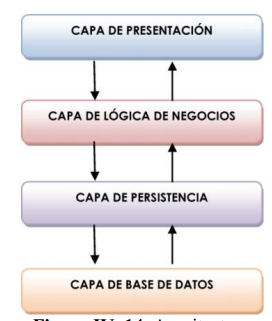


Figura IV. 2 Arquitectura del Sistema

* Capa de Presentación: Es la que el usuario final ve desplegada en el navegador e interactúa con la misma, esta capa se comunica con la capa de lógica de negocios y utiliza tecnología JSF.
* Capa de Lógica de Negocio: esta capa es la encargada de intercambiar información que luego será entregada a la capa de presentación (vista) controlando todo el flujo.
* Capa de Persistencia: esta capa es la encargada de mantener los datos accesibles todo el tiempo y desacoplar esta responsabilidad a la base de datos.
* Capa de Base de Datos: donde se almacenarán los datos una vez estos hayan sido recolectados por las capas que anteceden a esta.
  + 1. **Objetivos del Proyecto**

Los objetivos que se desea alcanzar con el desarrollo de Sistema de Evaluación al Docente del IPEC se detallan a continuación

* + - 1. **Objetivos del Negocio**

Mejorar la toma de decisiones sobre las evaluaciones a los encuestados de una manera los más certera y eficaz que beneficie al docente para el mejoramiento de sus desempeños y habilidades en el ámbito docente.

* + - 1. **Objetivos de Diseño**
* Administrar y gestionar la creación de usuarios con permisos administrativos para el sistema.
* Reducir los tiempos en la elaboración de resultados sobre el estado de las habilidades del docente encuestado.
* Administrar las preguntas y respuestas que serán usadas para la evaluación del docente.
* Brindar un ambiente amigable y fácil de usar con los usuarios que usarán el aplicativo.
  + 1. **Factores críticos**

Los principales factores críticos en el desarrollo de un software son los que se encuentran detallados en el análisis de riesgos[[8]](#footnote-8), los cuales pueden afectar al desarrollo o la calidad de los requisitos y las acciones para confrontarlos y evitarlos permitiéndonos minimizar el impacto que estos supondrían.

Al identificar los riesgos, se va un paso adelante evitándolos en lo posible y controlarlos según sea necesario y evitar que se conviertan en problemas en el futuro inmediato.

* + 1. **Planificación Inicial**

Aquí detallaremos el equipo de trabajo, los perfiles de usuario y cronograma inicial de trabajo, planteados para la realización del proyecto.

* + - 1. **Equipo de trabajo**

La presente Tabla IV. 2, se detallan la asignación de roles MSF, los cuales conforman el equipo de trabajo a implementar el Sistema de Evaluación Docente del IPEC

Tabla IV. 2 Equipo de trabajo

|  |  |  |
| --- | --- | --- |
| **ROL MSF** | **RESPONSABLE** | **CARGO** |
| Product Manager | José Luis Gavilanes | Tesista |
| Program Manager | José Luis Gavilanes | Tesista |
| Development Manager | José Luis Gavilanes | Tesista |
| Testing Manager | Ing. Diego Caisaguano | Técnico Informático del IPEC |
| User Experience Manager | José Luis Gavilanes | Tesista |
| Release Manager | Ing. Diego Caisaguano | Técnico Informático del IPEC |

* + - 1. **Perfiles de usuario**

En la toma de requisitos y después del análisis respectivo para la creación del Sistema de Evaluación Docente del IPEC, se obtuvo que para el uso del aplicativo es necesario los perfiles de usuario tales como: Administrador, Docente, Estudiante, Coordinador, Director. En la Tabla IV.3, se describe las funciones de los usuarios

|  |  |  |
| --- | --- | --- |
| **PERFIL** | **TIPO DE ACCESO** | **DESCRIPCION** |
| Administrador | Total | Es el encargado de la administración del sistema, donde puede crear, modificar y eliminar preguntas y respuestas según sea solicitado por el director del IPEC |
| Docente | Limitado | Acceso a realizar su auto-evaluación (encuesta) |
| Estudiante | Limitado | Acceso a realizar la evaluación al Docente (hetero-evaluación) |
| Coordinador | Limitado | Acceso a realizar la evaluación al Docente (co-evaluación) |
| Director | Limitado | Acceso a los reportes emitidos por el sistema |

* + - 1. **Cronograma de trabajo.**

El cronograma inicial de trabajo[[9]](#footnote-9) nos permite evaluar el tiempo para la realización del proyecto, al concluir la construcción del cronograma de trabajo se estimó que la duración e implementación del Sistema de Evaluación al Docente del IPEC (SEDIPEC), tendrá una duración de 138 días, donde se consideró 8 horas de trabajo diarias, el cronograma inicia el 27 de Octubre del 2014 y finaliza el viernes 01 de Mayo del 2015.

* 1. **FASE II. Planificación**

Al iniciar el desarrollo de una aplicación se requiere la definición de un adecuado plan de trabajo que facilite a cumplir con las necesidades y expectativas del sistema dentro de los tiempos estimados por ello las actividades deben ser controladas para evitar retrasos y perdidas de recursos.

La correcta planificación es vital al momento de elaborar un proyecto por ello la realización del plan de trabajo ayuda a refinar dichas actividades para llegar a la eficiencia deseada.

* + 1. **Especificación Funcional**

Este apartado comprende los requisitos necesarios para el desarrollo del aplicativo mediante acuerdos razonables entre el técnico y director del IPEC para con el equipo de trabajo del proyecto.

