



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

**“DESARROLLO DE UNA METODOLOGÍA PARA LA CONSTRUCCIÓN DE
UNA APLICACIÓN WEB MEDIANTE LA INTEGRACIÓN DE JSF Y EJB.
CASO PRÁCTICO: HOSPITAL PROVINCIAL GENERAL DOCENTE
RIOBAMBA”**

TESIS DE GRADO

Previa a la obtención del título de

INGENIERO EN SISTEMAS INFORMÁTICOS

Presentado por:

VICENTE ANÍBAL CARRILLO TIXE

ALEX IGNACIO ERAZO VIVERO

RIOBAMBA – ECUADOR

2012

INDICE GENERAL.

INDICE GENERAL

INDICE DE TABLAS

INDICE DE FIGURAS

AGRADECIMIENTO

DEDICATORIA

FIRMAS RESPONSABLES Y NOTAS

ÍNDICE DE ABREVIATURAS

CAPÍTULO I.

| | |
|----------------------------------|-----------|
| 1. MARCO REFERENCIAL. | 25 |
| 1.1. ANTECEDENTES. | 25 |
| 1.2. JUSTIFICACIÓN. | 27 |
| 1.2.1. JUSTIFICACIÓN TEÓRICA. | 27 |
| 1.2.2. JUSTIFICACIÓN APLICATIVA. | 28 |
| 1.3. OBJETIVOS. | 30 |
| 1.3.1. GENERAL. | 30 |
| 1.3.2. ESPECÍFICOS. | 30 |
| 1.4. HIPÓTESIS. | 30 |

CAPÍTULO II.

| | |
|---|-----------|
| 2. JAVA SERVER FACES Y ENTERPRISE JAVA BEAN. | 31 |
| 2.1. JAVA SERVER FACES. | 31 |
| 2.1.1. INTRODUCCIÓN A JSF. | 31 |
| 2.1.2. APLICACIONES JAVA SERVER FACES. | 32 |
| 2.1.2.1. CARACTERÍSTICAS. | 34 |
| 2.1.2.2. BENEFICIOS DE JSF. | 35 |
| 2.1.2.3. ETIQUETAS JSF. | 37 |
| 2.1.2.4. BACKBEANS. | 38 |
| 2.1.2.4.1. ESTRUCTURA DE LAS PÁGINAS. | 38 |

| | | |
|-------------|---|-----------|
| 2.1.2.5. | JSF Y AJAX. | 39 |
| 2.1.2.6. | EL FUTURO DE JSF. | 39 |
| 2.1.3. | MODELO VISTA CONTROLADOR EN JSF. | 40 |
| 2.1.3.1. | INTRODUCCIÓN. | 40 |
| 2.1.3.2. | MODELO. | 40 |
| 2.1.3.3. | VISTA. | 41 |
| 2.1.3.4. | CONTROLADOR. | 41 |
| 2.1.4. | MANAGED BEANS Y NAVEGACIÓN EN JSF. | 41 |
| 2.1.4.1. | MANAGED BEANS. | 41 |
| | ATRIBUTOS. | 42 |
| | EXPRESIONES PARA VALORES INMEDIATOS Y DIRECTOS. | 43 |
| | ÁMBITO DE LOS BEANS. | 43 |
| 2.1.4.2. | NAVEGACIÓN. | 44 |
| | NAVEGACIÓN ESTÁTICA. | 45 |
| | NAVEGACIÓN DINÁMICA. | 45 |
| 2.2. | ENTERPRISE JAVA BEAN. | 46 |
| 2.2.1. | INTRODUCCION. | 46 |
| 2.2.2. | DEFINIÓN DE EJB. | 48 |
| 2.2.3. | VENTAJAS DE EJB. | 48 |
| 2.2.3.1. | SERVICIOS MIDDLEWARE. | 48 |
| 2.2.3.2. | DIVISIÓN DE TRABAJO. | 49 |
| 2.2.3.3. | DIVERSOS VENDEDORES. | 49 |
| 2.2.3.4. | PROCEDIMIENTOS REMOTOS RMI. | 50 |
| 2.2.3.5. | DIVERSOS CLIENTES. | 50 |
| 2.2.4. | COMPONENTES DE LA ARQUITECTURA. | 50 |
| 2.2.4.1. | OBJETO EJB. | 50 |
| 2.2.4.2. | SERVIDOR DE EJB (EJB SERVER). | 53 |
| 2.2.4.3. | EJB CONTAINER. | 53 |
| 2.2.5. | FUNCIONAMIENTO DE LOS EJB. | 53 |
| 2.2.5.1. | INTERFAZ REMOTA. | 54 |
| 2.2.5.2. | INTERFAZ EJB HOME. | 54 |
| 2.2.5.3. | MANEJO DE ESTADOS. | 55 |
| 2.2.5.4. | SEGURIDAD. | 55 |
| 2.2.5.5. | TRANSACCIONES. | 55 |
| 2.2.5.6. | PERSISTENCIA. | 56 |
| 2.2.6. | TIPOS DE EJB. | 56 |

| | | |
|----------|----------------------------|----|
| 2.2.6.1. | EJB DE ENTIDAD. | 56 |
| 2.2.6.2. | EJB DE SESIÓN. | 56 |
| 2.2.6.3. | EJB DIRIGIDOS POR MENSAJE. | 57 |
| 2.2.7. | ARCHIVO EJB-JAR. | 57 |

CAPÍTULO III.

| | | |
|-------------|--|-----------|
| 3. | INTEGRACIÓN DE JSF Y EJB. | 59 |
| 3.1. | INTRODUCCIÓN. | 59 |
| 3.2. | ANTECEDENTES JSF Y EJB. | 61 |
| 3.3. | APLICACIÓN CON JSF Y EJB. | 61 |
| 3.3.1. | REQUISITOS PARA CONSTRUIR LA APLICACIÓN. | 62 |
| 3.3.2. | CREANDO EL PROYECTO. | 63 |
| 3.3.3. | CREANDO LAS CLASES ENTIDADES A PARTIR DE LA BASE DE DATOS. | 66 |
| 3.3.3.1. | CREACIÓN DE LOS EJB (SESSION Y MESSAGE DRIVEN) | 70 |
| 3.3.4. | USANDO JSF Y PRIMEFACES. | 77 |
| 3.3.4.1. | CREANDO EL CLIENTE BEAN GESTIONADO. | 82 |
| 3.3.4.2. | CREACIÓN DE LA PÁGINA WEB CLIENTE. | 85 |

CAPÍTULO IV.

| | | |
|-------------|---|-----------|
| 4. | DESARROLLO DE METODOLOGÍA ERCA WEB | 92 |
| 4.1. | ¿QUÉ ES UNA METODOLOGÍA? | 92 |
| 4.2. | METODOLOGÍAS DE DESARROLLO DE SOFTWARE. | 92 |
| 4.2.1. | METODOLOGÍAS TRADICIONALES O PESADAS. | 93 |
| 4.2.1.1. | METODOLOGÍA MICROSOFT SOLUTION FRAMEWORK (MSF). | 94 |
| 4.2.2. | METODOLOGÍAS ÁGILES. | 95 |
| 4.3. | METODOLOGÍA PROPUESTA ERCA WEB. | 96 |
| 4.3.1. | FASES DE LA METODOLOGÍA ERCA WEB. | 96 |
| 4.3.1.1. | Fase 1: Planificación. | 97 |
| 4.3.1.2. | Fase 2: Desarrollo de Base de Datos. | 97 |
| 4.3.1.3. | Fase 3: Desarrollo de Módulo EJB. | 98 |
| 4.3.1.4. | Fase 4: Integración de Frameworks. | 98 |
| 4.3.1.5. | Fase 5: Construcción Final de la Aplicación. | 98 |
| 4.3.1.6. | Fase 6: Implementación e Implantación. | 98 |

| | | |
|----------|--|-----|
| 4.3.2. | FASE 1: PLANIFICACIÓN. | 98 |
| 4.3.2.1. | ANÁLISIS DE REQUERIMIENTOS. | 98 |
| 4.3.2.2. | BOSQUEJO DE BASE DE DATOS. | 101 |
| 4.3.2.3. | APRENDIZAJE DE FRAMEWORK JSF. | 101 |
| 4.3.3. | FASE 2: DESARROLLO DE BASE DE DATOS. | 101 |
| 4.3.3.1. | DISEÑO DE BASE DE DATOS. | 102 |
| 4.3.3.2. | CONSTRUCCIÓN DE PROTOTIPO EN JSF. | 102 |
| 4.3.3.3. | PRESENTACIÓN DE PROTOTIPO. | 102 |
| 4.3.3.4. | DISEÑO FINAL DE BASE DE DATOS. | 103 |
| 4.3.4. | FASE 3: DESARROLLO DE MÓDULO EJB. | 103 |
| 4.3.4.1. | APRENDIZAJE DE FRAMEWORK EJB. | 103 |
| 4.3.4.2. | CREACIÓN DEL MÓDULO EJB. | 103 |
| 4.3.4.3. | GENERACIÓN DE LA CAPA DE ACCESO A DATOS Y LÓGICA DE NEGOCIO INICIAL. | 104 |
| 4.3.5. | FASE 4: INTEGRACIÓN DE FRAMEWORKS. | 106 |
| 4.3.5.1. | CREACIÓN DE MÓDULO JSF. | 106 |
| 4.3.5.2. | INTEGRACIÓN DE JSF Y EJB. | 107 |
| 4.3.5.3. | DESARROLLO FINAL DE CAPA DE LÓGICA DE NEGOCIO. | 108 |
| 4.3.6. | FASE 5: CONSTRUCCIÓN FINAL DE LA APLICACIÓN. | 108 |
| 4.3.6.1. | PREPARACIÓN DE TEMPLATES. | 108 |
| 4.3.6.2. | DESARROLLO FINAL DE INTERFACES. | 109 |
| 4.3.6.3. | PRESENTACIÓN FINAL. | 109 |
| 4.3.6.4. | SUGERENCIAS Y MEJORAMIENTO. | 109 |
| 4.3.7. | FASE 6: IMPLEMENTACIÓN E IMPLANTACIÓN. | 110 |
| 4.3.7.1. | INSTALACIÓN DE SERVIDORES. | 110 |
| 4.3.7.2. | ALOJAMIENTO DE LA APLICACIÓN EN EL SERVIDOR. | 112 |
| 4.3.7.3. | PRUEBAS Y FUNCIONAMIENTO. | 115 |

CAPÍTULO V.

5. APLICACIÓN DE METODOLOGÍA ERCA WEB PARA EL DESARROLLO DEL SOFTWARE DE GESTIÓN INDUSTRIAL. 117

5.1. PLANIFICACIÓN. 117

| | | |
|------------|-----------------------------|-----|
| 5.1.1. | ANÁLISIS DE REQUERIMIENTOS. | 117 |
| 5.1.1.1. | OBJETIVOS DEL SRS. | 118 |
| 5.1.1.1.1. | Ámbito. | 118 |
| 5.1.1.1.2. | Referencias. | 118 |
| 5.1.1.1.3. | Visión general. | 119 |

| | | |
|-------------|---------------------------------------|------------|
| 5.1.1.2. | DESCRIPCIÓN GENERAL | 119 |
| 5.1.1.2.1. | Perspectiva del producto. | 119 |
| 5.1.1.2.2. | Funciones del producto. | 119 |
| 5.1.1.2.2. | Características del usuario. | 120 |
| 5.1.1.2.3. | Limitaciones generales. | 120 |
| 5.1.1.2.4. | Supuestos y dependencias. | 121 |
| 5.1.1.3. | REQUERIMIENTOS ESPECIFICOS | 121 |
| 5.1.1.3.1. | Requerimientos Funcionales. | 121 |
| 1. | Requerimiento 1. | 123 |
| 2. | Requerimiento 2. | 124 |
| 3. | Requerimiento 3. | 126 |
| 4. | Requerimiento 4. | 128 |
| 5. | Requerimiento 5. | 129 |
| 6. | Requerimiento 6. | 131 |
| 7. | Requerimiento 7. | 133 |
| 8. | Requerimiento 8 | 134 |
| 9. | Requerimiento 9. | 135 |
| 10. | Requerimiento 10. | 137 |
| 11. | Requerimiento 11. | 138 |
| 5.1.1.3.2. | Requerimientos de Interfaces Externas | 140 |
| 5.1.1.3.3. | Requerimientos de Rendimiento. | 140 |
| 5.1.1.3.4. | Limitaciones de Diseño. | 141 |
| 5.1.1.3.5. | Otros requerimientos | 142 |
| 5.1.2. | BOSQUEJO DE BASE DE DATOS. | 142 |
| 5.1.3. | APRENDIZAJE DE FRAMEWORK JSF. | 143 |
| 5.2. | DESARROLLO DE BASE DE DATOS. | 144 |
| 5.2.1. | DISEÑO DE BASE DE DATOS. | 144 |
| 5.2.2. | CONSTRUCCIÓN DE PROTOTIPO EN JSF. | 144 |
| 5.2.3. | PRESENTACIÓN DE PROTOTIPO. | 144 |
| 5.2.4. | DISEÑO FINAL DE BASE DE DATOS. | 145 |
| 5.2.4.1. | DIAGRAMA DE BASE DE DATOS. | 145 |
| 5.2.4.2. | PROCEDIMIENTOS ALMACENADOS. | 145 |
| 5.3. | DESARROLLO DE MÓDULO EJB. | 146 |
| 5.3.1. | APRENDIZAJE DE FRAMEWORK EJB. | 146 |
| 5.3.2. | CREACIÓN DEL MÓDULO EJB. | 146 |

| | | |
|-------------------------|---|------------|
| 5.3.3. | GENERACIÓN DE LA CAPA DE ACCESO A DATOS Y LÓGICA DE NEGOCIO INICIAL. | 147 |
| 5.3.3.1. | ACCESO A DATOS. | 147 |
| 5.3.3.2. | LÓGICA DE NEGOCIO. | 149 |
| 5.4. | INTEGRACIÓN DE FRAMEWORKS. | 150 |
| 5.4.1. | CREACIÓN DE MÓDULO JSF. | 151 |
| 5.4.2. | INTEGRACIÓN DE JSF Y EJB. | 151 |
| 5.4.3. | DESARROLLO FINAL DE LA LÓGICA DE NEGOCIO. | 152 |
| 5.5. | CONSTRUCCIÓN FINAL DE LA APLICACIÓN. | 152 |
| 5.5.1. | PREPARACIÓN DE TEMPLATES. | 152 |
| 5.5.2. | DESARROLLO FINAL DE INTERFACES. | 153 |
| 5.5.2. | PRESENTACIÓN FINAL. | 153 |
| 5.5.3. | SUGERENCIAS Y MEJORAMIENTO. | 153 |
| 5.6. | IMPLEMENTACIÓN E IMPLANTACIÓN. | 153 |
| 5.6.1. | INSTALACIÓN DE SERVIDORES. | 153 |
| 5.6.2. | ALOJAMIENTO DE SITIO WEB. | 160 |
| 5.6.3. | PRUEBAS Y FUNCIONAMIENTO. | 160 |
| CAPITULO VI. | | |
| 6. | RESULTADOS Y DISCUSIÓN. | 161 |
| 6.1. | HIPÓTESIS. | 161 |
| 6.1.1. | INTRODUCCIÓN. | 161 |
| 6.1.2. | DEFINICIÓN DE LOS PARÁMETROS DE COMPARACIÓN. | 162 |
| 6.1.3. | DEFINICIÓN DE LOS INDICADORES QUE MIDEN EL TIEMPO EN LA CONSTRUCCIÓN Y MANTENIMIENTO DE APLICACIONES WEB. | 164 |
| 6.1.4. | CRITERIOS DE EVALUACIÓN. | 167 |
| 6.1.5. | ANALISIS DE LOS PARÁMETROS DE COMPARACIÓN PARA LAS METODOLOGÍAS. | 168 |
| 6.1.5.1. | REUSABILIDAD. | 168 |
| 6.1.5.1.1. | PORTABILIDAD. | 168 |
| 6.1.5.1.2. | INDEPENDENCIA DE COMPONENTES. | 169 |
| 6.1.5.1.3. | FACILIDAD DE INTEGRACIÓN. | 171 |
| 6.1.5.1.4. | INTERPRETACIÓN. | 174 |
| 6.1.5.2. | GENERACIÓN DE CÓDIGO. | 175 |
| 6.1.5.2.1. | CREACIÓN DE CAPA DE ACCESO A DATOS. | 175 |
| 6.1.5.2.2. | GENERACIÓN DE CAPA DE LÓGICA DE NEGOCIO. | 177 |

| | | |
|------------|--|-----|
| 6.1.5.2.3. | LINEAS DE CÓDIGO. | 178 |
| 6.1.5.2.4. | CODIGO NO UTILIZABLE. | 182 |
| 6.1.5.2.5. | COMPLEJIDAD. | 185 |
| 6.1.5.2.6. | INTERPRETACIÓN. | 189 |
| 6.1.5.3. | MODULARIDAD. | 189 |
| 6.1.5.3.1. | MODELO VISTA CONTROLADOR | 189 |
| 6.1.5.3.2. | DISEÑO ESTRUCTURADO | 190 |
| 6.1.5.3.3. | CAPACIDAD DE DESCOMPOSICIÓN | 191 |
| 6.1.5.3.4. | COMPRESIÓN. | 193 |
| 6.1.5.3.5. | INTERPRETACIÓN. | 197 |
| 6.1.5.4. | MANTENIBILIDAD. | 197 |
| 6.1.5.4.1. | ESCALABILIDAD. | 197 |
| 6.1.5.4.2. | DISPONIBILIDAD DE DOCUMENTACIÓN. | 198 |
| 6.1.5.4.3. | CÓDIGO ENTENDIBLE. | 200 |
| 6.1.5.4.4. | ADAPTABILIDAD DE NUEVOS REQUERIMIENTOS. | 202 |
| 6.1.5.4.5. | INTERPRETACIÓN. | 205 |
| 6.1.5.5. | PRESENTACIÓN. | 205 |
| 6.1.5.5.1. | TEMPLATES PERSONALIZADOS. | 205 |
| 6.1.5.5.2. | COMPONENTES AJAX. | 208 |
| 6.1.5.5.3. | VALIDACIONES. | 212 |
| 6.1.5.5.4. | MENOR USO DE JAVA SCRIPT. | 214 |
| 6.1.5.5.5. | INTERPRETACIÓN. | 219 |
| 6.1.6. | ANÁLISIS GENERAL DE COMPROBACIÓN. | 219 |
| 6.1.6.1. | HIPÓTESIS. | 219 |
| 6.1.6.1.1. | DETERMINACIÓN DE VARIABLES. | 219 |
| 6.1.6.1.2. | OPERACIONALIZACIÓN DE LAS VARIABLES. | 219 |
| 6.1.6.2. | COMPARACIÓN DE LOS RESULTADOS OBTENIDOS. | 221 |
| 6.1.6.2.1. | APLICACIÓN DE CHI CUADRADO. | 223 |
| 6.1.6.2.2. | CÁLCULOS OBTENIDOS CON SOFTWARE ESTADÍSTICO SPSS | 227 |

CONCLUSIONES.

RECOMENDACIONES.

RESUMEN.

ABSTRACT.

ANEXOS.

INTERFACES DE APLICACIÓN SGM PRO.

ENCUESTA REALIZADA A EXPERTOS.

GLOSARIO.

BIBLIOGRAFÍA.

INDICE DE TABLAS.

| | |
|---|-----|
| <i>Tabla IV.I: Métricas de Requerimientos no Funcionales.</i> | 100 |
| <i>Tabla V.II: Características de los Usuarios.</i> | 120 |
| <i>Tabla V.III: Entradas Requerimiento 1.</i> | 123 |
| <i>Tabla V.IV: Procesos Requerimiento 1.</i> | 123 |
| <i>Tabla V.V: Salidas Requerimiento 1.</i> | 123 |
| <i>Tabla V.VI: Entradas Requerimiento 2.</i> | 124 |
| <i>Tabla V.VII: Procesos Requerimiento 2.</i> | 125 |
| <i>Tabla V.VIII: Salidas Requerimiento 2.</i> | 125 |
| <i>Tabla V.IX: Entradas Requerimiento 3.</i> | 126 |
| <i>Tabla V.X: Procesos Requerimiento 3.</i> | 126 |
| <i>Tabla V.XI: Salidas Requerimiento 3.</i> | 126 |
| <i>Tabla V.XII: Entradas Requerimiento 4.</i> | 128 |
| <i>Tabla V.XIII: Procesos Requerimiento 4.</i> | 128 |
| <i>Tabla V.XIV: Salidas Requerimiento 4.</i> | 129 |
| <i>Tabla V.XV: Entradas Requerimiento 5.</i> | 130 |
| <i>Tabla V.XVI: Procesos Requerimiento 5.</i> | 130 |
| <i>Tabla V.XVII: Salidas Requerimiento 5.</i> | 130 |
| <i>Tabla V.XVIII: Entradas Requerimiento 6.</i> | 131 |
| <i>Tabla V.XIX: Procesos Requerimiento 6.</i> | 132 |
| <i>Tabla V.XX: Salidas Requerimiento 6.</i> | 132 |
| <i>Tabla V.XXI: Entradas Requerimiento 7.</i> | 133 |
| <i>Tabla V.XXII: Procesos Requerimiento 7.</i> | 133 |
| <i>Tabla V.XXIII: Salidas Requerimiento 7.</i> | 133 |
| <i>Tabla V.XXIV: Entradas Requerimiento 8.</i> | 134 |
| <i>Tabla V.XXV: Procesos Requerimiento 8.</i> | 135 |
| <i>Tabla V.XXVI: Salidas Requerimiento 8.</i> | 135 |
| <i>Tabla V.XXVII: Entradas Requerimiento 9.</i> | 136 |
| <i>Tabla V.XXVIII: Procesos Requerimiento 9.</i> | 136 |
| <i>Tabla V.XXIX: Salidas Requerimiento 9.</i> | 136 |
| <i>Tabla V.XXX: Entradas Requerimiento 10.</i> | 137 |
| <i>Tabla V.XXXI: Procesos Requerimiento 10.</i> | 137 |
| <i>Tabla V.XXXII: Salidas Requerimiento 10.</i> | 138 |
| <i>Tabla V.XXXIII: Entradas Requerimiento 11.</i> | 139 |
| <i>Tabla V.XXXIV: Procesos Requerimiento 11.</i> | 139 |
| <i>Tabla V.XXXV: Salidas Requerimiento 11.</i> | 139 |

| | |
|--|-----|
| <i>Tabla VI.XXXVI: Lista de Parámetros.</i> | 162 |
| <i>Tabla VI.XXXVII: Lista de Parámetros que determinan el Tiempo.</i> | 164 |
| <i>Tabla VI.XXXVIII: Parámetro 1 – Reusabilidad.</i> | 165 |
| <i>Tabla VI.XXXIX: Parámetro 2 – Generación de Código.</i> | 165 |
| <i>Tabla VI.XL: Parámetro 3 – Modularidad.</i> | 166 |
| <i>Tabla VI.XLI: Parámetro 4 – Mantenibilidad.</i> | 166 |
| <i>Tabla VI.XLII: Parámetro 5 – Presentación.</i> | 167 |
| <i>Tabla VI.XLIII: Criterios de Evaluación General.</i> | 167 |
| <i>Tabla VI.XLIV: Evaluación del Índice de Portabilidad.</i> | 169 |
| <i>Tabla VI.XLV: Evaluación del Índice de Independencia de Componentes.</i> | 170 |
| <i>Tabla VI.XLVI: Evaluación del Índice Facilidad de Integración.</i> | 172 |
| <i>Tabla VI.XLVII: Tabla General del Parámetro de Reusabilidad.</i> | 173 |
| <i>Tabla VI.XLVIII: Evaluación del Índice de Creación de Capa de Acceso a Datos.</i> | 176 |
| <i>Tabla VI.XLIX: Evaluación del Índice de Creación de Capa de Lógica de Negocio.</i> | 178 |
| <i>Tabla VI.L: Criterio de Evaluación para medir las Líneas de Código.</i> | 179 |
| <i>Tabla VI.LI: Evaluación del Índice de Líneas de Código.</i> | 181 |
| <i>Tabla VI.LII: Criterio de Evaluación para medir las Líneas de Código no utilizable.</i> | 182 |
| <i>Tabla VI.LIII: Evaluación del Índice de Código no utilizable.</i> | 184 |
| <i>Tabla VI.LIV: Criterio de Evaluación para medir la Complejidad.</i> | 185 |
| <i>Tabla VI.LV: Evaluación del Índice de Complejidad.</i> | 186 |
| <i>Tabla VI.LVI: Valores de cada índice del Parámetro de Generación de Código.</i> | 187 |
| <i>Tabla VI.LVII: Evaluación del Índice de Modelo Vista Controlador.</i> | 189 |
| <i>Tabla VI.LVIII: Evaluación del Índice de Diseño Estructurado.</i> | 191 |
| <i>Tabla VI.LIX: Evaluación del Índice de Capacidad de Descomposición.</i> | 193 |
| <i>Tabla VI.LX: Evaluación del Índice de Comprensión del Código.</i> | 195 |
| <i>Tabla VI.LXI: Índices - Parámetro Modularidad.</i> | 195 |
| <i>Tabla VI.LXII: Evaluación del Índice de Escalabilidad.</i> | 198 |
| <i>Tabla VI.LXIII: Evaluación del Índice de Disponibilidad de Documentación.</i> | 199 |
| <i>Tabla VI.LXIV: Evaluación del Índice de Código Entendible.</i> | 201 |
| <i>Tabla VI.LXV: Evaluación del Índice de Adaptabilidad de Nuevos Requerimientos.</i> | 202 |
| <i>Tabla VI.LXVI: Tabla general de índices - parámetro Mantenibilidad.</i> | 203 |
| <i>Tabla VI.LXVII: Evaluación del Índice de Plantillas Personalizados.</i> | 207 |
| <i>Tabla VI.LXVIII: Evaluación del Índice de Componentes AJAX.</i> | 211 |
| <i>Tabla VI.LXIX: Evaluación del Índice de Validaciones.</i> | 213 |
| <i>Tabla VI.LXX: Evaluación del Índice de Menor Uso de JavaScript.</i> | 216 |
| <i>Tabla VI.LXXI: Tabla general de índices - parámetro Presentación</i> | 217 |
| <i>Tabla VI.LXXII: Operacionalización Conceptual.</i> | 219 |

| | |
|---|-----|
| <i>Tabla VI.LXXIII: Operacionalización Metodológica.</i> | 220 |
| <i>Tabla VI.LXXIV: Paramaros de Optimización de tiempo y cuantificación</i> | 221 |
| <i>Tabla VI.LXXV: Valores Observados y Totales.</i> | 223 |
| <i>Tabla VI.LXXVI: Valores Esperados y Totales.</i> | 224 |
| <i>Tabla VI.LXXVII: Resumen Valores Observados y valores Esperados.</i> | 225 |

INDICE DE FIGURAS.

| | |
|---|----|
| <i>Figura II.1: Diagrama de una aplicación JSF.</i> | 35 |
| <i>Figura II.2: Arquitectura MVC.</i> | 40 |
| <i>Figura II.3: Función de los objetos EJB.</i> | 52 |
| <i>Figura II.5: Archivo EJB-jar.</i> | 58 |
| <i>Figura III.6: Distribución por capas de la Aplicación.</i> | 62 |
| <i>Figura III.7: Base de Datos de muestra en NetBeans.</i> | 63 |
| <i>Figura III.8: Creando el Proyecto => Seleccionar Proyecto.</i> | 64 |
| <i>Figura III.9: Creando el Proyecto => Nombre y Ubicación del Proyecto.</i> | 64 |
| <i>Figura III.10: Creando el Proyecto => Servidor y Configuraciones.</i> | 65 |
| <i>Figura III.11: Creando el Proyecto => Proyectos creados por el IDE NetBeans.</i> | 66 |
| <i>Figura III.12: Creando las clases entidades => Seleccionar Otro.</i> | 67 |
| <i>Figura III.13: Creando las clases entidades => Seleccionar "Clase Entidad a partir de Base de Datos".</i> | 67 |
| <i>Figura III.14: Creando las clases entidades => Seleccionar Tablas de la Base de Datos.</i> | 68 |
| <i>Figura III.15: Creando las clases entidades => Seleccionando Tablas.</i> | 69 |
| <i>Figura III.16: Creando las clases entidades => Clases Entidad.</i> | 69 |
| <i>Figura III.17: Creando las clases entidades => Opciones de Mapeo.</i> | 70 |
| <i>Figura III.18: Creando las clases entidades => Clases entidades creadas.</i> | 70 |
| <i>Figura III.19: Creación de los MDB => Nuevo Session Bean.</i> | 71 |
| <i>Figura III.20: Creación de los MDB => Nombre y Localización.</i> | 72 |
| <i>Figura III.21: Creación de los MDB => Nuevo Message Drive Bean.</i> | 72 |
| <i>Figura III.22: Creación de los MDB => Nombre y Localización de los MDB.</i> | 73 |
| <i>Figura III.23: Creación de los MDB => Paquetes y sus respectivas clases.</i> | 73 |
| <i>Figura III.24: Creación de los MDB => Sessions Beans creados.</i> | 74 |
| <i>Figura III.25: Creación de los MDB => Creando la Persistencia.</i> | 74 |
| <i>Figura III.26: Creación de los MDB => Código insertado de Persistencia.</i> | 75 |
| <i>Figura III.27: Creación de los MDB => Menú para agregar nuevos métodos.</i> | 75 |
| <i>Figura III.28: Creación de los MDB => Creación de Nuevos Métodos.</i> | 76 |
| <i>Figura III.29: Creación de los MDB => Código de los nuevos métodos.</i> | 76 |
| <i>Figura III.30: Creación de los MDB => Código para métodos update y retrieve.</i> | 77 |
| <i>Figura III.31: Usando JSF y PrimeFaces => Seleccionando Framework JavaServer Faces.</i> | 78 |
| <i>Figura III.32: Usando JSF y PrimeFaces => Agregando Biblioteca Global.</i> | 78 |
| <i>Figura III.33: Usando JSF y PrimeFaces => Agregando archivo .jar a la biblioteca.</i> | 79 |
| <i>Figura III.34: Usando JSF y PrimeFaces => Agregando la Biblioteca al proyecto.</i> | 80 |
| <i>Figura III.35: Usando JSF y PrimeFaces => Administrador de Plantillas.</i> | 81 |
| <i>Figura III.36: Usando JSF y PrimeFaces => Crear Nuevo JSF Managed Bean.</i> | 83 |

| | |
|---|-----|
| <i>Figura III.37: Usando JSF y PrimeFaces => Call Enterprise Bean.</i> | 84 |
| <i>Figura III.38: Usando JSF y PrimeFaces => Nuevo Archivo XHTML.</i> | 86 |
| <i>Figura III.39: Usando JSF y PrimeFaces => Tabla JSF desde la Entidad.</i> | 87 |
| <i>Figura III.40: Usando JSF y PrimeFaces => Aplicaciones del IDE NetBeans.</i> | 87 |
| <i>Figura III.41: Usando JSF y PrimeFaces => Pantalla Cruda de la Lista de Clientes.</i> | 88 |
| <i>Figura III.42: Usando JSF y PrimeFaces => Pantalla de Lista de Clientes usando JSF.</i> | 89 |
| <i>Figura III.43: Usando JSF y PrimeFaces => JSF Table From Entity.</i> | 89 |
| <i>Figura III.44: Usando JSF y PrimeFaces => Archivo faces-config.xml con PageFlow.</i> | 90 |
| <i>Figura III.45: Usando JSF y PrimeFaces => Formulario de Ingreso de Clientes.</i> | 91 |
| <i>Figura IV.46: Fases de la Metodología ERCA Web.</i> | 97 |
| <i>Figura IV.47: Creación Módulo EJB – Creando Nuevo Proyecto</i> | 104 |
| <i>Figura IV.48: Acceso a Datos => Crear Clases Entidad a partir de base de datos.</i> | 105 |
| <i>Figura IV.49: Lógica de Negocio => Session Beans para Clases Entidad.</i> | 106 |
| <i>Figura IV.50: Seleccionando Framework JSF.</i> | 107 |
| <i>Figura IV.51: Integración de EJB a JSF.</i> | 108 |
| <i>Figura IV.52: Logo XAMPP</i> | 110 |
| <i>Figura IV.53: Servidor de Base de Datos MySQL.</i> | 111 |
| <i>Figura IV.54: Ubicación del Servidor GlassFish en el Menú Inicio.</i> | 112 |
| <i>Figura IV.55: Consola GlassFish.</i> | 113 |
| <i>Figura IV.56: Configuración para subir Aplicaciones.</i> | 113 |
| <i>Figura IV.57: Lista de Aplicaciones subidas al servidor.</i> | 114 |
| <i>Figura IV.58: Pantalla Inicial de Software de Gestión SGM Pro</i> | 114 |
| <i>Figura IV.59: Funcionamiento de Aplicación.</i> | 115 |
| <i>Figura V.60: Funciones de SGM Pro</i> | 119 |
| <i>Figura V.61: Lista de Estrategias.</i> | 124 |
| <i>Figura V.62: Aplicación SGM Pro.</i> | 125 |
| <i>Figura V.63: Insertar Estrategias</i> | 127 |
| <i>Figura V.64: Editar Estrategias</i> | 127 |
| <i>Figura V.65: Confirmación Eliminar Estrategia.</i> | 128 |
| <i>Figura V.66: Asignar Actividades</i> | 129 |
| <i>Figura V.67: Historial Plan de Mantenimiento.</i> | 131 |
| <i>Figura V.68: Plan de Mantenimiento.</i> | 131 |
| <i>Figura V.69: Notificar Actividad.</i> | 132 |
| <i>Figura V.70: Ingresar Fallas.</i> | 134 |
| <i>Figura V.71: Asignar Estrategias a Equipos.</i> | 135 |
| <i>Figura V.72: Buscar Estrategia Asignada a Tecnico</i> | 137 |
| <i>Figura V.73: Lista de Tecnicos en la Industria.</i> | 138 |

| | |
|---|-----|
| <i>Figura V.74: Actividades de Tecnico determinado.</i> | 138 |
| <i>Figura V.75: Actividades de Tecnico.</i> | 140 |
| <i>Figura V.76: Bosquejo de Base de Datos.</i> | 143 |
| <i>Figura V.77: Diseño de Base de Datos.</i> | 144 |
| <i>Figura V.78: Prototipo en JSF de Lista de Estrategias.</i> | 144 |
| <i>Figura V.79: Modelo Final de Base de Datos.</i> | 145 |
| <i>Figura V.80: Creando Aplicación Empresarial.</i> | 146 |
| <i>Figura V.81: Aplicación Empresarial.</i> | 147 |
| <i>Figura V.82: Clases entidad a partir de base de Datos.</i> | 147 |
| <i>Figura V.83: Selección de Tablas para el acceso a datos.</i> | 148 |
| <i>Figura V.84: Capa de Acceso a Datos.</i> | 148 |
| <i>Figura V.85: Código Generado por la capa de Acceso a Datos.</i> | 149 |
| <i>Figura V.86: Creando los Session Bean</i> | 149 |
| <i>Figura V.87: Seleccionando Clases Entidad</i> | 150 |
| <i>Figura V.88: Lógica de Negocio</i> | 150 |
| <i>Figura V.89: Creación de módulo JSF.</i> | 151 |
| <i>Figura V.90: EJB Integrado a JSF.</i> | 152 |
| <i>Figura V.91: Templates Utilizados.</i> | 152 |
| <i>Figura V.92: Interfaz Final.</i> | 153 |
| <i>Figura V.93: Panel de Control de XAMPP.</i> | 154 |
| <i>Figura V.94: Consola de Administración de GlassFish.</i> | 155 |
| <i>Figura V.95: Conexión a MySQL en GlassFish.</i> | 155 |
| <i>Figura V.96: Configuración de Conexión a MySQL.</i> | 156 |
| <i>Figura V.97: Parámetros de Conexión a MySQL.</i> | 156 |
| <i>Figura V.98: Lista de Conexiones.</i> | 157 |
| <i>Figura V.99: Lista de Recursos JDBC</i> | 157 |
| <i>Figura V.100: Nuevo Recurso JDBC.</i> | 158 |
| <i>Figura V.101: Lista de Recursos JDBC.</i> | 158 |
| <i>Figura V.102: Consola de GlassFish.</i> | 159 |
| <i>Figura V.103: Servidor Web GlassFish.</i> | 159 |
| <i>Figura V.104: Alojamiento de Sitio Web.</i> | 160 |
| <i>Figura VI.105: Índice de comparación Portabilidad.</i> | 169 |
| <i>Figura VI.106: Independencia de Componentes.</i> | 171 |
| <i>Figura VI.107: Índice Facilidad de Integración.</i> | 172 |
| <i>Figura VI.108: Parámetro de Reusabilidad de acuerdo a cada índice.</i> | 173 |
| <i>Figura VI.109: Reusabilidad en las dos metodologías.</i> | 174 |
| <i>Figura VI.110: Índice Creación de Capa de Acceso a Datos.</i> | 176 |

| | |
|--|-----|
| <i>Figura VI.111: CRUD de Entidad Ubicación Técnica.</i> | 177 |
| <i>Figura VI.112: Gráfica del Índice Generación de Capa de Lógica de Negocios.</i> | 178 |
| <i>Figura VI.113: Líneas de Código Método Inserción Técnicos Prototipo 1.</i> | 179 |
| <i>Figura VI.114: Líneas de Código Método Eliminación Técnicos Prototipo 1.</i> | 180 |
| <i>Figura VI.115: Líneas de Código Método Inserción Técnicos Prototipo 2.</i> | 180 |
| <i>Figura VI.116: Líneas de Código Método Eliminación Técnicos Prototipo 2.</i> | 181 |
| <i>Figura VI.117: Gráfica del Índice Líneas de Código.</i> | 182 |
| <i>Figura VI.118: Código no utilizable en la Metodología Propuesto en la Capa de Acceso a Datos.</i> | 183 |
| <i>Figura VI.119: Código no utilizable en la Metodología Propuesto en la Capa de Lógica de Negocios.</i> | 183 |
| <i>Figura VI.120: Gráfica del Índice Código no Utilizable.</i> | 184 |
| <i>Figura VI.121: Gráfica del Índice Complejidad.</i> | 186 |
| <i>Figura VI.122: Gráfica de Valores del Parámetro Generación de Código.</i> | 188 |
| <i>Figura VI.123: Gráfica Porcentual del Parámetro Generación de Código.</i> | 188 |
| <i>Figura VI.124: Índice de comparación Modelo Vista Controlador.</i> | 190 |
| <i>Figura VI.125: Índice de comparación Diseño Estructurado.</i> | 191 |
| <i>Figura VI.126: Página HTML prototipo 1.</i> | 192 |
| <i>Figura VI.127: Página XHTML y Backbean.</i> | 192 |
| <i>Figura VI.128: Índice de comparación Capacidad de Descomposición.</i> | 193 |
| <i>Figura VI.129: Comprensión de código prototipo 1.</i> | 194 |
| <i>Figura VI.130: Comprensión de código prototipo 2.</i> | 194 |
| <i>Figura VI.131: Índice de comparación – Comprensión del Código.</i> | 195 |
| <i>Figura VI.132: Índices de comparación – Modularidad.</i> | 196 |
| <i>Figura VI.133: Porcentajes totales - Modularidad.</i> | 196 |
| <i>Figura VI.134: Índice de comparación Templates Personalizados.</i> | 198 |
| <i>Figura VI.135: Índice de comparación Disponibilidad de Documentación.</i> | 199 |
| <i>Figura VI.136: Código de Método Insertar Prototipo 1.</i> | 200 |
| <i>Figura VI.137: Código Generado por Framework.</i> | 201 |
| <i>Figura VI.138: Índice de comparación Código Entendible.</i> | 201 |
| <i>Figura VI.139: Índice de comparación Adaptabilidad de Nuevos Requerimientos.</i> | 203 |
| <i>Figura VI.140: Índices de comparación - Mantenibilidad</i> | 204 |
| <i>Figura VI.141: Porcentajes totales – Mantenibilidad.</i> | 204 |
| <i>Figura VI.142: Hojas de Estilo prototipo 1.</i> | 206 |
| <i>Figura VI.143: Referencias - Hojas de Estilo prototipo 1.</i> | 206 |
| <i>Figura VI.144: Templates prototipo 2.</i> | 207 |
| <i>Figura VI.145: Referencias - Templates prototipo 2.</i> | 207 |
| <i>Figura VI.146: Índice de comparación Templates Personalizados.</i> | 208 |
| <i>Figura VI.147: Ajax en JSP.</i> | 209 |

| | |
|--|-----|
| <i>Figura VI.148: Función AJAX.</i> | 209 |
| <i>Figura VI.149: AJAX HTML.</i> | 210 |
| <i>Figura VI.150: AJAX XHTML prototipo 2.</i> | 210 |
| <i>Figura VI.151: AJAX PRIMEFACES.</i> | 211 |
| <i>Figura VI.152: Índice de comparación Componentes AJAX.</i> | 211 |
| <i>Figura VI.153: Validaciones prototipo 1.</i> | 212 |
| <i>Figura VI.154: Validaciones prototipo 2.</i> | 213 |
| <i>Figura VI.155: Índice de comparación Validaciones.</i> | 214 |
| <i>Figura VI.156: JavaScript prototipo 1.</i> | 215 |
| <i>Figura VI.157: JavaScript prototipo 2.</i> | 216 |
| <i>Figura VI.158: Índice de comparación Menor Uso de Javascript.</i> | 217 |
| <i>Figura VI.159: Índices de comparación -Presentación.</i> | 218 |
| <i>Figura VI.160: Porcentajes Totales - Presentación.</i> | 218 |
| <i>Figura VI.161: Tabla General de Valores.</i> | 222 |
| <i>Figura VI.162: Porcentaje General.</i> | 223 |
| <i>Figura VI.163: Tabla de Chi-Cuadrado.</i> | 226 |
| <i>Figura VI.164: Ingreso de datos al Spss.</i> | 227 |
| <i>Figura VI.165: Optimización del tiempo – Metodologías.</i> | 227 |
| <i>Figura VI.166: Resultado Chi cuadrado SPSS.</i> | 228 |

AGRADECIMIENTO

Agradezco primero a Dios por su misericordia, amor y por permitirme cumplir uno de mis sueños más anhelados dándome las fuerzas necesarias para avanzar en el camino hacia mi superación, convirtiéndose en el pilar más importante en mi vida. A mis padres y hermanos por su apoyo incondicional en todo momento, siendo así las columnas fundamentales en mi vida, por sus consejos, enseñanzas y principios, los cuales han contribuido en todo momento para convertirme en la persona que soy hoy en día. A la Escuela Superior Politécnica de Chimborazo por acogerme en sus aulas e impartirme todos los conocimientos que hoy hacen de mi un hombre de bien para la sociedad

Alex Erazo

Doy gracias al ser más importante DIOS Que guía cada momento de mi vida. A la Escuela Superior Politécnica de Chimborazo que me permitió estudiar e investigar para el bien propio y de la sociedad. Un especial agradecimiento a los Ingenieros Danny Velasco y Raúl Rosero Miembros del Tribunal por la ayuda y colaboración para la realización de esta tesis. Al Hospital General Docente de Riobamba por permitirnos trabajar conjuntamente en el Área de Mantenimiento.