* + - 1. **Diseño Conceptual**

El diseño conceptual trata de la abstracción de los requerimientos funcionales para el aplicativo donde comprende los actores, diagramas de casos de uso, escenarios y glosario de términos, mismo que se detallan a continuación

* Requerimientos: la especificación de requerimientos de software (SRS)[[10]](#footnote-10) donde se define de forma precisa el producto que se pretende construir, es por ellos que se ha implementado el SRS aplicando la norma IEEE-830 versión 1, donde se logró definir 15 requerimientos funcionales, adicional los requerimientos de interfaces externas, restricciones de diseño y atributos.
* Actores: Los actores que se identificaron para el sistema se detallan en la Tabla IV.3

Tabla IV. 3 Actores del sistema

|  |  |
| --- | --- |
| **PERFIL** | **DESCRIPCION** |
| Administrador | Es el encargado de la administración del sistema, donde puede crear, modificar y eliminar preguntas y respuestas según sea solicitado por el director del IPEC |
| Docente | Acceso a realizar su auto-evaluación (encuesta) |
| Estudiante | Acceso a realizar la evaluación al Docente (hetero-evaluación) |
| Coordinador | Acceso a realizar la evaluación al Docente (co-evaluación) |
| Director | Acceso a los reportes emitidos por el sistema |

* Diagramas de caso de uso: mediante los diagramas de casos de uso[[11]](#footnote-11) se representa gráficamente los requisitos funcionales, estos diagramas salen de la definición de requerimientos previamente establecidos, es por ello que se consideraron los siguientes módulos: Administración, Evaluación al Docente, Reportes.

El modelo de casos de uso que describe el módulo de Evaluación al Docente por parte del estudiante se ilustra en la Figura IV.3



Figura IV. 3 Diagrama de Casos de Uso - Evaluación al Docente por el estudiante

* Casos de uso formato extendido: una vez realizados o definidos los diagramas de casos de uso, a continuación se detalla cada uno de ellos[[12]](#footnote-12), donde para el módulo de Administración, Evaluación al Docente, Reportes se crearon x, x y x casos de uso extendido respectivamente

En la Tabla IV.4 tenemos el caso de uso donde se realiza una encuesta al Docente por parte del estudiante, donde se define: (Fernández Alarcón, 2010, p. 147)

* + Identificador del caso de uso con el nombre del mismo anteponiéndole “cu”
  + Nombre del caso de uso, el cual corresponde a una descripción abreviada al requerimiento funcional respectivo.
  + El actor o actores que desempeñan roles.
  + Propósito que corresponde a la intención del caso de uso.
  + La visión general que es un resumen o repetición del caso de uso de alto nivel.
  + Tipo que puede ser primario, secundario u opcional, también esencial o real.
  + Referencia que se relaciona con el requisito del sistema.
  + Curso típico de eventos se describe la interacción entre los actores y el sistema.
  + Cursos alternativos describe los puntos que surgen de una alternativa junto al número de la excepción.

Tabla IV. 4 Caso de Uso, Evaluación al Docente por el Estudiante

|  |  |
| --- | --- |
| **IDENTIFICADOR DEL CASO DE USO** | cuLlenarEncuesta |
| **NOMBRE DEL CASO DE USO** | Llenar encuesta |
| **ACTORES** | Estudiante |
| **VISION GENERAL** | Realiza el llenado de la encuesta, en base a las preguntas con diferentes opciones de respuesta |
| **TIPO** | Primario y esencial |
| **REFERENCIA** | Requisito Funcional 2 |
| **CURSO TIPICO DE EVENTOS** | |
| **ACCION DEL ACTOR** | **RESPUESTA DEL SISTEMA** |
| 1. Este caso es iniciado cuando el estudiante ingresa al sistema y elije evaluar a un determinado docente de los varios disponibles dentro de la maestría que está cursando.  3. Ingresa la información contestando todas las preguntas de la encuesta | 2. Muestra un pantalla con las preguntas de la encuesta.  4. Verifica que todas las preguntas estén contestadas con al menos una opción de las posibles respuestas  5. Se ingresa todos los datos de la encuesta y se regresa a la página inicial para que se elija otro docente a evaluar |
| **CURSO ALTERNATIVO** | |
| 1. Si alguna de las preguntas falta de contestar, se despliega un mensaje de información que faltan preguntas por contestar. | |

* Escenarios.

Para la implantación del Sistema de Evaluación Docente del IPEC (SEDIPEC), el Departamento de Tecnologías de la Información y Comunicación cuenta con un servidor HP StorageWorks 440 Enterprise Virtual Array (EVA), las características de cada máquina virtual se detalla en la Tabla IV.5, 6 y 7

Tabla IV. 5 Características del Escenario - Servidor de aplicaciones virtualizado

|  |  |
| --- | --- |
| **DEFINICION** | **CARACTERISITICA** |
| Procesador | 4 Procesadores Xeon 3,16 GHz |
| Memoria RAM | 4 GB |
| Disco duro | 65 GB |

Tabla IV. 6 Características del Escenario - Servidor de base de datos virtualizado

|  |  |
| --- | --- |
| **DEFINICION** | **CARACTERISITICA** |
| Procesador | 4 Procesadores Xeon 3,16 GHz |
| Memoria RAM | 8 GB |
| Disco duro | 84 GB |

Tabla IV. 7 Características del Escenario - Firewall virtualizado

|  |  |
| --- | --- |
| **DEFINICION** | **CARACTERISITICA** |
| Procesador | 4 Procesadores Xeon 3,16 GHz |
| Memoria RAM | 4 GB |
| Disco duro | 65 GB |

Por lado del cliente al tener múltiples máquinas de diferentes características que pudieran hacer uso del aplicativo, se detalla en la Tabla IV.8 una maquina con los requisitos mínimos para usar el aplicativo adecuadamente.

Tabla IV. 8 Maquina con requisitos mínimos

|  |  |
| --- | --- |
| **DEFINICION** | **CARACTERISITICA** |
| Procesador | Procesador Dual Core 2 Duo |
| Memoria RAM | 1 GB |
| Disco duro | 10 GB |
| Red | Tarjeta de Red Ethernet 10/100 Mbps |

Tomando en consideración lo planteado anteriormente en la Figura IV. Se ilustra un escenario de implantación del Sistema de Evaluación Docente IPEC, donde se puede observar que nos comunicamos mediante un web services para traer los datos de los estudiantes y docentes que se encuentran ya en el sistema académico SISEPEC del IPEC



Figura IV. 4 Escenario planteado

* Glosario de Términos

Ver la sección “Glosario” al final del documento, revisar el nuevo formato (es opcional ahora esto)

* + - 1. **Diseño Lógico**

Esta etapa trata de describir la solución en términos de la organización, como esta estructurada y la interacción de sus partes desde la perspectiva del equipo del proyecto.

* + - * 1. **Tecnología a usar en el proyecto**

El departamento de Tecnologías de la Información y Comunicación (DTIC) de la ESPOCH propuso trabajar en la misma línea de tecnología que esta implementado el sistema académico del IPEC (SISEPEC), bajo software libre Tomcat y motor de base de datos PostgreSQL con ello se apega al Decreto Ejecutivo número 1014 del 10 de Abril del 2008donde se dispone el uso de Software Libre en los sistemas y equipamientos informáticos de la Administración Pública de Ecuador. (ESPOCH, 2008).

En base a lo descrito anteriormente se detalla las tecnologías a usarse en el proyecto informático:

* Apache Tomcat 8.0.3 como servidor de aplicaciones
* PostgreSQL 9.3 para el almacenamiento de los datos
* Netbeans 8.0 como IDE para el desarrollo
* Framework Hibernate 4.3.5 como framework de persistencia
* JasperReport 5.5.0 para la generación de reportes
* ArgoUML para creación de los diferentes diagramas utilizados en el presente documento
* Microsoft Project para la planificación del proyecto
* Toad Data Modeler para el diseñar el modelo entidad – relacional
* Kit de desarrollo Java JDK 1.7
* MSF (Microsoft Solution Framework) como metodología para el desarrollo del proyecto de software
  + - * 1. **Diagramas de Secuencia**

Los diagramas de secuencia[[13]](#footnote-13) nos ayudan a ver las clases necesarias durante el desarrollo del Sistema de Evaluación Docente del IPEC (SEDIPEC), adicional nos da una visualización dinámica de los procesos necesarios de la aplicación como los métodos que se deberían implementar en cada clase. En el desarrollo del sistema informático se identificaron X diagramas, en la Figura IV. 5 se aprecia el diagrama de secuencia del caso de uso “Llenar la encuesta por el estudiante”

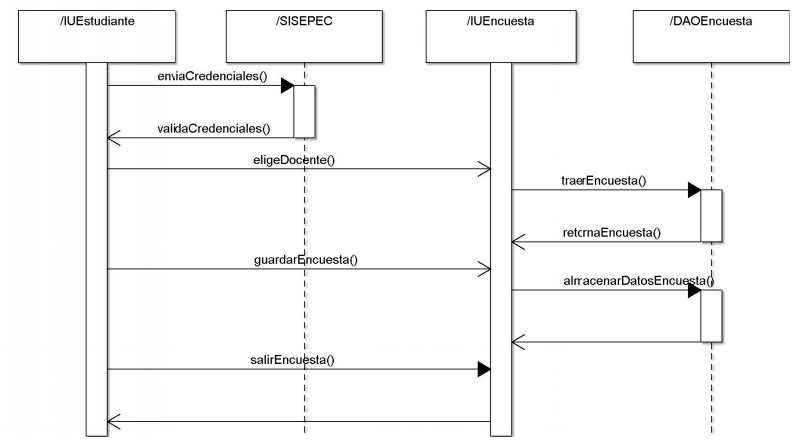


Figura IV. 5 Diagrama de Secuencia cuLlenarEncuesta

* + - * 1. **Diagramas de Clase**

Una vez que hemos podido definir los diagramas de secuencia[[14]](#footnote-14) en el apartado anterior especificaremos los diagramas de clase para el proyecto informático donde se han identificado varios paquetes:

Los paquetes compuestos algunos para sesiones de Hibernate y los DAOs para la gestión de la persistencia con sus respectivos POJOs, el diseño de clases conformado por la factoria de Hibernate y la herencia para las diferentes clases. Al momento de construir el modelo se lo realizo para los módulos de Administración, Evaluación y Reportes.

* + - * 1. **Modelo Lógico de la Base de datos**

Una vez obtenido los diagramas de clases los cuales nos servirán para la creación de los POJOs, se crea el modelo lógico de la base de datos[[15]](#footnote-15), la siguiente Figura IV. nos muestra parte del diseño de la misma.



Figura IV. 6 Modelo Lógico de la Base de dato

* + - * 1. **Diseño de Interfaces de Usuario**

Las interfaces de usuario están diseñadas en base de los requerimientos funcionales previamente establecidos, es por ello que para la implementación del aplicativo se ha considerado las siguiente interfaces.

* Interfaz de autentificación: permite al usuario ingresar sus credenciales que son usadas en el SISEPEC como los son la cedula y contraseña.
* Interfaz Principal: se mostrarán las funcionalidades principales de acuerdo a los permisos de cada usuario las cuales pueden ser accedidas a través de vínculos que estarán identificadas por un nombre descriptivo que al seleccionar llevará al usuario a otro lugar del sistema según lo escogido. Todas las interfaces se implementan basando su estructura como lo muestra la Figura IV donde la sección de contenido cambia según las acciones realizadas.

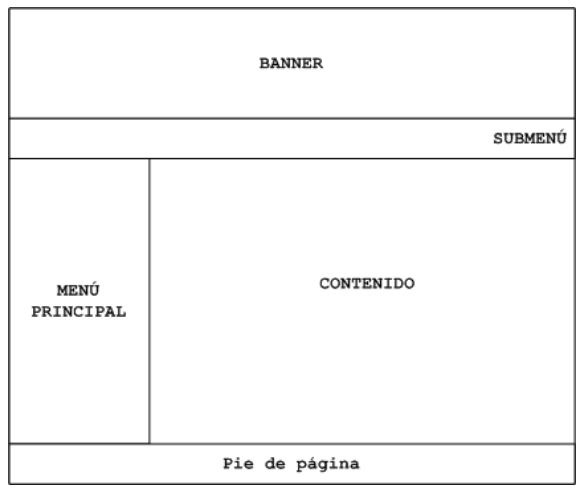


Figura IV. 7 Interfaz principal

* Interfaz para administración de usuarios: este funcionalidad permite la creación de nuevos usuarios, adicional el criterio de búsqueda se considera mediante la cedula o nombres, luego se puede modificar o eliminar el usuario encontrado.
* Interfaz de evaluaciones: nos muestra las preguntas de la encuesta
* Interfaz de administración de cuestionarios: permite al usuario Administrador ingresar preguntas o administrar nuevas categorías de preguntas.
* Interfaz de resultados: de tallan los resultados obtenidos de los datos de las encuestas ya realizadas, según lo especificado en los requerimientos.

De lo mencionado con anterioridad, se realizó el prototipo de las interfaces que se usarían en el sistema, mismas que se pueden observar en el SRS[[16]](#footnote-16).