Vicente Carrillo

DEDICATORIA

*Dedico este trabajo investigativo y toda mi carrera
politécnica, primero a Dios, ya que por medio de Él he
alcanzado este logro tan anhelado, renovando mis fuerzas
en todo momento y estando conmigo siempre.*

*Dedico también el presente trabajo a mis padres, hermanos
y amigos, quienes nunca dudaron de mis capacidades, más
bien, fueron un puntal de motivación en mi vida,
demostrando en toda ocasión su confianza depositada en mi*

Alex Erazo

*A **DIOS**, a mis padres, hermanos, abuelitos y de manera
exclusiva a mis sobrinos Pablito y Sebastián que con el
andar cotidiano, me fueron aprovisionando de
herramientas especiales, para que el diario construir de
mi vida me sea más interesante, a la experiencia de la vida
misma, a todas y cada una de mis amistades y a los que me
han ofrecido sus experiencias para construir las mías, a
todo ser humano que cree en sí mismo y en sus propósitos.*

Vicente Carrillo

FIRMAS RESPONSABLES Y NOTAS.

| NOMBRES | FIRMAS | FECHA |
|---|---------------|--------------|
| Ing. Iván Menes | ----- | ----- |
| DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA | | |
| Ing. Raúl Rosero Miranda | ----- | ----- |
| DIRECTOR DE LA ESCUELA DE INGENIERÍA EN SISTEMAS | | |
| Ing. Danny Velasco | ----- | ----- |
| DIRECTOR DE TESIS | | |
| Ing. Raúl Rosero | ----- | ----- |
| MIEMBRO DE TESIS | | |
| Tec. Carlos Rodriguez | ----- | ----- |
| DIR. DPTO DOCUMENTACION | | |

Nota:

ÍNDICE DE ABREVIATURAS.

| | |
|--------|---|
| ADF | Application Development Framework |
| API | Application Programming Interface. |
| ASP | Active Server Page |
| AWT | Abstract Window Toolkit |
| CGI | Common Gateway Interface |
| CRUD | Create Read Update and Delete |
| DI | Dependence Injection |
| EJB | Enterprise Java Beans. |
| ERP | Planificación de Recursos Empresariales |
| ESPOCH | Escuela Superior Politécnica de Chimborazo. |
| HQL | Hibernate Query Language |
| HTML | Hipertext Markup Language |
| HTTP | Hipertext Transfer Protocol |
| IBM | International Business Machines |
| IDE | Integrated Development Environment |
| IoC | Inversion of Control |
| ISO | Organización Internacional de Normalización |
| J2EE | Java Enterprise Edition |
| J2SE | Java Standard Edition |
| JAAS | Java Authorization and Authentication Service |
| jBPM | JBoss Business Process Manager |

| | |
|------|---------------------------------------|
| JDBC | Java Database Connectivity |
| JDK | Java Development Kit |
| JDO | Java Data Objects |
| JMS | Java Message Service |
| JPA | Java Persistence API |
| JPOX | Java Persistent Objects |
| JSF | Java Server Faces |
| JSP | Java Server Pages |
| LDAP | Lightweight Directory Access Protocol |
| MDB | Message Driven Bean |
| MSF | Microsoft Solution Framework |
| MVC | Modelo Vista Controlador |
| ODBC | Open Database Connectivity |
| OJB | Object Relational Bridge |
| ORM | Object-Relational Mapping |
| PHP | Personal Home Page |
| POA | Programación Orientada a Aspectos |
| POJO | Plain Object Java Old |
| RMI | Remote Method Invocation |
| RUP | Rational Unified Process |
| SQL | Estructured Query Language |
| UI | Interfaz de Usuario |

| | |
|-----|-----------------------------------|
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| XML | Extensible Markup Language |
| XUL | XML-based User-Interface Language |
| XP | Extreme Programming |

“Nosotros, Vicente Aníbal Carrillo Tixe, Alex Ignacio Erazo Vivero, somos los responsables del contenido, ideas y resultados planteados en el presente proyecto de tesis, y el patrimonio intelectual del mismo pertenecen a la Escuela Superior Politécnica de Chimborazo”.

Vicente Anibal Carrillo Tixe

Alex Ignacio Erazo Vivero

CAPÍTULO I.

1. MARCO REFERENCIAL.

1.1.ANTECEDENTES.

El Internet y la Web han influido enormemente tanto en el mundo de la informática como en la sociedad en general. La web, en menos de 10 años ha transformado los sistemas informáticos: ha roto las barreras físicas (debido a la distancia), económicas y lógicas (debido al empleo de distintos sistemas operativos, protocolos, etc.), y ha abierto todo un abanico de nuevas posibilidades. Una de las áreas que más expansión está teniendo en la Web en los últimos años son las aplicaciones web.

Las aplicaciones web permiten la generación automática de contenido, la creación de páginas personalizadas según el perfil del usuario o el desarrollo del comercio electrónico. Además permite interactuar con los sistemas informáticos de gestión de una empresa, como puede ser de gestión de clientes, contabilidad o inventario a través de una página web.

Estas aplicaciones se encuadran dentro de las arquitecturas cliente / servidor, donde un ordenador solicita servicios (el cliente) y otro está a la espera de recibir solicitudes y las responde (el servidor).

Existen multitud de tecnologías que se pueden emplear para programar las aplicaciones web.

A través de la historia se ha tenido varias tecnologías para realizar las diferentes aplicaciones web; así por ejemplo desde el inicio únicamente existía información estática utilizando HTML básico y obteniendo como resultado aplicaciones pobres.

Posteriormente ha evolucionado la tecnología web teniendo mejoras como son: las presentaciones adecuadas por lado del cliente, introduciendo HTML dinámico y JAVASCRIPT.

Estos cambios fueron el inicio de una gran evolución dentro del ámbito de aplicaciones web; así aparecieron nuevas tecnologías como son: CGIs, Java Server Pages, Serverlets, etc. las cuales implementaron codificación por lado del servidor.

El CGIs (Common Gateway Interface) es la primera tecnología de programación de aplicaciones web, un Script CGI se ejecutaba en el servidor, y tenía la capacidad de retornar información.

Posteriormente aparecieron otras tecnologías para solucionar los puntos débiles de los CGIs, así surgieron alternativas como: JSP, Serverlets, ASP, PHP. Mediante estas tecnologías se dota a los servidores un intérprete de algún lenguaje de programación que permita incluir el código en las páginas de forma que lo ejecute el servidor, reduciendo el intervalo de respuesta; En consecuencia también surge la aparición de nuevas arquitecturas, la más utilizadas eran las que permiten mezclar los 2 sistemas: un lenguaje integrado que permita al servidor interpretar comandos "incrustados" en las páginas HTML y, además, un sistema de ejecución de programas mejor enlazado con el servidor, que no implique los problemas de rendimiento propios de los CGIs.

En la actualidad han surgido tecnologías que permiten realizar las aplicaciones web de una forma rápida y sencilla permitiendo el ahorro de recursos y tiempo, así como la tecnología Java Server Faces. Esta tecnología trabaja conjuntamente con otras tecnologías como Java Server Page permitiendo simplificar de forma significativa la construcción y el mantenimiento de una aplicación web.

Así como también tecnologías como Enterprise Java Beans que se encargan de manejar módulos concretos de una aplicación web como son: la Seguridad, Sesiones, Persistencia Transacciones, etc.

1.2.JUSTIFICACIÓN.

1.2.1. JUSTIFICACIÓN TEÓRICA.

La constante, casi frenética, evolución que está sufriendo el desarrollo de Internet y el aumento del número de usuarios que lo utiliza, está causando un gran auge en el desarrollo de aplicaciones Web. Este tipo de software permite que un determinado usuario pueda acceder a información almacenada en servidores Web a través de Internet o de una intranet. El factor clave radica en la posibilidad de poder acceder a una determinada aplicación, de forma transparente al usuario y desde cualquier lugar. Este valor añadido hace que muchas empresas opten, por ejemplo, realizar intranets corporativas, permitiendo el despliegue de aplicaciones Web para compartir y gestionar información.

Desde el punto de vista de los profesionales que se dedican al mundo de la informática y en particular, al desarrollo de aplicaciones Web, la creciente evolución del sector conlleva la mejora e innovación de los lenguajes y herramientas disponibles para desarrollar este tipo de software. Además, muchos de los problemas que se presentan en el desarrollo de aplicaciones Web no existen en las aplicaciones de escritorio y se debe, fundamentalmente, a que el protocolo HTTP, en el cual se basa la comunicación cliente-servidor, es un protocolo sin estado. Con lo que cada vez más, aparecen nuevas herramientas que hacen que la construcción y diseño de un entorno Web sea más fácil y rápida.

Una de los lenguajes de programación que proporcionan las características establecidas anteriormente es Java, ya que es un lenguaje altamente extendido, independiente de la plataforma hardware/software, orientado a objetos, con gran cantidad de herramientas que dan soporte a su programación y además es relativamente sencillo de utilizar y mantener.

Uno de los aspectos más interesantes de Java es el ritmo al que evolucionan tanto sus tecnologías como las arquitecturas diseñadas para soportarlas. Cada poco tiempo aparecen nuevas librerías, herramientas, frameworks, patrones, etc., y se hace muy difícil elegir entre ellos y, sobre todo, combinarlos de una manera eficiente que aproveche toda su flexibilidad y potencia.

Es importante el estudio y selección de un conjunto de frameworks J2EE de código libre. La arquitectura final de aplicación Web depende de la separación en capas y partes que se decida, así como de los patrones de diseño utilizados por las diferentes (una o varias) herramientas que se unen para el desarrollo del producto final.

Este trabajo investigativo pretende establecer una metodología para el desarrollo de una aplicación web, de modo que la construcción de ésta, sea de una manera más rápida, ordenada, mantenible y reutilizable, utilizando nuevas tecnologías para el desarrollo de aplicaciones web, que proporcionen estas funcionalidades como son Java Server Faces y Enterprise Java Bean

Con el desarrollo de esta metodología se proporcionará una guía para el desarrollo de aplicaciones web de forma óptima y ágil a los nuevos desarrolladores.

Además se presentará un caso práctico en el cual se utilizará la metodología plasmada de modo que logre alcanzar los beneficios propuestos.

1.2.2. JUSTIFICACIÓN APLICATIVA.

En la actualidad, las grandes industrias, en su laborar diario, utilizan un sin número de máquinas para la producción de los productos que se desarrollan, así como de técnicos experimentados en la producción, sin embargo muchas de estas industrias no toman en cuenta el mantenimiento que se debe llevar diariamente en los elementos que intervienen en esta producción o incluso algunas industrias las llevan manualmente. Por esto la necesidad de desarrollar un software que permita automatizar estos procesos de mantenimiento, permitiendo una optimización de los recursos de la institución, aumentando la productividad y rentabilidad de la misma.

La implementación del Sistema Web De Gestión de Mantenimiento, en el HOSPITAL PROVINCIAL GENERAL DOCENTE RIOBAMBA, permitirá solucionar los problemas de control y Gestión Preventivo del Mantenimiento Industrial.

Además que en la actualidad se requiere que en empresas de este tipo se lleve el Control y la Gestión del Mantenimiento Industrial de manera Informática para que dicha empresa se pueda certificar con las Normas ISO 9000 que son normas de Calidad.

Lugar de la aplicación:

HOSPITAL PROVINCIAL GENERAL DOCENTE RIOBAMBA

Alcance:

Este tema de tesis se fundamenta en el desarrollo de una metodología para la construcción de una aplicación web mediante la Integración del Framework JSF y EJB, con lo cual se desarrollará el Sistema Web de Gestión de Mantenimiento.

Limitación:

Para la parte aplicativa se limitará en los siguientes módulos:

- Módulo de Cuentas de Usuario.
 - Registro de Usuarios.
- Módulo De Gestión. En este módulo se tienen varios sub-módulos como son:
 - Técnicos.
 - Equipos.
 - Componentes.
 - Fallas Equipos.
 - Estrategias.
 - Ubicaciones Técnicas.

Todos estos módulos constaran de Ingresar, Modificar, Eliminar, Seleccionar.

- Módulo de Planificación y Control de Mantenimiento Industrial
 - Asignación de Estrategias a Equipos.
 - Asignación de Técnicos a Actividades.
 - Asignación de Componentes a equipos.
- Reportes.
 - Reporte Plan de Mantenimiento.
 - Reporte de Tareas Asignadas

1.3.OBJETIVOS.

1.3.1. GENERAL.

- Desarrollar una metodología para construir una aplicación web mediante la integración de Java Server Face y Enterprise Java Beans, y aplicar en el desarrollo de un sistema web para el Hospital Provincial General Docente Riobamba.

1.3.2. ESPECÍFICOS.

- Analizar el framework Java Server Faces y el API Enterprise Java Bean.
- Estudiar y desarrollar la integración de Java Server Faces con Enterprise Java Bean.
- Desarrollar la metodología para la construcción de una aplicación web mediante la integración de Java Server Face y Enterprise Java Bean.
- Desarrollar el módulo de Plan de Mantenimiento en un software de gestión de mantenimiento, utilizando la tecnología de Java Server Faces integrado con Enterprise Java Bean.

1.4.HIPÓTESIS.

Mediante el desarrollo de una metodología para la construcción de una aplicación web a través de la integración de las tecnologías JSF y EJB se minimizará el tiempo en la construcción y mantenimiento de aplicaciones web.

CAPÍTULO II.

2. JAVA SERVER FACES Y ENTERPRISE JAVA BEAN.

2.1.JAVA SERVER FACES.

2.1.1. INTRODUCCIÓN A JSF.

Tradicionalmente, las aplicaciones web se han codificado mediante páginas JSP¹ que reciben peticiones a través de formularios y construyen como respuesta páginas HTML² mediante ejecución directa o indirecta, a través de bibliotecas de etiquetas de código JAVA, lo que permite acceder a bases de datos para obtener los resultados a mostrar, con el fin de realizar operaciones marginales como insertar o modificar registros en tablas relacionales.

Java Server Faces (JSF) pretende facilitar la construcción de estas aplicaciones proporcionando un entorno de trabajo (framework) vía web que gestione las acciones producidas por el usuario en su página HTML y las traduce a eventos que son enviados al servidor con el objetivo de regenerar la página original y reflejar los cambios pertinentes provocados por dichas acciones.

Cualquier evento realizado sobre una página JSF incurre en una carga sobre la red, ya que el evento debe enviarse a través de ésta al servidor, y la respuesta de éste debe devolverse al cliente; por ello, el diseño de aplicaciones JSF debe hacerse con cuidado

¹ JSP: Tecnología Java que permite generar contenido dinámico para web.

² HTML: Lenguaje de Marcado predominante para la elaboración de páginas web

cuando se pretenda poner las aplicaciones a disposición del mundo entero a través del internet. Aquellas aplicaciones que vayan a ser utilizadas en una intranet podrán aprovecharse de un mayor ancho de banda y producirán una respuesta mucho más rápida.

2.1.2. APLICACIONES JAVA SERVER FACES.

En su mayoría, las aplicaciones JSF son como cualquier otra aplicación web Java. Se ejecutan en un contenedor de servlets³ de Java y, típicamente, contienen:

- Componentes JavaBeans (llamados objetos del modelo en tecnología JSF) conteniendo datos y funcionalidades específicas de la aplicación.¹
- Oyentes de Eventos.
- Páginas, (principalmente páginas JSP).
- Clases de utilidad del lado del servidor, como beans⁴ para acceder a las bases de datos.

Además posee:

- Una librería de etiquetas personalizadas para dibujar componentes UI en una página.
- Una librería de etiquetas personalizadas para representar manejadores de eventos, validadores y otras acciones.
- Componentes UI representados como objetos con estado en el servidor.

Toda aplicación JSF debe incluir una librería de etiquetas personalizadas que define las etiquetas que representan componentes UI, así como una librería de etiquetas para controlar otras acciones importantes, como validadores y manejadores de eventos. La implementación de JSF de Sun proporciona estas dos librerías. La librería de etiquetas de componentes elimina la necesidad de codificar componentes UI en HTML u otro lenguaje de marcas, lo que se traduce en el empleo de componentes completamente reutilizables. Y la librería principal (core) hace fácil registrar eventos, validadores y otras acciones de los componentes.

³ Servlets: Permiten generar todas las páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

⁴ Beans: Componente hecho en software que se puede reutilizar.

Como librería de etiquetas de componentes puede usarse la librería **html_basic** incluida con la implementación de referencia de la tecnología JSF, aunque también es posible definir una librería de etiquetas personalizadas que dibuje componentes propios o que proporcione una salida distinta a HTML.

Otra ventaja importante de las aplicaciones JSF es que los componentes UI de la página están representados en el servidor como objetos con estado. Esto permite a la aplicación manipular el estado del componente y conectar los eventos generados por el cliente a código en el lado del servidor.

Finalmente, la tecnología JSF permite convertir y validar datos sobre componentes individuales e informar de cualquier error antes de que se actualicen los datos en el lado del servidor.

Debido a la división de labores que permite el diseño de la tecnología JSF, el desarrollo y mantenimiento de una aplicación JSF se puede realizar muy rápida y fácilmente.

Aunque en muchos equipos, los desarrolladores individuales pueden interpretar más de uno de esos roles, resulta muy útil considerar la tecnología JSF desde varias perspectivas basadas en la responsabilidad principal que tiene cada participante:

- **Autores de páginas**, que utilizan un lenguaje de marcas, como HTML, para construir páginas para aplicaciones web. Cuando se utiliza la tecnología JSF, los autores de páginas casi siempre usarán exclusivamente la librería de etiquetas.
- **Desarrolladores de aplicaciones**, que programan los objetos del modelo, los manejadores de eventos, los validadores, y la navegación de páginas. Los desarrolladores de aplicaciones también pueden proporcionar las clases de utilidad necesarias.
- **Escritores de componentes**, que tienen experiencia en programar interfaces de usuario y prefieren crear componentes personalizados utilizando un lenguaje de programación. Esta gente puede crear sus propios componentes desde cero, o puede extender los componentes estándares proporcionados por JSF.

- **Vendedores de herramientas**, que proporcionan herramientas que mejoran la tecnología JSF para hacer incluso más sencilla la construcción de interfaces de usuario en el lado servidor.

Para las aplicaciones web pueden ser muy convenientes frameworks como JSF, Struts, Spring, etc., pero éstos no servirán (a no ser a costa de un gran esfuerzo) para hacer portales. Para este segundo caso sería más adecuado usar gestores de contenidos como Lenya, OpenCMS, etc.

2.1.2.1. CARACTERÍSTICAS.

La tecnología JSF constituye un marco de trabajo de interfaces de usuario del lado del servidor para aplicaciones web basadas en tecnología Java y en el patrón MVC (Modelo Vista Controlador).

Los principales componentes de la tecnología JSF son:

- Una API y una implementación de referencia para:
 - Representar componentes de interfaz de usuario y manejar su estado.
 - Manejar eventos, validar en el lado del servidor y convertir datos.
 - Definir la navegación entre páginas.
 - Soportar internacionalización y accesibilidad.
 - Proporcionar extensibilidad para todas estas características.
- Una librería de etiquetas Java Server Pages personalizadas para dibujar componentes de interfaces de usuario dentro de una página JSP.

Este modelo de programación bien definido y la librería de etiquetas para componentes de interfaz de usuario facilita de forma significativa la tarea de la construcción y mantenimiento de aplicaciones web con interfaces de usuario en el lado del servidor.

Con un mínimo esfuerzo es posible:

- Conectar eventos generados en el cliente a código de la aplicación en el lado servidor.
- Mapear componentes de interfaz de usuario a una página de datos en el lado servidor.
- Construir una interfaz de usuario con componentes reutilizables y extensibles.

La interfaz de usuario que se crea con la tecnología JSF se ejecuta en el servidor y se renderiza en el cliente. Ver figura II.1.

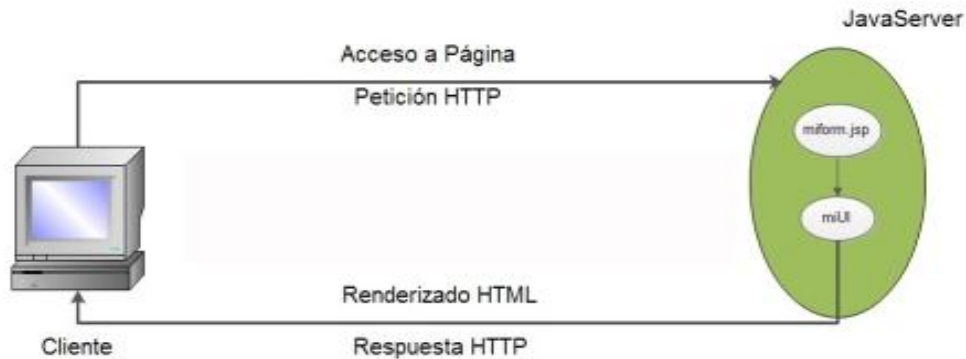


Figura II.1: Diagrama de una aplicación JSF.

La página JSP **miForm.jsp**, especifica los componentes de la interfaz de usuario mediante etiquetas personalizadas definidas por la tecnología JSF. La interfaz de usuario de la aplicación web (representada por miUI en la figura) maneja los objetos referenciados por la página JSP, que pueden ser de los siguientes tipos:

- Objetos componentes que mapean las etiquetas sobre la página JSP.
- Oyentes de eventos, validadores y conversores registrados y asociados a los componentes.
- Objetos del modelo que encapsulan los datos y las funcionalidades de los componentes específicos de la aplicación (lógica del negocio).

2.1.2.2.BENEFICIOS DE JSF.

Una de las ventajas de que JSF sea una especificación estándar es que pueden encontrarse implementaciones de distintos fabricantes. Esto permite no vincularse exclusivamente con un proveedor concreto, y poder seleccionar el que más interese en cada caso según el número de componentes que suministra, el rendimiento de éstos, soporte proporcionado, precio, política de evolución, etc.

JSF trata la vista (la interfaz de usuario) de una forma algo diferente a lo que se está acostumbrado en aplicaciones web, ya que este tratamiento es mucho más cercano al estilo de Java Swing, Visual Basic o Delphi, donde la programación de la interfaz se

hacer a través de componentes y está basada en eventos (pulsación de un botón, cambio en el valor de un campo, etc.).

JSF es muy flexible. Por ejemplo, permite crear nuestros propios componentes, y/o crear nuestros propios renderizadores para pintar los componentes en la forma que más convenga.

Una de las grandes ventajas de la tecnología JSF es que ofrece una clara separación entre el comportamiento y la presentación. Las aplicaciones web construidas con tecnología JSP conseguían parcialmente esta separación. Sin embargo, una aplicación JSP no puede mapear peticiones HTTP⁵ al manejo de eventos específicos de los componentes o manejar elementos UI como objetos con estado en el servidor. La tecnología JSF permite construir aplicaciones web que introducen realmente una separación entre el comportamiento y la presentación, separación sólo ofrecida tradicionalmente por arquitecturas UI del lado del cliente.

Separar la lógica de negocio de la presentación también permite que cada miembro del equipo de desarrollo de la aplicación web se centre en su parte asignada del proceso de diseño, y proporciona un modelo sencillo de programación para enlazar todas las piezas. Por ejemplo, personas sin experiencia en programación pueden construir páginas JSF usando las etiquetas de componentes UI que ésta tecnología ofrece, y luego enlazarlas con código de la aplicación sin escribir ningún script ni algo parecido.

Otro objetivo importante de la tecnología JSF es mejorar los conceptos familiares de componente-UI y capa-web sin limitarnos a una tecnología de script particular o un lenguaje de marcas. Aunque la tecnología JSF incluye una librería de etiquetas JSP personalizadas para representar componentes en una página JSP, las APIs⁶ de JSF se han creado directamente sobre el API Java Servlet. Esto permite, teóricamente, hacer algunas cosas avanzadas: usar otra tecnología de presentación junto a JSP, crear nuestros propios componentes personalizados directamente desde las clases de componentes, y generar salida para diferentes dispositivos cliente, entre otras.

⁵ HTTP: Método más común de intercambio de información en la www, el método mediante el cual se transfieren las páginas web a un ordenador.

⁶ API: Conjunto de funciones y procedimientos para ser utilizado por otro software como una capa de abstracción.

En otras palabras, la tecnología JSF proporciona una rica arquitectura para manejar el estado de los componentes, procesar los datos, validar la entrada del usuario, y manejar eventos.

2.1.2.3. ETIQUETAS JSF.

JSF dispone de un conjunto básico de etiquetas que permiten crear fácilmente componentes dinámicos en las páginas web. Estas etiquetas son:

- **h:commandButton.** Un botón al que se puede asociar una acción.
- **h:commandLink.** Un enlace hipertexto al que se puede asociar una acción.
- **h:dataTable.** Crea una tabla de datos dinámica con los elementos de una propiedad de tipo Array o Map del bean.
- **h:form.** Define el formulario JSF en la página JSP.
- **h:graphicImage.** Muestra una imagen jpg o similar.
- **h:inputHidden.** Incluye un campo oculto del formulario.
- **h:inputSecret.** Incluye un campo editable de tipo contraseña (no muestra lo que se escribe).
- **h:inputText.** Incluye un campo de texto normal.
- **h:inputTextarea.** Incluye un campo de texto multilínea.
- **h:message.** Imprime un mensaje de error en la página (si se ha producido alguno).
- **h:messages.** Imprime varios mensajes de error en la página, si se han producido.
- **h:outputFormat.** Muestra texto parametrizado. Utiliza la clase `java.text.MessageFormat` de formateo.
- **h:outputLabel.** Muestra un texto fijo.
- **h:outputLink.** Crea un enlace hipertexto.
- **h:outputText.** Crea una casilla de texto.
- **h:panelGrid.** Crea una tabla con los componentes incluidos en el `panelGrid`.
- **h:panelGroup.** Agrupa varios componentes para que cierto componente los trate como un único componente (por ejemplo para meter varios componentes en una celda de un `panelGrid`).
- **h:selectBooleanCheckbox.** Crea una casilla con dos estados: activado y desactivado.

- **h:selectManyCheckbox.** Crea un conjunto de casillas activables.
- **h:selectManyListbox.** Crea una lista que permite seleccionar múltiples elementos.
- **h:selectManyMenu.** Crea una lista desplegable de selección múltiple.
- **h:selectOneListbox.** Crea una lista en la que se puede seleccionar un único elemento.
- **h:selectOneMenu.** Crea una lista desplegable de selección.
- **h:selectOneRadio.** Crea una lista de botones, redondos normalmente, excluyentes.

2.1.2.4.BACKBEANS.

A las clases java que se asocian a los formularios JSF se les denomina backend beans ya que son los beans (clases java) que están detrás del formulario. Estos beans se referencian en el fichero de configuración de JSF en el apartado de managed beans, ya que son beans gestionados por el controlador JSF. Éste se encarga de su construcción y destrucción automáticas cuando es necesario.

2.1.2.4.1. ESTRUCTURA DE LAS PÁGINAS.

En su versión más sencilla, cada página JSF está formada por una página JSP que contiene un formulario (HTML FORM) y un backbean.

El controlador JSF registra en el servidor de aplicaciones un tipo especial de petición, típicamente *.jsf, que estará asociado a estas páginas.

El primer caso comienza cuando el usuario realiza en su navegador una petición de navegación a una URL de tipo *.jsf. Cuando al servidor web llega una petición del tipo página JSF, el controlador JSF entra en funcionamiento.

Primero comprueba si es la primera vez que se accede a dicha página. Si es así, carga la página JSP asociada pagina.jsp y la procesa construyendo en memoria la representación de los controles de la página. Tras esta etapa JSF sabe cómo construir el código HTML de salida y la lista de controles de usuario que la cumplen, es decir, sabe lo que contiene y cómo pintarla.

El siguiente paso es asociarle los backbeans. Para ello, del procesamiento de la página JSP, el controlador ha obtenido la lista de backbeans asociados, por lo que procede a buscarlos en sus correspondientes ámbitos de la aplicación como la request y la session. Los beans que no existan se crean llamando a los constructores de sus clases, definidos en la sección de managed beans del fichero de configuración de JSF.

El tercer paso es dar valores a las propiedades de los elementos JSF de la página. Aquí juega un papel fundamental el lenguaje de expresiones de JSF, que es parecido al lenguaje de expresiones que se permite en las páginas JSP normales. En su versión más sencilla una expresión JSF sería del tipo `#{mibackbean.propiedad}`.

Finalmente el servidor devuelve al usuario una página creada a partir de una página JSP que incluye normalmente etiquetas JSF, cuyos valores se extraerán del backbean asociado, ahora ya actualizados.

2.1.2.5.JSF Y AJAX.

JSF es un framework que lanza muchas peticiones al servidor. Para optimizar dicho diálogo están empezando a aparecer implementaciones de JSF que incorporan AJAX en sus etiquetas. Esto permite actualizar los componentes en el navegador del usuario de manera selectiva, sin necesidad de recargar la página completa. La combinación JSF y AJAX dota a las páginas de gran dinamismo sin complicar el desarrollo, evitando el uso de javascripts codificado a mano asegurando un mayor soporte a los navegadores web.

2.1.2.6.EL FUTURO DE JSF.

El framework JSF forma parte importante del estándar java J2EE⁷. De hecho se está preparando una nueva versión que traerá numerosas novedades, sobre todo en lo que se refiere a su integración con AJAX. También se está comenzando a utilizar en numerosas aplicaciones empresariales, ya que permite crear pantallas de usuario bastante complejas con una cierta facilidad, aunque desde luego no es sencillo la primera vez que te enfrentas a este framework. En la nueva versión se espera una mejora sobre el control de las fases del ciclo de vida de la petición que faciliten la creación de componentes JSF complejos que se usan de manera simple.

⁷ J2EE: Plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java.

2.1.3. MODELO VISTA CONTROLADOR EN JSF.

2.1.3.1. INTRODUCCIÓN.

El patrón MVC (Modelo Vista Controlador), ver **Figura II.2**, permite separar la lógica de control (qué cosas hay que hacer pero no cómo), la lógica de negocio (cómo se hacen las cosas) y la lógica de presentación (cómo interactuar con el usuario).

Utilizando este tipo de patrón es posible conseguir más calidad, un mantenimiento más fácil, perder el miedo al folio en blanco (existe un patrón de partida por el que empezar un proyecto), etc. al margen de todo esto, una de las cosas más importantes que permite el uso de este patrón consiste en normalizar y estandarizar el desarrollo de Software.

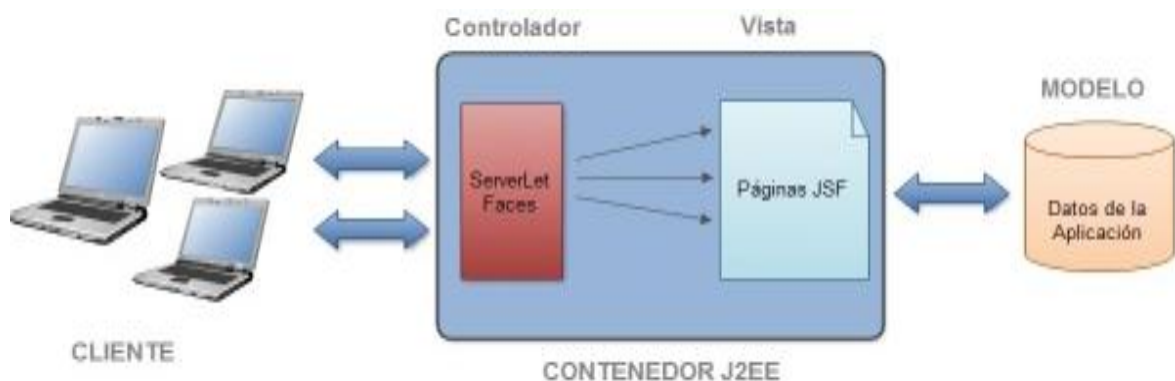


Figura II.2: Arquitectura MVC.

Además, este modelo de arquitectura presenta otras importantes ventajas:

- Hay una clara separación entre los componentes de un programa; lo cual permite implementarlos por separado.
- Hay una API muy bien definida; cualquiera que use la API, podrá reemplazar el modelo, la vista o el controlador, sin demasiada dificultad.
- La conexión entre el modelo y sus vistas (ya que puede haber varias) es dinámica: se produce en tiempo de ejecución, no en tiempo de compilación.

2.1.3.2. MODELO.

Todas las aplicaciones software dejan a los usuarios manipular ciertos datos que proceden de una realidad sobre la que se pretende actuar, como supermercados,

itinerarios de viaje, o cualquier dato requerido en un dominio problemático particular. A estos datos en estado puro, que representan el estado de la realidad se les llama modelo: modelan la parte de la realidad sobre la que se desea actuar.

El modelo, pues, es el objeto que representa y trabaja directamente con los datos del programa: gestiona los datos y controla todas sus transformaciones. El modelo no tiene conocimiento específico de los diferentes controladores y/o vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas, y notificar a las vistas cuándo deben reflejar un cambio en el modelo.

2.1.3.3.VISTA.

La vista es el objeto que maneja la presentación visual de los datos gestionados por el Modelo. Genera una representación visual del modelo y muestra los datos al usuario. Interacciona con el modelo a través de una referencia al propio modelo.

2.1.3.4.CONTROLADOR.

El controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el modelo. Entra en acción cuando se realiza alguna operación, ya sea un cambio en la información del modelo o una interacción sobre la Vista. Se comunica con el modelo y la vista a través de una referencia al propio modelo.

Además, JSF opera como un gestor que reacciona ante los eventos provocados por el usuario, procesa sus acciones y los valores de estos eventos, y ejecuta código para actualizar el modelo o la vista.

2.1.4. MANAGED BEANS Y NAVEGACIÓN EN JSF.

2.1.4.1.MANAGED BEANS.

Un apartado importante en el diseño de aplicaciones web es la separación de la presentación y la lógica de negocio. JSF usa beans para lograr esta separación. Las páginas JSF se refieren a las propiedades del bean, y la lógica de programa está contenida en el código de implementación del bean. Los beans son fundamentales para programar JSF.

Según la especificación de los JavaBeans, un Java bean es “un componente reutilizable del software, que puede ser manipulado”⁸. Esta es una definición bastante imprecisa y, ciertamente, los beans sirven para una gran variedad de propósitos.

A primera vista, un bean parece ser similar a cualquier otro objeto. Sin embargo, los beans se manejan de una forma más concreta. Cualquier objeto se crea y se manipula dentro de un programa Java llamando a los constructores e invocando a los métodos. Sin embargo, los beans pueden ser configurados y manipulados sin programar, a través de entornos de trabajo (frameworks) o entornos de desarrollo integrados (IDE-Integrated Development Environment), que los utilizan mediante técnicas de introspección. En el contexto de JSF, los beans no se utilizan para nada relacionado con la interfaz de usuario: los beans se utilizan cuando se necesita conectar las clases Java con páginas web o archivos de configuración.

Una vez que un bean ha sido definido, puede ser accedido a través de etiquetas JSF. Por ejemplo, la siguiente etiqueta lee y actualiza el atributo password del bean usuario:

```
<h:inputSecret value="#{usuario.password}"/>
```

ATRIBUTOS.

Las características más importantes de un bean son los atributos que posee, también llamados propiedades. Cada uno de éstos tiene:

- Un nombre.
- Un tipo.
- Métodos para obtener y establecer los valores de atributo.

La especificación de los JavaBeans impone una sola exigencia en una clase bean: debe tener un constructor predeterminado, sin parámetros. Además, para que los entornos de trabajo o de desarrollo puedan acceder a sus atributos mediante introspección, una clase bean debe declarar métodos get y/o set para cada uno de ellos, o debe definir descriptores utilizando la clave **java.beans.BeanDescriptor**.