* + - 1. **Diseño Físico**

Esta etapa es la final en la fase de planificación, en ella se describen los diagramas de actividades, componentes, servicios y tecnologías de solución.

* + - * 1. **Diagrama de Actividades**

A través de los diagrama de actividades[[17]](#footnote-17) se describe el flujo de trabajo desde el inicio hasta el final de la tarea, detallándose todas las posibles opciones durante el proceso de la aplicación. En este apartado se detallarán los diagramas para la realización de una encuesta por parte del Estudiante, creación de usuarios para uso del sistema.

* + - * 1. **Diagrama de Componentes**

En la Figura IV.8, se muestra el diagrama que nos detalla la estructura del Sistema de Evaluación Docente del IPEC SEDIPEC, nos podemos observar el paquete de gestión de usuarios, el cual está compuesto por el componente Usuario, de igual manera los demás paquetes con sus correspondientes paquetes.



Figura IV. 8 Diagrama de componentes

**Diagrama de Despliegue**

La aplicación fue diseñada en base a los requisitos previos y la solución tanto de la capa de presentación, negocio y acceso a datos se ejecuten en el servidor de aplicaciones Apache Tomcat, como lo demuestra la Figura IV.9



Figura IV. 9 Diagrama de despliegue

* + - 1. **Modelo físico de la Base de Datos**

Al momento de crear el modelo lógico de la base de datos del sistema de evaluación al Docente del IPEC por medio del software Toad Data Modeles, las relaciones N:M al convertir el modelo entidad – relación en un modelo relacional normal, nos da un total de 20 tablas que se implementaran en la base de datos “EVALUACION”.

En el momento de la generación del script[[18]](#footnote-18) con la herramienta de software antes mencionada se obtuvo un archivo con sentencias SQL con todas las tablas y tipos de datos para poder ejecutar en PostgreSQL.

* 1. **FASE III. Desarrollo**

En esta etapa es donde se creará la aplicación como tal, dado que se incluye la construcción del código, asi como su documentación y la implementación de la solución.

* + 1. **Nomenclaturas y Estándares**

Este apartado nos ayuda a como se definirán los estándares o convenciones de programación a ser tomados en cuenta para el desarrollo del Sistema de Evaluación Docente del IPEC, patrones basados en recomendaciones de Sun Microsystenes

La nomenclatura será usada en la organización de ficheros fuente java, declaraciones de clases, estándar Java bean, declaraciones generales, especificación XHTML y creación de las tablas de las bases de datos.

* Organización de ficheros: las clases se agrupan en paquetes y estos son organizados de manera jerárquica de tal manera que todo el código fuente para el sistema estará incluido dentro del paquete “espoch.sedipec” donde contendrá las clases y subpaquetes en función de la sección a la cual pertenecen, ejemplo la implementación de acceso a datos DAOs irán como “espoch.sedipec.dao”
* Ficheros fuente Java: estos deben contener una única clase o interfaz pública, donde el nombre de la clase coincide con el nombre del fichero, en toso fichero fuente java se debe distinguir adecuadamente la sentencia paquete, la sentencia de importación y declaración de clases e interfaces/
* Declaración de clases

Aquí se incluyen la clase, métodos, variables y constantes, más adelante detallaremos brevemente los elementos que componen una clase o interfaz.

* + Clases: para toda nombre de clase, la primera debe ser mayúscula, al tratar con una combinación de clases se debe intercalar mayúsculas y minúsculas, ejemplo EncuestaEstudiante.java
  + Métodos: aquí la primera letra va en minúscula, si son varias palabras se intercalan minúsculas y mayúsculas.
  + Variables; también se aplica el concepto como para los métodos, donde la primera letra empieza en minúscula.
  + Constantes: estas deben ser escritas completamente en mayúsculas y la separación entre palabras se debe colocar carácter guion bajo/undercode ( \_ ), ejemplo VALOR\_MINIMO\_PI.
  + Estándar Java bean: los Java vean todos ellos contienen los métodos setter y getter la características de estos es que son privados, ejemplo getUsuario().
  + Declaraciones generales: la declaración de la variable se debe realizar una por línea, debe ser inicializada en el momento de su declaración o por algún valor calculado previamente
  + Especificación XHTML: las reglas de sintaxis que se aplican para los documentos XHTML del sistema informático son:
    - Todo se escribe en minúsculas
    - Los tags deben llevar a cierre incluyendo los vacíos
    - Los parámetros de los tags deben ser colocados entre comillas, ejemplo <table border=3 >
    - Los elementos de cierre, que al no ser usados correctamente pueden ocasionar futuros problemas, ejemplo <p></p>
  + Estándares del diseño de la base de datos: los nombres y campos de la tabla se escribirán en letras minúsculas, letras acentuadas se remplazara si es el caso ( ñ) por ( ni ), los nombres de las tablas deben ser los más descriptivos posibles
    1. **Capa de Presentación**

En esta capa se desarrolla los componentes relacionados con las interfaces de usuario, el desarrollo del sistema se basó en las páginas XHTML de PrimaFaces y estas se integran a los java beans

* + - 1. **Implementación de interfaces de usuario con XTMHL y PrimeFaces**:

para el desarrollo de la pantalla principal y el contenido, el contenido de ésta según sea el usuario bloqueado al sistema tiene diferentes funcionalidades. Página de inicio, diseñada para la validación de credenciales por parte del usuario como se aprecia en la Fig IV.10



Figura IV. 10 Interfaz de autentificación de usuarios

Pantalla principal que se le mostraría en caso de loguearse un Administrador, donde está dividido en tres partes como son banner, menú y contenido, referirse a la Figura IV.11