⁸ <http://www.sc.ehu.es/sbweb/fisica/cursoJava/applets/javaBeans/fundamento.htm>

Un método get() no posee parámetros mientras que un método set() posee un parámetro y no devuelve ningún valor. Por supuesto, una clase bean puede tener otros métodos además de los get() y set().

EXPRESIONES PARA VALORES INMEDIATOS Y DIRECTOS.

Muchos componentes de interfaz de usuario JSF tienen un valor de atributo que permite especificar, ya sea un valor inmediato o un valor directo obtenido del atributo de un bean. Por ejemplo, se puede especificar un valor inmediato de la forma: **<h:outputText value="Hola mundo!"/>** o se puede especificar un valor directo: **<h:inputText value="#{usuario.nombre}"/>**

ÁMBITO DE LOS BEANS.

Para comodidad del programador de aplicaciones web, un contenedor de servlets suministra diferentes ámbitos: de petición, de sesión y de aplicación. Estos ámbitos normalmente mantienen beans y otros objetos que necesitan estar disponibles en diferentes componentes de una aplicación web.

Ámbito de Tipo Petición.

Es el de vida más corta. Empieza cuando una petición HTTP, comienza a tramitarse y acaba cuando la respuesta se envía al cliente. Por ejemplo, en la siguiente línea de código: **<f:loadBundle basename="mensajes" var="msjs"/>** la etiqueta f:loadBundle hace que la variable BUNDLE solo exista mientras dura la petición. Un objeto debe tener un ámbito de este tipo sólo si lo que se quiere es renviarlo a otra fase de procesado.

Ámbito de tipo Sesión.

El navegador envía una petición al servidor, el servidor devuelve una respuesta, y entonces ni el navegador ni el servidor tiene cualquier obligación para conservar cualquier memoria de la transacción. Este acomodamiento simple marcha bien para recuperar información básica, pero es poco satisfactorio para aplicaciones del lado del servidor. Por ejemplo, en una aplicación de un carrito compras, necesita al servidor para recordar los contenidos del carrito de compras.

Por esa razón, los contenedores servlet amplían el protocolo de HTTP para seguir la pista a una sesión, esto se consigue repitiendo conexiones para el mismo cliente. Hay diversos métodos para el rastreo de sesión. El método más simple es usar cookies: La pareja nombre/valor la envía el servidor a un cliente, esperando que regresen en subsiguientes peticiones.

Mientras el cliente no desactive las cookies, el servidor recibe un identificador de sesión por cada petición siguiente. Los servidores de aplicación usan estrategias de retirada, algo semejante como al URL rewriting, para tratar con esos clientes que no devuelven cookies. El URL rewriting añade un identificador de sesión a la URL, con lo cual se parece algo a esto: **http://ejemploBasico/index.jsp;jsessionid=64C28D1FC...D28**

El rastreo con cookies es completamente transparente al desarrollador web, y las etiquetas estándar JSF automáticamente rescriben URL si un cliente no usa cookies. El ámbito de sesión permanece desde que la sesión es establecida hasta que ésta termina. Una sesión termina si la aplicación web invoca el método invalidate en el objeto HttpSession o si su tiempo expira.

Las aplicaciones Web típicamente colocan la mayor parte de sus bean dentro de un ámbito de sesión. Por ejemplo, un bean UsuarioBean puede contener información acerca de usuarios que son accesibles a lo largo de la sesión entera. Un bean CarritoCompraBean puede irse llenando gradualmente durante las demandas que levantan una sesión.

Ámbito de Tipo Aplicación.

Persiste durante toda la aplicación web. Este ámbito es compartido entre todas las peticiones y sesiones.

2.1.4.2.NAVEGACIÓN.

Las aplicaciones JSF usan las reglas de navegación para controlar la navegación entre páginas. Cada regla de navegación especifica cómo ir de una página a las demás dentro de la aplicación. En la arquitectura MVC, la navegación de la página es una de las responsabilidades del controlador. Las reglas de navegación de las aplicaciones JSF están contenidas en el archivo faces-config.xml bajo el directorio WEB-INF.

Existen dos tipos diferenciados de navegación: navegación estática y dinámica.

NAVEGACIÓN ESTÁTICA.

Considere el caso en el que un usuario rellena un formulario de una página web. El usuario puede escribir en campos del texto, puede hacer clic sobre enlaces, pulsar botones o seleccionar elementos de una lista, entre otras muchas cosas.

Todas estas acciones ocurren dentro del navegador del cliente. Cuando, por ejemplo, el usuario pulsa un botón, envía los datos del formulario y éstos son gestionados por el servidor. Al mismo tiempo, el servidor JSF analiza la entrada del usuario y debe decidir a qué página ir para dar la respuesta.

En una aplicación web simple, la navegación es estática. Es decir, pulsar sobre un botón suele redirigir al navegador a una misma página para dar la respuesta. En este caso, simplemente, a cada botón se le da un valor para su atributo de acción (action), por ejemplo: `<h:commandButton label="Aceptar" action="login"/>` Esta acción desencadenante, debe concordar con la etiqueta outcome del fichero **faces-config.xml**, dentro de sus reglas de navegación.

En esta simple regla de navegación, se indica que tras la acción login, se navegará a la página hola.jsp, si esta acción ocurre dentro de la página index.jsp.

NAVEGACIÓN DINÁMICA.

En la mayoría de aplicaciones web, la navegación no es estática. El flujo de la página no depende de qué botón se pulsa, sino que también depende de los datos que el cliente introduce en un formulario. Por ejemplo, una página de entrada al sistema puede tener dos resultados: El éxito o el fracaso.

El resultado depende de una computación, sea cual sea el nombre y si la contraseña es legítima. Para implementar una navegación dinámica, el botón de aceptar debe tener un método referencia, por ejemplo: `<h:commandButton label="Aceptar" action="#{loginControlador.verificarUsuario}"/>`

En este caso, loginControlador, referencia un bean, y éste debe tener un método denominado verificarUsuario.

Un método de referencia, en un atributo de acción, no tiene parámetros de entrada y devuelve una cadena de caracteres, que será usada para activar una regla de navegación.

2.2. ENTERPRISE JAVA BEAN.

2.2.1. INTRODUCCION.

Hoy en día, los desarrolladores pueden construir aplicaciones distribuidas, transaccionales, seguras y confiables basadas en la tecnología de servidores de aplicación.

Cada vez, las aplicaciones deben ser diseñadas, construidas y producidas por menos dinero, a más velocidad y utilizando menor cantidad de recursos.

Para reducir costos y acelerar el proceso de desarrollo, Java Platform Enterprise Edition (J2EE) provee un enfoque basado en componentes para el diseño, ensamblaje e instalación de aplicaciones. La plataforma J2EE provee un modelo de aplicación multicapa, componentes reutilizables, un modelo de seguridad unificado, control flexible de transacciones, soporte de Web Services⁹ a través de intercambio de datos integrados por estándares y protocolos abiertos XML.

Las soluciones basadas en componentes de la plataforma J2EE son desarrolladas en forma independiente de la implementación, por lo tanto no se encuentran asociadas a los productos particulares y a las APIs de los proveedores; llegando en algunos casos a perder el soporte debido a la falta de actualización de la versión del producto.

El servidor de aplicación (application server) nace para resolver estos problemas. Provee servicios comunes de middleware¹⁰ como persistencia, transacciones, distribución de objetos, administración de conexiones, etc. Los proveedores implementan estas interfaces estándares para integrar y proveer los servicios a sus productos mediante el servidor de aplicaciones. Por lo tanto los desarrolladores ahora pueden centrarse en la problemática central de negocio, o los requerimientos funcionales, que debe resolver la aplicación utilizando APIs estándares y para invocar al middleware y decidiendo luego

⁹ Web Services: Tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones

¹⁰ Middleware: Software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas.

qué implementaciones específicas utilizar para resolver mejor los requerimientos de tipo no funcionales.

Los Enterprise JavaBeans (también conocidos por sus siglas EJB) son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE de Oracle Corporation.

EJB corresponde a una tecnología Java que permite definir un modelo para el desarrollo y despliegue de componentes servidor reusables. EJB soporta el desarrollo de aplicaciones basadas en una arquitectura de objetos distribuidos, en múltiples capas y en donde la mayor parte de la lógica de la aplicación se mueve desde el cliente hasta el servidor. EJB extiende el modelo de componentes que definieron los JavaBeans.

Un "JavaBean" es un componente utilizado en Java que permite agrupar funcionalidades para formar parte de una aplicación. Un "EJB" también agrupa funcionalidades para una aplicación, sin embargo, implica que existe un ambiente de ejecución ("deployable component"), "EJB Container. Un "JavaBean" requiere ser integrado con otros componentes para que éste sea funcional, mientras un "Enterprise Java Bean" a través de un "EJB Container" puede ser activado ("deployed").

EJB es una arquitectura de componentes de servidor que simplifica el proceso de construcción de aplicaciones de componentes empresariales distribuidos en Java.

Con su utilización es posible escribir aplicaciones escalables, fiables y seguras sin escribir código de infraestructura. La existencia de infraestructura permite un desarrollo más rápido de la parte servidora.

Dado que son componentes, permiten desarrollar aplicaciones portables entre distintas plataformas (son Java) y servidores de aplicaciones (especificación estándar).

Su especificación detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor que son, precisamente, los EJB:

- Comunicación remota utilizando CORBA¹¹.
- Transacciones.

¹¹ CORBA: Standard que permite que diversos componentes de software escritos en múltiples lenguajes de programación y que corren en diferentes computadoras puedan trabajar juntos.

- Control de la concurrencia.
- Eventos utilizando JMS (Java messaging service).
- Servicios de nombres y de directorio.
- Seguridad.
- Ubicación de componentes en un servidor de aplicaciones.

La especificación de EJB define los papeles jugados por el contenedor de EJB y, además de disponer los EJB en un contenedor.

2.2.2. DEFINIÓN DE EJB.

Los EJB proporcionan un modelo de componentes distribuido estándar del lado del servidor. El objetivo de los EJB es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial (concurrencia, transacciones, persistencia, seguridad, etc.) para centrarse en el desarrollo de la lógica de negocio en sí.

El hecho de estar basado en componentes permite que éstos sean flexibles y sobre todo reutilizables.

No hay que confundir los EJB con los JavaBeans. Los JavaBeans también son un modelo de componentes creado por Oracle - Sun Microsystems para la construcción de aplicaciones, pero no pueden utilizarse en entornos de objetos distribuidos al no soportar nativamente la invocación remota (RMI).

2.2.3. VENTAJAS DE EJB.

Un EJB a través de un "EJB Container" ofrece varios servicios y funcionalidades no disponibles en un "Java Bean", algunas son las siguientes:

2.2.3.1. SERVICIOS MIDDLEWARE.

Esta posiblemente sea la mayor ventaja de un EJB. Cuando se diseña un componente de software se deben definir varios servicios para su funcionamiento, algunos pueden ser:

- ¿Si ocurre un error, qué procedimiento debe ejecutarse?
- ¿Si la base de datos especificada se encuentra desactivada, existe otra alternativa?

- No fue posible cumplir exitosamente "x" procedimiento ¿se deben retractar sus acciones parciales o re-invocar la transacción?

Estos servicios (comúnmente llamados "Middleware") por lo general son requeridos además de la lógica contenida en los componentes principales; obviamente estos servicios ("Middleware") aún deben ser diseñados, sin embargo, mediante un "EJB Container" se ofrecen estos servicios y es a través de un "Enterprise Java Bean" que es posible desarrollar los componentes principales ("lógica de negocios").

2.2.3.2.DIVISIÓN DE TRABAJO.

La posibilidad de dividir "Servicios" (EJB Container) de "Componentes Principales" (EJB's) permite una clara división de trabajo, esto es, que un diseñador de "componentes" (EJB's) puede concentrar sus esfuerzos en la "lógica de proceso" sin preocuparse del diseño de servicios. Y de la misma manera un "diseñador" de servicios ("Middleware") concentrarse en su área.

Esta división de trabajo trae consigo otra pregunta: *¿Cómo se logra la interoperabilidad?* La interoperabilidad entre "Servicios" y "Componentes" se debe a la existencia de especificaciones para EJB's, estas especificaciones (parte primordial de J2EE) definen los requerimientos para un "EJB Container" y los requisitos para un "Enterprise Java Bean".

2.2.3.3.DIVERSOS VENDEDORES.

El uso de especificaciones para EJB's permite que existan diversos vendedores tanto de "EJB Containers" los cuales son incluidos en un java application server, así como "Enterprise Java Beans" los cuales resuelven algún tipo de lógica.

Lo anterior permite ejecutar *cualquier "EJB" en cualquier "EJB Container"*, esto es, puede adquirir un conjunto de EJB's producidos por Inprise o inclusive desarrollarlos dentro de su empresa y estos podrán ser ejecutados en un "EJB Container" de IBM, Inprise o JBoss.

En lo que se refiere a diferencias en costo y calidad, generalmente estas dependen de las funcionalidades que van más allá de las especificaciones EJB. El "EJB Container" de IBM ofrece un integración más eficiente con sus productos (Domino, DB2), mientras

Inspire puede desarrollar su "EJB Container" orientado hacia ambientes financieros, Oracle hacía manufactura y sus productos en "Bases de Datos", etc.

2.2.3.4. PROCEDIMIENTOS REMOTOS RMI.

Debido a la solución que intenta ofrecer EJB ("Enterprise Java Beans") su diseño gira alrededor de procedimientos remotos.

2.2.3.5. DIVERSOS CLIENTES.

Un EJB puede interactuar con una gran gama de clientes desde: JSP o Servlets, bases de datos, Applets, sistemas ERP (SAP, JDEdward's).

2.2.4. COMPONENTES DE LA ARQUITECTURA.

Con respecto a la arquitectura EJB, se pueden encontrar los siguientes componentes que la estructuran:

2.2.4.1. OBJETO EJB.

Los Enterprise Beans no son estrictamente objetos remotos. Cuando un cliente necesita utilizar una instancia de una clase Enterprise Bean, el cliente nunca invoca el método directamente en la instancia real. La invocación es interceptada por el contenedor EJB y delegada a la instancia del Enterprise Bean. Interceptando las invocaciones, el contenedor EJB puede realizar en forma automática las invocaciones de middleware en forma implícita. Por lo tanto el desarrollador de componentes no necesita escribir, corregir o mantener invocaciones a las APIs de middleware. Algunos de los servicios que se pueden obtener en forma implícita mediante la intercepción son:

- **Administración implícita de transacciones distribuidas.** Las transacciones permiten realizar operaciones en forma robusta y determinística en ambientes distribuidos. El contenedor EJB provee un servicio de transacciones, que consta de una implementación a bajo nivel de administración de transacciones a bajo nivel. El servicio de transacciones debe ser expuesto a través de la API de alto nivel JTA¹². JTA es una interfaz de alto nivel que permite controlar las transacciones.

¹² JTA: Establece una serie de Interfaces java entre el manejador de transacciones y las partes involucradas en el sistema de transacciones distribuidas.

- **Seguridad implícita.** La seguridad suele ser una consideración importante para el desarrollo de aplicaciones multicapa distribuida. La plataforma Java Edición Estándar (J2SE), define un servicio de seguridad robusto que permite autorizar y autenticar usuarios, asegurando los componentes instalados. EJB agrega a este concepto la noción de seguridad transparente o declarativa, que permite obtener los beneficios de seguridad sin necesidad de escribir las invocaciones a la API de JAAS. (Java Authorization and Authentication Service).

- **Administración implícita de recursos y ciclos de vida de los componentes.** El contenedor EJB administra implícitamente los recursos utilizados por los Enterprise Beans, como threads, sockets o conexiones a la base de datos. El ciclo de vida de los Enterprise Beans también es administrado, permitiendo al contenedor EJB reutilizar las instancias de los Enterprise Beans cuando sea necesario.

- **Persistencia implícita.** La persistencia de datos es un requerimiento de cualquier aplicación que requiere almacenamiento permanente. EJB ofrece asistencia para almacenar los datos persistentes de los objetos en las capas subyacentes de almacenamiento y luego obtener esa información.

- **Acceso remoto implícito.** La clase Enterprise Bean no puede ser invocada a través de la red directamente porque es una clase sin soporte para redes. El contenedor EJB soporta invocación remota mediante el encapsulamiento de los beans en objetos con soporte para red. Los objetos con soporte para red reciben invocaciones desde el cliente y las delegan a las instancias correspondientes de los beans. Esto ahorra al programador tener que preocuparse por temas referidos al uso de redes.

- **Soporte implícito.** Los contenedores EJB automáticamente reciben invocaciones concurrentes de los clientes. El contenedor EJB provee soporte de threads, instanciando las múltiples copias necesarias de los componentes, instanciando múltiples Enterprise Beans y alojando las instancias en threads. De

esta forma se obtiene concurrencia robusta sin necesidad de escribir código multithread.

- **Transparencia de ubicación de componentes en forma implícita.** Los clientes de los componentes se encuentran desacoplados de la ubicación específica del componente que se encuentra siendo utilizado.
- **Monitoreo implícito.** El contenedor EJB puede realizar un seguimiento de los métodos invocados y recopilar información estratégica para optimización y balance de carga inteligente.

El contenedor EJB actúa como una capa de dirección entre el código del cliente y el del Enterprise Bean. Esta capa de indirección se manifiesta como un objeto con soporte de red llamado *objeto EJB* (EJB object) que actúa como el interceptor de invocaciones.

El objeto EJB es un objeto que conoce sobre redes, transacciones, persistencia y más. Tiene la inteligencia para saber cómo realizar la lógica intermedia que el contenedor EJB requiere antes de la invocación a la instancia de la clase Enterprise Bean. El objeto EJB replica y expone todos los métodos de negocios expuestos por el Enterprise Bean y delega todas las invocaciones del cliente a los Enterprise Beans. La **figura II.3** muestra la función de los objeto EJB.

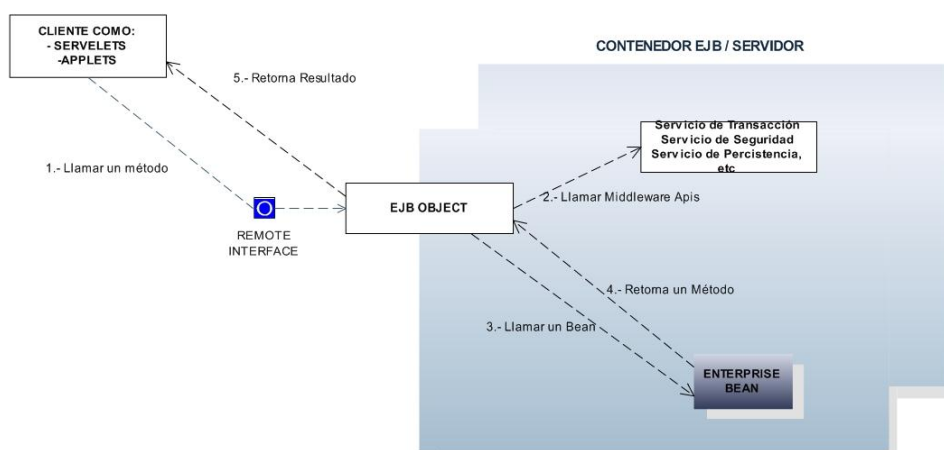


Figura II.3: Función de los objetos EJB.

El Objeto EJB puede pensarse como una parte física del contenedor. Todos tienen código específico del container dentro. Cada contenedor puede manejar internamente el middleware en forma diferente, el proveedor del contenedor (vendedor) genera las clases para los objetos EJB en forma automática. Existen varias estrategias para la generación de los EJB Object.

2.2.4.2.SERVIDOR DE EJB (EJB SERVER).

El servidor EJB proporciona el ambiente o entorno necesario para soportar la ejecución de aplicaciones que han sido desarrolladas utilizando la tecnología de EJB. Este servidor administra y coordina la ubicación de los recursos para que sean utilizados por las aplicaciones.

2.2.4.3.EJB CONTAINER.

Un servidor EJB debe ser capaz de proveer uno o más contenedores EJB. El concepto de *container* es análogo al concepto de hogar, esto queda más claro al constatar que es el EJB container quien administra a los beans contenidos en él.

Por cada EJB, el contenedor es responsable de registrarlo, de proporcionar una interfaz remota para él, de crear o destruir instancias del objeto, realizar chequeos de seguridad para el objeto, manejar su estado activo y coordinar transacciones distribuidas.

Como una funcionalidad opcional, el EJB container puede incluso llegar a manejar toda la información persistente dentro del objeto.

2.2.5. FUNCIONAMIENTO DE LOS EJB.

Los EJB se disponen en un contenedor EJB dentro del servidor de aplicaciones. La especificación describe cómo el EJB interactúa con su contenedor y cómo el código cliente interactúa con la combinación del EJB y el contenedor.

Cada EJB debe facilitar una clase de implementación Java y dos interfaces Java. El contenedor EJB creará instancias de la clase de implementación Java para facilitar la implementación EJB. Las interfaces Java son utilizadas por el código cliente del EJB. Las dos interfaces, conocidas como interfaz "home" e interfaz "remota", especifican las firmas de los métodos remotos del EJB. Los métodos remotos se dividen en dos grupos:

- **Métodos que no están ligados a una instancia específica**, por ejemplo aquellos utilizados para crear una instancia EJB o para encontrar una entidad EJB existente. Estos métodos se declaran en la interfaz "home".
- **Métodos ligados a una instancia específica**. Se ubican en la interfaz remota.

Dado que se trata simplemente de interfaces Java y no de clases concretas, el contenedor EJB genera clases para esas interfaces que actuarán como un proxy en el cliente. El cliente invoca un método en los proxies generados que a su vez sitúa los argumentos del método en un mensaje y envía dicho mensaje al servidor EJB. Los proxies usan RMI-IIOP¹³ para comunicarse con el servidor EJB.

El servidor llamará a un método correspondiente a una instancia de la clase de implementación Java para manejar la llamada del método remoto.

2.2.5.1.INTERFAZ REMOTA.

Proporciona el acceso a los métodos dentro del bean. Un objeto EJB representa la visión del usuario del bean. El objeto EJB revela todas las interfaces de la aplicación vinculadas al objeto, pero no aquellas interfaces que permiten al contenedor administrar y controlar al objeto.

El wrapper¹⁴ de objeto EJB permite que el contenedor EJB intercepte todas las operaciones que se hagan sobre el bean. Dicho, de otra forma, cada vez que el cliente invoca un método sobre un objeto EJB, la petición pasa por el EJB container antes de ser traspasada al bean.

El contenedor implementa la administración de estados, el control de transacciones, seguridad y servicios de persistencia en forma transparente tanto para el cliente como para el bean.

2.2.5.2.INTERFAZ EJB HOME.

Esta interfaz proporciona el acceso a los servicios del ciclo de vida del EJB. Los clientes pueden usar esta interfaz para crear o destruir instancias del bean en materia de

¹³ RMI-IIOP: Denota la interfaz RMI de Java sobre el sistema CORBA.

¹⁴ Wrapper: Clases que modelan los tipos de datos primitivos tales como enteros y flotantes

servicios. El modelo EJB provee soporte para un determinado número de servicios implícitos (**Ver Figura II.4**).

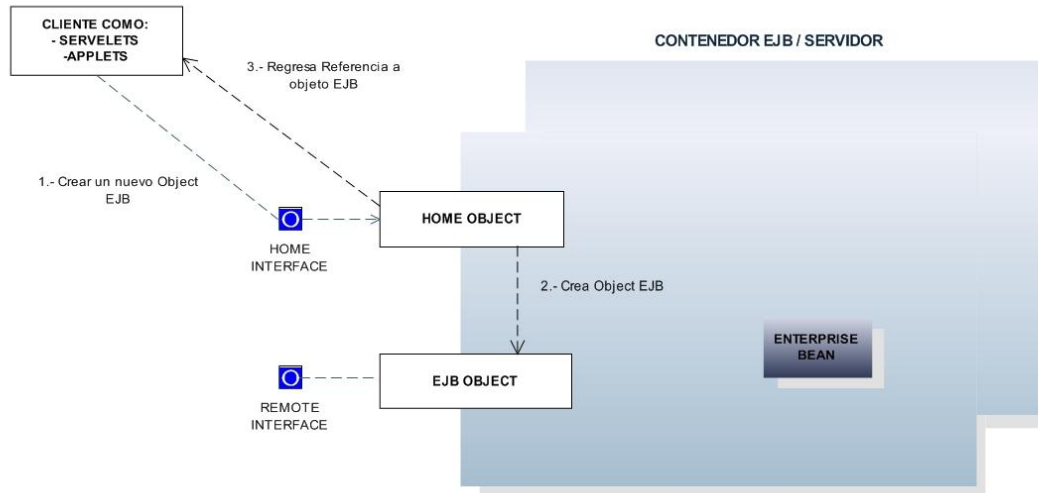


Figura II.4: Interfaz EJB Home.

2.2.5.3.MANEJO DE ESTADOS.

Los EJB no requieren guardar o recuperar en forma explícita los estados del objeto entre llamado y llamado de los métodos, debido a que el contenedor EJB se encarga de llevar a cabo estas tareas en beneficio del mismo bean.

2.2.5.4.SEGURIDAD.

Todas las tareas involucradas con este punto (chequeos de seguridad), como por ejemplo, la autenticación de usuarios o el chequeo de niveles de autorización, quedan en manos del container del bean.

2.2.5.5.TRANSACCIONES.

Los Beans no necesitan explicitar individualmente el código de demarcación transaccional para poder participar en transacciones distribuidas. Es el EJB container quien se encarga de manejar la partida, el enroollment, el compromiso y el rollback de transacciones.

2.2.5.6.PERSISTENCIA.

Un bean no necesita retirar o almacenar objetos de datos persistentes desde una base de datos, ya que el EJB container se encarga de manejar datos persistentes.

2.2.6. TIPOS DE EJB.

Existen 3 tipos de EJB:

1. EJB de Entidad.
2. EJB de Sesión.
3. EJB dirigidos por mensaje.

2.2.6.1.EJB DE ENTIDAD.

Son objetos persistentes, que representan una vista de objeto de una determinada fuente de datos. Esto permite que el EJB manipule información residente en sistemas ajenos al "EJB Container". En otras palabras, un "Entity EJB" manipula una copia/reflejo de información que reside en otro sistema. Cada instancia de un bean de identidad se identifica a través de una única clave primaria. Las beans de entidad pueden crearse bien creando un objeto (Create()) o insertando directamente datos en la fuente de datos. Dado que el estado del bean se mantiene en la fuente de datos, este tipo de beans pueden recuperarse ante una caída del sistema.

Un bean de entidad a diferencia de un bean de sesión trabaja en conjunción con un depósito de información (generalmente una base de datos), esto permite que el EJB manipule la información residente en sistemas ajenos al EJB Container.

2.2.6.2.EJB DE SESIÓN.

Un Bean de Sesión permite realizar cierta lógica solicitada por un cliente ya sea un JSP/servlet, Applet¹⁵ e inclusive otro EJB. Existen dos tipos de Beans de Sesión:

- 1. Beans de Sesión sin Estado (Stateless Session EJB's):** Este tipo de EJB como su nombre lo indica no mantiene estado ("Stateless") en el "EJB Container". Estos EJB's son utilizados para realizar tareas rutinarias que no requieren identificar o rastrear al cliente. Dado que este tipo de bean no

¹⁵ Applet: Componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web.

mantiene estado, los beans de sesión no están ligados a un cliente particular, de manera que cualquier instancia de un bean de esta clase puede usarse para dar servicio a un cliente. Algunos EJB's de este tipo son: operaciones matemáticas complejas, búsquedas generales, etc. Cualquier información de estado, en caso de que fuera necesario, se mantiene por el cliente.

2. **Beans de Sesión con Estado (Statefull Session EJB's):** Permiten mantener la sesión del cliente en el "EJB Container", de esta manera el cliente puede trabajar con cierto juego de datos específico administrado por el "EJB Container". Dado que el estado se mantiene en este tipo de EJB, el servidor de aplicación maneja pares cliente-bean. Cada instancia de un cierto EJB se crea bajo petición de un cliente concreto, y en principio es un recurso privado para ese cliente (aunque puede compartirse entre clientes, usando el handle del bean). En esencia, un bean con estado actúa como una extensión del cliente, con la excusa de que cierta carga del cliente se distribuye en el servidor de aplicaciones. Los datos de estado no sobreviven a una caída del servidor, salvo que la implementación particular soporte persistencia frente a caídas del servidor. Los beans con estado pueden acceder a recursos persistentes (como bases de datos o ficheros) pero, a diferencia de los beans de entidad, no representan datos.

2.2.6.3.EJB DIRIGIDOS POR MENSAJE.

Son los únicos beans con funcionamiento asíncrono. Usando el Java Messaging System¹⁶ (JMS), se suscriben a un tema (topic) o a una cola (queue) y se activan al recibir un mensaje dirigido a dicho tema o cola. No requieren de su instanciación por parte del cliente.

2.2.7. ARCHIVO EJB-JAR.

Una vez creadas las clases Enterprise Bean, las interfaces home, la interfaces remotas y los descriptores, es necesario empaquetarlos en un módulo EJB mediante un archivo EJB-jar con extensión ".jar" (**Ver Figura II.5**).

¹⁶ Java Messaging System: Estándar de mensajería que permite a los componentes de aplicaciones basados en la plataforma Java2 crear, enviar, recibir y leer mensajes.

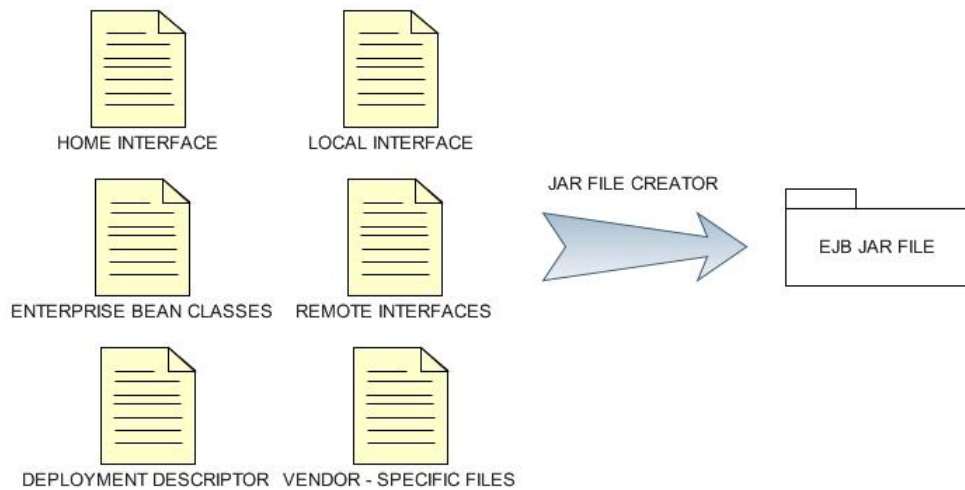


Figura II.5: Archivo EJB-jar.

Una vez construido el archivo EJB-jar, el Enterprise Bean está completo, y es una unidad instalable en cualquier servidor de aplicación J2EE. Una vez instalado, las herramientas provistas por el proveedor del contenedor EJB son las responsables de descomprimir, leer, y extraer la información contenida en el archivo EJB-jar. Luego el instalador realiza tareas específicas del proveedor, como generar los objetos EJB, los objetos home, importar el Enterprise Bean dentro del container, y configurarlo. El soporte para los archivos EJB-jar es un estándar y es una funcionalidad requerida por todas las herramientas EJB.

Es importante destacar que más de un Enterprise Bean puede estar contenido en un mismo archivo EJB-jar, permitiendo empaquetar módulos o aplicaciones enteras en un solo archivo.

CAPÍTULO III.

3. INTEGRACIÓN DE JSF Y EJB.

3.1.INTRODUCCIÓN.

EJB3 es la última versión de EJB (Enterprise Java Beans). EJB es una especificación de un framework que define unos componentes software (los Enterprise Java Beans), que debidamente diseñados y configurados por el desarrollador, y alojados en un servidor de aplicaciones J2EE, conforman la capa del modelo en el patrón MVC (a la capa del modelo también se le llama lógica del negocio - bussiness logic - en el entorno de la gestión empresarial).

EJB3 provee una gran cantidad de funcionalidad al desarrollador a la hora de implementar el modelo, funcionalidad que de una u otra forma se tendría que implementar por sí mismo. Entre otras se encuentran las siguientes características:

- **Proceso de Transacciones.** El servidor J2EE se encarga de que las modificaciones que se realicen al modelo y que estén encerradas dentro de una "transacción" se realicen todas o ninguna.
- **Integración con los servicios de persistencia que ofrece Java Persistence API (JPA).** JPA integra el mapeo ORM como una forma de persistencia para los 'beans de entidad' de EJB. Su equivalente en Ruby on Rails es el ActiveRecord. En realidad, la parte de persistencia de EJB3 es la integración en

el API de EJB de los conceptos de ORM implementados con gran éxito por Hibernate¹⁷.

- **Control de concurrencia.**
- **Piscinas de recursos y clustering**, que asegura la escalabilidad del producto.
- **Seguridad.**

JSF (Java Server Faces), por otro lado, es un framework de desarrollo de aplicaciones web en Java, también para ser alojado en un servidor de aplicaciones JEE, cuyo diseño implementa el patrón MVC al completo. Es decir, un desarrollador puede implementar una aplicación Web completa sólo usando las herramientas de JSF. El problema es que JSF le deja al desarrollador casi todo el trabajo de implementar la capa de modelo, y la capa de presentación que incluye es muy pobre. Sin embargo, la capa del controlador es excelente, y además - y esto es lo más importante de JSF - da muchas facilidades para ser extendido.

JSF implementa la conexión del controlador con la capa de modelo a través de los JSF Managed Java Beans. El sistema es que el desarrollador asocia los inputs del usuario con objetos y métodos de los Managed Java Beans.

Si un desarrollador quiere realizar una aplicación JSF + EJB3, la forma correcta es crear el modelo en EJB3, y realizar las llamadas a éste modelo desde los JSF Managed Java Beans de la aplicación.

En la mayoría de las aplicaciones, esto genera mucha duplicación del trabajo.

Las longitudes de los ciclos de vida de los componentes de una aplicación viene dada por los ámbitos de aplicación (scopes) disponibles en el framework para dichos componentes.

Los ámbitos de aplicación de los JSF Java Beans pueden ser:

- Aplicación.
- Sesión.
- Página.

¹⁷ Hibernate: Herramienta JAVA que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación

Es decir, el controlador JSF es capaz de manejar componentes del modelo con ciclos de vida de dichas longitudes.

Cualquier otro ciclo de vida de componentes del modelo tiene que ser implementado por el programador.

3.2.ANTECEDENTES JSF Y EJB.

A pesar del esfuerzo realizado por el grupo de expertos encargado de definir las especificaciones que forman parte de JEE por simplificar el desarrollo de aplicaciones JEE, hay que tener en cuenta que JEE se trata de una plataforma compleja, alrededor de la cual se concentran una serie de APIs y de conceptos, y que sin un conocimiento medio de éstos será difícil sacar el máximo partido a JEE, por lo que aunque a primera vista parezca sencillo el desarrollo de aplicaciones se tendrá que intentar conocer al máximo todo aquello que se mueve a su alrededor si se quiere sacar rendimiento a JEE.

En este capítulo se mostrará cómo desarrollar un prototipo de aplicación en la que se integrará dos tecnologías que forman parte de esta renovada plataforma, EJB y Java Server Faces.

Para ello se utilizará NetBeans, cuyo equipo también ha realizado grandes esfuerzos por simplificar al máximo el desarrollo de aplicaciones JEE con este IDE.

3.3.APLICACIÓN CON JSF Y EJB.

En este capítulo se demostrará un prototipo de integración entre las dos tecnologías Java Server Face y Enterprise Java Beans, en el Entorno de Desarrollo Netbeans.

El prototipo ilustrado a continuación se fundamenta en demostrar la facilidad de uso de las diversas tecnologías J2EE y ponerlas juntas para crear una aplicación web empresarial.

Aunque la aplicación que se va a demostrar a continuación no profundiza temas complejos ni se va a desarrollar una aplicación completa, su arquitectura representa las mejores prácticas en el desarrollo de una aplicación empresarial, por lo que la modularidad, escalabilidad y reusabilidad se tienen en cuenta.

La **figura III.6** muestra la arquitectura como se distribuye la aplicación por capas:

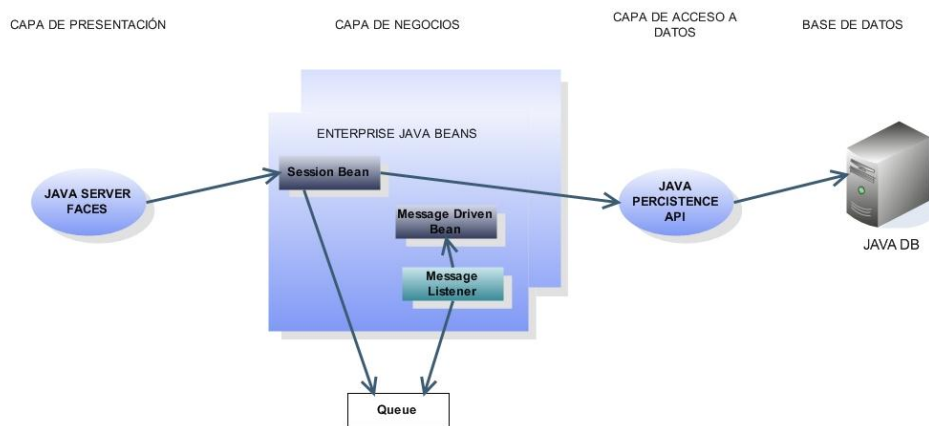


Figura III.6: Distribución por capas de la Aplicación.

La arquitectura de la aplicación se muestra en la figura III.6, en el que se organizan en distintos niveles: Presentación, lógica de negocios, acceso a datos y de datos, donde cada uno tiene un papel importante que desempeñar y se separa el uno del otro.

Esta arquitectura favorece una clara separación de responsabilidades, la reutilización y escalabilidad mediante el uso de Enterprise Java Beans. Con el uso de JSF y JPA, el desarrollo de interfaz gráfica de usuario se convierte en una brisa y los desarrolladores ya no están empantanados por las tareas tediosas y propensas a errores de conversión de datos relacionales orientada a objetos, lo cual es natural en Java y viceversa. Así, todo-en-todo, esta demostración no sólo muestra el uso de las diversas tecnologías J2EE, también demuestra la arquitectura de la aplicación mejor de su clase que puede ser utilizada en un sistema de producción.

3.3.1. REQUISITOS PARA CONSTRUIR LA APLICACIÓN.

Para construir una aplicación web integrando las tecnologías de JSF y EJB, es necesario tener conocimientos previos sobre:

- JavaServer Faces (JSF) con Facelets¹⁸.
- Enterprise Java Beans (EJB) 3/3.1
- Java Persistence API (JPA)

¹⁸ Facelets: Framework ligero que permite el uso de plantillas en aplicaciones JSF.

- Conocimientos básicos de uso de NetBeans IDE.

Para construir esta aplicación, es necesario tener el siguiente software instalado en el equipo:

- NetBeans IDE 6.5 o superior.
- GlassFish Enterprise Server v3.
- Biblioteca de Componentes PrimeFaces¹⁹.
- Base de Datos de muestra instalada junto con NetBeans.