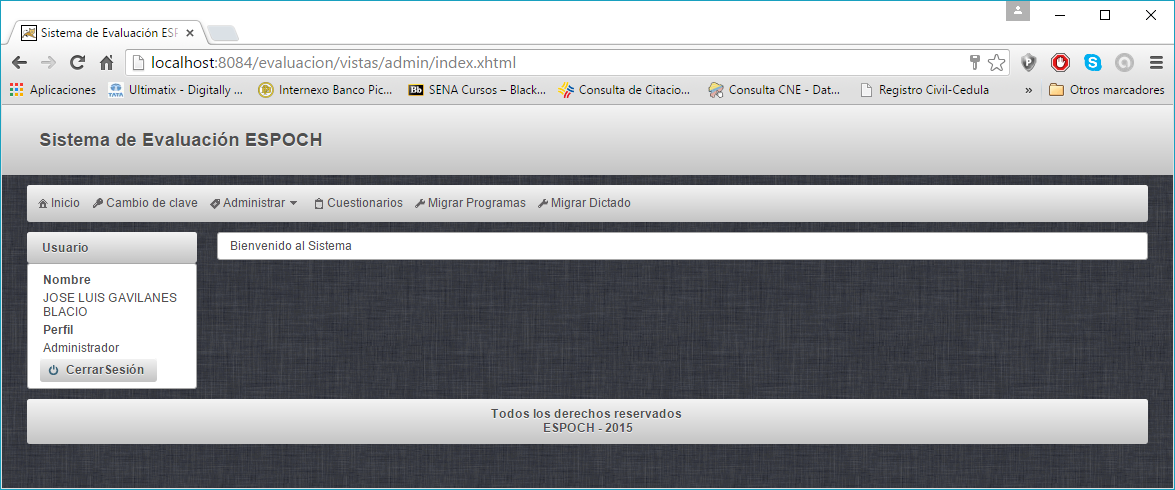


Figura IV. 11 Página principal para usuario Administrador

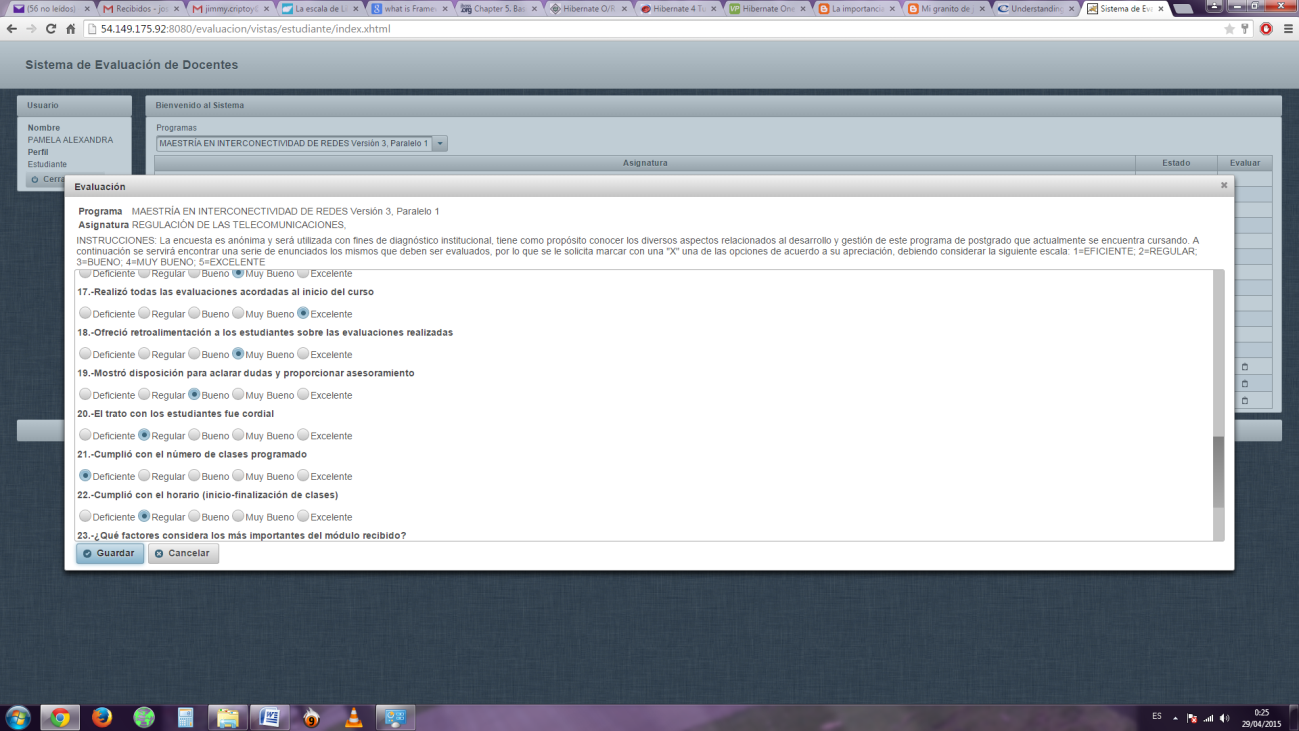
Página donde el Estudiante llena la encuesta de acuerdo a la maestría la cual está estudiando y si el Docente este disponible para ser evaluado. Figura IV.12 

Figura IV. 12 Interfaz para Encuesta

* + - 1. **Implementación de Beans**

Los elementos Java beans son característicos de una aplicación JSF, donde encapsula los datos procedentes del cliente, donde también a través de los métodos da respuesta al que solicita, al implementar el sistema de Evaluación al Docente del IPEC se obtuvo un total de 30 clases

* + 1. **Capa lógica de Negocios**

Esta etapa nos permite conocer la implementación de los servicios para centralizar la obtención de información, que actúan desde la capa de presentación y la capa de persistencia, para el sistema se desarrollaron 15 eventos.

* + 1. **Capa de Persistencia**

Para poder acceder a los datos se lo hizo a través del patrón Data Access Object (DAO) donde se definieron los correspondientes DAOs para que puedan trabajar con el modelo, de igual formar la persistencia se implementó con Hibernate en base al estudio comparativo realizado en el Capítulo III del presente documento, para lo cual fueron necesarios la creación de los archivos correspondiente como son los POJOs y archivos de configuración.