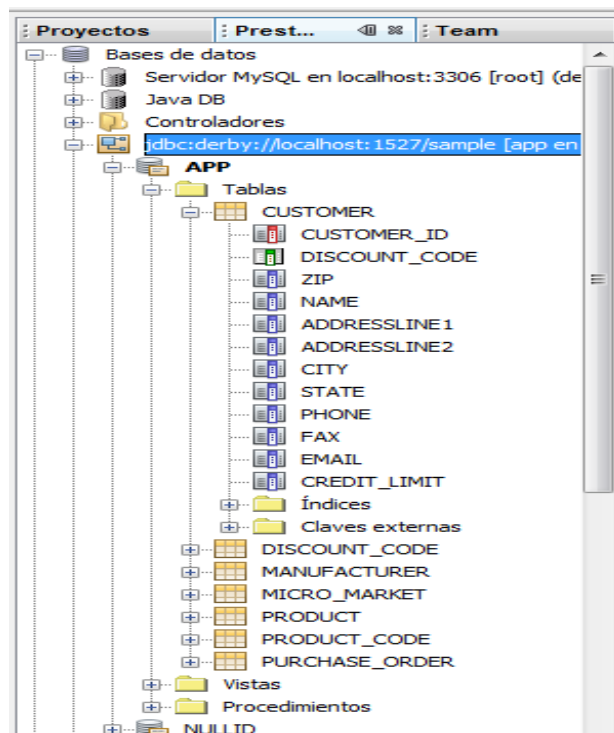


Figura III.7: Base de Datos de muestra en NetBeans.

3.3.2. CREANDO EL PROYECTO.

Crear un nuevo proyecto con el nombre de CustomerApp.

- Seleccione en la barra de menú la opción “Archivo => Proyecto Nuevo...”. En el cuadro de diálogo, seleccione **Java EE** en la parte de Categorías, en virtud de los proyectos, seleccione **Enterprise Application**. Haga clic en Siguiente.

¹⁹ PrimeFaces: Componente para JavaServer Faces que cuenta con un conjunto de componentes ricos que facilitan la creación de las aplicaciones web

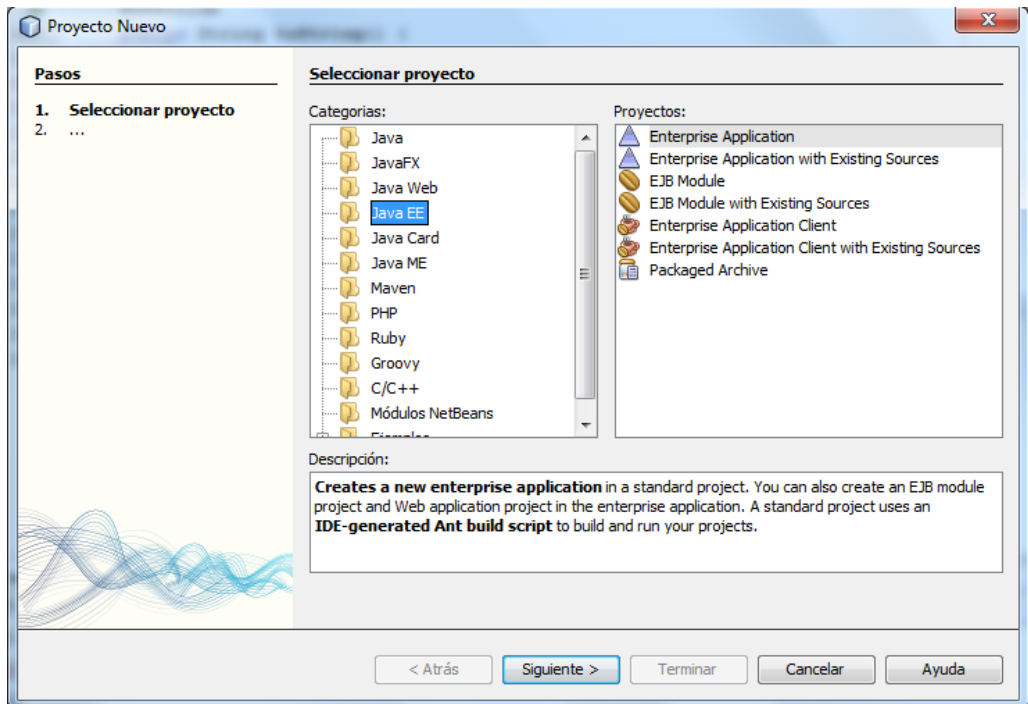


Figura III.8: Creando el Proyecto => Seleccionar Proyecto.

- Seleccione la ubicación del proyecto y el nombre del proyecto, **CustomerApp**, y haga clic en Siguiete.

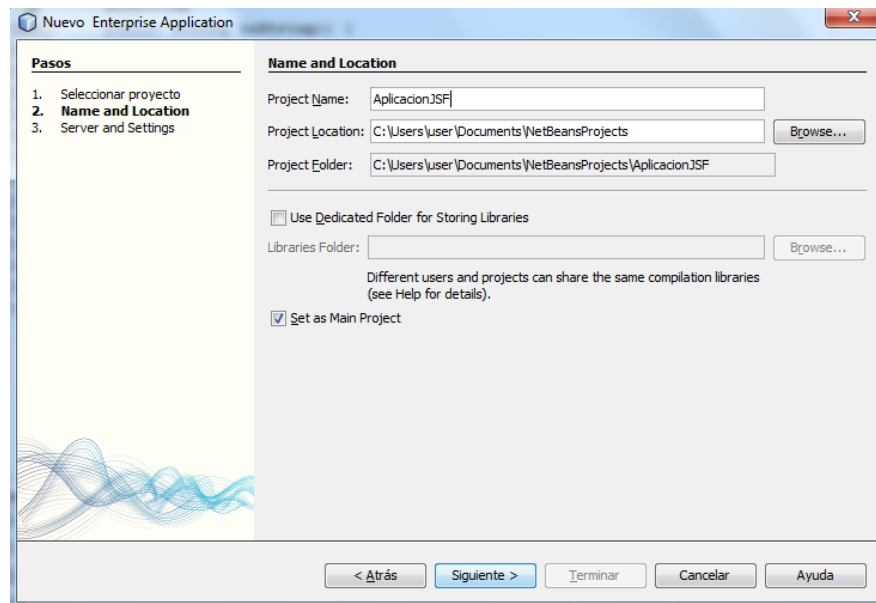


Figura III.9: Creando el Proyecto => Nombre y Ubicación del Proyecto.

- Seleccione **GlassFish v3** como el servidor, y **Java EE 6** como la versión de Java EE, y haga clic en Terminar.

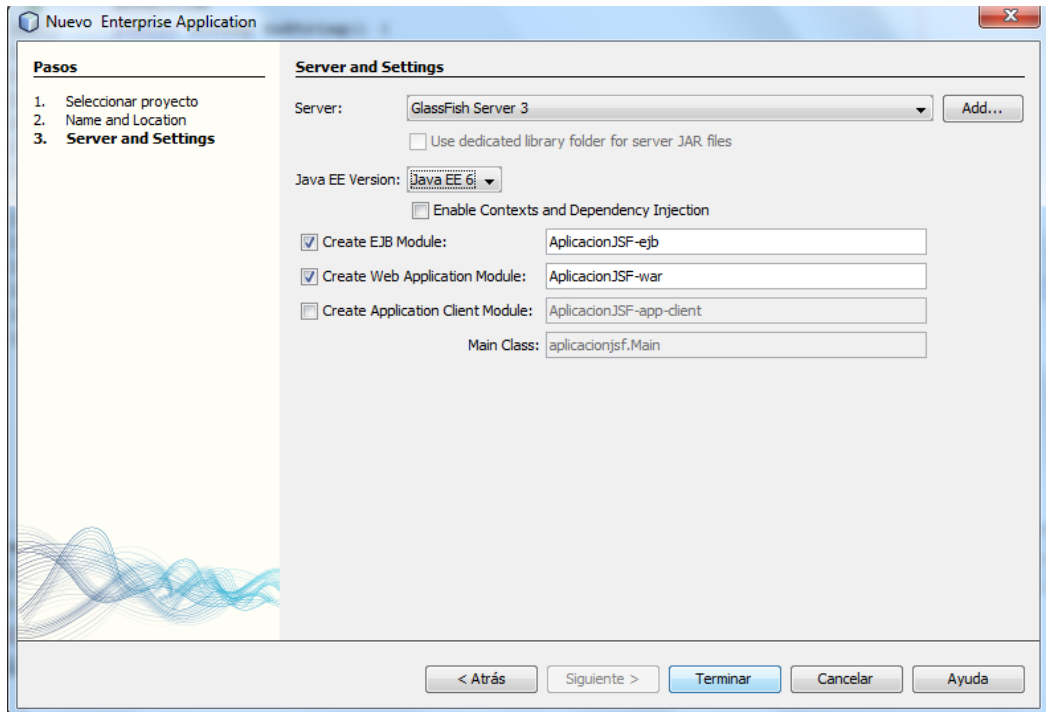


Figura III.10: Creando el Proyecto => Servidor y Configuraciones.

NetBean creará **3** proyectos, a saber, **CustomerApp** (proyecto de aplicaciones empresariales), **CustomerApp-ejb** (EJB del proyecto), y **CustomerApp-war** (Web del proyecto).

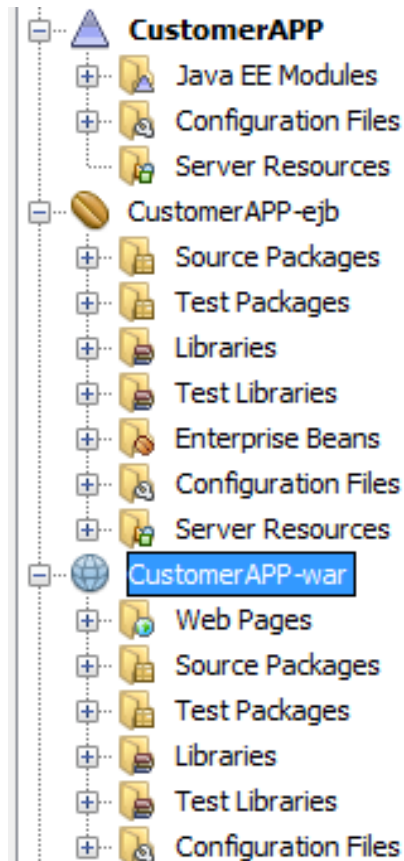


Figura III.11: Creando el Proyecto => Proyectos creados por el IDE NetBeans.

3.3.3. CREANDO LAS CLASES ENTIDADES A PARTIR DE LA BASE DE DATOS.

- En el proyecto “**CustomerApp-ejb**” dar click derecho, seleccionar “Nuevo” y seleccionar la opción “Otro..”

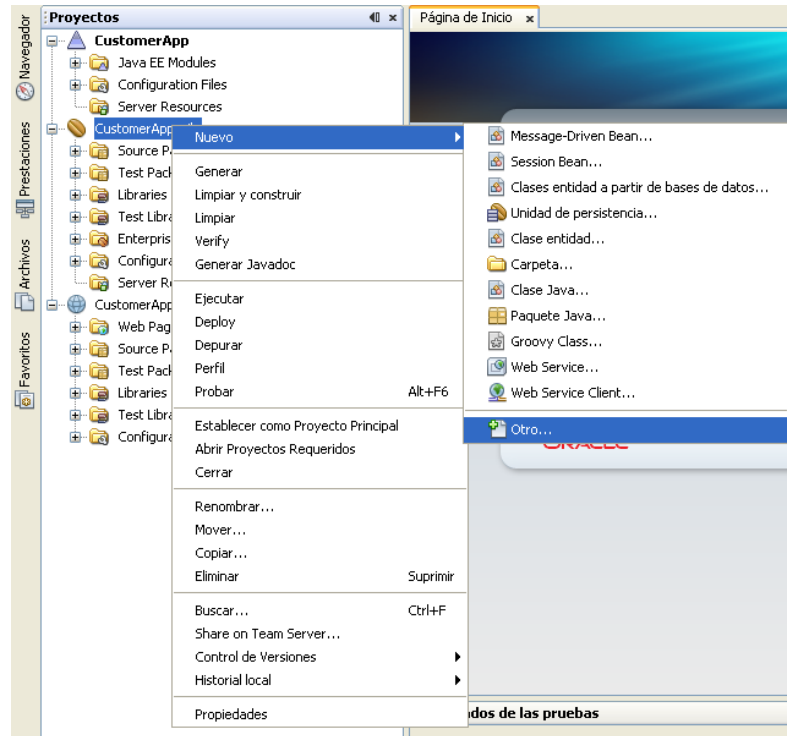


Figura III.12: Creando las clases entidades => Seleccionar Otro.

- En la Ventana, se selecciona en la pestaña de “Persistencia” seleccionar en “Clases Entidad a partir de bases de datos”, y dar siguiente.

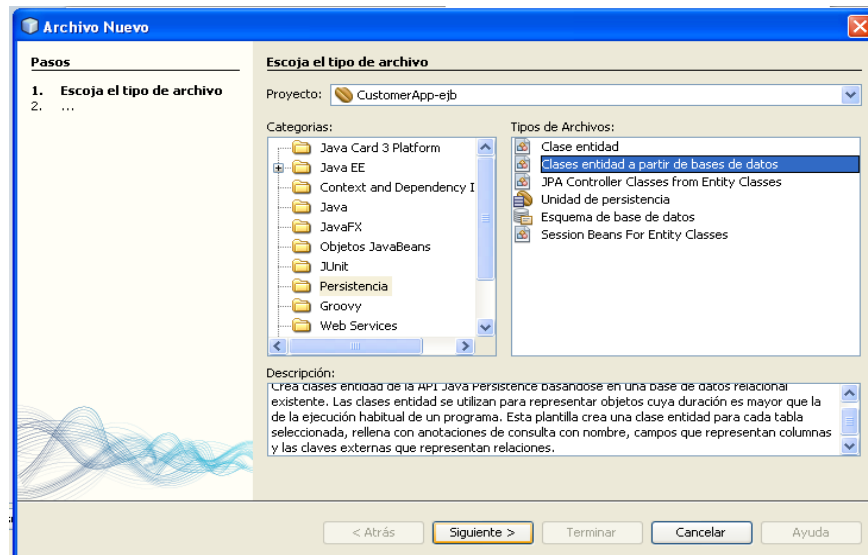


Figura III.13: Creando las clases entidades => Seleccionar “Clase Entidad a partir de Base de Datos”.

- En la siguiente ventana en la opción “*Fuente de Datos*”, seleccionar “*Jndi/sample*”

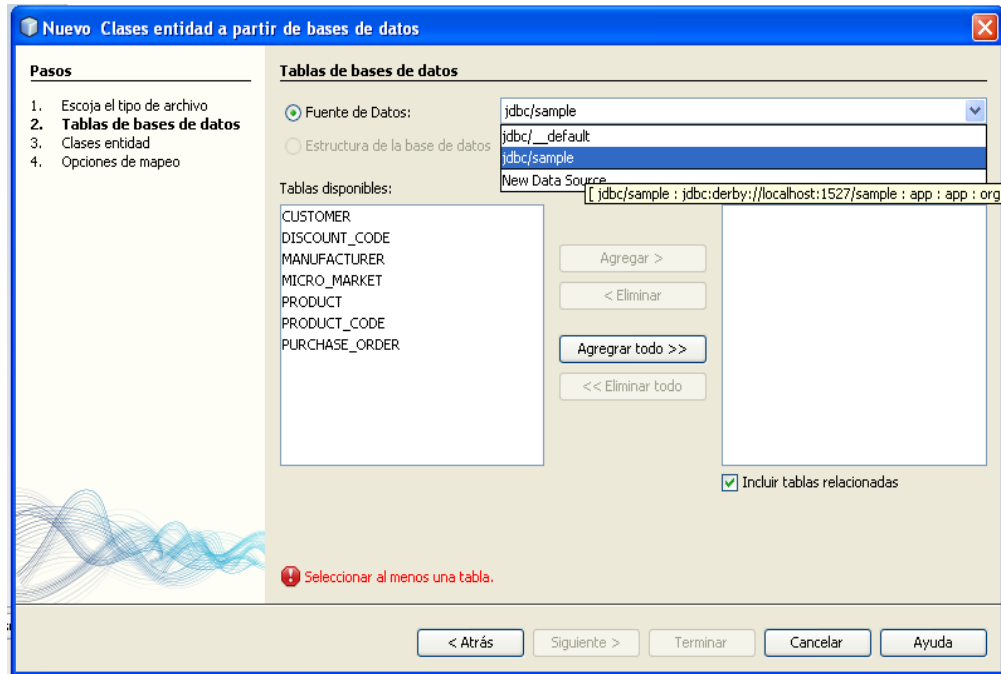


Figura III.14: Creando las clases entidades => Seleccionar Tablas de la Base de Datos.

- Seleccionar una tabla para agregar, se marca en “*Customer*”, y se notó que se agrega la tabla relacionada que es “*Discount_Code*”, dar en Siguiente

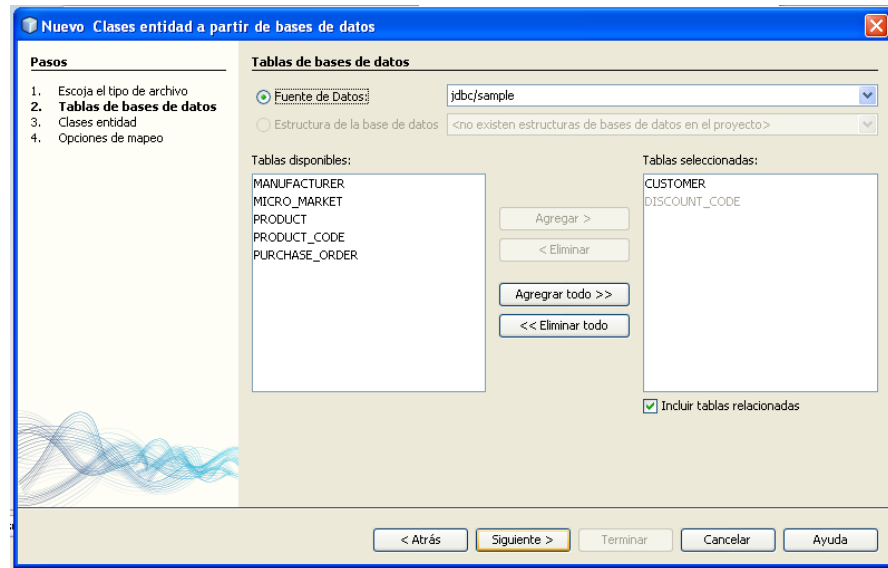


Figura III.15: Creando las clases entidades => Seleccionando Tablas.

- En la siguiente venta poner en el nombre de paquete “*com.customerapp.entity*”, dar siguiente.

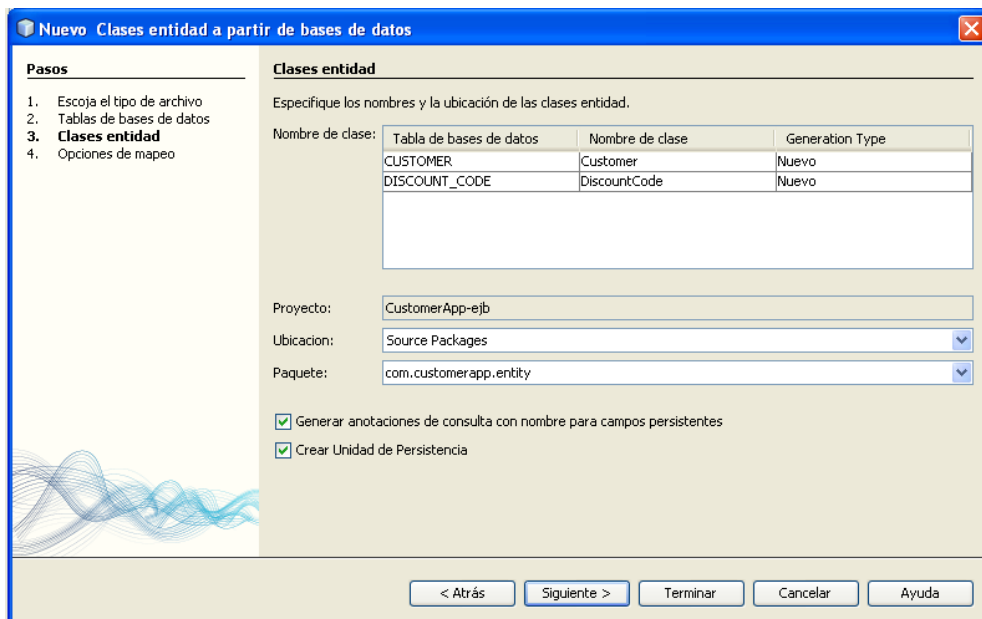


Figura III.16: Creando las clases entidades => Clases Entidad.

- En la ventana siguiente seleccionar en el tipo de colección "*java.util.List*", y finalizar.

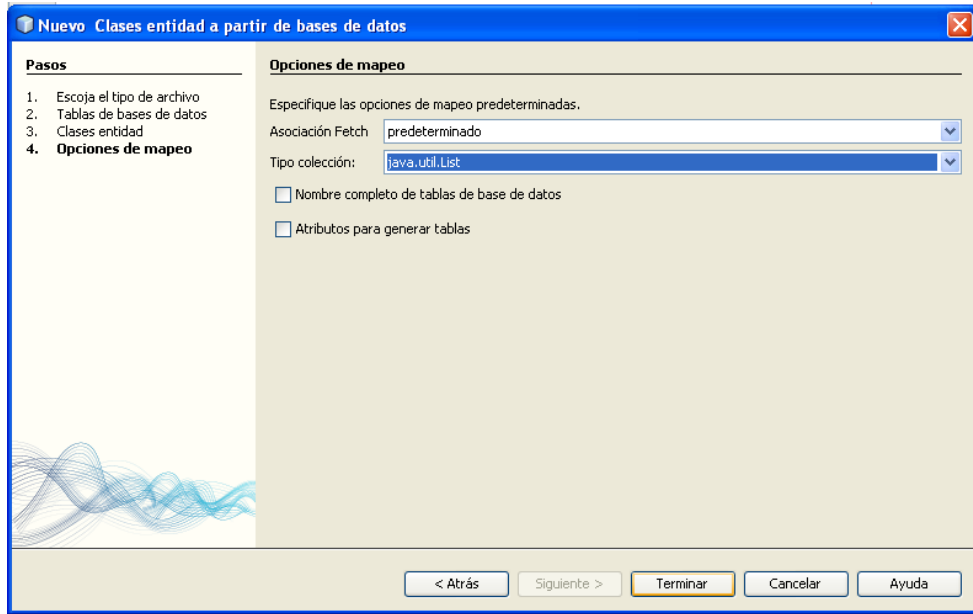


Figura III.17: Creando las clases entidades => Opciones de Mapeo.

- Se puede notar la creación de las clases “*Customer.java*” y “*DiscountCode.java*”

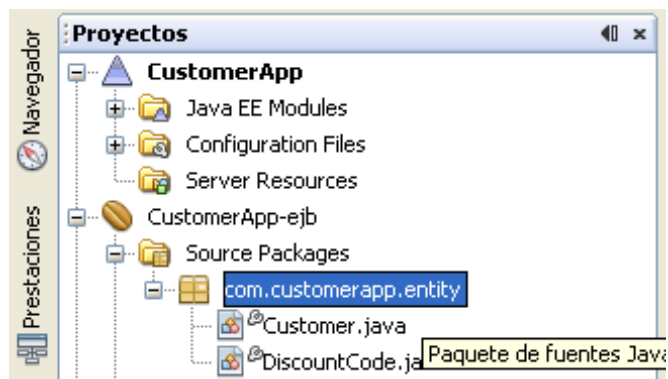


Figura III.18: Creando las clases entidades => Clases entidades creadas.

3.3.3.1. CREACIÓN DE LOS EJB (SESSION Y MESSAGE DRIVEN)

Ahora que se tienen las clases de entidad, la siguiente tarea es la creación del Session Bean (sin estado), *CustomerSession* para manipular y contar con las funciones RU en los objetos del cliente. En esta demostración, el cliente que utiliza estas funciones son las páginas JSF. Uno de los beneficios de hacer esto (es decir, contar con las funciones de la capa EJB) es la reutilización, porque las mismas funciones pueden ser utilizadas

por más de mil páginas JSF, objetos EJB, otros clientes de aplicaciones empresariales y clientes de servicios web cuando se exponen como servicios web. Otros beneficios incluyen la capacidad de ampliación debido a que el contenedor EJB puede ser fácilmente afinado y ampliado cuando aumenta la carga.

También se va a crear un Message-Driven Bean (MDB), NotificationBean aquí para demostrar su uso para la mensajería asincrónica. En esta ilustración, se quiere enviar las notificaciones sobre la correcta actualización de un registro de cliente. Para simplificar, sólo se está enviando el objeto de cliente actualizado a la cola, de modo que el MDB puede recoger y procesar en un subproceso independiente.

Para crear los MDB, se seguirán los siguientes pasos.

- Seleccionar en “*CustomerApp-ejb*” click derecho, escoger “*Nuevo*” y seleccionar “*Session Bean*”.

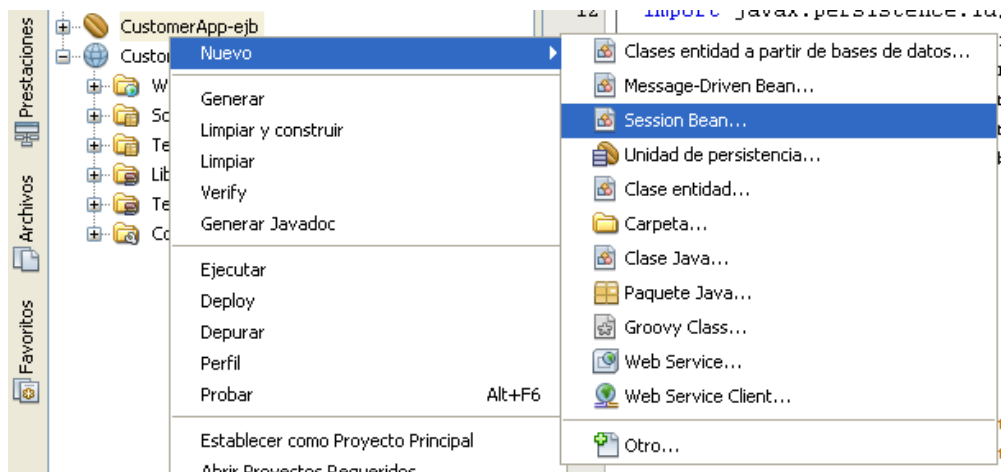


Figura III.19: Creación de los MDB => Nuevo Session Bean.

- En la ventana que aparece, poner los siguientes datos, **Ejb Name** → *CustomerSession*, **Package** → *com.customerapp.ejb*, en **Session Type** → *Stateless*, y terminar.

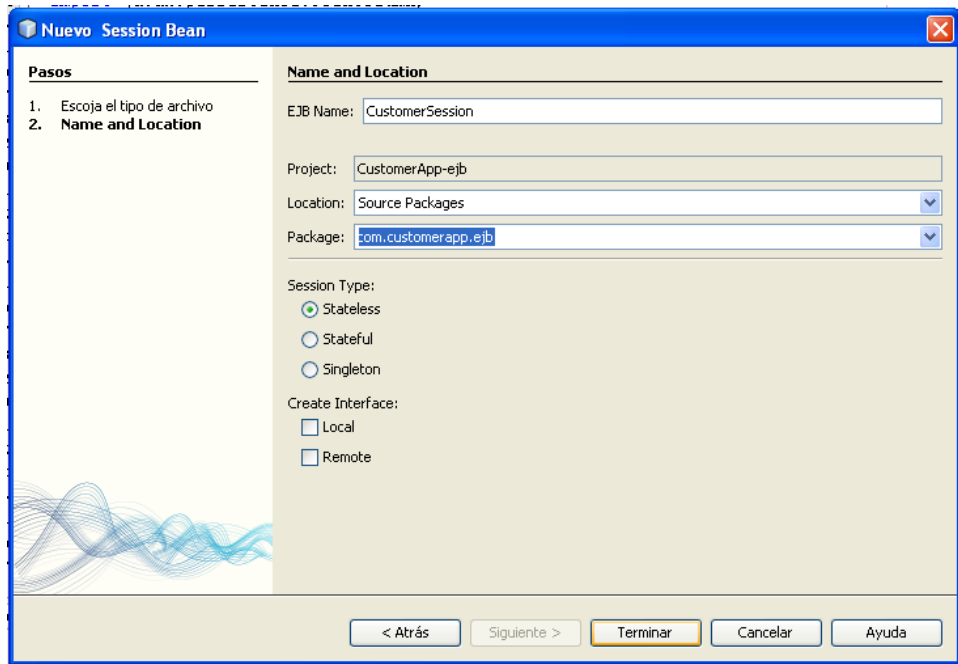


Figura III.20: Creación de los MDB => Nombre y Localización.

- Seleccionar en “*CustomerApp-ejb*” y dar clic derecho, escoger “*Nuevo*” y seleccionar “*Message-Driven Bean*”.

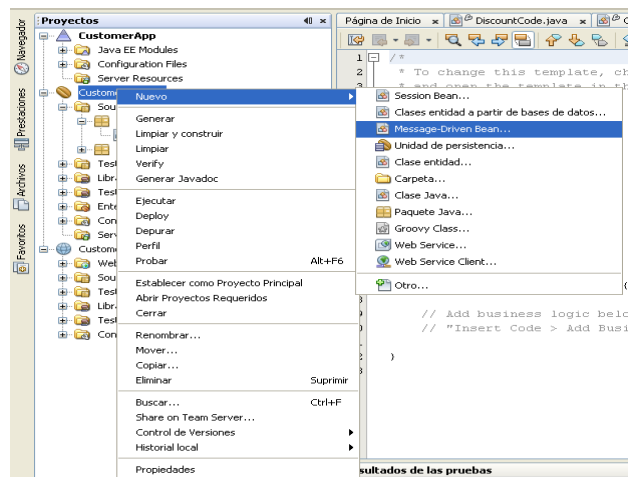


Figura III.21: Creación de los MDB => Nuevo Message Drive Bean.

- En la ventana que aparece poner los siguientes datos, **Ejb Name** → *NotificationBean*, **Package** → *com.customerapp.mdb*, en **Project** →

Destinations, seleccionar “Add”, en **Destination Name** → *NotificationQueue*, y **Destination Type** → *Queue* y terminar

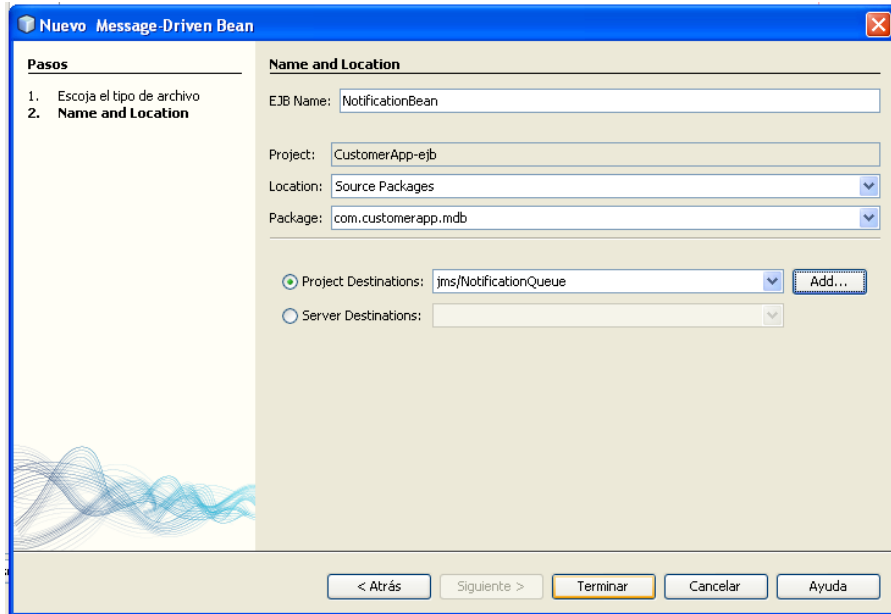


Figura III.22: Creación de los MDB => Nombre y Localización de los MDB.

- En el explorador de proyectos se observan los paquetes que se ha creado con sus respectivas clases.

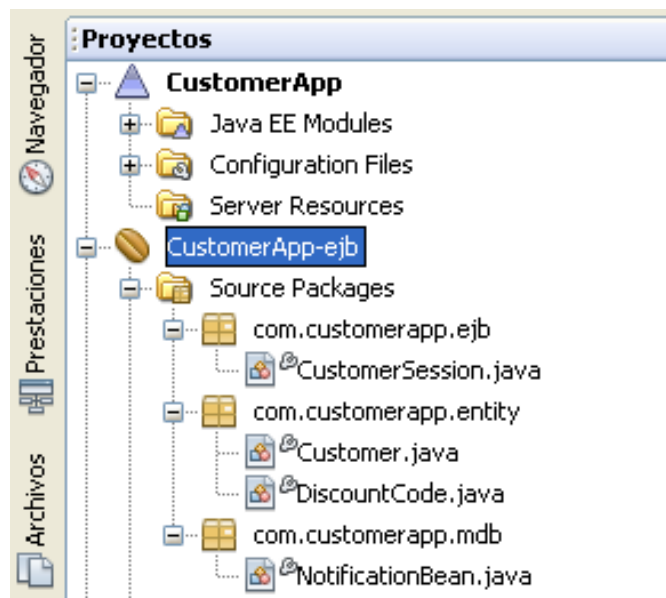


Figura III.23: Creación de los MDB => Paquetes y sus respectivas clases.

- Dirigirse a la Fuente de Session Beans, haciendo click sobre “*CustomerSession*”

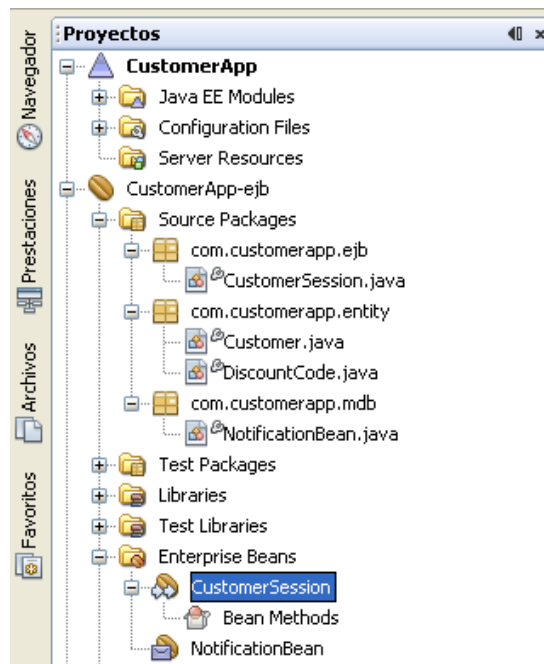


Figura III.24: Creación de los MDB => Sessions Beans creados.

- En el editor de código, hacer click derecho, seleccionar en “Persistencia”, “*Utilizar el administrador de entidades*”

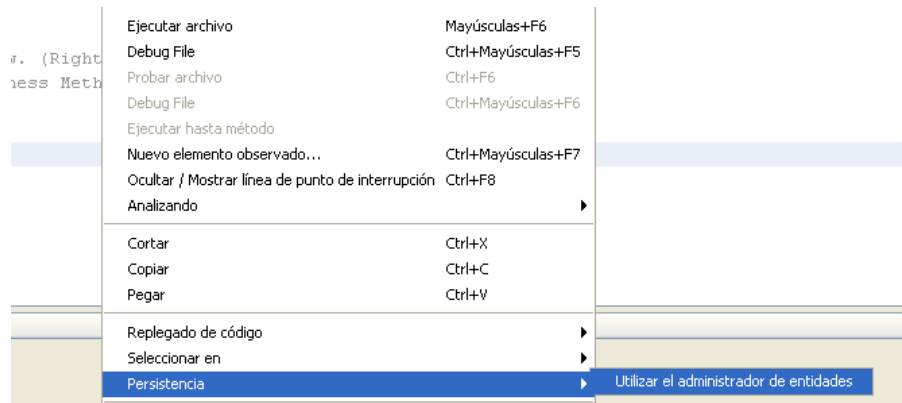
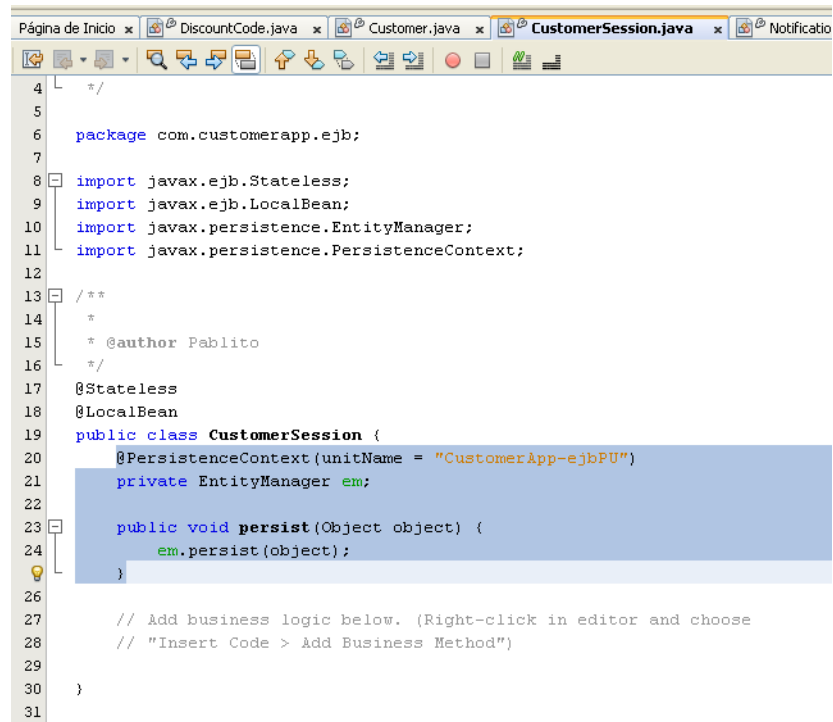


Figura III.25: Creación de los MDB => Creando la Persistencia.

- Se puede observar que se inserta código.



```
4  L  */
5
6  package com.customerapp.ejb;
7
8  import javax.ejb.Stateless;
9  import javax.ejb.LocalBean;
10 import javax.persistence.EntityManager;
11 import javax.persistence.PersistenceContext;
12
13 /**
14  *
15  * @author Pablito
16  */
17 @Stateless
18 @LocalBean
19 public class CustomerSession {
20     @PersistenceContext(unitName = "CustomerApp-ejbPU")
21     private EntityManager em;
22
23     public void persist(Object object) {
24         em.persist(object);
25     }
26
27     // Add business logic below. (Right-click in editor and choose
28     // "Insert Code > Add Business Method")
29
30 }
31
```

Figura III.26: Creación de los MDB => Código insertado de Persistencia.

- En el mismo editor de código, dar click derecho, seleccionar en **“Insertar Código”** y seleccionar en **“Add Bussines Method”**

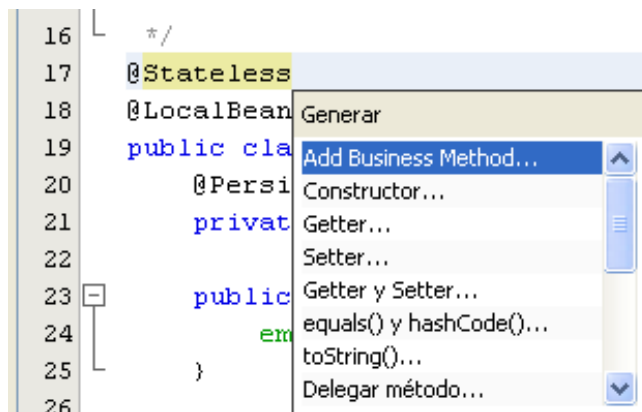


Figura III.27: Creación de los MDB => Menú para agregar nuevos métodos.

- A continuación proporcionar el nombre del método **“retrive”**, en El tipo de regreso digitar **“List <Customer>”**, también hacer lo mismo para el método **“update”** y Aceptar

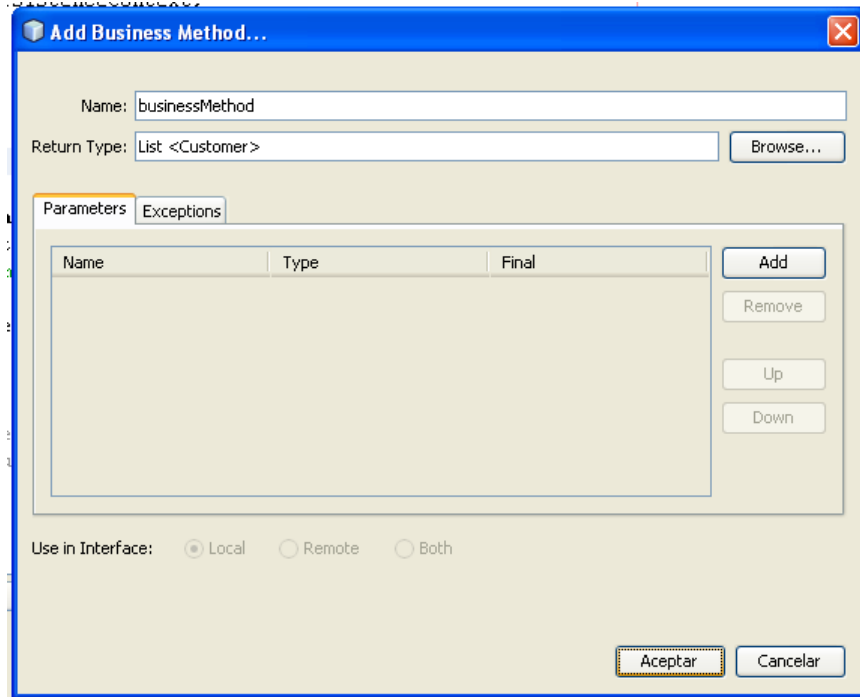


Figura III.28: Creación de los MDB => Creación de Nuevos Métodos.

➤ Y aparece el siguiente código.

```
Página de Inicio x DiscountCode.java x Customer.java x CustomerSession.java
10 import javax.persistence.EntityManager;
11 import javax.persistence.PersistenceContext;
12
13 /**
14  *
15  * @author Pablito
16  */
17 @Stateless
18 @LocalBean
19 public class CustomerSession {
20     @PersistenceContext(unitName = "CustomerApp-ejbPU")
21     private EntityManager em;
22
23     public void persist(Object object) {
24         em.persist(object);
25     }
26
27     public List <Customer> retrieve() {
28         return null;
29     }
30
31     public List <Customer> update() {
32         return null;
33     }
34
35     // Add business logic below. (Right-click in editor and
36     // "Insert Code > Add Business Method")
37
38 }
```

Figura III.29: Creación de los MDB => Código de los nuevos métodos.

➤ Editar los métodos, como se muestra a continuación.

```
@Stateless
@LocalBean
public class CustomerSessionBean
{
    @PersistenceContext
    private EntityManager em;

    /**
     * Returns a list of Customer objects in the database
     * @return List<Customer>
     */
    public List<Customer> <b>retrieve</b>()
    {
        Query query = em.createNamedQuery("Customer.findAll");
        return query.getResultList();
    }

    /**
     * Update the customer record
     * @param customer object to be updated
     * @return Customer
     */
    public Customer <b>update</b>(Customer customer)
    {
        return em.merge(customer);
    }
}
```

Figura III.30: Creación de los MDB => Código para métodos update y retrieve.

3.3.4. USANDO JSF Y PRIMEFACES.

Debido a que JSF es nuevo, no hay muchas opciones de framework basados en Ajax. Sin embargo, **PrimeFaces** es el más completo y adecuado para esta aplicación, ya que ha puesto en marcha el componente **dataTable** de interfaz de usuario, y parece ser el más fácil de integrar en el IDE NetBeans.

Antes de crear las páginas web, asegúrese de que el framework **JSF** se agregue al proyecto Web, **CustomerApp-war**.

- En la ventana de Proyectos, haga clic en el proyecto Web, **CustomerApp-war**, y seleccione Propiedades.
- En el cuadro de diálogo de propiedades, seleccione Frameworks, y luego presione en Add, para agregar el framework con el que se va a trabajar, seleccione JavaServer Faces.

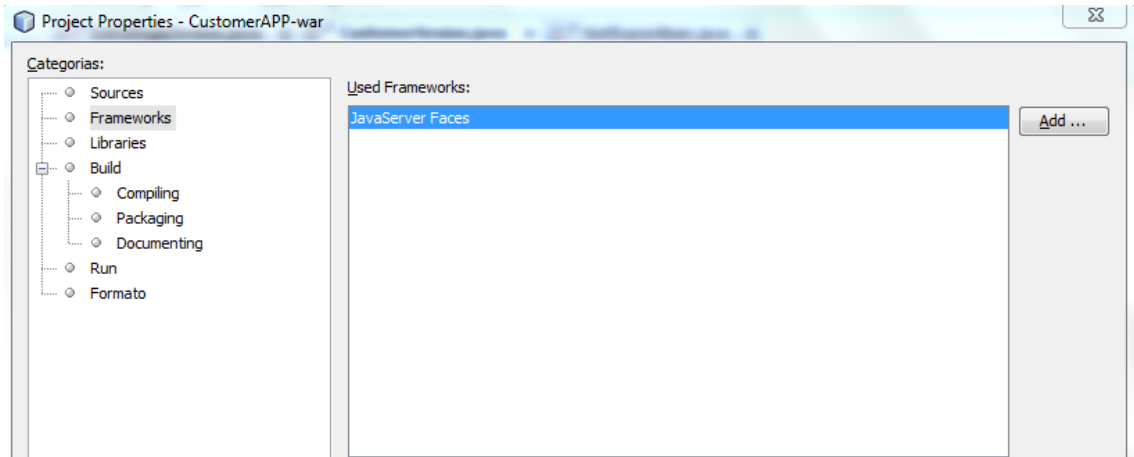


Figura III.31: Usando JSF y PrimeFaces => Seleccionando Framework JavaServer Faces.

- Antes de utilizar los componentes PrimeFaces en nuestras Facelets, es necesario incluir la biblioteca en el IDE NetBeans y realizar un par de cosas.
 - Descargar la biblioteca PrimeFaces (primefaces-2.0.0.RC.jar) de <http://www.primefaces.org/downloads.html> y guardarlo en algún lugar del disco local.
 - Para permitir que los proyectos futuros usen PrimeFaces, cree una **biblioteca mundial** en NetBeans para PrimeFaces.
 - Seleccione "Herramientas > Bibliotecas" en el menú principal de NetBeans.
 - En el cuadro de diálogo Administrador de Biblioteca, elegir la opción "Biblioteca Nueva" y proporcione un nombre para la biblioteca, por ejemplo, "PrimeFaces3".

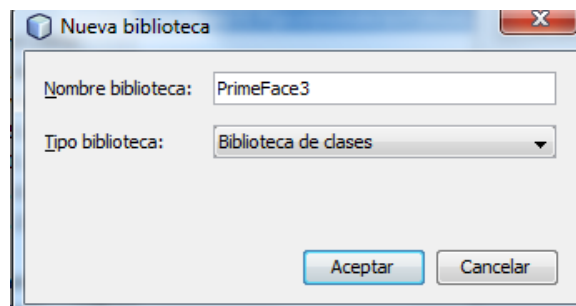


Figura III.32: Usando JSF y PrimeFaces => Agregando Biblioteca Global.

- Con la nueva biblioteca "PrimeFaces2" creada, haga clic en el botón "Add JAR / carpeta..." y seleccione el archivo .jar que se ha descargado antes y haga clic en Aceptar para terminar.

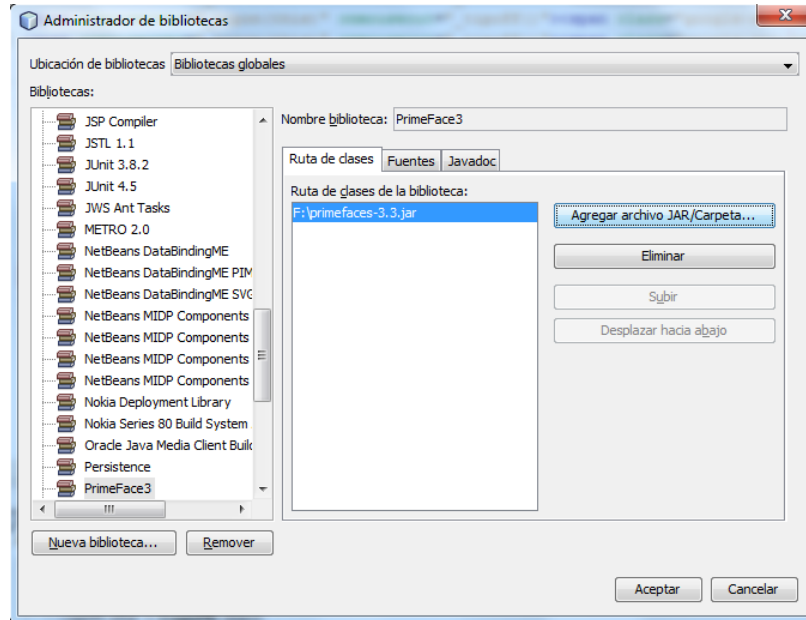


Figura III.33: Usando JSF y PrimeFaces => Agregando archivo .jar a la biblioteca.

- A continuación, se tiene que añadir la biblioteca PrimeFaces3 al proyecto Web.
 - Seleccione el proyecto Web, CustomerApp-war, desde la ventana de proyecto, haga clic derecho y seleccione "Propiedades".
 - En la categoría Bibliotecas, haga clic en "Add Library..." y seleccione la biblioteca PrimeFaces3 y haga clic en Aceptar para terminar.

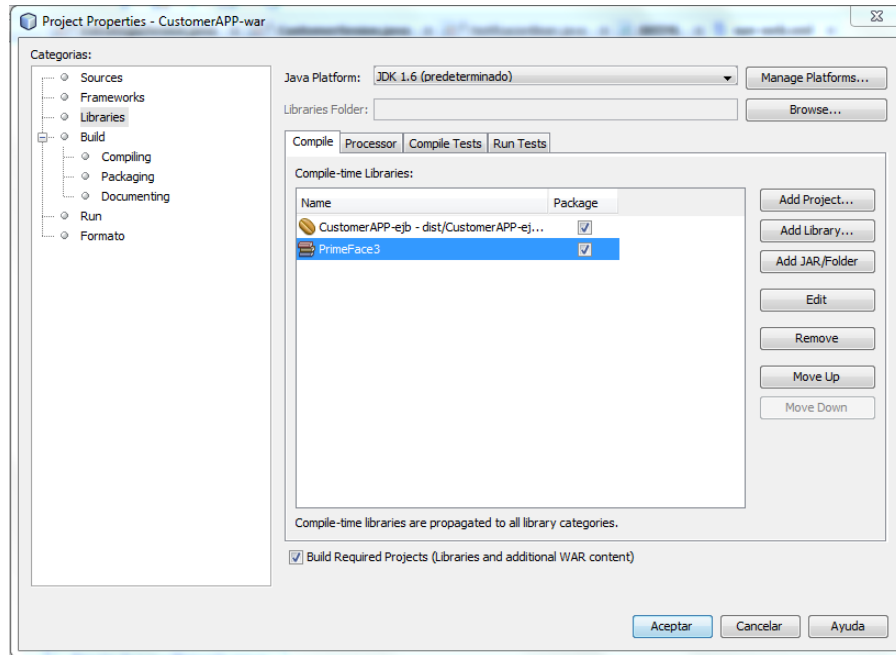


Figura III.34: Usando JSF y PrimeFaces => Agregando la Biblioteca al proyecto.

- Debido a que se va a utilizar **Facelets** en nuestra aplicación, se tiene que actualizar la plantilla de **XHTML** en NetBeans para que todos los archivos XHTML creados posteriormente dispongan de los **espacios de nombres** y los **recursos** necesarios para el desarrollo.
- Seleccione "Herramientas> Plantillas" en el menú de NetBeans.
- En el cuadro de diálogo Administrador de plantillas, seleccione "Web> XHTML" y haga clic en la opción "Abrir en el Editor de".

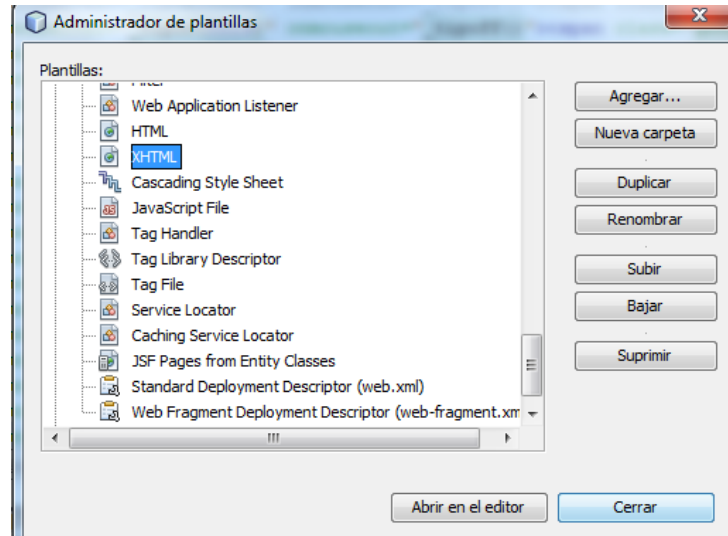


Figura III.35: Usando JSF y PrimeFaces => Administrador de Plantillas.

- Editar el contenido de este archivo con este código:

```
<?xml version="1.0" encoding="{encoding}"?>
<#assign licenseFirst = "<!--"><#assign licensePrefix = ""><#assign
licenseLast = "--"><# Asignar licenseLast = "--">
<#include "../Licenses/license-{$project.license}.txt">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <Xhtmlns html = "http://www.w3.org/1999/xhtml"
xmlns:f="http://java.sun.com/jsf/core" xmlns: f =
"http://java.sun.com/jsf/core" xmlns:h="http://java.sun.com/jsf/html"
  xmlns: h = "http://java.sun.com/jsf/html"
xmlns:p="http://primefaces.prime.com.tr/ui" xmlns: p =
"http://primefaces.prime.com.tr/ui">
  <h:head>
    <meta http-equiv="Content-Type" content="text/html;
charset={encoding}"/></span>
    <title>TODO supply a title</title></span>
    <p:resources />
  </h:head>
  <h:body>
    <p>
      TODO write content
    </p>
  </h:body>
</html>
```

- Por último, se añade las siguientes declaraciones en el archivo **web.xml** del proyecto Web para que PrimeFaces funcione correctamente:

```
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>/faces/* </url-pattern>
  <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
<servlet>
  <servlet-name>Resource Servlet</servlet-name>
  <servlet-
class>org.primefaces.resource.ResourceServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Resource Servlet</servlet-name>
  <url-pattern>/primefaces_resource/*</url-pattern>
</servlet-mapping>
<context-param>
  <param-name>com.sun.faces.allowTextChildren</param-
name>
  <param-value>>true</param-value>
</context-param>
```

- En este punto, se ha terminado la instalación y configuración del entorno para PrimeFaces para trabajar en NetBeans. A continuación se creará el listado de clientes y puntos de vista Detalles.

3.3.4.1. CREANDO EL CLIENTE BEAN GESTIONADO.

Antes de crear las páginas JSF, primero se crea el bean gestionado que será la prestación de los servicios necesarios para las páginas JSF que se crearán más tarde.

- En la ventana de Proyectos, haga clic en el proyecto Web, **CustomerApp-war**, y selecciona "Nuevo> JSF Managed Bean...", especifique **CustomerMBean** como nombre de clase ", **com.customerapp.web**" como el nombre del paquete, el **cliente** como el nombre, y en Scope **Session**.

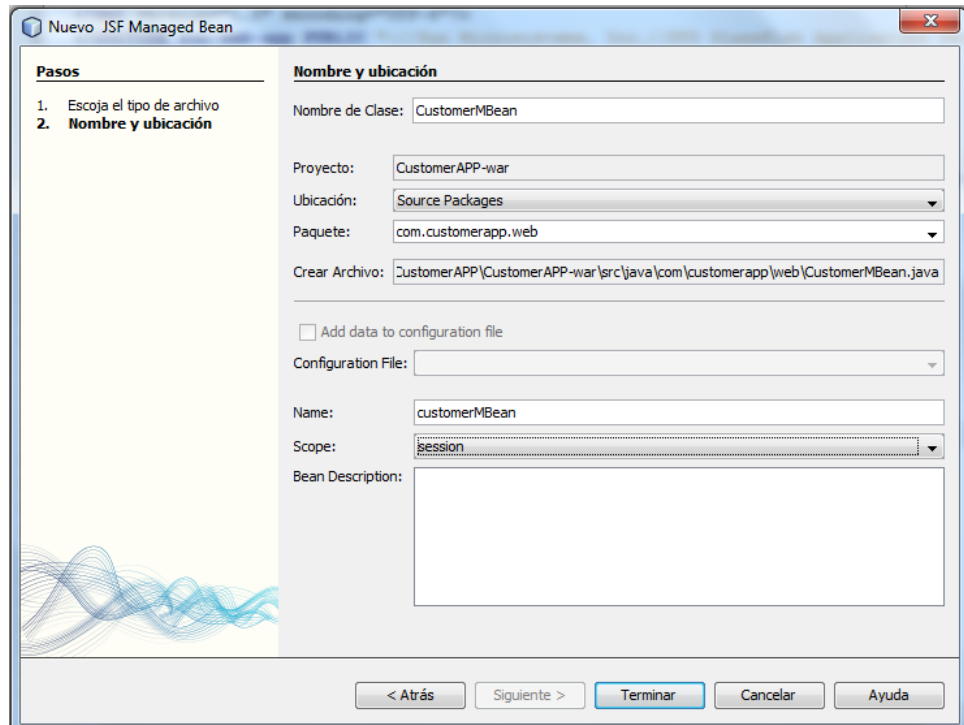


Figura III.36: Usando JSF y PrimeFaces => Crear Nuevo JSF Managed Bean.

- En el editor de código de la clase recién creada, CustomerMBean, haga clic derecho y seleccionar "Insertar Código...", y "Call Enterprise Bean...".
- En el cuadro de diálogo "Call Enterprise Bean", expanda el proyecto **CustomerApp-ejb** y seleccione el **CustomerSessionBean** y seleccione la opción de **No interfaz** (de todos modos las opciones locales y remotos deben ser desactivados, porque se creó el bean de sesión sin interfaz) para la **interfaz de referencia**, y haga clic en Aceptar.

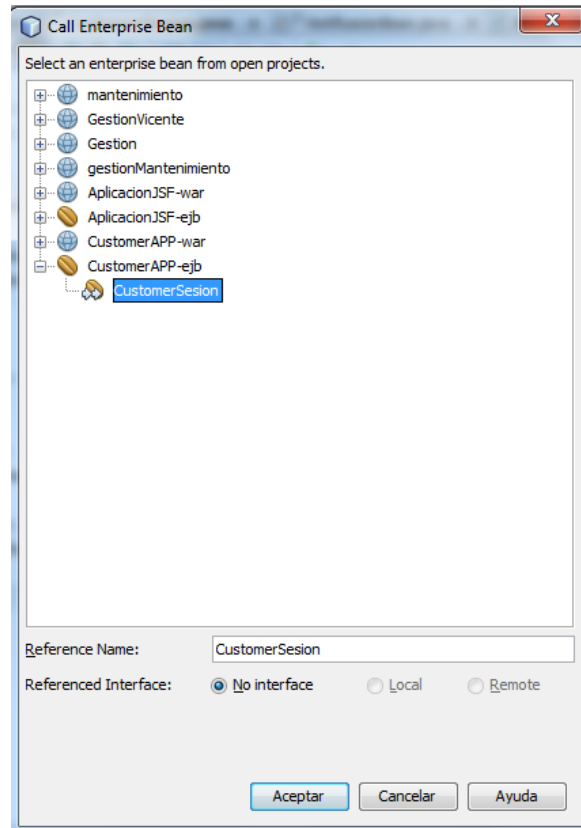


Figura III.37: Usando JSF y PrimeFaces => Call Enterprise Bean.

- Tenga en cuenta la variable que se genera automáticamente, **customerSessionBean** que representa una instancia del bean de sesión, a principios de la declaración de clase.

Añadir el resto de los métodos (propiedades y controladores de acción) y su implementación en la clase como se muestra a continuación, que serán utilizados por las páginas JSF más tarde:

```
package com.customerapp.entity;

import java.util.List;
import javax.ejb.Stateless;
import javax.ejb.LocalBean;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**
 *
 * @author user
```

```
*/
@Stateless
@LocalBean
public class CustomerSesion {
    @PersistenceContext(unitName = "CustomerAPP-ejbPU")
    private EntityManager em;

    public void persist(Object object) {
        em.persist(object);
    }

    public List<Customer> retrieve() {
        return null;
    }

    public Customer update(Customer customer) {
        return null;
    }

    // Add business logic below. (Right-click in editor and choose
    // "Insert Code > Add Business Method")
}
```

3.3.4.2. CREACIÓN DE LA PÁGINA WEB CLIENTE.

Ahora, se debe crear la primera página web que muestra los registros de clientes en la base de datos en forma de tabla.

- En la ventana de Proyectos, haga clic en el proyecto Web, **CustomerApp-war**, y selecciona "Nuevo> XHTML ...", especifique **CustomerList** como nombre de archivo.

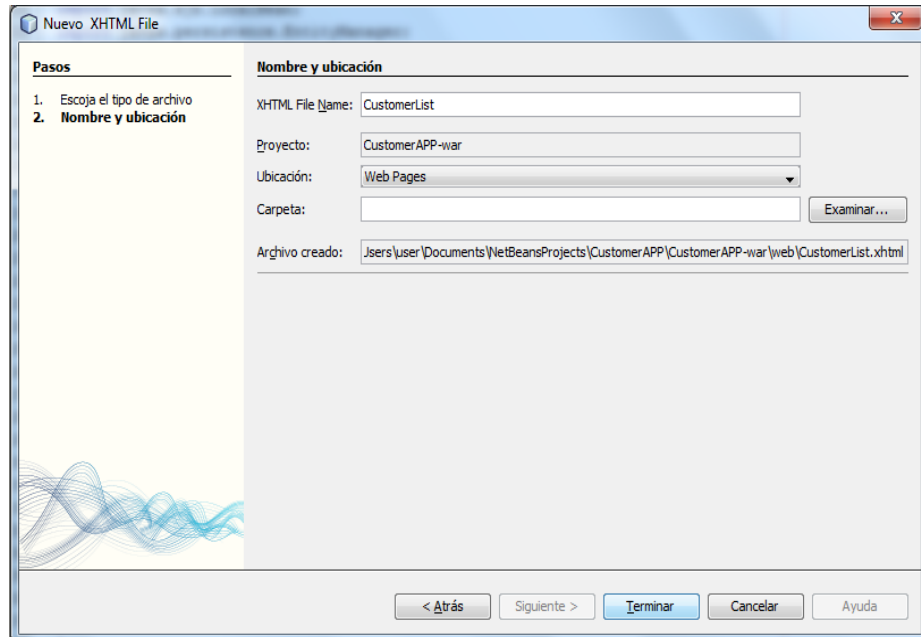


Figura III.38: Usando JSF y PrimeFaces => Nuevo Archivo XHTML.

Nota: Si el elemento "XHTML..." no aparece en la lista del menú, seleccione "Nuevo> Otros..." en su lugar, luego en el diálogo **Nuevo archivo**, seleccione **Web** en Categorías y aparecerá el tipo de archivo **XHTML** a la derecha.

- Por razones de conveniencia, se creará la tabla inicial de la gama de colores en vez de codificar desde cero. En el editor de código, arrastre, "JSF data table from entity" desde la paleta y colóquelo entre el `<h:body>` `</ h: body>` etiquetas del archivo recién generado, **CustomerList.xhtml**.
- Un cuadro de diálogo con el título, "JSF Data Table from Entity" aparece, seleccione "com.customerapp.entity.Customer" como el bean de entidad, y "customer.customers" como la propiedad de un Bean administrado y haga clic en Aceptar.

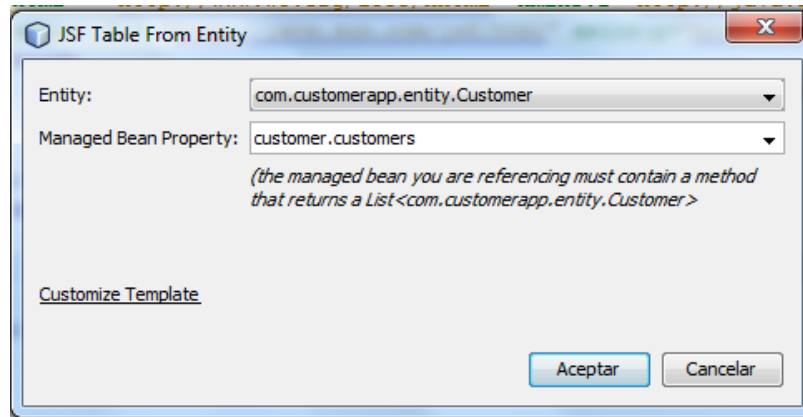


Figura III.39: Usando JSF y PrimeFaces => Tabla JSF desde la Entidad.

Nota: El resultado de esto son las líneas de los códigos generados automáticamente para mostrar una lista predeterminada de los objetos Customer.

- En este punto, se va a ver el resultado de la primera página web creada hasta ahora. En la ventana de Proyectos, haga clic derecho en el proyecto **CustomerApp** y seleccione limpiar y construir, y seleccione Implementar. Para confirmar que el despliegue se realiza correctamente, vaya a la carpeta **Aplicaciones** en el servidor Glassfish en la ventana de Servicios y compruebe si la aplicación, **CustomerApp**, existe.

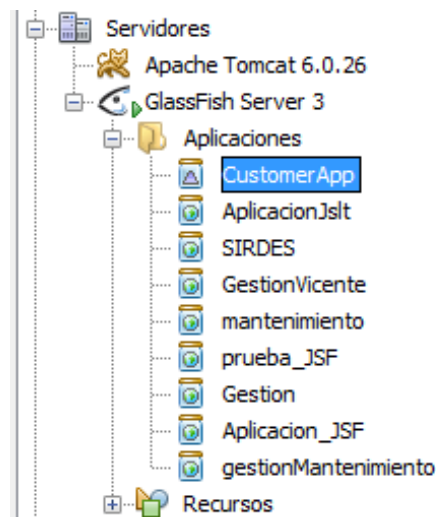


Figura III.40: Usando JSF y PrimeFaces => Aplicaciones del IDE NetBeans.

- Abra el navegador y vaya a la URL, <http://localhost:8080/CustomerApp-war/CustomerList.jsf> y usted debería ver la siguiente pantalla:

Lista de Clientes

| Id | Zip | Name | Addressline1 | Addressline2 | City | State | Phone | Fax | Email | CreditLimit | DiscountCode |
|-----------|------------|-------------------------|----------------------|---------------------|-----------------|--------------|--------------|--------------|---------------------|--------------------|---|
| 1 | 33015 | JumboCom | 111 E. Las Olas Blvd | Suite 51 | Fort Lauderdale | FL | 305-777-4632 | 305-777-4635 | jumbocom@gmail.com | 100000 | com.customerapp.entity.DiscountCode[discountCode=N] |
| 2 | 33055 | Livermore Enterprises | 9754 Main Street | P.O. Box 567 | Miami | FL | 305-456-8888 | 305-456-8889 | www.tsoftt.com | 50000 | com.customerapp.entity.DiscountCode[discountCode=M] |
| 25 | 75200 | Oak Computers | 8989 Qume Drive | Suite 9897 | Houston | TX | 214-999-1234 | 214-999-5432 | www.oakc.com | 25000 | com.customerapp.entity.DiscountCode[discountCode=M] |
| 3 | 12347 | Nano Apple | 8585 Murray Drive | P.O. Box 456 | Alanta | GA | 555-275-9900 | 555-275-9911 | www.nanoapple.net | 90000 | com.customerapp.entity.DiscountCode[discountCode=L] |
| 36 | 94401 | HostProCom | 65653 El Camino | Suite 2323 | San Mateo | CA | 650-456-8876 | 650-456-1120 | www.hostprocom.net | 65000 | com.customerapp.entity.DiscountCode[discountCode=H] |
| 106 | 95035 | CentralComp | 829 Flex Drive | Suite 853 | San Jose | CA | 408-987-1256 | 408-987-1277 | www.centralcomp.com | 26500 | com.customerapp.entity.DiscountCode[discountCode=L] |
| 149 | 95117 | Golden Valley Computers | 4381 Kelly Ave | Suite 77 | Santa Clara | CA | 408-432-6868 | 408-432-6899 | www.gvc.net | 70000 | com.customerapp.entity.DiscountCode[discountCode=L] |
| 863 | 94401 | Top Network Systems | 456 4th Street | Suite 45 | Redwood City | CA | 650-345-5656 | 650-345-4433 | www.hpsys.net | 25000 | com.customerapp.entity.DiscountCode[discountCode=N] |
| 777 | 48128 | West Valley Inc. | 88 North Drive | Building C | Dearborn | MI | 313-563-9900 | 313-563-9911 | www.westv.com | 100000 | com.customerapp.entity.DiscountCode[discountCode=L] |
| 753 | 48128 | Ford Motor Co | 2267 Michigan Ave | Building 21 | Dearborn | MI | 313-787-2100 | 313-787-3100 | www.parts@ford.com | 5000000 | com.customerapp.entity.DiscountCode[discountCode=H] |
| 722 | 48124 | Big Car Parts | 52963 Outer Dr | Suite 35 | Detroit | MI | 313-788-7682 | 313-788-7600 | www.sparts.com | 50000 | com.customerapp.entity.DiscountCode[discountCode=N] |

Figura III.41: Usando JSF y PrimeFaces => Pantalla Cruda de la Lista de Clientes.

Nota: La pantalla es muy cruda y sin ningún tipo de embellecimiento porque hasta ahora, sigue siendo una página JSF sencilla. Así que se va a modificar la página para mostrar sólo las columnas de interés y utilizar **PrimeFaces dataTable**.

- Para utilizar **PrimeFaces dataTable** en la página CustomerList.xhtml, basta con sustituir las etiquetas `<h:dataTable>` y `<h:column>` por `<p:dataTable>` y `<p:column>`, respectivamente, y modificar el resto del código.
- El resultado de estos cambios deberían dar a la página este aspecto:

Lista de Clientes

| Customer ID | Name | CreditLimit | Discount % | State | City | Zip | Phone | Email |
|-------------|-------------------------|-------------|------------|-------|-----------------|-------|--------------|---------------------|
| <u>1</u> | JumboCom | 100000 | 0.00 | FL | Fort Lauderdale | 33015 | 305-777-4632 | jumbocom@gmail.com |
| <u>2</u> | Livermore Enterprises | 50000 | 11.00 | FL | Miami | 33055 | 305-456-8888 | www.tsoftt.com |
| <u>25</u> | Oak Computers | 25000 | 11.00 | TX | Houston | 75200 | 214-999-1234 | www.oakc.com |
| <u>3</u> | Nano Apple | 90000 | 7.00 | GA | Alanta | 12347 | 555-275-9900 | www.nanoapple.net |
| <u>36</u> | HostProCom | 65000 | 16.00 | CA | San Mateo | 94401 | 650-456-8876 | www.hostprocom.net |
| <u>106</u> | CentralComp | 26500 | 7.00 | CA | San Jose | 95035 | 408-987-1256 | www.centralcomp.com |
| <u>149</u> | Golden Valley Computers | 70000 | 7.00 | CA | Santa Clara | 95117 | 408-432-6868 | www.gvc.net |
| <u>863</u> | Top Network Systems | 25000 | 0.00 | CA | Redwood City | 94401 | 650-345-5656 | www.hpsys.net |
| <u>777</u> | West Valley Inc. | 100000 | 7.00 | MI | Dearborn | 48128 | 313-563-9900 | www.westv.com |
| <u>753</u> | Ford Motor Co | 5000000 | 16.00 | MI | Dearborn | 48128 | 313-787-2100 | www.parts@ford.com |

Figura III.42: Usando JSF y PrimeFaces => Pantalla de Lista de Clientes usando JSF.

- Ahora se procede a crear la página de detalles.
- En el editor de código, arrastre, "JSF data table from entity" desde la paleta y colóquelo entre el `<h:body>` `</ h: body>` etiquetas del archivo recién generado, **CustomerDetails.xhtml**.
- Aparece un cuadro de diálogo con el título "JSF Table from Entity" , seleccione "com.customerapp.entity.Customer" como la Entidad, y "customer.showDetails()" como la propiedad de un Bean administrado y haga clic en Aceptar:

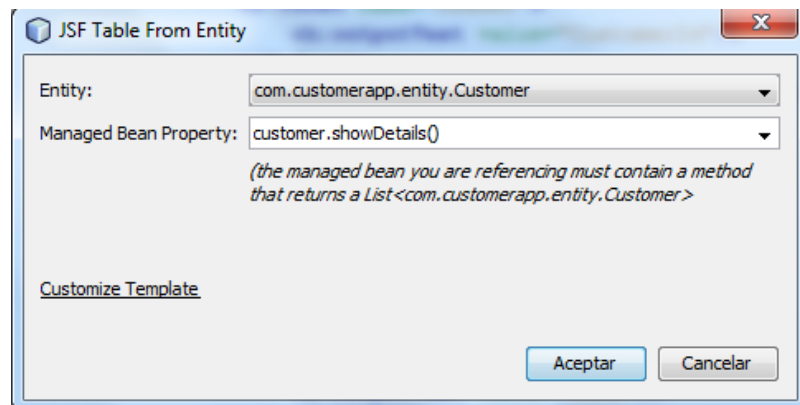


Figura III.43: Usando JSF y PrimeFaces => JSF Table From Entity.

Nota: El resultado de esto son las líneas de código generadas automáticamente para mostrar la etiqueta y el campo de entrada de todos los atributos del objeto de cliente en una cuadrícula.

- Para permitir la navegación de la página cliente con la de detalles, y viceversa, es necesario modificar el **faces-config.xml** con el editor de **PageFlow** y conecte las 2 páginas, como se muestra en el siguiente diagrama:

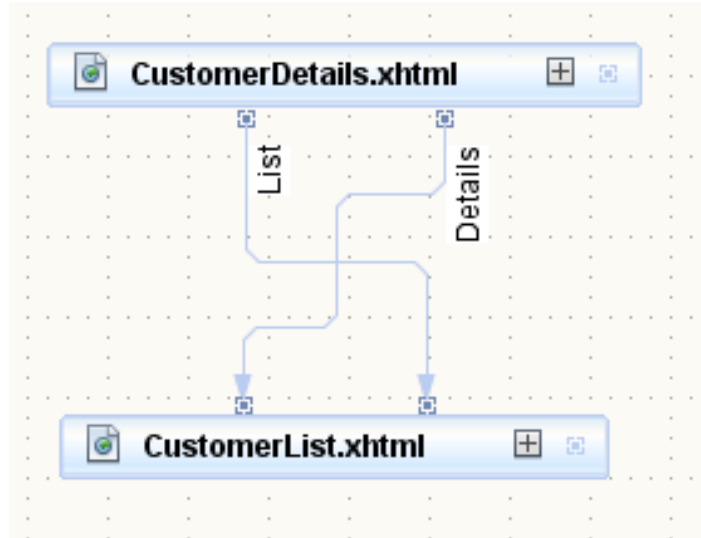


Figura III.44: Usando JSF y PrimeFaces => Archivo *faces-config.xml* con PageFlow.

Nota: La **LISTA DE DETALLES** debe coincidir con la cadena de retorno de los métodos **List** y **ShowDetails** en el **CustomerMBean**.

- Para ver el resultado, guarde y despliegue la aplicación, vaya a la página del listado de clientes en la URL, <http://localhost:8080/CustomerAppwar/CustomerList.jsf>, y haga clic en el ID de cliente en la primera fila de la tabla:

Create/Edit

| | |
|---------------|--|
| CustomerId: | <input type="text" value="25"/> |
| Zip: | <input type="text" value="75200"/> |
| Name: | <input type="text" value="Oak Computers"/> |
| Addressline1: | <input type="text" value="8989 Qume Drive"/> |
| Addressline2: | <input type="text" value="Suite 9897"/> |
| City: | <input type="text" value="Houston"/> |
| State: | <input type="text" value="TX"/> |
| Phone: | <input type="text" value="214-999-1234"/> |
| Fax: | <input type="text" value="214-999-5432"/> |
| Email: | <input type="text" value="www.oakc.com"/> |
| CreditLimit: | <input type="text" value="25000"/> |
| DiscountCode: | <input type="text" value="▼"/> |

Figura III.45: Usando JSF y PrimeFaces => Formulario de Ingreso de Clientes.

De este modo se ha enlazado el ID de cada registro a la tabla de detalles, creando así la navegación entre páginas.

De esta manera se ha podido integrar el framework EJB con JSF.

CAPÍTULO IV.

4. DESARROLLO DE METODOLOGÍA ERCA WEB

4.1.¿QUÉ ES UNA METODOLOGÍA?

Una metodología es un conjunto de etapas formalmente estructuradas, de manera que brinden a los interesados parámetros de acción en el desarrollo de sus proyectos: plan general y detallado, tareas y acciones, tiempos, aseguramiento de la calidad, involucrados, etapas, revisiones de avance, responsables, recursos requeridos, entre otros²⁰.

La metodología es el enlace entre el sujeto y el objeto de conocimiento. Sin ella es prácticamente imposible lograr el camino que conduce al conocimiento científico²¹.

La metodología es necesaria para que un equipo de profesionales alcance un resultado homogéneo tal como si lo hiciera uno solo, por lo que resulta habitual el uso de metodologías para el desarrollo de aplicaciones informáticas.

4.2.METODOLOGÍAS DE DESARROLLO DE SOFTWARE.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, lo que se obtiene son clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos. Sin embargo, muchas veces no se toma en

²⁰ <http://www.angelfire.metodologia.doc>

²¹ <https://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r33282.PDF>

cuenta el utilizar una metodología adecuada, sobre todo cuando se trata de proyectos pequeños de dos o tres meses. Lo que se hace con este tipo de proyectos es separar rápidamente el aplicativo en procesos, cada proceso en funciones, y por cada función determinar un tiempo aproximado de desarrollo.