Los POJOs se crean al momento de asociarse con cada tabla de la base de datos, donde gracias a los archivos de mapeo indican a Hibernate que tabla de la base de datos será accedida y que columnas deberán ser usadas y es donde los DAOs nos facilitan la gestión de la persistencia con la base de datos PostgreSQL

* + 1. **Capa de Base de Datos**

La creación de los POJOs para el manejo de la persistencia se lo realizo con el framework Hibernate, según los resultados y conclusiones realizadas en el Capítulo III del presente documento, donde los parámetros de conexión a la base de datos se configuro el archivo “hibernate.cfg.xml”, haciendo transparente para el desarrollador permitiéndole en el futuro modificar estos parámetros sin modificar líneas de código adicionales. El modelo de la base de datos también cuenta con vistas y procedimientos almacenados (funciones) para su mejor funcionamiento

* 1. **Fase IV. Estabilización**

Esta fase es donde se realizan las pruebas de carga e integración de componentes de la solución planteada, por medio de los escenarios de prueba se hace la identificación, priorización y resolución de problemas de tal forma que la solución quede lista para su posterior publicación

* + 1. **Revisión general del sistema’**

Para proceder con los test del sistema se prepara los diferentes escenarios planteados, para ello antes se revisa que el código fuente del aplicativo cumpla con los estándares anteriormente definidos, de igual manera se comprueba que el script de la base de datos se ejecute sin inconvenientes y los documentos de instalación sea un asistente intuitivo para el futuro administrador del sistema

* + - 1. **Código fuente**

Se verifica que el código fuente del Sistema de Evaluación Docente del IPEC[[19]](#footnote-19) cumpa con la nomenclatura antes especificada, donde para ellos se observa la organización de ficheros, las clases, los beans, métodos, variables existentes de cada clase y los ficheros XHTML

* + - 1. **Script de Base de datos**

Se valida que los nombres de tablas y los campos de la base de datos SEDIPEC, cumpla de igual manera con los estándares antes mencionados, se ejecuto el script [[20]](#footnote-20) en Administrador de bases de datos PostgreSQL para corroborar su adecuado funcionamiento

* + - 1. **Documentación de la instalación**

Con esta documentación se capacito al administrador del SEDIPEC usando el manual de instalación[[21]](#footnote-21), donde se verifico el proceso de instalación de PostgreSQL, Apache Tomcat y despliegue de la aplicación

* + 1. **Pruebas**

Las pruebas es una etapa de mucha importancia en la fase de estabilización de la metodología MSF, pues de ello se entregará un software que cumpla con todo lo estipulado en los requerimientos funcionales para el aplicativo.

Las pruebas se la realizo los días 6, 7 y 8 de Mayo del 2015 con el Ing. Diego Caisaguano (Testing Manager), donde las mismas nos permitieron verificar el correcto funcionamiento de los flujos básicos y alternativos de los casos de uso a implementar en las iteraciones.

Las pruebas de test al aplicativo, y para desarrollar esta actividad se eligió los casos de prueba (anexo de los casos de prueba), como el modelo que se muestra en la Tabla IV.# donde se define el código como un consecutivo, el nombre el cual corresponde al nombre del caso de uso, las condiciones baja las cuales se ejecuta, pasos para dicha ejecución, lista ordenada de los pasos que permiten verificar el caso de prueba, resultados esperados, el estado obtenido al finalizar el caso de prueba.

En la culminación de la construcción de todos los casos de prueba se obtuvo un total de 30, la **Tabla IV.8** está definido el caso de prueba para crear un usuario en su flujo básico.

**Tabla IV.8 Caso de Prueba 1 Crear Usuario**

|  |  |  |  |
| --- | --- | --- | --- |
| **PRUEBA DE ACEPTACIÓN** | | | |
| **CÓDIGO:** | 1 | **NOMBRE:** | Crear Usuario |
| **DESCRIPCIÓN:** | Probar el flujo básico para crear un usuario | | |
| **CONDICIONES:** | Para crear un usuario se escribieron los datos del mismo | | |
| **PASOS DE EJECUCIÓN:** | 1. Ingresar con credenciales de Administrador. 2. Seleccionar nuevo usuario 3. Ingresar los valores correctamente 4. Clic en Guardar | | |
| **RESULTADO ESPERADO:** | Se muestra un mensaje de confirmación que se ha creado el usuario | | |
| **EVALUACIÓN DE PRUEBA:** | Exitosa | | |

En los casos de pruebas realizados se obtuvo un 86% de exitosos, mientras que el 14% restantes no fueron exitosos, de los cuales se verificarán nuevamente para obtener mejores resultados cercanos al 95%.

* 1. **FASE V. Fase de Implementación**

En esta fase se implementa la solución como tal y se valida que sea estable y utilizable, donde para la implementación de la solución se cuenta con un plan de operaciones y soporte

* + 1. **Estrategia de implantación**

Se hizo el despliegue de la aplicación con el técnico informático del IPEC, con su debido manual de instalación[[22]](#footnote-22)

* + 1. **Soporte**

Se informó a los futuros usuarios del sistema acerca del despliegue de la solución y la dirección de acceso a la misma, la presentar inconvenientes para acceder al aplicativo por parte de los usuarios se brindó el soporte adecuado

* + 1. **Coordinación del entrenamiento**

Se capacito al personal de soporte técnico, de tal manera que obtenga los conocimientos necesarios para brindar un adecuado soporte

* + 1. **Transferencias a operaciones**

Al finalizar el despliegue de la solución, se firmó el documento de entrega del sistema, manuales técnico y manual de usuario al técnico informático del IPEC. Con este se da formalidad de transferencia de responsabilidad de la aplicación al Administrador del sitio.

**CONCLUSIONES**

Al finalizar el estudio comparativo entre los frameworks Hibernate y Mybatis aplicados el Sistema de Evaluación Docente del IPEC se han obtenido las siguientes conclusiones:

* La selección del framework ideal para la persistencia de objetos entre Hibernate y Mybatis no pretende desplazarlos entre sí, la elección se basara en las características que tiene cada uno y de estas cuales se acogen mejor para el desarrollo del Sistema de Evaluación Docente del IPEC.
* Con el pasar del tiempo donde las aplicaciones se vuelven cada vez más complejas y para tener un adecuado manejo de los datos algunas herramientas han evolucionado y mejorado de una manera trasparente y eficiente el manejo de la persistencia de objetos, dando así la posibilidad que el desarrollador ocupe menos tiempo en el acceso a los datos.
* Al culminar el estudio comparativo se obtuvo que Hibernate alcanzo el 62,17% y Mybatis el 37,84%, con un nivel de confianza 95%
* La implementación del Sistema de Evaluación al Docente del IPEC con el framework de persistencia Hibernate permitió al desarrollador realizar ágilmente el aplicativo, por las diferentes ventajas que éste presta, en cuanto a la disponibilidad de documentación, soporte a través de comunidades en línea y por la suma de componentes reutilizables que el framework proporciona.
* Es necesario tener un conocimiento previo de las tecnologías a usarse para el desarrollo de una aplicación, como el caso de Hibernate donde su rendimiento es más eficiente con una correcta configuración de sus archivos y el diseño adecuado de la base de datos