Cuando los proyectos que se van a desarrollar son de mayor envergadura, ahí si toma sentido el basarnos en una metodología de desarrollo, y se busca cual sería la más apropiada para nuestro caso. Lo cierto es que muchas veces no se encuentra la más adecuada y se termina por hacer o diseñar nuestra propia metodología, algo que por supuesto no está mal, siempre y cuando cumpla con el objetivo.

Las Metodologías de Desarrollo de Software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software. Se podrían clasificar en dos grandes grupos:

1. **Metodologías Orientadas al Control de los Procesos.** Establecen rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas metodologías son llamadas Metodologías Tradicionales o Pesadas.

2. **Metodologías Orientadas a la Interacción con el Cliente y el Desarrollo Incremental del Software.** Muestra versiones parcialmente del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Estas son llamadas Metodologías Ágiles.

4.2.1. METODOLOGÍAS TRADICIONALES O PESADAS.

Son las más tradicionales, se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, ya que pretende prever todo de antemano. Este tipo de metodologías son mas eficaces y necesarias cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que

son necesarios emplear, donde una gran organización es requerida. Una de las metodologías tradicionales más conocidas y utilizadas es la Metodología MSF (Microsoft Solution Framework).

4.2.1.1.METODOLOGÍA MICROSOFT SOLUTION FRAMEWORK (MSF).

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

MSF tiene las siguientes características:

- **Adaptable:** es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- **Escalable:** puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- **Flexible:** es utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología Agnóstica:** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

- ***Modelo de Arquitectura del Proyecto:*** Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.
- ***Modelo de Equipo:*** Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles.
- ***Modelo de Proceso:*** Diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto,

describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de equipo.

- **Modelo de Gestión del Riesgo:** Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.
- **Modelo de Diseño del Proceso:** Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.
- **Modelo de Aplicación:** Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores.

La Metodología MSF se adapta a proyectos de cualquier dimensión y de cualquier tecnología.

4.2.2. METODOLOGÍAS ÁGILES.

Este tipo de metodologías nace en febrero del 2001 en una reunión celebrada en Utah-EEUU.

Las principales ideas de la metodología ágil son:

- Se encarga de valorar al individuo y las iteraciones del equipo más que a las herramientas o los procesos utilizados.
- Se hace mucho más importante crear un producto software que funcione que escribir mucha documentación.
- El cliente está en todo momento colaborando en el proyecto.
- Es más importante la capacidad de respuesta ante un cambio realizado que el seguimiento estricto de un plan.

Como una metodología ágil se tiene a la Programación Extrema o XP (Extreme Programming).

4.3.METODOLOGÍA PROPUESTA ERCA WEB.

Esta metodología se centra únicamente en el desarrollo de aplicaciones web de tipo empresarial utilizando tecnologías actuales para este tipo de software, tales como son Java Server Faces (JSF) y Enterprise Java Bean (EJB).

Mediante la integración de éstas tecnologías se proporcionará un modelo para la construcción de aplicaciones web que permitirá a los interesados desarrollar aplicaciones web empresariales sin mayor esfuerzo.

4.3.1. FASES DE LA METODOLOGÍA ERCA WEB.

La metodología ERCA Web está organizada en seis fases las mismas que se describen en el siguiente gráfico junto con las principales actividades.



Figura IV.46: Fases de la Metodología ERCA Web.

El objetivo de cada una de las fases se describe a continuación:

4.3.1.1. Fase 1: Planificación.

Definir concretamente, con los interesados relevantes, un plan inicial para el desarrollo de la aplicación determinada, realizando el análisis de los requerimientos del usuario.

4.3.1.2. Fase 2: Desarrollo de Base de Datos.

Analizar y diseñar la base de datos de acuerdo a los requerimientos establecidos previamente por parte del cliente, con el fin de presentar a estos un prototipo inicial de la aplicación a desarrollar.

4.3.1.3.Fase 3: Desarrollo de Módulo EJB.

En esta fase se va a utilizar el framework EJB con el fin de generar la capa de acceso a datos y la lógica de Negocios.

4.3.1.4.Fase 4: Integración de Frameworks.

Creación de la capa de Presentación mediante JSF, la integración con el módulo EJB y el desarrollo complementario de la capa de lógica de negocio.

4.3.1.5.Fase 5: Construcción Final de la Aplicación.

En esta fase se desarrollará la parte final de la aplicación, haciendo uso de templates para el correcto desarrollo de las interfaces. Luego de esto se deberá presentar al cliente el producto final, de modo que pueda dar nuevas sugerencias del software y adaptarlas posteriormente.

4.3.1.6.Fase 6: Implementación e Implantación.

Esta etapa corresponde a la implementación e implantación de la aplicación realizada, en las máquinas de los usuarios, con el fin de verificar si se han cumplido a cabalidad todos los requerimientos establecidos en un principio por parte de los usuarios.

4.3.2. FASE 1: PLANIFICACIÓN.

4.3.2.1.ANÁLISIS DE REQUERIMIENTOS.

Esta etapa es la primera actividad que se debe realizar en el desarrollo de un Sistema de Información. Comienza después de que un usuario ha detectado una ausencia, falla o falta de oportunidad de la información o simplemente, luego que la organización ha determinado un cambio en sus políticas, reglas o tecnologías a aplicar.

En esta etapa, se debe responder a una pregunta fundamental: ¿Qué es lo que quiere el cliente? y para ello, se debe diagnosticar la situación actual, recopilar los requerimientos del cliente, tanto en relación al sistema, como generales respecto del área informática, es decir la situación ideal, para así poder definir alternativas de solución, según las cuales se podrá avanzar desde lo que hoy se posee, hacia el punto que se pretende llegar.

Como parte del trabajo, se debe señalar cuál de las alternativas, es a nuestro juicio la más conveniente (y justificarlo) en la propuesta.

En esta etapa se logra claridad sobre lo que desea el usuario y la forma en la cual se le va a presentar la solución que se está buscando.

Los requerimientos son parte fundamental en un sistema de información ya que estos representan el conjunto completo de resultados a ser obtenidos una vez que se utilice el sistema. Los requerimientos de sistemas deben mostrar todo lo que el sistema debe hacer más todas las restricciones sobre la funcionalidad. Los requerimientos forman un modelo completo, representando el sistema total a algún nivel de abstracción.

De este modo se puede decir que un requerimiento es una condición o capacidad a la que el sistema (siendo construido) debe conformar. Un requerimiento de software puede ser definido como:

- Una capacidad del software necesaria por el usuario para resolver un problema o alcanzar un objetivo.
- Una capacidad del software que debe ser reunida o poseída por un sistema o componente del sistema para satisfacer un contrato, especificación, estándar, u otra documentación formal.

Los requerimientos de usuario representan el conjunto completo de resultados a ser obtenidos utilizando el sistema²². Estos se dividen así mismos en: Funcionales y no Funcionales.

Requerimientos no Funcionales.

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema, como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en las interfaces del sistema. Sin embargo, estos requerimientos no siempre se refieren al sistema de software a desarrollar.

²² Requerimientos
<http://www.galeon.com/zuloaga/Doc/AnalisisRequer.pdf>

Para poder especificar los requerimientos no funcionales, es necesario establecer algunas métricas:

Tabla IV.I: Métricas de Requerimientos no Funcionales.

| PROPIEDAD | MEDIDA |
|--------------------------|---|
| <i>Rapidez.</i> | <ul style="list-style-type: none">➤ Transacciones procesadas por segundo.➤ Tiempo de respuesta al usuario y a eventos.➤ Tiempo de actualización de la pantalla. |
| <i>Tamaño.</i> | <ul style="list-style-type: none">➤ Kilobytes.➤ Tamaño de RAM. |
| <i>Facilidad de uso.</i> | <ul style="list-style-type: none">➤ Tiempo de capacitación.➤ Número de ventanas de ayuda. |
| <i>Fiabilidad.</i> | <ul style="list-style-type: none">➤ Tiempo promedio entre fallas.➤ Probabilidad de no disponibilidad.➤ Tasa de ocurrencia de las fallas.➤ Disponibilidad. |
| <i>Robustez.</i> | <ul style="list-style-type: none">➤ Tiempo de reinicio después de fallas.➤ Porcentaje de eventos que provocan las fallas.➤ Probabilidad de corrupción de los datos después de las fallas. |
| <i>Portabilidad.10</i> | <ul style="list-style-type: none">➤ Porcentaje de declaraciones dependientes del objetivo.➤ Número de sistemas objetivo. |

Requerimientos Funcionales.

Describen la interacción entre el sistema y su ambiente independientemente de su implementación. El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema.

Un requisito funcional define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo

los casos de uso serán llevados a la práctica. Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación.

4.3.2.2.BOSQUEJO DE BASE DE DATOS.

En esta actividad se toma los requerimientos y especificaciones de la etapa de análisis y se determina la mejor manera de satisfacerlos, según las apreciaciones del Usuario y del que lo desarrolla, a través de un diseño inicial de base de datos.

La finalidad de este bosquejo es establecer, a partir del trabajo con los usuarios, las líneas básicas del proyecto, principalmente en lo que respecta a funcionalidad y estructura de la base de datos.

Este bosquejo deberá exponer claramente las entidades involucradas, los atributos, las relaciones y los procedimientos almacenados.

Una vez realizado el bosquejo, el coordinador podrá determinar si la idea ha sido correcta, o si se ha presentado algún desvío con respecto a las metas fijadas, pudiendo en este momento corregir y “repensar” sin ninguna dificultad el modelo a desarrollar.

4.3.2.3.APRENDIZAJE DE FRAMEWORK JSF.

Esta actividad se realiza con el fin de que los interesados se familiaricen con el framework JSF desde la planificación del proyecto. Los desarrolladores tienen que realizar un estudio minucioso sobre las ventajas y desventajas del framework JSF, sobre los componentes y las características principales de este framework.

4.3.3. FASE 2: DESARROLLO DE BASE DE DATOS.

Una vez realizada la fase de planificación que es la parte previa al desarrollo, se procederá a desarrollar la base de datos, que es la parte modular y fundamental de esta metodología.

En esta fase se debe dedicar un gran tiempo a la base de datos, ya que desde aquí se asegurará que el proyecto sea realizado sin ningún contratiempo e inconveniente.

Las actividades que corresponden a esta fase son las siguientes:

4.3.3.1.DISEÑO DE BASE DE DATOS.

Toda gestión empresarial requiere, hoy en día, manejar grandes cantidades de datos. Cualquier empresa que se precie debe tener almacenados todos estos datos en una base de datos para poder realizarlos mediante una aplicación profesional. Sin esta funcionalidad resultaría imposible tratar y manejar en su totalidad los datos que lleva a cabo la empresa y se perdería un tiempo y un dinero muy valiosos.

Uno de los pasos cruciales en la construcción de una aplicación que maneje una base de datos, es sin duda, el diseño de la base de datos. Si las tablas no son definidas apropiadamente, se tendrá muchos problemas al momento de ejecutar consultas a la base de datos para tratar de obtener algún tipo de información.

Por lo cual es muy importante asegurarse que la base de datos está correctamente diseñada para que tenga eficiencia y que se pueda seguir utilizando por largo tiempo.

Dependiendo de los requerimientos de la aplicación, el diseño de la base de datos puede ser algo complejo, pero con algunas reglas simples que se tengan será mucho más fácil crear una base de datos perfecta para un determinado proyecto.

4.3.3.2.CONSTRUCCIÓN DE PROTOTIPO EN JSF.

Una vez diseñada la base de datos, se desarrolla un prototipo utilizando el framework JSF, el cual se estudió previamente en la fase de planificación. El objetivo de esta actividad es poder tener una visión clara de lo que se pretende hacer, además de presentar al cliente los primeros avances del proyecto, logrando así una interacción con el o ellos.

Cabe indicar que un prototipo no es el diseño final de la aplicación, más bien es una muestra de cómo puede quedar la aplicación. Este prototipo está sujeto a cambios y utiliza la base de datos previamente diseñada, la cual también puede ser cambiada o modificada si el cliente así lo desea.

4.3.3.3.PRESENTACIÓN DE PROTOTIPO.

En esta actividad, como su nombre lo indica, se deberá presentar el prototipo construido anteriormente al cliente. Al presentar el prototipo, el cliente podrá establecer nuevos cambios, nuevos requerimientos, etc., de modo que se establezca una idea más clara de

lo que se quiere. Con esto se logra involucrar más al cliente haciéndolo parte fundamental del desarrollo de la aplicación. De esta manera se garantiza que se cumpla efectivamente con todo lo que realmente quiere el cliente.

4.3.3.4.DISEÑO FINAL DE BASE DE DATOS.

En esta actividad se realiza un remodelamiento de la base de datos, de acuerdo con los cambios o sugerencias que haya establecido el cliente. Como se dijo anteriormente, la base de datos es una parte medular en el desarrollo de una aplicación, utilizando la metodología ERCA Web.

En este remodelamiento se debe diseñar correctamente la base de datos, construir procedimientos almacenados, funciones, triggers, si fuere necesario.

Una vez remodelada la base de datos, no se deberá modificar en las actividades posteriores, a no ser la inclusión de un nuevo procedimiento o función, sin alterar el diseño lógico de la base de datos, sus relaciones, etc.

4.3.4. FASE 3: DESARROLLO DE MÓDULO EJB.

4.3.4.1.APRENDIZAJE DE FRAMEWORK EJB.

En esta actividad se debe realizar un estudio completo del framework EJB, de modo que se puede aprovechar la mayoría de sus funcionalidades y ventajas. El framework EJB es el componente que permite generar la capa de acceso a datos de la aplicación y parte de la lógica de negocio, por lo cual es muy importante que se conozcan sus componentes y su correcto funcionamiento.

4.3.4.2.CREACIÓN DEL MÓDULO EJB.

El framework EJB, a su vez, está compuesto por tres componentes básicos que son: los EJB Entity, los EJB Sessions y los Message Drive Bean. Cada uno de estos componentes forma lo que es el acceso a datos de nuestra aplicación.

En esta actividad se procederá a crear el módulo EJB, a través del IDE NetBeans.

Primeramente crear un Nuevo Proyecto de tipo empresarial, así como se muestra en la figura:

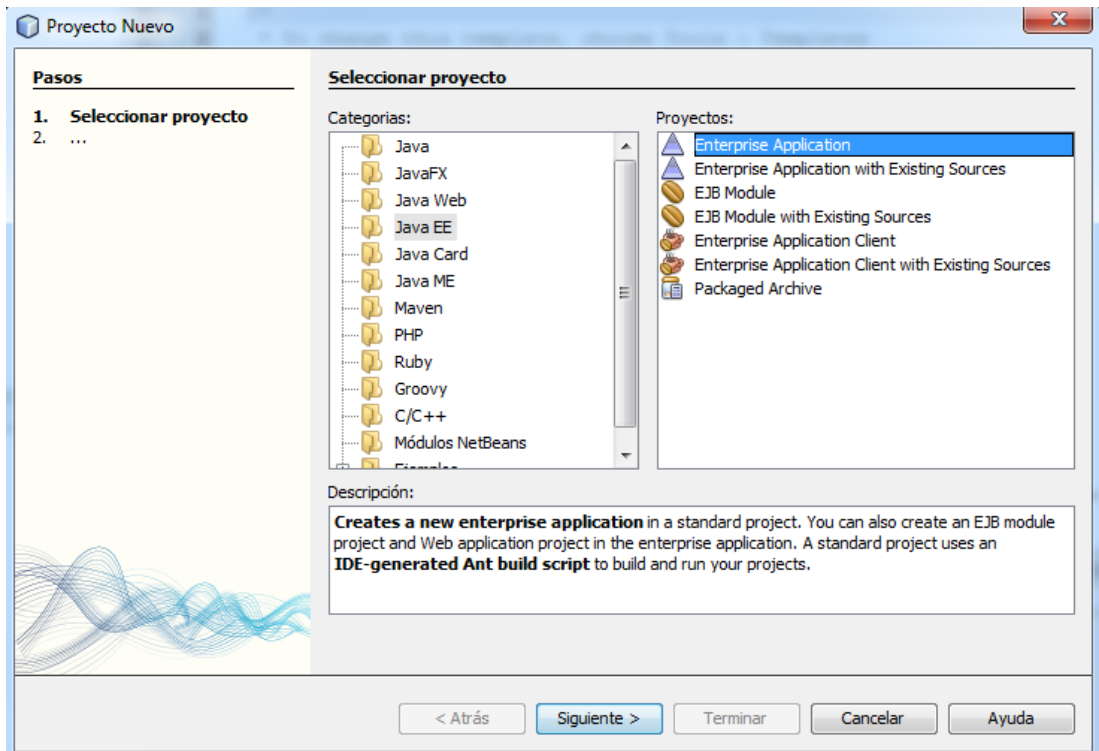


Figura IV.47: Creación Módulo EJB – Creando Nuevo Proyecto

Al seleccionar una aplicación empresarial, el IDE automáticamente creará el módulo EJB como otro proyecto.

4.3.4.3. GENERACIÓN DE LA CAPA DE ACCESO A DATOS Y LÓGICA DE NEGOCIO INICIAL.

Una vez creado el módulo EJB, se debe generar la capa de acceso a datos y parte de la lógica de negocio.

Por lo cual, en el módulo creado previamente, se deberá establecer los EJB Entity, los cuales hacen incapie al acceso a datos, y los EJB Sessions que se refieren a la lógica de negocio.

Para establecer los EJB Entity, se debe dar clic derecho en el módulo EJB generado y seleccionar en *Nuevo -> Clase entidad a partir de Base de Datos*. De este modo aparece un cuadro diálogo donde se debe ubicar la conexión a utilizar.

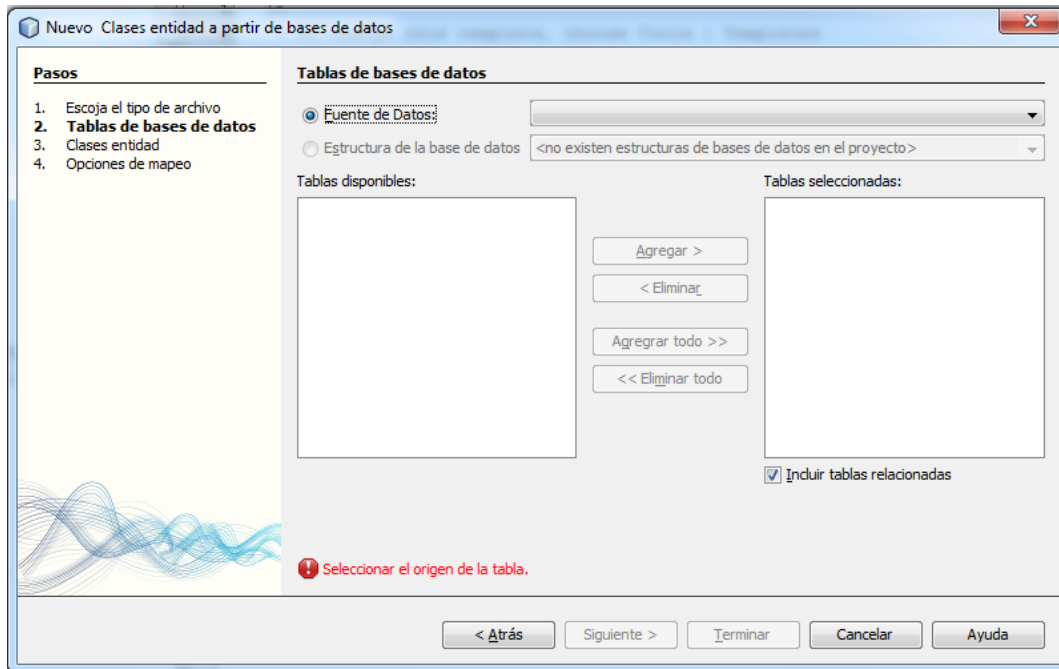


Figura IV.48: Acceso a Datos => Crear Clases Entidad a partir de base de datos.

Luego se debe seleccionar las tablas de las cuales se van a generar las clases entidad del módulo EJB. De este modo se genera los EJB Entity que son nada más que una copia/reflejo de la información de establecida en la base de datos.

Luego, para generar los EJB de Sesión, de igual forma dar clic derecho en el módulo EJB y seleccionar *Nuevo->Session Bean for Entity Classes*. Se generarán las clases donde se establecerá la lógica de negocio. La creación de los EJB de Session permite la reutilización de estos mismos métodos en otras páginas.

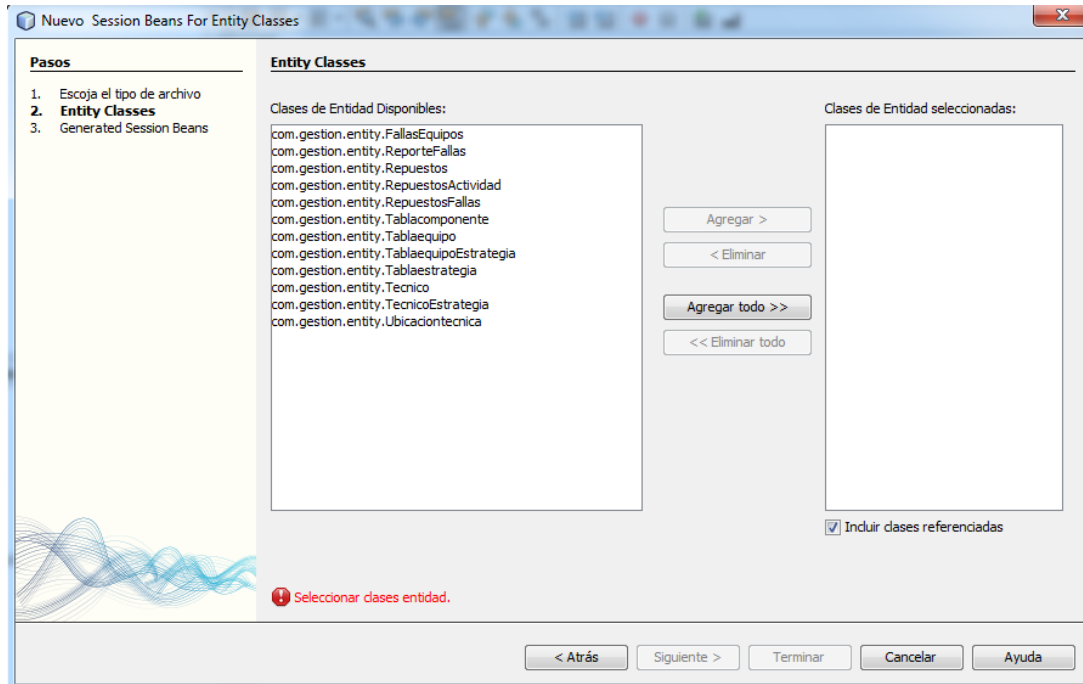


Figura IV.49: Lógica de Negocio => Session Beans para Clases Entidad.

Agregar todas la entidades deseadas para que se generen los beans de sesión, así se obtendrán nuevas clases según la entidad en donde se establecerá la programación modular para la aplicación, así como el CRUD de cada entidad.

4.3.5. FASE 4: INTEGRACIÓN DE FRAMEWORKS.

4.3.5.1. CREACIÓN DE MÓDULO JSF.

En esta actividad se creará el módulo JSF, el cual se lo integrará al módulo EJB. JSF se centra en el desarrollo de las interfaces de usuario, simplificando su construcción.

Como anteriormente se creó una aplicación empresarial, esta a su vez genera dos módulos; el primero es el módulo EJB y el segundo es un módulo tipo web. En este segundo módulo, se debe seleccionar el framework JSF para activar las características principales del mismo. Una vez seleccionado el framework se activarán todas las etiquetas correspondientes a éste y se está listo para implementarlo en nuestra aplicación.

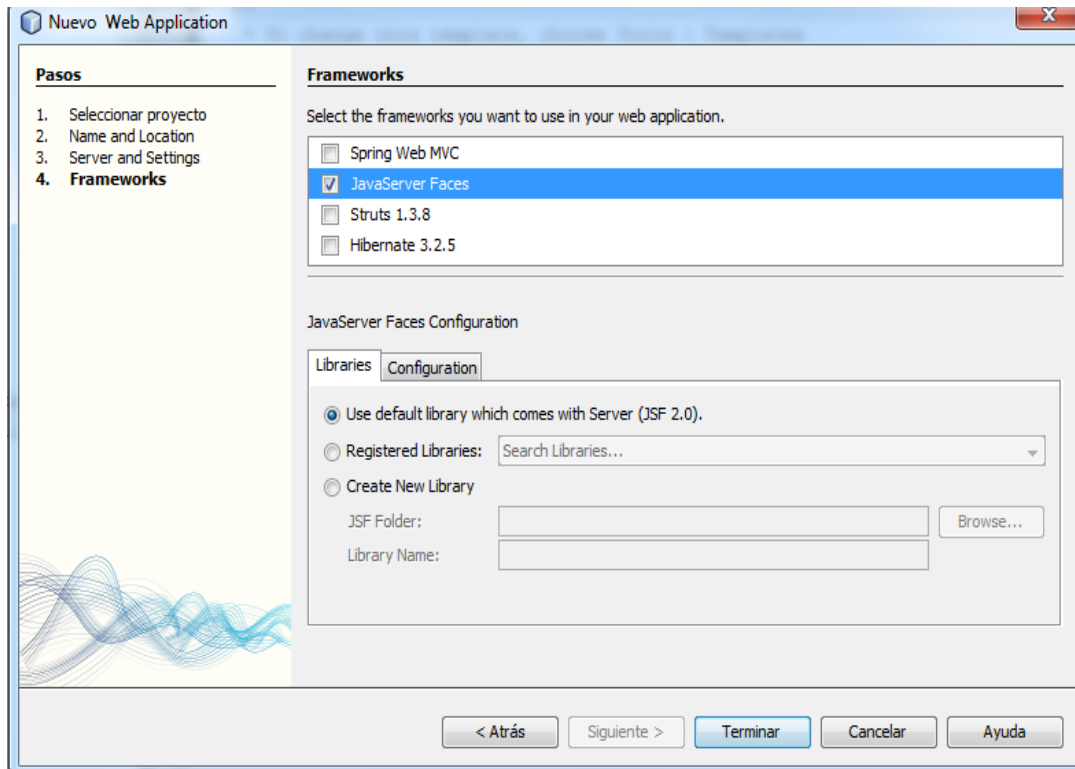


Figura IV.50: Seleccionando Framework JSF.

4.3.5.2. INTEGRACIÓN DE JSF Y EJB.

Una vez creado los módulos EJB y JSF, es necesario realizar la integración de estos, con el fin de crear una aplicación robusta de tipo empresarial.

El framework EJB funciona como un módulo separado el cual puede ser integrado a otros módulos de una manera muy sencilla, simplemente haciendo referencia al EJB desde cualquier otro módulo.

La arquitectura de esta integración representa las mejores prácticas en el desarrollo de una aplicación empresarial, por lo que la modularidad, escalabilidad y reusabilidad se tienen en cuenta, y además se organiza en distintos niveles: Presentación, lógica de negocios, acceso a datos y de datos, donde cada uno tiene un papel importante que desempeñar y se separa el uno del otro.

Si un desarrollador quiere realizar una aplicación JSF + EJB, la forma correcta es crear el modelo EJB, y realizar las llamadas a éste modelo desde los JSF Managed Java Beans de la aplicación.

Para integrar ambos módulos, simplemente desde el módulo JSF se hace referencia, en las librerías, al módulo EJB, así como se muestra en la figura:

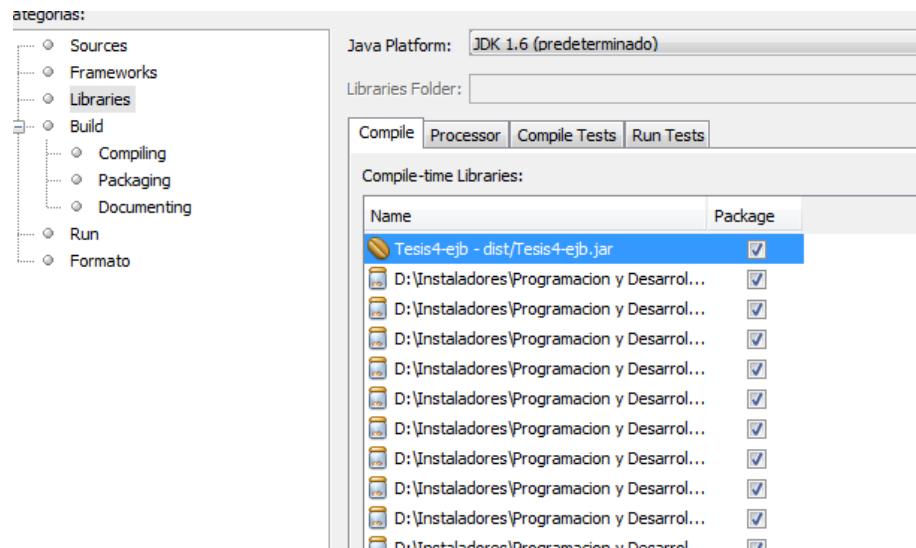


Figura IV.51: Integración de EJB a JSF.

4.3.5.3. DESARROLLO FINAL DE CAPA DE LÓGICA DE NEGOCIO.

Anteriormente se dijo que el módulo EJB solo genera una parte de la lógica de negocio, pues bien, en esta actividad se realizará totalmente la lógica de negocio, conforme a los requerimientos establecidos por el cliente.

4.3.6. FASE 5: CONSTRUCCIÓN FINAL DE LA APLICACIÓN.

Siguiendo correctamente las actividades anteriores, la parte lógica de la aplicación ya estaría totalmente desarrollada, sin embargo, esto para el cliente no representa nada, ya que lo que el cliente desea es poder interactuar con el software desarrollado a través de las interfaces de usuarios. Por lo cual en esta fase, se dedicará totalmente al diseño de las interfaces, de modo que la aplicación sea presentada al cliente.

En esta fase se desarrollarán las siguientes actividades:

4.3.6.1. PREPARACIÓN DE TEMPLATES.

Los templates son plantillas de interfaces de usuario que una vez implementados en una aplicación generan interfaces de usuario amigables.

El uso de templates trae consigo las ventajas de tener un código ordenado (sobre todo en aplicaciones grandes), fácil de desarrollar y reusar.

Por lo general JSF hace uso de estas plantillas para un mejor diseño de interfaces. En JSF existen un sin número de templates, de los cuales se deberá seleccionar el que más conveniente. Entre los templates que usa JSF se tienen:

- MyFaces.
- ICEFaces.
- RichFaces.
- RC Faces.
- PrimeFaces.

Los templates son archivos .jar que son referenciados desde las librerías del módulo JSF.

En esta actividad se deberá seleccionar y preparar los templates necesarios para el desarrollo de la aplicación.

4.3.6.2.DESARROLLO FINAL DE INTERFACES.

Una vez preparado los templates, se deberán referenciar desde las librerías. En esta actividad, se realiza el diseño final de las interfaces, creando las páginas de interacción con el usuario, los comandos, colores, etc.

4.3.6.3.PRESENTACIÓN FINAL.

Al desarrollar las interfaces, se puede decir que la aplicación está prácticamente terminada, por lo cual, en esta actividad se realiza una presentación final al cliente, el cual dará nuevas sugerencias, si las hubiera, sobre cómo deberían estar organizadas las interfaces, los colores, etc.

4.3.6.4.SUGERENCIAS Y MEJORAMIENTO.

De acuerdo a la presentación final al cliente, este dará nuevas sugerencias sobre la aplicación. Estas sugerencias no deberán afectar en nada a la lógica de negocio de la aplicación, es decir, el cliente no podrá sugerir la adición de un nuevo requerimiento o el cambio de alguno, mas bien estas sugerencias se centran en las interfaces.

Una vez dada las sugerencias se procederá al mejoramiento y acoplo de estas sugerencias en la aplicación.

4.3.7. FASE 6: IMPLEMENTACIÓN E IMPLANTACIÓN.

Como su nombre lo indica, esta etapa corresponde a la implementación de la aplicación realizada, en las máquinas de los usuarios, con el fin de verificar si se han cumplido a cabalidad todos los requerimientos establecidos en un principio por parte de los usuarios.

Las actividades correspondientes a esta fase son las siguientes:

4.3.7.1. INSTALACIÓN DE SERVIDORES.

En esta actividad se deberán instalar los servidores necesarios para que la aplicación web funcione correctamente. Los servidores más importantes son: El servidor de Base de Datos y el servidor de Aplicaciones Web.

Un servidor de base de datos es, como su nombre lo indica, un programa que provee servicios de base de datos a otros programas u otras computadoras, como es definido por el modelo cliente/servidor. También puede hacer referencia a aquellas computadoras (servidores) dedicadas a ejecutar esos programas, prestando el servicio.

Entre los servidores más comunes se tiene el Servidor MySQL. Este servidor es muy rápido, seguro, y fácil de usar. Si eso es lo que se está buscando, se le debe dar una oportunidad a MySQL.

Una de las herramientas más sencillas para instalar el servidor MySQL es el denominado XAMPP que es un servidor independiente de plataforma que consiste principalmente en la Base de Datos MySQL, el servidor Apache, PHP y Perl.



Figura IV.52: Logo XAMPP

Comúnmente, para crear una base de datos en MySQL es necesario establecer un conjunto de comandos SQL. Debido a que no se conocen todos los comandos

necesarios, es muy difícil crear una base de datos con MySQL, sin embargo, existen alternativas que permiten configurar una base de datos en MySQL utilizando una interfaz gráfica. Estas alternativas se denominan: MySQL Workbench, MySQL SQL-Front, y otras más.

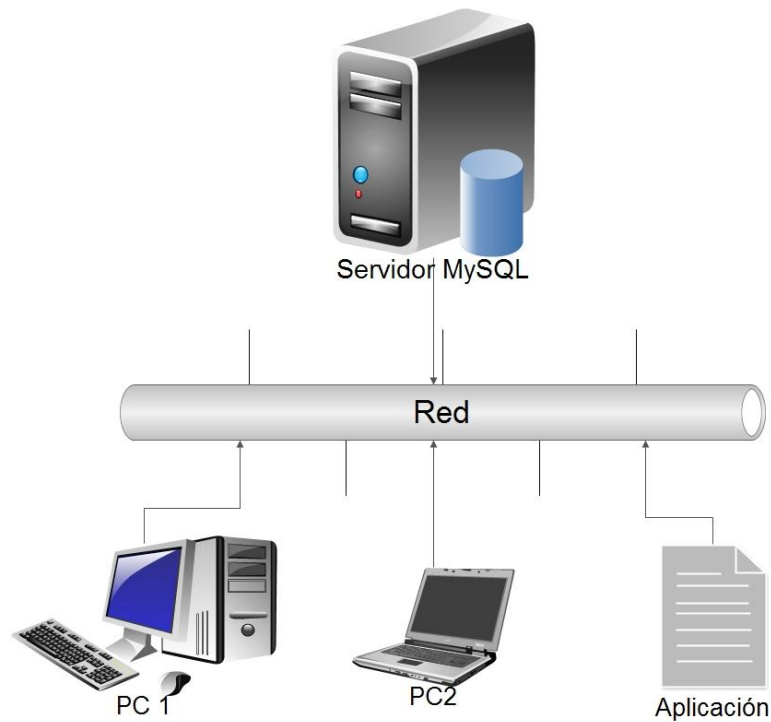


Figura IV.53: Servidor de Base de Datos MySQL.

Un servidor web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados.

Un servidor utilizado comúnmente es el denominado GlassFish que es un servidor libre para aplicaciones Java EE de código abierto. Usualmente GlassFish es uno de los primeros Servidores de aplicaciones en soportar las últimas novedades de Java EE.

4.3.7.2.ALOJAMIENTO DE LA APLICACIÓN EN EL SERVIDOR.

En esta actividad se deberá alojar la aplicación en el servidor web instalado, en nuestro caso GlassFish.

Para publicar una aplicación en servidor Glassfish se siguen los siguientes pasos:

- Primero se debe dirigir a Inicio todos los programas y en la carpeta de Glassfish le dar clic en Iniciar Servidor de Aplicaciones.

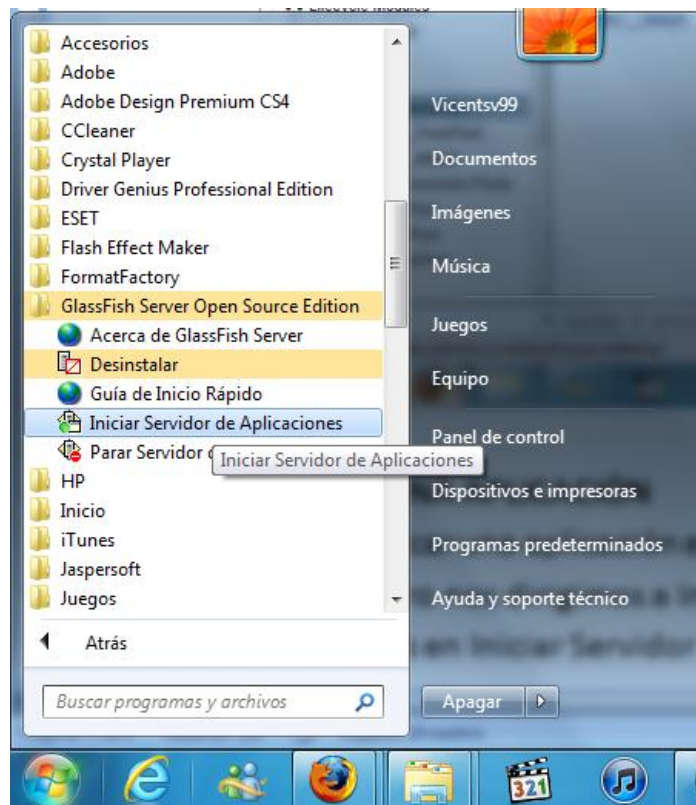


Figura IV.54: Ubicación del Servidor GlassFish en el Menú Inicio.

- Una vez iniciado el servidor dirigirse al explorador y teclear `http://localhost:4848`, y aparece la consola de configuración.

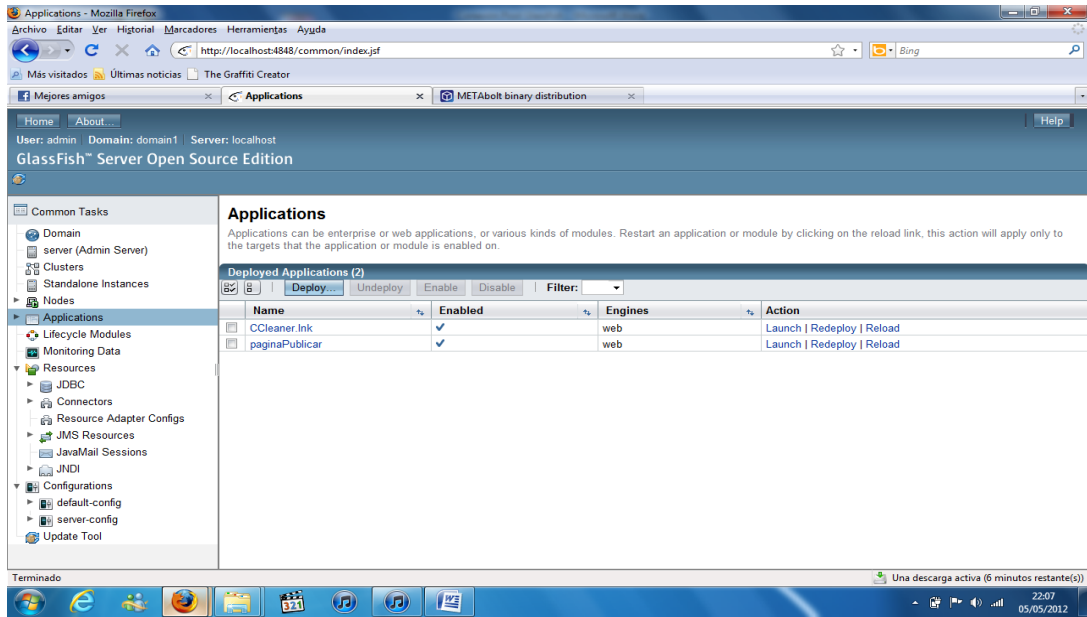


Figura IV.55: Consola GlassFish.

- Dar clic en Applications y en Deploy, para subir nuestra aplicación web.

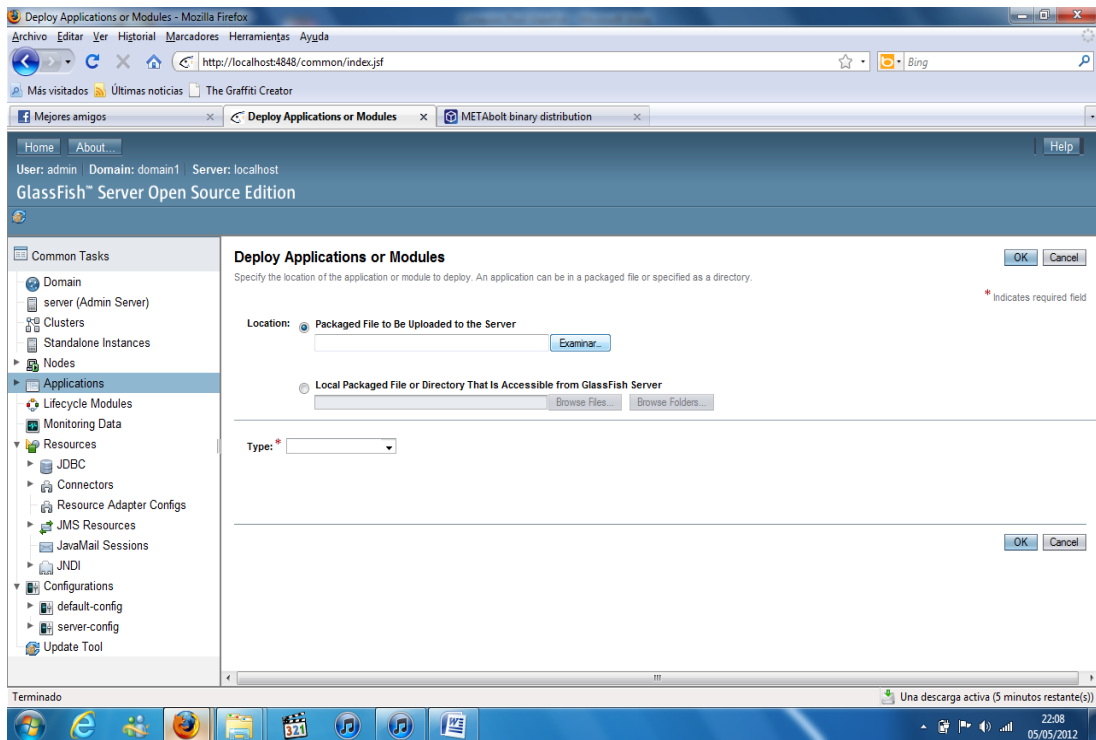


Figura IV.56: Configuración para subir Aplicaciones.

- Y escoger la aplicación que debe estar en el formato .war
- Y la aplicación aparecerá en la lista.

| Deployed Applications (3) | | | | | | |
|---------------------------|----------------------|---------------------------|--------------------------|--|-------------------------|------------------------------|
| <input type="checkbox"/> | | Deploy... | Undeploy | Enable | Disable | Filter: <input type="text"/> |
| <input type="checkbox"/> | Name | Enabled | Engines | Action | | |
| <input type="checkbox"/> | CCleaner.Ink | ✓ | web | Launch Redeploy Reload | | |
| <input type="checkbox"/> | gestionMantenimiento | ✓ | web | Launch Redeploy Reload | | |
| <input type="checkbox"/> | paginaPublicar | ✓ | web | Launch Redeploy Reload | | |

Figura IV.57: Lista de Aplicaciones subidas al servidor.

- Dar clic en Launch y aparece la aplicación



Figura IV.58: Pantalla Inicial de Software de Gestión SGM Pro

El funcionamiento de la aplicación a través de una red, estaría representado en la siguiente figura:

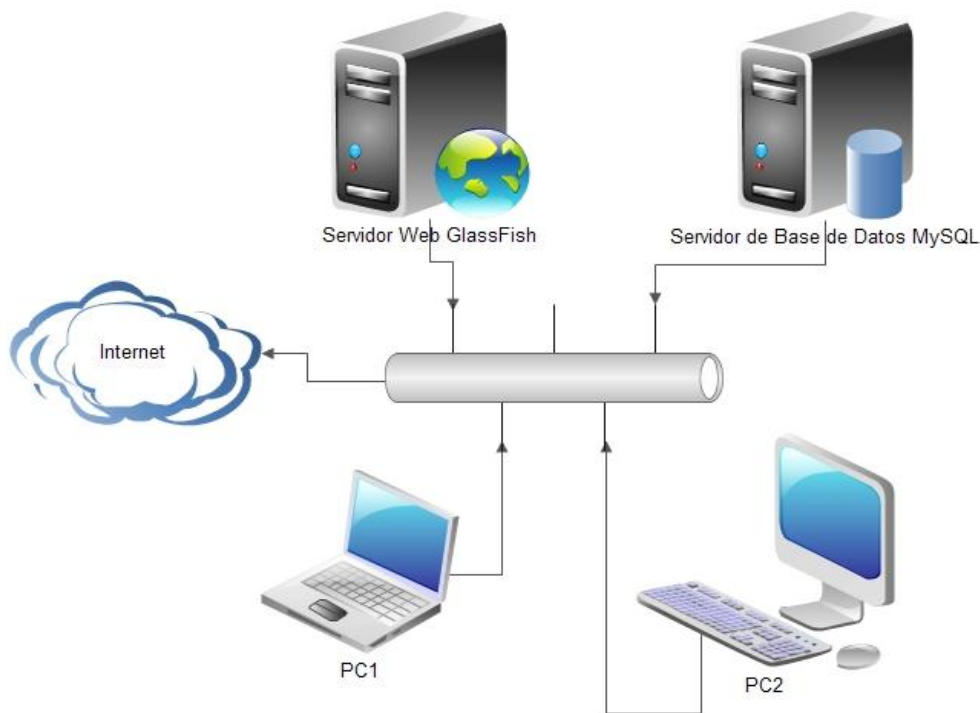


Figura IV.59: Funcionamiento de Aplicación.

4.3.7.3. PRUEBAS Y FUNCIONAMIENTO.

Esta actividad, da inicio luego de que las diferentes unidades de diseño han sido desarrolladas y probadas por separado. Durante su desarrollo, el sistema se emplea de forma experimental para asegurar que el software no falle, es decir, que funcione de acuerdo a sus especificaciones y a la manera que los usuarios esperan que lo haga, y de esta forma poder detectar cualquier anomalía, antes de que el sistema sea puesto en marcha y se dependa de él. Para evaluar el desenvolvimiento del sistema, en esta fase se llevan a cabo varios niveles de prueba:

- **Funcional:** Prueba desde el punto de vista de los requerimientos funcionales.
- **De Sistema:** Prueba desde el punto de vista de los niveles de calidad del sistema y de desempeño.
- **De Integración:** Prueba de interfaces.
- **De Aceptación Técnica:** Prueba de manejo de condiciones extremas.

Si el Sistema cumple de forma satisfactoria con estos niveles mencionados anteriormente, se procede a realizar la carga de los archivos, base de datos y tablas del

nuevo sistema, para de esta forma dar inicio al proceso de aceptación final, durante el cual, el sistema comenzará a funcionar bajo la responsabilidad del departamento de operaciones y del usuario, por un lapso determinado de tiempo llamado Periodo de Aceptación.

Finalizado el Periodo de Aceptación, se le dará al sistema la aprobación final, para que pase a ser el sistema oficial.

CAPÍTULO V.

5. APLICACIÓN DE METODOLOGÍA ERCA WEB PARA EL DESARROLLO DEL SOFTWARE DE GESTIÓN INDUSTRIAL.

Para el desarrollo del software de gestión de mantenimiento industrial para el Hospital Docente General Riobamba SGM Pro, se aplicó la metodología propuesta denominada ERCA Web, la cual consta de las siguientes fases y actividades.

5.1.PLANIFICACIÓN.

5.1.1. ANÁLISIS DE REQUERIMIENTOS.

En esta actividad se desarrolló la Especificación de Requerimientos de Software (SRS) para determinar los requerimientos de usuario que debe cumplir el software creado.

La Especificación de Requerimientos de Software SRS es una de las piezas más importantes en un proyecto de Desarrollo de Software, es necesario entonces establecer las condiciones o necesidades del usuario para resolver un problema o alcanzar un objetivo, de esta manera satisfacer un contrato a través de un documento formal.

Los requerimientos que se establecen en el presente entregable son funcionales los cuales definen las funciones que el sistema será capaz de realizar además describen las transformaciones que el sistema realiza sobre las entradas para producir salidas y los no

funcionales que tiene que ver con características que de una u otra forma pueden limitar el sistema.

Las características de un requerimiento son sus propiedades principales que debe presentar tanto individualmente como en grupo. Éstas deben satisfacer una necesidad, ser completo, verificable y no ser ambiguas.

5.1.1.1.OBJETIVOS DEL SRS.

Los objetivos del SRS son los siguientes:

- Determinar los requerimientos funcionales y no funcionales para el sistema SGM PRO.
- Describir de la forma más clara posible cada uno de los requerimientos del usuario, actividades, métodos, procesos, etc.
- Realizar algoritmos de seguimiento para el buen uso y manejo del sistema por parte de los usuarios, propuestas de interfaces, etc.
- Presentar un prototipo de las interfaces de usuario que contendrá el Sistema.

5.1.1.1.1. Ámbito.

El Sistema SGM PRO servirá de ayuda para las pequeñas y medianas empresas que realicen actividades de mantenimiento, ya que le permitirá llevar un mejor control de las actividades que se hacen por parte de sus técnicos ya que el plan de mantenimiento se llevaba de forma manual, Por tal razón este sistema aporta a la efectividad y rapidez de estos controles para un mejor control de las actividades.

5.1.1.1.2. Referencias.

La siguiente documentación de especificación de requerimiento se lo realizó:

- Software Requirements Specifications basado en el estándar 830 IEEE.
- Documentos de estatutos, acuerdos, de contratación de personal.

5.1.1.1.3. Visión general.

El documento de SRS describe los requerimientos del software a construir. Se incluyen los requerimientos funcionales y no funcionales, así como también las el posible diseño de las interfaces de SW. Su especificación existe en otros documentos, a los que se hace referencia.

5.1.1.2. DESCRIPCIÓN GENERAL

El proyecto constará de una aplicación desarrollada en el lenguaje de programación JAVA, utilizando los frameworks JSF y EJB, con el motor de Base de Datos denominado MySQL. Se utilizará el IDE NetBeans para el desarrollo de la propuesta, además del servidor web Glassfish.

5.1.1.2.1. Perspectiva del producto.

SGM PRO permitirá llevar un mejor control de las actividades que se realizan por parte de los técnicos, además de determinar un reporte de fallas permitiendo llevar una planificación anual de las actividades de acuerdo al número de semana.

5.1.1.2.2. Funciones del producto.

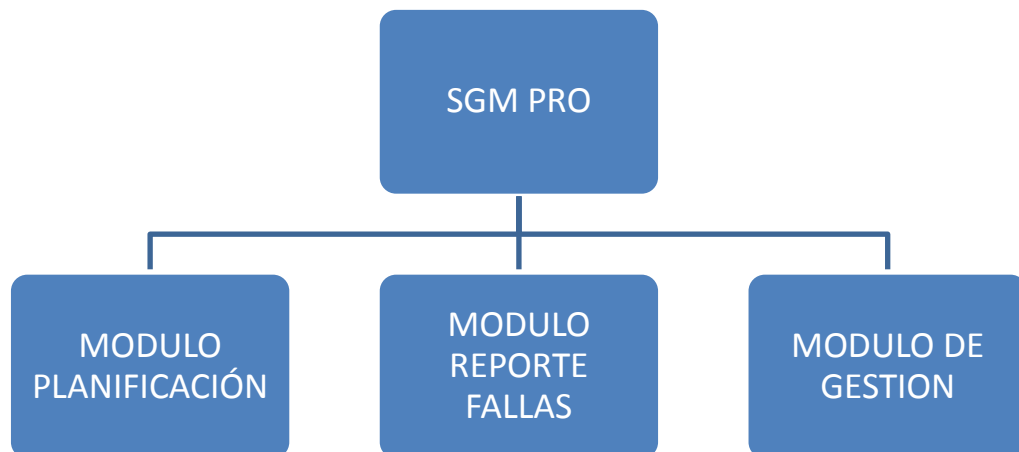


Figura V.60: Funciones de SGM Pro

Módulo de Planificación: En éste módulo se realizará la organización de las actividades que debe realizar cada técnico en la empresa de acuerdo al número

de semanas, definiendo así las actividades que se deben desarrollar en la semana correspondiente y el técnico que las debe desarrollar.

Módulo Reporte Fallas: Este módulo permite a los técnicos de la empresa reportar fallas en el desarrollo de las actividades, ya sean fallas en los equipos de mantenimiento o problemas al desarrollar alguna actividad. Se podrá establecer el tiempo de para de los equipos determinando así costos por la falla de los equipos.

Módulo de Gestión. En este módulo se realizarán todas las operaciones con respecto a la inserción, modificación y eliminación de cada una de las entidades que se manejan en el sistema.

5.1.1.2.2. Características del usuario.

Tabla V.II: Características de los Usuarios.

| FUNCIÓN | NIVEL | EXPERIENCIA | CONOCIMIENTOS TÉCNICOS |
|-------------|-----------|-------------|---|
| Programador | Superior | 1 año | <ul style="list-style-type: none">• Ing. en Sistemas• Programador en Sistemas• Analista en Sistemas |
| Estudiantes | Bachiller | | <ul style="list-style-type: none">• Conocimiento en programación básica.• Conocimientos en aplicaciones web básico |
| Gerente | Superior | | Conocimiento en computación básica. Conocimiento en programación básica |

5.1.1.2.3. Limitaciones generales.

- El sistema no podrá funcionar al máximo de su capacidad en equipos que tengan menos de 256 MB de memoria RAM.

- El sistema no funciona en máquinas que no tengan acceso a internet.
- La aplicación va a ser desarrollada utilizando el Motor de Base de Datos de MySQL, utilizando el lenguaje de programación de Java, utilizando el servidor web denominado GlassFish.
- Para el correcto funcionamiento del sistema se debe implantar en una arquitectura cliente-servidor en su totalidad.
- Necesita algún tipo de sistema operativo (Windows XP, Vista, Seven, etc;).

5.1.1.2.4. Supuestos y dependencias.

- Si la solución de la aplicación en JAVA requiere de alguna aplicación en VisualStudio se lo hará pero producirá retrasos en el desarrollo y avance.
- Los cambios en los requerimientos en algún instante de la construcción del software producirán retraso en el desarrollo e incluso puede causar una reprogramación total del producto de software.
- El cambio de algún miembro del equipo de trabajo de desarrollo podrían afectar en gran manera el desarrollo del producto.
- Si el DBMS sufre algún desperfecto este debería ser cambiado por otro de idénticas características.

5.1.1.3. *REQUERIMIENTOS ESPECIFICOS*

5.1.1.3.1. Requerimientos Funcionales.

Req. 1. Como gerente quiero gestionar los datos a través de una base de datos de modo que exista un control organizado de la información.

Req. 2. Como gerente quiero que la información sea visualizada a través de la web para que nuestros clientes puedan acceder a nuestros servicios desde cualquier parte.

Req. 3. Como un gerente quiero que se establezca la inserción, modificación y eliminación de estrategias, técnicos y equipos para llevar un control eficaz de las actividades que se realiza.

Req. 4. Como gerente quiero llevar el proceso automático de asignación de actividades a los técnicos.

Req. 5. Como gerente quiero que se muestre un plan de mantenimiento de todas las actividades que se desarrollarán durante el año de acuerdo al número de semana correspondiente.

Req. 6. Como gerente quiero que se permita a los técnicos notificar las actividades que vaya realizando de acuerdo a la semana que toque.

Req. 7. Como gerente quiero que se permite reportar las fallas de algún equipo además de poder determinar el tiempo que estuvo parado ese equipo hasta su arreglo.

Req. 8. Como gerente quiero llevar el proceso automático de asignación de actividades a los equipos.

Req. 9. Como gerente quiero que se permita establecer búsquedas de estrategias, técnicos, equipos o fallas.

Req. 10. Como gerente quiero que se presente por cada técnico, las actividades que éste debe desarrollar.

Req. 11. Como gerente quiero que se presente por cada equipo, las actividades que se deben desarrollar en él.

1. **Requerimiento 1.**

Como gerente quiero gestionar los datos a través de una base de datos de modo que exista un control organizado de la información.

1.1.Introducción.

La información que se maneje en la empresa será organizada e ingresada en una base de datos con el fin llevar un control eficiente de los datos.

1.2.Entradas.

Tabla V.III: Entradas Requerimiento 1.

| | |
|-----------------------------------|--|
| Fuente: | Base de Datos SGM PRO. |
| Cantidad: | Relacionado con las necesidades de la empresa. |
| Frecuencia: | Depende de los requerimientos del usuario. |
| Rango de entradas válidas: | Disponible todo el tiempo |

1.3.Procesos.

Tabla V.IV: Procesos Requerimiento 1.

| |
|--|
| 1.- El usuario utiliza la aplicación. |
| 2.- Realiza los requerimientos de gestión de acuerdo a la necesidad. |

1.4.Salidas.

Tabla V.V: Salidas Requerimiento 1.

| | |
|--------------------|--|
| Destino: | Pantalla |
| Cantidad: | Número de registros solicitados. |
| Frecuencia: | Relacionado con las necesidades del usuario. |

| | |
|---------------------------|--|
| Mensajes de error: | |
|---------------------------|--|

1.5. Interfaces de Usuario.

ESTRATEGIAS

Gestión

➤ Agregar Estrategia
➤ E. Realizadas
➤ E. Revisadas
★ E. de Medición

Generar Reportes

| ID ESTRATEGIA | ACTIVIDAD | DURACIÓN | FRECUENCIA | UNIDAD | TIPO | | | |
|---------------|--|----------|------------|---------|------|--------|----------|---|
| 1 | Cambio de rodamientos | 1.0 | 4 | Semanas | M | Editar | Eliminar | ⌵ |
| 2 | Cambio de valvulas | 3.0 | 6 | Semanas | M | Editar | Eliminar | ⌵ |
| 3 | Mantenimiento de inyectores electricos | 2.0 | 10 | Semanas | E | Editar | Eliminar | ⌵ |
| 4 | Cambio de aceite de motor | 1.0 | 1000 | Horas | M | Editar | Eliminar | ⌵ |

Figura V.61: Lista de Estrategias.

2. Requerimiento 2.

Como gerente quiero que la información sea visualizada a través de la web para que nuestros clientes puedan acceder a nuestros servicios desde cualquier parte.

2.1. Introducción.

Para una mejor visualización de la información de desea diseñar un sistema informativo, en el cual cualquier cliente podrá ver nuestra información pero solo podrá reservar un vuelo si se registra.

2.2. Entradas.

Tabla V.VI: Entradas Requerimiento 2.

| | |
|-----------------------------------|--|
| Fuente: | Aplicación web. |
| Cantidad: | Relacionado con las necesidades del administrador. |
| Frecuencia: | Depende de los requerimientos del usuario administrador. |
| Rango de entradas válidas: | Disponible todo el tiempo |

2.3. Procesos.

Tabla V.VII: Procesos Requerimiento 2.

| |
|---|
| 1. El usuario utiliza la aplicación. |
| 2. Digita www.sgmpro.com . |
| 3. Se presenta la página web. |
| 4. El usuario podrá obtener todo tipo de información que la página dispone. |

2.4. Salidas.

Tabla V.VIII: Salidas Requerimiento 2.

| | |
|---------------------------|--|
| Destino: | Pantalla |
| Cantidad: | Número de registros solicitados. |
| Frecuencia: | Relacionado con las necesidades del usuario. |
| Mensajes de error: | |

2.5. Interfaces de Usuario.



Figura V.62: Aplicación SGM Pro.

3. **Requerimiento 3.**

Como un gerente quiero que se establezca la inserción, modificación y eliminación de estrategias, técnicos y equipos para llevar un control eficaz de las actividades que se realiza.

3.1. **Introducción.**

Este requerimiento tiene como finalidad permitir los métodos de inserción, modificación y eliminación de cada una de las entidades que en el sistema se manejan.

3.2. **Entradas.**

Tabla V.IX: Entradas Requerimiento 3.

| | |
|-----------------------------------|------------------------------|
| Fuente: | Base de Datos SGM PRO |
| Cantidad: | Uno por usuario. |
| Frecuencia: | Demanda alta. |
| Rango de entradas válidas: | Disponible todo el tiempo |

3.3. **Procesos.**

Tabla V.X: Procesos Requerimiento 3.

| |
|--|
| 1.- Seleccionar la entidad a gestionar. |
| 2.- Seleccionar lo que desea realizar en esta entidad, ya sea agregar un nuevo registro, modificarlo o eliminarlo. |
| 3.- Si selecciona agregar, insertar la entidad. |
| 4.- Si selecciona modificar, editar la entidad. |
| 5.- Si selecciona eliminar, eliminar la entidad. |

3.4. **Salidas.**

Tabla V.XI: Salidas Requerimiento 3.

| | |
|------------------|--------------------|
| Destino: | Pantalla |
| Cantidad: | Número de usuarios |

| | |
|---------------------------|----------------------------|
| | que solicitan un servicio |
| Frecuencia: | Número de acceso a la red. |
| Mensajes de error: | Datos mal ingresados |

3.5. Interfaces de Usuario.

Ingrese una nueva Estrategia

Los campos con asterisco (*) son campos requeridos.

Actividad * :

Duración (Horas) * :

Frecuencia * :

Unidad:

Tipo:

Figura V.63: Insertar Estrategias

Editar Estrategia

Los campos con asterisco (*) son campos requeridos.

IdEstrategia:

Actividad * :

Duración * :

Frecuencia * :

Unidad:

Tipo:

Figura V.64: Editar Estrategias

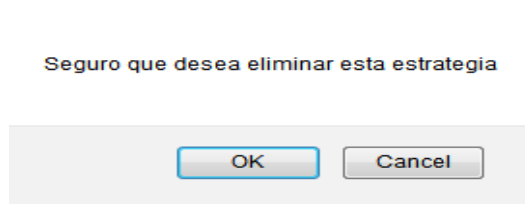


Figura V.65: Confirmación Eliminar Estrategia.

4. Requerimiento 4.

Como gerente quiero llevar el proceso automático de asignación de actividades a los técnicos.

4.1.Introducción.

Cada actividad que se desarrolla en la empresa debe asignarse a un técnico para que la complete por lo cual es lo que permite este requerimiento.

4.2.Entradas.

Tabla V.XII: Entradas Requerimiento 4.

| | |
|-----------------------------------|---------------------------|
| Fuente: | Base de Datos SGM PRO. |
| Cantidad: | Una por usuario. |
| Frecuencia: | Posee una demanda alta. |
| Rango de entradas válidas: | Disponible todo el tiempo |

4.3.Procesos.

Tabla V.XIII: Procesos Requerimiento 4.

| |
|--|
| 1.- El usuario accede a la aplicación. |
| 2.- Visualiza el plan de mantenimiento. |
| 3.- Verifica las actividades a realizar en la semana. |
| 4.- Asigna las actividades de esa semana a un técnico determinado. |

4.4.Salidas.

Tabla V.XIV: Salidas Requerimiento 4.

| | |
|---------------------------|----------------------------------|
| Destino: | Pantalla |
| Cantidad: | Uno por usuario |
| Frecuencia: | Número de acceso a la aplicación |
| Mensajes de error: | |

4.5.Interfaces de Usuario



Figura V.66: Asignar Actividades

5. Requerimiento 5.

Como gerente quiero que se muestre un plan de mantenimiento de todas las actividades que se desarrollarán durante el año de acuerdo al número de semana correspondiente.

5.1.Introducción.

El usuario necesita conocer las actividades que se deberán desarrollar durante el año por lo cual debe visualizar el plan de mantenimiento anual.

5.2.Entradas.

Tabla V.XV: Entradas Requerimiento 5.

| | |
|-----------------------------------|-----------------------------|
| Fuente: | Base de Datos SGM Pro. |
| Cantidad: | Una a la vez. |
| Frecuencia: | Demanda alta. |
| Rango de entradas válidas: | Disponibile todo el tiempo. |

5.3.Procesos.

Tabla V.XVI: Procesos Requerimiento 5.

| |
|---|
| 1.- El usuario accede a la aplicación. |
| 2.- Selecciona la opción Actividades en Reportes. |
| 3.- Selecciona el año en el que desea que se visualice el plan. |
| 4.- Visualiza el plan de mantenimiento. |

5.4.Salidas.

Tabla V.XVII: Salidas Requerimiento 5.

| | |
|---------------------------|----------------------------------|
| Destino: | Pantalla |
| Cantidad: | Una por usuario. |
| Frecuencia: | Número de acceso a la red. |
| Mensajes de error: | Usuario o password mal ingresado |

5.5. Interfaces de Usuario.



Figura V.67: Historial Plan de Mantenimiento.

PLAN MANTENIMIENTO AÑO 2012

OPCIONES

| ESTRATEGIA ▾ | --- EQUIPO --- ▾ | FRECUENCIA | INICIO SEMANA | -1- | -2- | -3- | -4- | -5- | -6- | -7- | -8- | -9- | -10- | -11- | -12- | -13- | -14- | -15- | -16- | -17- |
|-----------------------|------------------|------------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|
| Cambio de rodamientos | Lavadora - LAV1 | 4 | 2 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| TOTAL HORAS: | | | | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |

RESULTADOS

| | |
|-------------------------------|---------------------|
| TOTAL DE HORAS EN EL AÑO : | 13.0 horas |
| PROMEDIO DE HORAS POR SEMANA: | 0.25 horas / Semana |

Figura V.68: Plan de Mantenimiento.

6. **Requerimiento 6.**

Como gerente quiero que se permita a los técnicos notificar las actividades que vaya realizando de acuerdo a la semana que toque.

6.1. **Introducción.**

El técnico puede notificar de algún inconveniente al momento de completar la actividad.

6.2. **Entradas.**

Tabla V.XVIII: Entradas Requerimiento 6.

| | |
|------------------|-----------------------|
| Fuente: | Base de Datos SGM PRO |
| Cantidad: | Una a la vez. |

| | |
|-----------------------------------|-----------------|
| Frecuencia: | Demanda media. |
| Rango de entradas válidas: | Todo el tiempo. |

6.3. Procesos.

Tabla V.XIX: Procesos Requerimiento 6.

| |
|---|
| 1.- El usuario visualiza las actividades que ya han sido asignadas. |
| 2.- Selecciona la actividad que desea notificar. |
| 3.- Notifica la actividad si hubo inconvenientes en la realización de ésta. |
| 4.- Si Completa la actividad selecciona en Actividad Realizada. |

6.4. Salidas.

Tabla V.XX: Salidas Requerimiento 6.

| | |
|---------------------------|----------------------------|
| Destino: | Pantalla |
| Cantidad: | Una a la vez. |
| Frecuencia: | Número de acceso a la red. |
| Mensajes de error: | |

6.5. Interfaces de Usuario.

NOTIFICAR ACTIVIDAD

Notifique la Actividad

Tecnico: Alex Erazo Vivero
Actividad: Cambio de rodamientos
Equipo: Lavadora
Novedad:
Tiempo Real:
Fecha:

Figura V.69: Notificar Actividad.

7. Requerimiento 7.

Como gerente quiero que se permita reportar las fallas de algún equipo además de poder determinar el tiempo que estuvo parado ese equipo hasta su arreglo.

7.1.Introducción.

Los equipos de mantenimiento en la empresa sufren diversas fallas, por lo cual es necesario reportar esas fallas para su posterior arreglo y además poder determinar el tiempo de para que tuvo el equipo.

7.2.Entradas.

Tabla V.XXI: Entradas Requerimiento 7.

| | |
|-----------------------------------|--|
| Fuente: | Base de Datos SGM PRO |
| Cantidad: | Una a la vez. |
| Frecuencia: | Alta demanda. |
| Rango de entradas válidas: | Las entradas se realizan a cualquier hora del día. |

7.3.Procesos.

Tabla V.XXII: Procesos Requerimiento 7.

| |
|--|
| 1.- El usuario accede a la aplicación. |
| 2.- Selecciona la opción Reporte Fallas. |
| 3.- Agrega un nuevo reporte de fallas. |

7.4.Salidas.

Tabla V.XXIII: Salidas Requerimiento 7.

| | |
|--------------------------|----------------------|
| Destino: | Pantalla |
| Cantidad: | Una por usuario |
| Frecuencia: | Demanda Media – Alta |
| Mensaje de Error: | Ninguno |

7.5. Interfaces de Usuario.

NUEVA FALLA

Reportar nueva Falla

Novedad:

Fecha Inicial:

Hora Inicial:

Fecha Final:

Hora Final:

Tiempo Falla: 0.0 Horas

Equipos:

Figura V.70: Ingresar Fallas.

8. Requerimiento 8

Como gerente quiero llevar el proceso automático de asignación de actividades a los equipos.

8.1. Introducción.

Cada actividad se debe asignar a un equipo determinado con el fin de que los técnicos desarrollen estas actividades.

8.2. Entradas.

Tabla V.XXIV: Entradas Requerimiento 8.

| | |
|-----------------------------------|-----------------------|
| Fuente: | Base de Datos SGM PRO |
| Cantidad: | Una por usuario |
| Frecuencia: | Demanda Media. |
| Rango de entradas válidas: | Todo el día |

8.3. Proceso.

Tabla V.XXV: Procesos Requerimiento 8.

| |
|--|
| 1.- El usuario selecciona el equipo. |
| 2.- Selecciona la actividad que se va a asignar. |
| 3.- Se asigna la actividad al equipo. |

8.4. Salida.

Tabla V.XXVI: Salidas Requerimiento 8.

| | |
|--------------------------|----------------------|
| Destino: | Pantalla |
| Cantidad: | Demanda Media – Alta |
| Frecuencia: | Media |
| Mensaje de Error: | |

8.5. Interfaces de Usuario.

ASIGNACIÓN DE ACTIVIDADES A EQUIPO - 2012



Plan:

Fecha Asignación:

Equipo: Lavadora - LAV1

Actividad:

Figura V.71: Asignar Estrategias a Equipos.

9. Requerimiento 9.

Como gerente quiero que se permita establecer búsquedas de estrategias, técnicos, equipos o fallas.

9.1. Introducción.

En determinados momentos el usuario necesitará realizar búsquedas específicas de las entidades para acceder más rápido a registros determinados.

9.2.Entradas.

Tabla V.XXVII: Entradas Requerimiento 9.

| | |
|-----------------------------------|-----------------------|
| Fuente: | Base de Datos SGM PRO |
| Cantidad: | Una por usuario. |
| Frecuencia: | Demanda Alta. |
| Rango de entradas válidas: | Todo el día |

9.3.Procesos.

Tabla V.XXVIII: Procesos Requerimiento 9.

| |
|--|
| 1.- El usuario selecciona la entidad. |
| 2.-Ingresa los parámetros de búsqueda. |
| 3.- Busca el o los registros de acuerdo a los parámetros pre-establecidos. |

9.4.Salidas.

Tabla V.XXIX. Salidas Requerimiento 9.

| | |
|--------------------------|---------------------------------------|
| Destino: | Pantalla |
| Cantidad: | Una por usuario |
| Frecuencia: | Demanda Alta |
| Mensaje de Error: | Parámetro establecido no es correcto. |

9.5. Interfaces de Usuario.

| ID TECNICO | TÉCNICO | ESTRATEGIA | EQUIPO | ESTADO | NOVEDAD | FECHA | TIEMPO REAL |
|------------|-------------------|-----------------------|-----------------|--------|---------|-------|-------------|
| 1 | Alex Erazo Vivero | Cambio de rodamientos | LAV1 - Lavadora | Activa | | | 00:00:00 |

Figura V.72: Buscar Estrategia Asignada a Tecnico

10. Requerimiento 10.

Como gerente quiero que se presente por cada técnico, las actividades que éste debe desarrollar.

10.1. *Introducción.*

Se debe presentar a cada técnico las actividades que éste debe desarrollar para su posterior impresión.

10.2. *Entradas.*

Tabla V.XXX: Entradas Requerimiento 10.

| | |
|-----------------------------------|-----------------------|
| Fuente: | Base de Datos SGM PRO |
| Cantidad: | |
| Frecuencia: | Demanda Alta. |
| Rango de entradas válidas: | Todo el día |

10.3. *Procesos.*

Tabla V.XXXI: Procesos Requerimiento 10.

| |
|---|
| 1.- Selecciona el técnico. |
| 2.- Revisa sus actividades. |
| 3.- Verifica las actividades que ha realizado y las que se deben desarrollar. |

10.4. Salidas.

Tabla V.XXXII: Salidas Requerimiento 10.

| | |
|--------------------------|----------------------------|
| Destino: | Pantalla |
| Cantidad: | Demanda Media |
| Frecuencia: | Dependiendo de la consulta |
| Mensaje de Error: | |

10.5. Interfaces de Usuario.

LISTA DE TECNICOS

| ID TECNICO | CEDULA | NOMBRE | APELLIDO | |
|------------|------------|---------|-----------------|-----------------|
| 1 | 0803110451 | Alex | Erazo Vivero | Ver Actividades |
| 2 | 0803880756 | Jessica | Sanchez Verduga | Ver Actividades |

Figura V.73: Lista de Tecnicos en la Industria.

LISTA DE ACTIVIDADES

Gestión
Aceptar

| ACTIVIDAD | EQUIPO | FRECUENCIA | DURACION |
|-----------------------|-----------------|------------|----------|
| Cambio de rodamientos | LAV1 - Lavadora | 4 Semanas | 1.0 |

Figura V.74: Actividades de Tecnico determinado.

11. Requerimiento 11.

Como gerente quiero que se presente por cada equipo, las actividades que se deben desarrollar en él.

11.1. Introducción.

Se debe presentar por cada equipo las actividades que se deben desarrollar en éste.

11.2. Entradas.

Tabla V.XXXIII: Entradas Requerimiento 11.

| | |
|-----------------------------------|--------------------------|
| Fuente: | Base de Datos SGM PRO |
| Cantidad: | |
| Frecuencia: | Demanda Media. |
| Rango de entradas válidas: | Dependiendo la consulta. |

11.3. Procesos.

Tabla V.XXXIV: Procesos Requerimiento 11.

| |
|--|
| 1.- Selecciona el equipo. |
| 2.- Revisa sus actividades. |
| 3.- Verifica las actividades que se deben realizar en la semana correspondiente. |

11.4. Salidas.

Tabla V.XXXV: Salidas Requerimiento 11.

| | |
|--------------------------|----------------------------|
| Destino: | Pantalla |
| Cantidad: | Demanda Media |
| Frecuencia: | Dependiendo de la consulta |
| Mensaje de Error: | |

11.5. Interfaces de Usuario.



Figura V.75: Actividades de Tecnico.

5.1.1.3.2. Requerimientos de Interfaces Externas

5.1.1.3.2.1. Interfaces Hardware

- Access Point.
- Conectores RJ45.
- Computadora.
- Impresora.

5.1.1.3.2.2. Interfaces Software.

- SO (XP, VISTA, SEVEN, etc.).
- Servidor de BD MySQL.
- Servidor GlassFish

5.1.1.3.2.3. Interfaces de Comunicación

- TCP/IP.

5.1.1.3.3. Requerimientos de Rendimiento.

5.1.1.3.3.1. Número de Usuarios simultáneos:

Existen múltiples usuarios que van a manejar el sistema.

5.1.1.3.3.2. Estaciones de trabajo:

Existen muchas estaciones de trabajo.

5.1.1.3.3.3. Número de tablas y registros a manejar:

- Existen 12 tablas.
- Registros Rango (10-100).

5.1.1.3.4. Limitaciones de Diseño.

5.1.1.3.4.1. Obediencia a Estándares

- SRS IEEE830.
- ISO 9300.

5.1.1.3.4.2. Atributos.

Seguridad

El sistema proporcionará la integridad y la seguridad de toda la información que se maneje debido a la manera de implementación que se utilizará. Siendo el sistema eficaz, confiable y muy seguro.

Mantenibilidad.

El sistema debido a su implementación será fácil de darle mantenimiento, al mismo tiempo su mantenimiento será económico.

Escalabilidad.

El sistema es flexible ya que se pueden incrementar nuevas funciones de acuerdo a los requerimientos que puedan presentarse.

Interfaz amigable.

El sistema cuenta con una interfaz de fácil manejo para el usuario final.

Disponibilidad.

El sistema deberá estar disponible durante todo el día.

5.1.1.3.5. Otros requerimientos

5.1.1.3.5.1. *Adaptación al Sitio.*

Se requieren adaptaciones a nivel de infraestructura física, espacio y cambios necesarios para la adecuación de la red.

5.1.2. BOSQUEJO DE BASE DE DATOS.

En esta etapa se va a definir un bosquejo de la base de datos de la aplicación que se va a desarrollar. Para el desarrollo del sistema SGM Pro se ha desarrollado un bosquejo de la base de datos:

TABLAS Y ATRIBUTOS:

- Estrategias.
 - Id_Estrategia.
 - Descripción.
 - Duración.
 - Frecuencia.
- Tecnicos.
 - Id_Tecnico.
 - Cedula.
 - Nombre.
 - Apellidos.
 - Direccion.
 - Telefono.
- Equipos.
 - Id_equipo.
 - Codigo.
 - Descripcion.
 - Marca.
 - Modelo.
- Ubicación Técnica.
 - Id_ubicacion.
 - Codigo.

- Descripción.
- Componentes.
 - Id_componente.
 - Descripción.
- Fallas.
 - Id_Falla.
 - Descripción.
 - Fecha.
- Repuestos.
 - Id_Repuesto.
 - Descripción.
 - Stock.
 - Costo.