**RECOMENDACIONES**

* Las actualizaciones del framework Hibernate, donde se le agregan mejoras o corrigen fallos del mismo, son liberadas continuamente por lo que es necesario analizar la posibilidad de estar con las últimas versiones con el fin de que la aplicación sea más estable y segura.
* En los proyectos es necesaria una correcta planificación, por lo cual la elección de una metodología en la que permita minimizar los riesgos, mejorar la calidad y reducir el tiempo, ayuda a que la solución sea factible por el equipo de trabajo.
* Es necesario una adecuada recolección de los requisitos y que estos sean analizados adecuadamente por las partes interesadas estableciendo los alcances de los mismos, ayudándonos asi a minimizar el impacto de los riesgos en el futuro inmediato.
* Es adecuado que se estudie un ORM antes de hacer uso de las herramientas basadas en este concepto, posibilitando así al desarrollador tener una idea más clara de cómo funciona la herramienta internamente.
* Es recomendable que el areá técnica del IPEC utilice la documentación del sistema como referencia para el uso y solución de errores que se puedan dar en el futuro por parte del sistema.
* Se recomienda que se desarrolle un módulo de logs para el sistema, el cual ayudaría a entender más fácilmente y rápidamente donde el sistema o los usuarios tuvieron inconvenientes

**RESUMEN**

El estudio comparativo de los frameworks Hibernate y Mybatis se llevó acabo con la finalidad de seleccionar el más adecuado y eficiente que nos permita desarrollar rápidamente un sistema de Evaluaciones para que pueda ser implementado en Instituto de Postgrado y Educación Continua de Chimborazo.

Al evaluarse los diferentes frameworks mediante diferentes parámetros a base de experimentación e investigación directa al tener los datos y tabularlos a estos resultados se aplicó chi cuadrado para la demostración de la hipótesis, rechazando asi la hipótesis nula y aceptando la de la investigación.

Luego de aplicar el estudio comparativo se obtuvo que Hibernate alcanzo 62,39% y Mybatis 37,61% por lo que se seleccionó el framework Hibernate por las ventajas que este presta, en cuanto a la disponibilidad de documentación, calidad, robustez, aprendizaje, facilidad para el desarrollo y la cantidad de componentes de usuario que pone a disposición de los desarrolladores.

Se prototipo e implanto el Sistema de Evaluación al Docente IPEC, mismo que muestras que muestra algunas de las características y funcionalidades del framework Hibernate, siguiendo la metodología Microsoft Solution Framework.

Se recomienda el uso del framework a los desarrolladores para aplicaciones similares de media complejidad.

# ABSTRACT

The comparative study of Hibernate and Mybatis framework was held in order to select the appropriate and efficent to allow us to quickly develop a system of evaluations so it can be implemented at The Institute of Postgraduation and Continuing Education of Chimborazo.

The different framework were assessed by different parameters base don direct experimentation and it has the data and tabulate these research results with chi square test that was applied to demonstrate the hypothesis, then rejecting the null hypothesis and accepting the research.

After applying the comparative study, it was obtained that Hibernate reached the 62,39% and 37,61% Mybatis so the Hibernate framework is selected for the advantages it provides. As for the availability of documentation, quality, robustness, learning, ease of development and the use amount of components available to developers.

It was designed the prototype and implemented the Teacher’s evaluation System IPEC, same that shows some of the features and functionality of the Hibernate framework, following the methodology Microsoft Solution Framework.

It is recomended the use of framework developers for similiar of average complexity

# BIBLIOGRAFÍA

Payá, A. (14 de Septiembre de 2006). *ANÁLISIS Y USO DE FRAMEWORKS DE PERSISTENCIA EN JAVA.* Obtenido de UNIVERSIDAD PONTIFICIA COMILLAS: http://www.iit.upcomillas.es/pfc/resumenes/450955e7368ca.pdf

(12 de 05 de 2014). Recuperado el 25 de 10 de 2014, de http://google.com

Alegsa, L. (12 de Mayo de 2010). *Definición de SGBD*. Obtenido de ALEGSA: http://www.alegsa.com.ar/Dic/sgbd.php

Cando, O. (2013). *ANÁLISIS HIBERNATE COMO TECNOLOGÍA DE PERSISTENCIA DE OBJETOS.* Obtenido de Escuela Superior Politécnica de Chimborazo: http://dspace.espoch.edu.ec/handle/123456789/2427

CONSEJO DE EDUCACIÓN SUPERIOR. (31 de Octubre de 2012). *REGLAMENTO DE CARRERA Y ESCALAFÓN DEL PROFESOR E INVESTIGADOR DEL SISTEMA DE EDUCACIÓN SUPERIOR.* Obtenido de RPC-SO-037-No.265-2012 - CODIFICADA: http://www.ces.gob.ec/gaceta-oficial/actas-y-resoluciones/sesiones-ordinarias/2012/1064-rpc-so-037-no265-2012-codificada

Córdova Espinoza, R. F., & Cuzco Sarango, B. E. (2013). Análisis Comparativo entre Bases de datos relacionales con Bases de datos no relacionales. *Tesis de pregrado*. Cuenca, Azuay, Ecuador: Universidad Politécnica Salesiana.

Escuela Superior Politécnica de Chimborazo. (20 de Diciembre de 2012). *LÍNEAS DE INVESTIGACIÓN ESPOCH-2012.* Obtenido de RESOLUCIÓN 582.CP.2012: http://www.espoch.edu.ec/Descargas/facultadpub/RESOLUCION\_582\_5a97d\_3b492.pdf

ESPOCH. (10 de Abril de 2008). *ESPOCH.* Obtenido de ESPOCH: http://www.espoch.edu.ec/Descargas/programapub/Decreto\_1014\_software\_libre\_Ecuador\_c2d0b.pdf

ESPOCH. (2010). *Misión, Visión y Objetivos del IPEC*. Obtenido de Instituto de Posgrado y Educación Continua: http://sisepec.espoch.edu.ec/index.jsp?accion=MISION

Fernández Alarcón, V. (01 de Marzo de 2010). *una metodología basada en el modelado.* Obtenido de Desarrollo de sistemas de información: https://books.google.com.ec/books?id=Sqm7jNZS\_L0C&lpg=PA147&ots=Hx2EafqCBE&dq=curso%20t%C3%ADpico%20de%20eventos%20se%20describe%20la%20interacci%C3%B3n%20entre%20los%20actores%20y%20el%20sistema&pg=PA147#v=onepage&q=curso%20t%C3%ADpico%20de%20eventos%20s

Freire, T. A. (Octubre de 2008). *ORM (Mapeo Objeto Relacional).* Obtenido de Estudio de la Técnica ORM (Mapeo Objeto – Relacional): http://repositorio.utn.edu.ec/bitstream/123456789/571/1/Tesis.pdf

Gómez, E. J. (15 de Mayo de 2013). *¿Qué es un Framework?* Obtenido de Servicios Informáticos: http://edgargomez.es/que-es-un-framework/#.VE6JbiKG-E4

Gonzaga, M. (2013). *Especificación de Requerimientos*. Obtenido de Buscador web: https://git.taw.utpl.edu.ec/mjgonzaga/buscador/wikis/home

Gonzalez, A. (Febrero de 2011). *Persistencia de objetos.* Obtenido de LOGICA DE NEGOCIOS: http://ingenieria.uatx.mx/marva/files/2011/02/LOGICA-DE-NEGOCIOS.pptx.

Israel. (14 de Septiembre de 2013). *Datamapper, un ORM para codeigniter*. Obtenido de Codeigniter: http://uno-de-piera.com/datamapper-un-orm-para-codeigniter/

Junta de Andalucía. (10 de Mayo de 2012). *Comparación de las tecnologías de acceso a datos*. Obtenido de Capa de Persistencia: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/180

Mármol, J. (Septiembre de 2004). *Persistencia de Objetos JDO, Soluciones Java.* Obtenido de Universidad de Murcia: http://dis.um.es/~jmolina/Persistencia%20de%20Objeto%20JDO.pdf

Martínez, F. J. (4 de Mayo de 2006). *Filosofía y Objetivos de JDBC*. Obtenido de Introducción a JDBC: http://www.adictosaltrabajo.com/tutoriales/introjdbc/

Microsoft Corp. (10 de Marzo de 2010). *Serialización.* Obtenido de Visual Studio: https://msdn.microsoft.com/es-es/library/vstudio/7ay27kt9(v=vs.100).aspx

NAULA, J. P., & MAIQUIZA, W. I. (2012). *Importancia de un Framework.* Obtenido de ANÁLISIS DE LA TECNOLOGÍA TOPLINK, COMO FRAMEWORK DE PERSISTENCIA APLICADO AL SISTEMA DE EVALUACIÓN DE LAS CARRERAS DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA CON FINES DE ACREDITACIÓN.

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-sbdoo

Sanchéz, J. (2004). *Principios sobre Bases de datos relacionales*. Obtenido de Sitio web de Jorge Sanchez: http://www.jorgesanchez.net/bd/bdrelacional.pdf

SENPLADES. (2013). *Plan Nacional para el Buen Vivir 2013-2017.* Obtenido de SENPLADES: http:// www.buenvivir.gob.ec/documents/10157/96c82f1c-5dd8-4a60-8283-d95d9ed24f0e

Soy Programador. (25 de Noviembre de 2011). *MAPEO OBJETO-RELACIONAL ORM*. Obtenido de Talleres: http://soyprogramador.liz.mx/mapeo-objeto-relacional-orm/

TutorialsPoint. (2015). *What is JDBC?* Obtenido de JDBC - Introduction: http://www.tutorialspoint.com/jdbc/jdbc-introduction.htm

1. el cual es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. [↑](#footnote-ref-1)
2. Es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación [↑](#footnote-ref-2)
3. Es un framework de persistencia que soporta SQL, procedimientos almacenados y mapeos avanzados [↑](#footnote-ref-3)
4. Sitio consultado: http://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-for-2014/10/ [↑](#footnote-ref-4)
5. es una licencia de software creada por la Free Software Foundation que pretende garantizar la libertad de compartir y modificar el software cubierto por ella, asegurando que el software es libre para todos sus usuarios [↑](#footnote-ref-5)
6. permite al usuario del software la libertad de usarlo para cualquier propósito, distribuirlo, modificarlo, y distribuir versiones modificadas de ese software. [↑](#footnote-ref-6)
7. Es una tienda de online dedicada a la venta de todo tipo de artículos [↑](#footnote-ref-7)
8. Manual Tecnico Anexo D – Información CD SEDIPEC [↑](#footnote-ref-8)
9. Manual Tecnico Anexo L – Información CD SEDIPEC [↑](#footnote-ref-9)
10. Manual Tecnico Anexo C – Información CD SEDIPEC [↑](#footnote-ref-10)
11. Manual Tecnico Anexo E – Información CD SEDIPEC [↑](#footnote-ref-11)
12. Manual Tecnico Anexo F – Información CD SEDIPEC [↑](#footnote-ref-12)
13. Manual Tecnico Anexo G – Información CD SEDIPEC [↑](#footnote-ref-13)
14. Manual Tecnico Anexo H – Información CD SEDIPEC [↑](#footnote-ref-14)
15. Manual Tecnico Anexo I – Información CD SEDIPEC [↑](#footnote-ref-15)
16. Manual Tecnico Anexo C – Información CD SEDIPEC [↑](#footnote-ref-16)
17. Manual Tecnico Anexo J – Información CD SEDIPEC [↑](#footnote-ref-17)
18. Manual Tecnico Anexo K – Información CD SEDIPEC [↑](#footnote-ref-18)
19. Manual Tecnico Anexo L – Información CD SEDIPEC [↑](#footnote-ref-19)
20. Manual Tecnico Anexo L – Información CD SEDIPEC [↑](#footnote-ref-20)
21. Manual de Instalación – Información CD SEDIPEC [↑](#footnote-ref-21)
22. Manual de Instalación - Información CD SEDIPEC [↑](#footnote-ref-22)