RELACIONES:

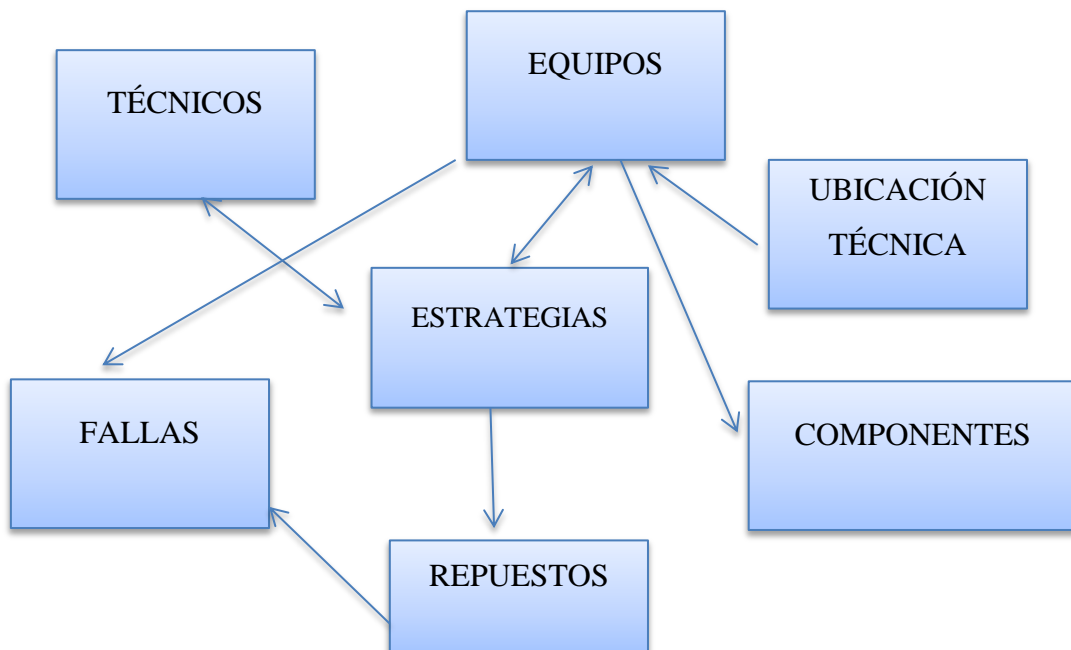


Figura V.76: Bosquejo de Base de Datos.

5.1.3. APRENDIZAJE DE FRAMEWORK JSF.

5.2. DESARROLLO DE BASE DE DATOS.

5.2.1. DISEÑO DE BASE DE DATOS.

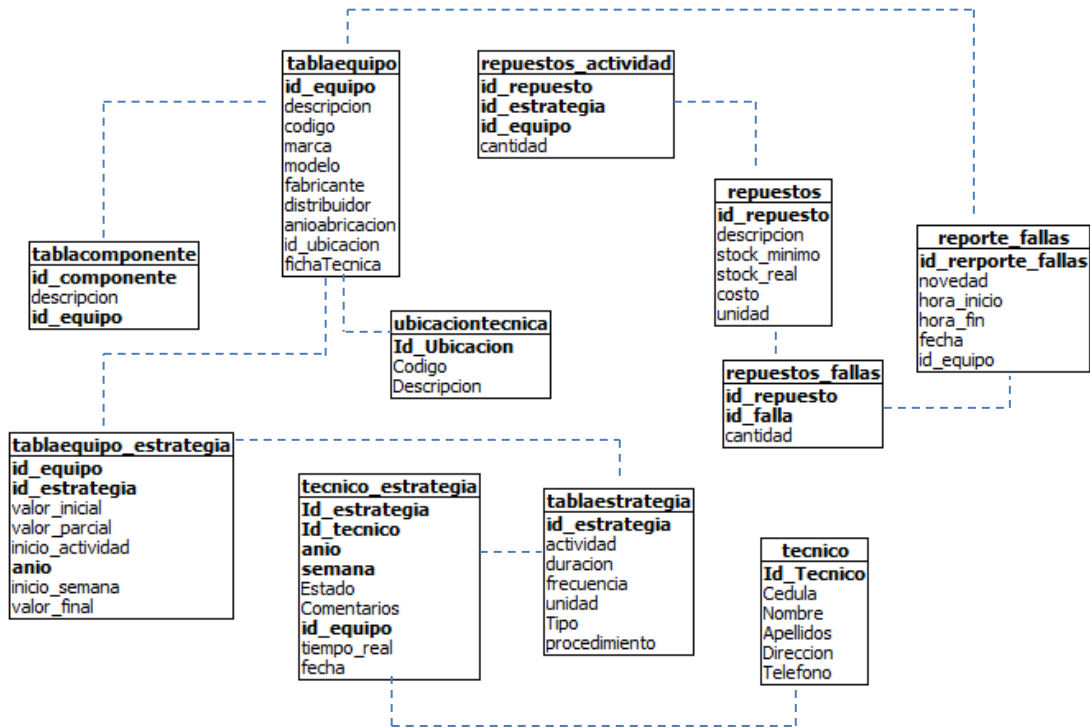


Figura V.77: Diseño de Base de Datos.

5.2.2. CONSTRUCCIÓN DE PROTOTIPO EN JSF.

List

1.4/4

| IdEstrategia | Actividad | Duracion | Frecuencia | Unidad | Tipo | Procedimiento | |
|--------------|--|----------|------------|---------|------|---------------|---|
| 1 | Cambio de rodamientos | 1.0 | 4 | Semanas | M | | View Edit Destroy |
| 2 | Cambio de valvulas | 3.0 | 6 | Semanas | M | | View Edit Destroy |
| 3 | Mantenimiento de inyectores electricos | 2.0 | 10 | Semanas | E | | View Edit Destroy |
| 4 | Cambio de aceite de motor | 1.0 | 1000 | Horas | M | | View Edit Destroy |

[Create New Tablaestrategia](#)

[Index](#)

Figura V.78: Prototipo en JSF de Lista de Estrategias.

5.2.3. PRESENTACIÓN DE PROTOTIPO.

Durante esta presentación se establecieron algunos cambios por parte del cliente en cuanto a la base de datos, es decir, se hizo la inclusión de una nueva tabla denominada

- *cambiarTecnicosPlan1.*
- *Cerrar_actividad.*
- *Comprar_repuesto.*
- *GenerarPlanMantenimiento.*

5.3. DESARROLLO DE MÓDULO EJB.

5.3.1. APRENDIZAJE DE FRAMEWORK EJB.

5.3.2. CREACIÓN DEL MÓDULO EJB.

Para crear el módulo EJB, primeramente se creó una aplicación de tipo empresarial en el IDE NetBeans.

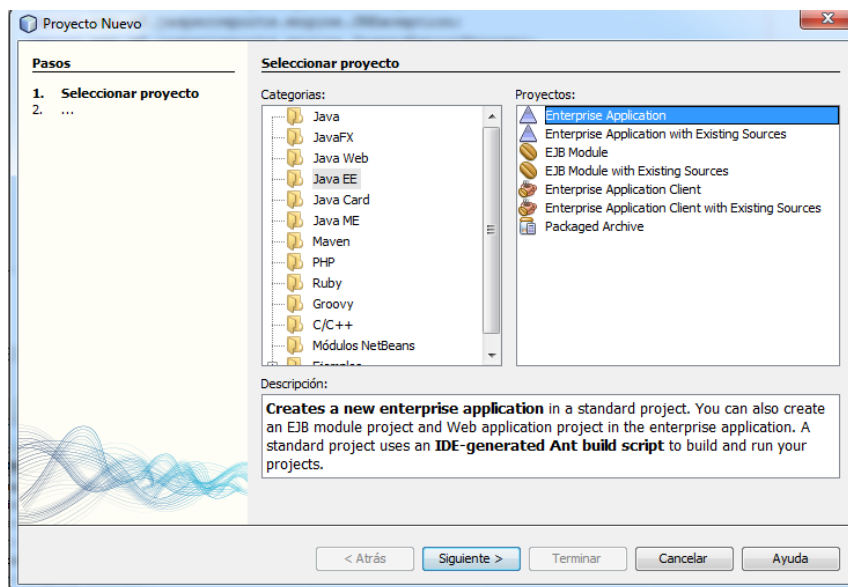


Figura V.80: Creando Aplicación Empresarial.

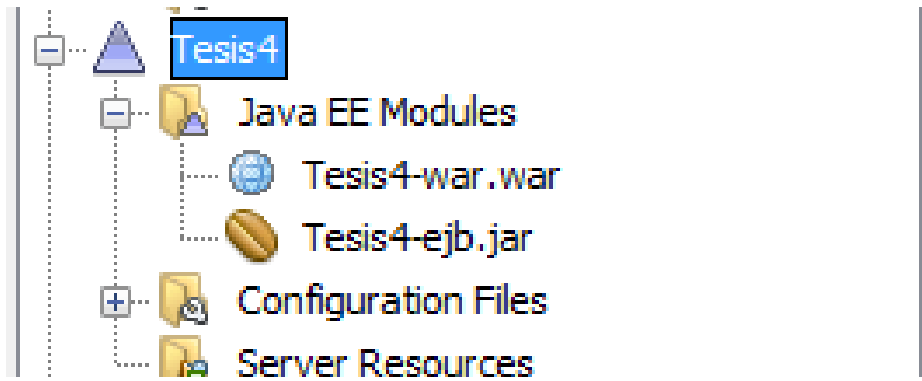


Figura V.81: Aplicación Empresarial.

A partir de esta se crean dos módulos: uno tipo web y el otro, tipo EJB.

5.3.3. GENERACIÓN DE LA CAPA DE ACCESO A DATOS Y LÓGICA DE NEGOCIO INICIAL.

El acceso a datos en EJB está definido por las clases entidades, sin embargo, la lógica de negocio es definida por los beans de sesión.

5.3.3.1. ACCESO A DATOS.

Para realizar la capa de acceso a datos primero se creó las clases entidad a partir de la base de datos:

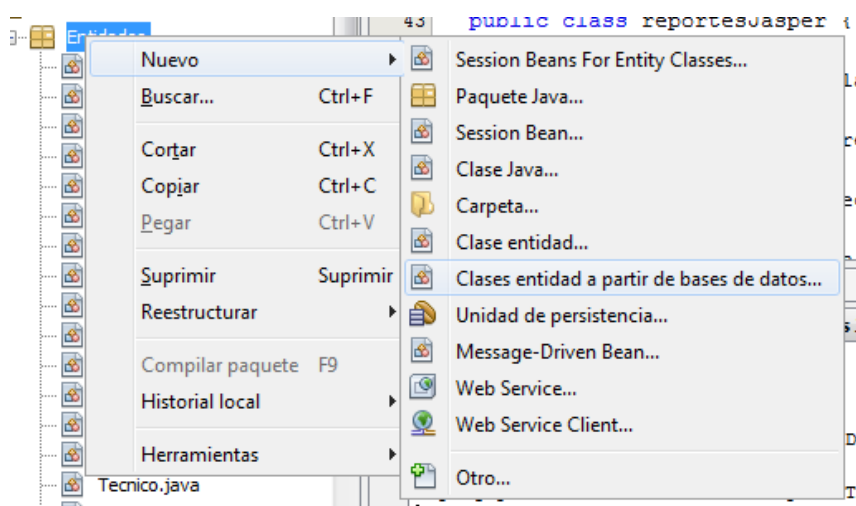


Figura V.82: Clases entidad a partir de base de Datos.

Para esto se deberá establecer la conexión a nuestra base de datos en el siguiente cuadro de diálogo y seleccionar las tablas de las cuales se van a tomar los datos.

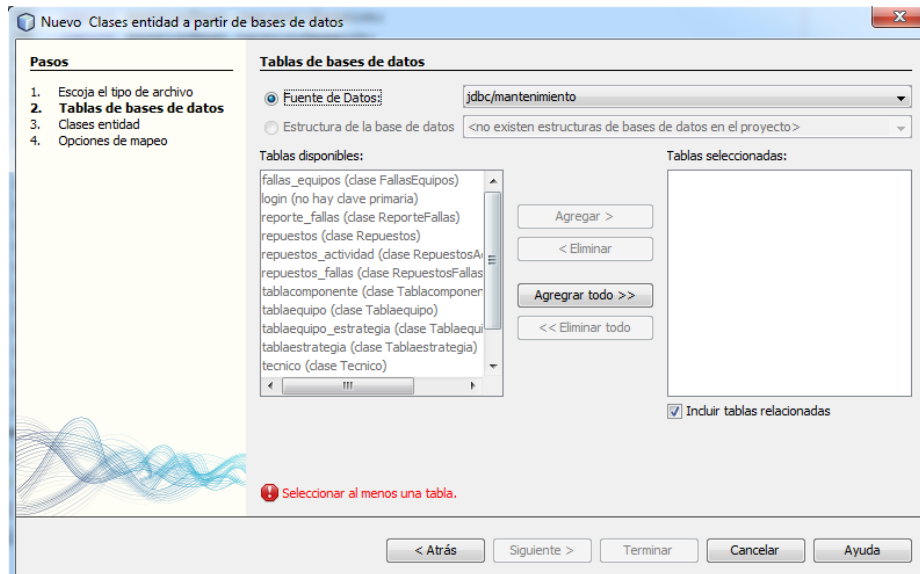


Figura V.83: Selección de Tablas para el acceso a datos.

Una vez realizado esto se genera la capa de acceso a datos con las siguientes clases:

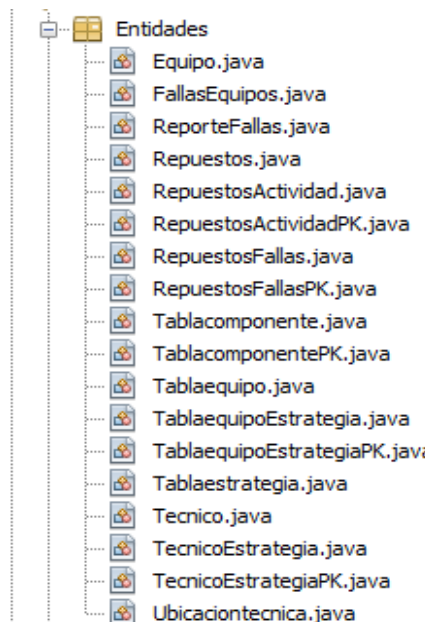


Figura V.84: Capa de Acceso a Datos.

El código que se genera en cada clase es parecido al siguiente:

```
@Entity
@Table(name = "equipo")
@NamedQueries({
    @NamedQuery(name = "Equipo.findAll", query = "SELECT e FROM Equipo e"),
    @NamedQuery(name = "Equipo.findById", query = "SELECT e FROM Equipo e WHERE e.id = :id"),
    @NamedQuery(name = "Equipo.findByName", query = "SELECT e FROM Equipo e WHERE e.nombre = :nombre"),
    @NamedQuery(name = "Equipo.findByCodigo", query = "SELECT e FROM Equipo e WHERE e.codigo = :codigo"),
    @NamedQuery(name = "Equipo.findByUbicacion tecnicaId", query = "SELECT e FROM Equipo e WHERE e.ubicacion tecnicaId = :ubicacion tecnicaId")
})
public class Equipo implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "ID")
    private Long id;
    @Column(name = "NOMBRE")
    private String nombre;
    @Column(name = "CODIGO")
    private String codigo;
    @Column(name = "UBICACION TECNICA_ID")
    private BigInteger ubicacion tecnicaId;
}
```

Figura V.85: Código Generado por la capa de Acceso a Datos.

5.3.3.2.LÓGICA DE NEGOCIO.

Aquí se genera un parte de la lógica de negocio. Para generarla se debe crear los Session Bean a partir de las clases entidad creadas anteriormente:

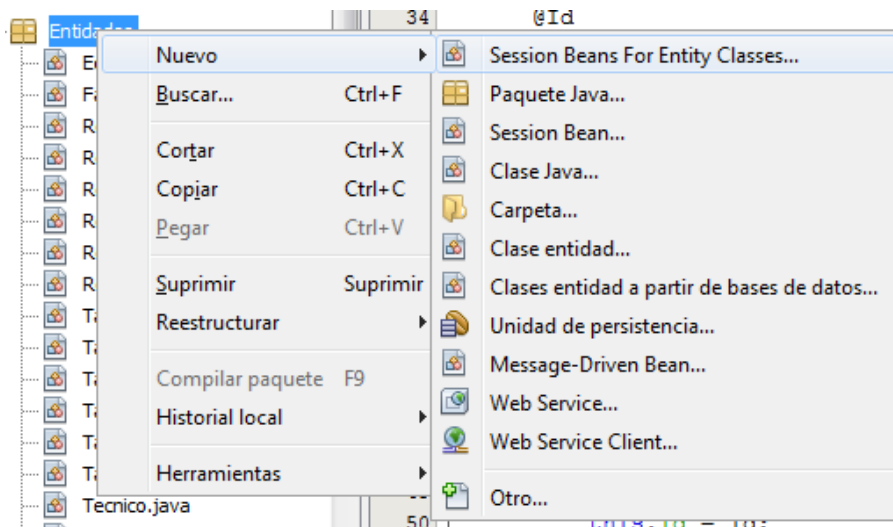


Figura V.86: Creando los Session Bean

Luego se seleccionan las clases entidad creadas y se deja que el modulo EJB genere las clases para la lógica de negocio de estas entidades.

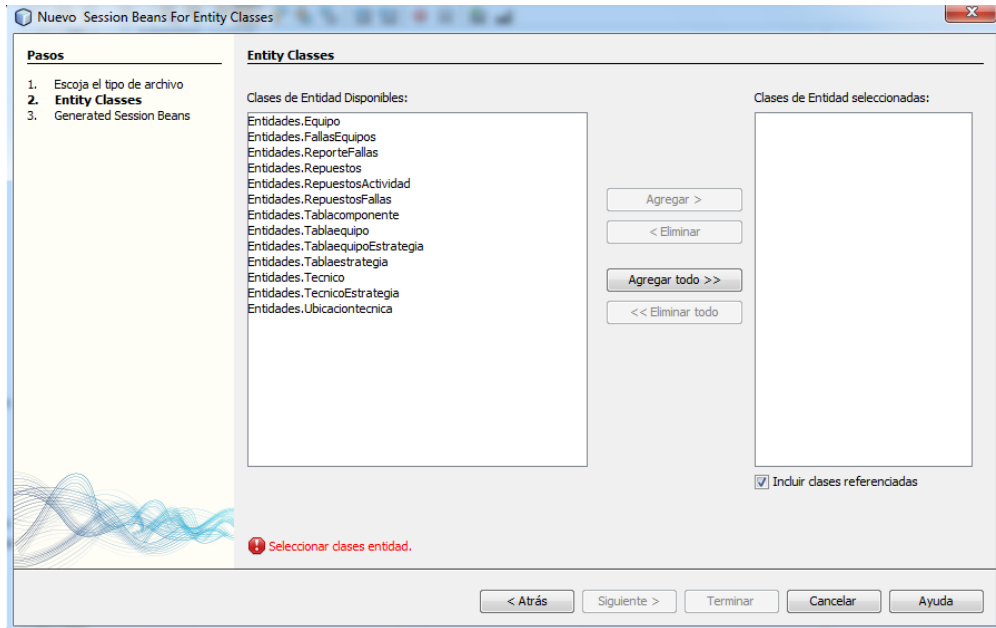


Figura V.87: Seleccionando Clases Entidad

Una vez hecho esto se genera las siguientes clases:

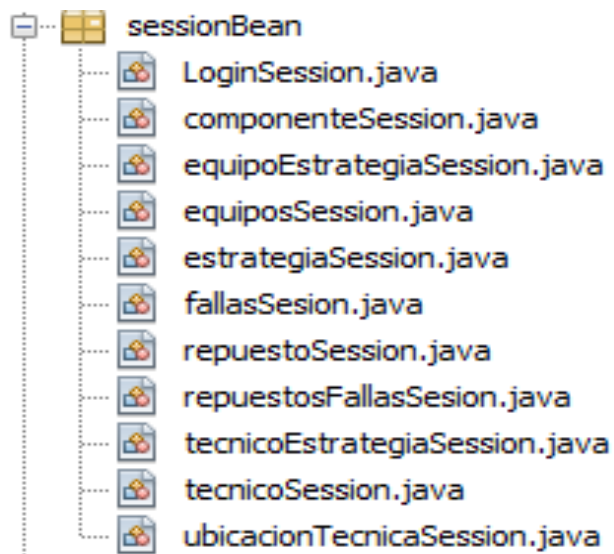


Figura V.88: Lógica de Negocio

5.4. INTEGRACIÓN DE FRAMEWORKS.

Ahora en esta fase se va a integrar los frameworks JSF y EJB, siguiendo estas actividades:

5.4.1. CREACIÓN DE MÓDULO JSF.

Este módulo se crea en el momento de crear la aplicación empresarial. En las actividades anteriores se creó una aplicación empresarial, esta aplicación creó dos módulos a su vez, que era el módulo EJB y el módulo de tipo web.

Para que el módulo de tipo web sea utilizado como un JSF simplemente se debe seleccionar en este módulo el framework JSF para desarrollo, haciendo clic derecho en el módulo web, luego en propiedades, apareciendo este cuadro de diálogo:

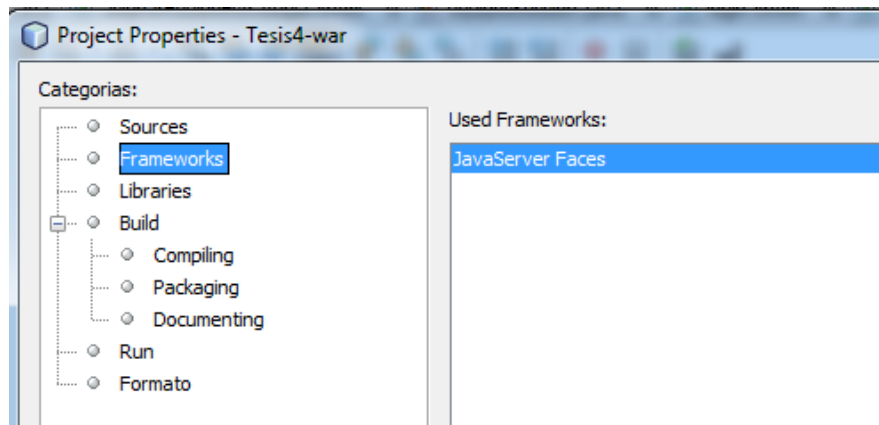


Figura V.89: Creación de módulo JSF.

En este cuadro de diálogo se selecciona el framework JavaServer Faces, creando de esta manera el módulo JSF.

5.4.2. INTEGRACIÓN DE JSF Y EJB.

Para integrar EJB a JSF, se debe agregar el módulo EJB al módulo JSF. Para esto, se añade a las librerías del proyecto en JSF, el módulo EJB, así como se muestra a continuación. De esta manera se integran ambos módulos, haciendo uso de los beneficios que otorgan ambos frameworks, desarrollando de esta manera una aplicación más robusta.

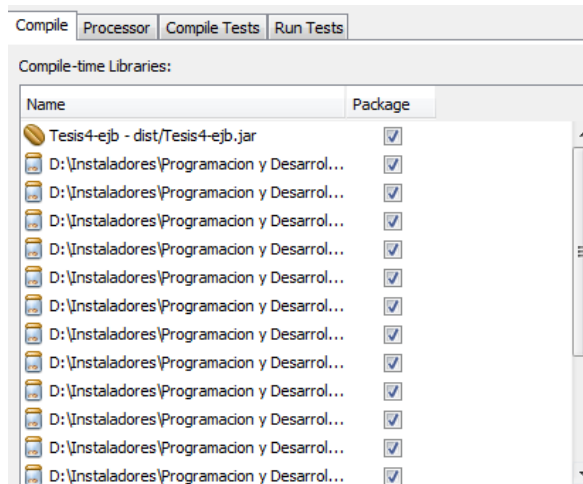


Figura V.90: EJB Integrado a JSF.

5.4.3. DESARROLLO FINAL DE LA LÓGICA DE NEGOCIO.

En esta actividad se desarrolló toda la lógica de negocio de la aplicación.

5.5. CONSTRUCCIÓN FINAL DE LA APLICACIÓN.

5.5.1. PREPARACIÓN DE TEMPLATES.

Los templates que se utilizaron son los siguientes:

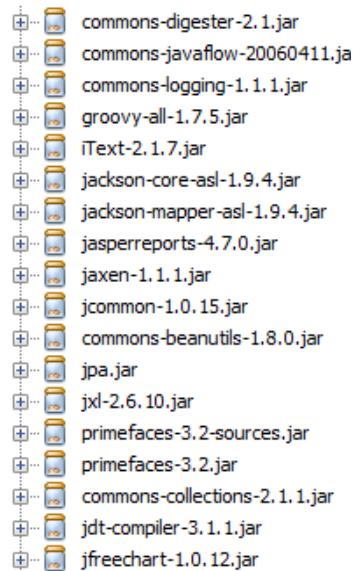


Figura V.91: Templates Utilizados.

5.5.2. DESARROLLO FINAL DE INTERFACES.

En esta actividad se desarrollaron 79 páginas aproximadamente, cada interfaz tiene un aspecto similar al siguiente:



Figura V.92: Interfaz Final.

5.5.2. PRESENTACIÓN FINAL.

5.5.3. SUGERENCIAS Y MEJORAMIENTO.

5.6. IMPLEMENTACIÓN E IMPLANTACIÓN.

5.6.1. INSTALACIÓN DE SERVIDORES.

El servidor de base de datos a utilizar es MySQL, para instalarlo, se hace uso de la herramienta XAMPP:

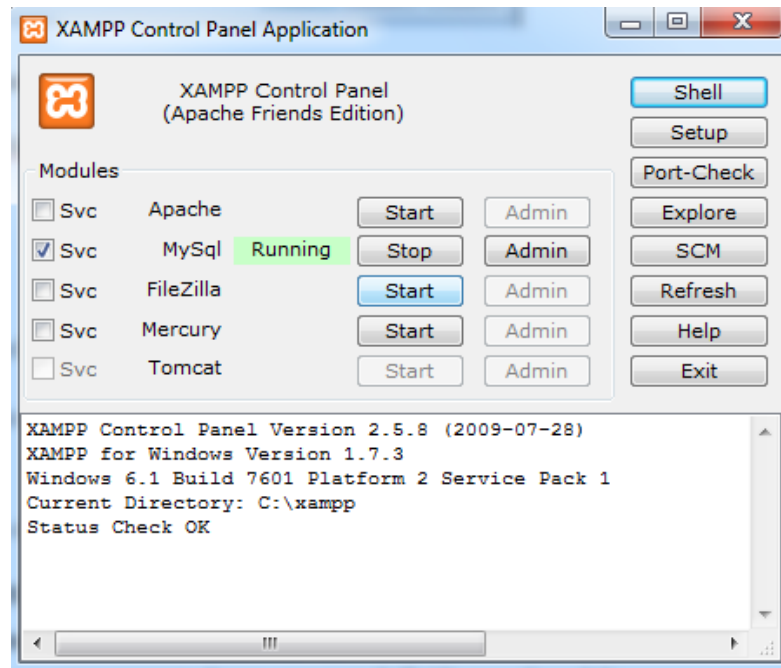


Figura V.93: Panel de Control de XAMPP.

Para instalar el servidor de aplicaciones en una máquina determinada, primero se debe descargar el instalador desde el siguiente enlace:

<http://dlc.sun.com.edgesuite.net/glassfish/3.1/release/glassfish-3.1-windows-ml.exe>

Una vez instalado glassfish se procede con los siguientes pasos:

- En el siguiente path “*C:\Program Files\glassfish-3.0.1\glassfish\domains\domain1\lib\ext*” se copia el archivo **mysql-connector-java.jar** (Lo puede descargar desde el siguiente link: <http://cdn.mysql.com/Downloads/Connector-J/mysql-connector-java-5.0.8.tar.gz>).
- Se Inicia en el explorador el administrador de Glassfish poniendo la url: **localhost:4848**

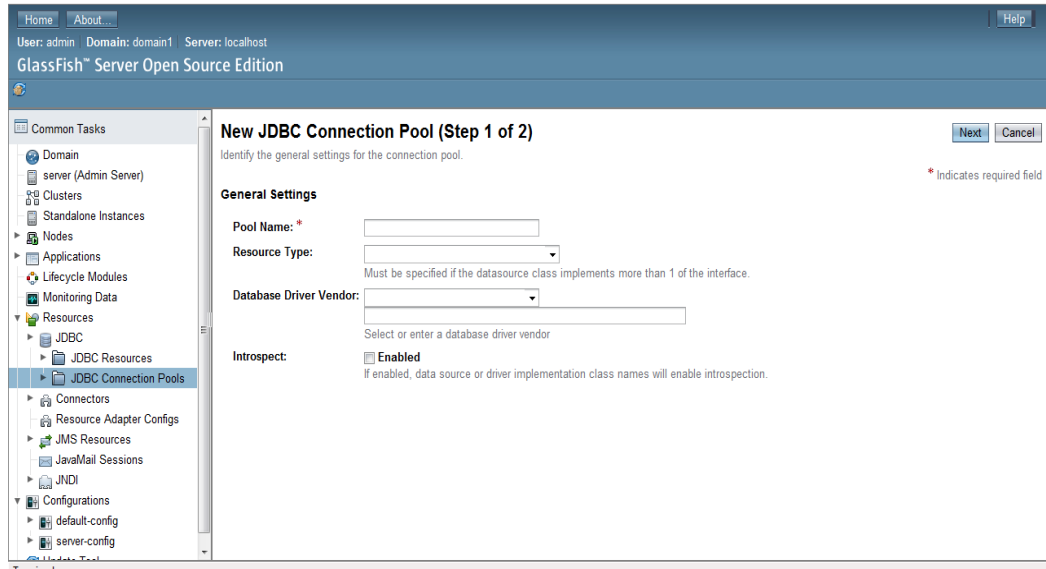


Figura V.94: Consola de Administración de GlassFish.

- Se dirige a la opción Resources JDBC – Jdbc Connection Pools

New JDBC Connection Pool (Step 1 of 2)

Identify the general settings for the connection pool.

General Settings

Pool Name: *

Resource Type:

Must be specified if the datasource class implements more than 1 of the interface.

Database Driver Vendor:

Select or enter a database driver vendor

Introspect: **Enabled**
If enabled, data source or driver implementation class names will enable introspection.

Figura V.95: Conexión a MySQL en GlassFish.

- Llenar los datos que se solicitan, de la siguiente manera:
 - **Pool Name:** Mysql_Pool
 - **Resource Type:** java.sql.Source
 - **Database Driver Vendor:** MySQL.

New JDBC Connection Pool (Step 1 of 2)

Identify the general settings for the connection pool.

General Settings

Pool Name: *

Resource Type:
Must be specified if the datasource class implements more than 1 of the interface.

Database Driver Vendor:
Select or enter a database driver vendor

Introspect: **Enabled**
If enabled, data source or driver implementation class names will enable introspection.

Figura V.96: Configuración de Conexión a MySQL.

- Configurar valores como el Nombre del Servidor, el Usuario, Password, etc.

| Name | Value | Description: |
|---|-----------------------------------|----------------------|
| <input type="checkbox"/> User | <input type="text"/> | <input type="text"/> |
| <input type="checkbox"/> EnableQueryTimeouts | true | <input type="text"/> |
| <input type="checkbox"/> UseInformationSchema | false | <input type="text"/> |
| <input type="checkbox"/> AutoReconnectForPools | false | <input type="text"/> |
| <input type="checkbox"/> NoAccessToProcedureBodies | false | <input type="text"/> |
| <input type="checkbox"/> ClobCharacterEncoding | <input type="text"/> | <input type="text"/> |
| <input type="checkbox"/> LoggerClassName | com.mysql.jdbc.log.StandardLogger | <input type="text"/> |
| <input type="checkbox"/> UltraDevHack | false | <input type="text"/> |
| <input type="checkbox"/> TrustCertificateKeyStoreType | <input type="text"/> | <input type="text"/> |
| <input type="checkbox"/> ServerName | localhost | <input type="text"/> |
| <input type="checkbox"/> AllowMultiQueries | false | <input type="text"/> |
| <input type="checkbox"/> ConnectionCollation | <input type="text"/> | <input type="text"/> |
| <input type="checkbox"/> SessionVariables | <input type="text"/> | <input type="text"/> |

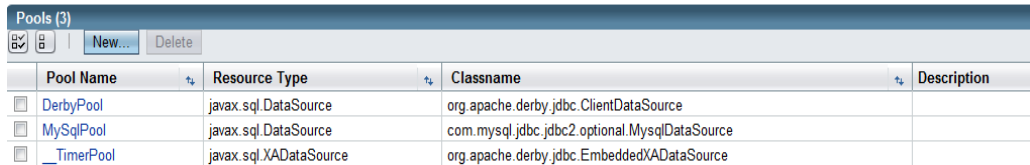
Figura V.97: Parámetros de Conexión a MySQL.

- Una vez llenado todos los valores se da clic en finalizar, y se hace una prueba de conexión, si todo está bien se mostrará una pantalla de **Ping Succeeded:**

✓ Ping Succeeded

JDBC Connection Pools

To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.



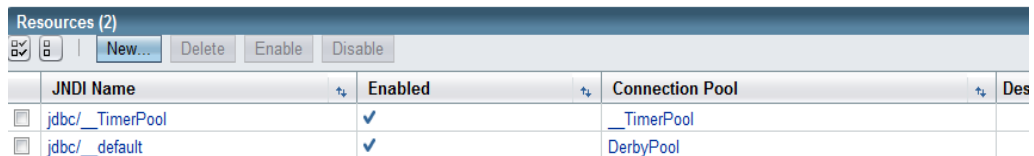
| Pool Name | Resource Type | Classname | Description |
|------------|------------------------|---|-------------|
| DerbyPool | javax.sql.DataSource | org.apache.derby.jdbc.ClientDataSource | |
| MySqlPool | javax.sql.DataSource | com.mysql.jdbc.jdbc2.optional.MysqlDataSource | |
| _TimerPool | javax.sql.XADataSource | org.apache.derby.jdbc.EmbeddedXADataSource | |

Figura V.98: Lista de Conexiones.

- Ahora se debe configurar el Data Set para eso, dirigirse a Resources/JDBC/JDBC Resources y dar en New.

JDBC Resources

JDBC resources provide applications with a means to connect to a database.



| JNDI Name | Enabled | Connection Pool | Des |
|-----------------|---------|-----------------|-----|
| jdbc/_TimerPool | ✓ | _TimerPool | |
| jdbc/_default | ✓ | DerbyPool | |

Figura V.99: Lista de Recursos JDBC

- En la página, llenar los siguientes datos.
 - Jndi Name: dstMySql
 - Pool name: MySqlPool

New JDBC Resource

Specify a unique JNDI name that identifies the JDBC resource you want to create. The name must contain only a

JNDI Name: *

Pool Name:
 Use the [JDBC Connection Pools](#) page to create new pools

Description:

Status: Enabled

Additional Properties (0)

| Name | Value | Description: |
|-----------------|-------|--------------|
| No items found. | | |

Figura V.100: Nuevo Recurso JDBC.

- Dar clic en Ok y se crea el dataSet.

JDBC Resources

JDBC resources provide applications with a means to connect to a database.

| Resources (3) | | | | |
|-------------------------------------|-----------------|-------------------------------------|-----------------|--|
| | JNDI Name | Enabled | Connection Pool | Description |
| <input checked="" type="checkbox"/> | dstMySql | <input checked="" type="checkbox"/> | MySqlPool | Conexion para MySql para la base basemantenimiento |
| <input type="checkbox"/> | jdbc/_TimerPool | <input checked="" type="checkbox"/> | _TimerPool | |
| <input type="checkbox"/> | jdbc/_default | <input checked="" type="checkbox"/> | DerbyPool | |

Figura V.101: Lista de Recursos JDBC.

De esta manera se ha configurado el servidor web para el alojamiento de la página y para conectarse a la base de datos de la aplicación.

El servidor web instalado es GlassFish 3.0

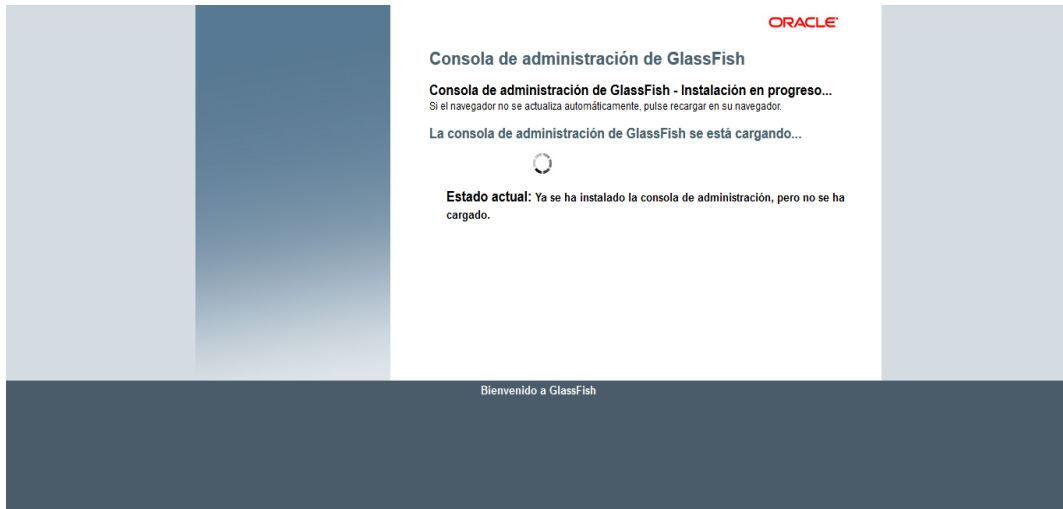


Figura V.102: Consola de GlassFish.

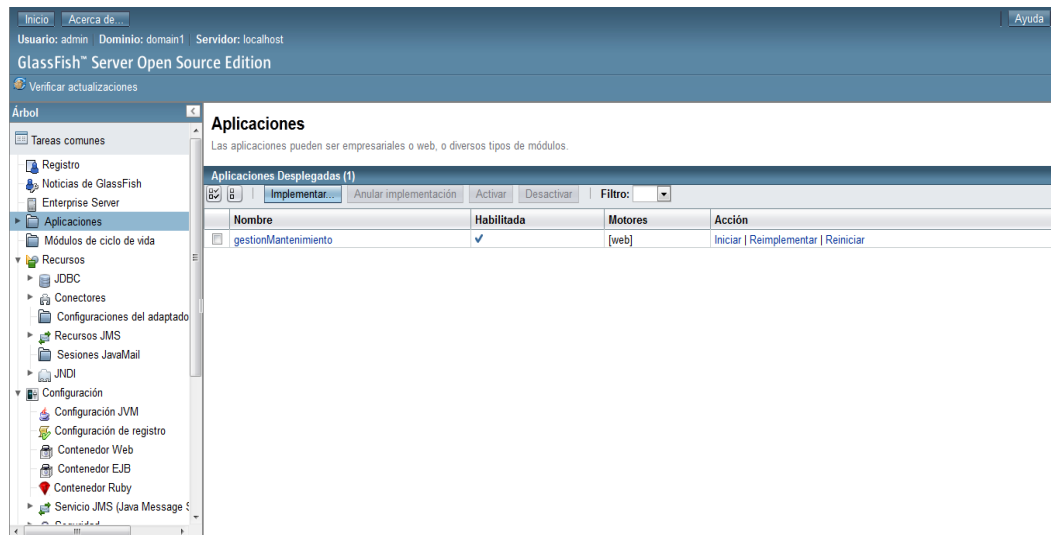


Figura V.103: Servidor Web GlassFish.

5.6.2. ALOJAMIENTO DE SITIO WEB.

Aplicaciones

Las aplicaciones pueden ser empresariales o web, o diversos tipos de módulos.

| Aplicaciones Desplegadas (1) | | | |
|---|-------------------------------------|---------|-------------------------------------|
| Nombre | Habilitada | Motores | Acción |
| <input type="checkbox"/> gestionMantenimiento | <input checked="" type="checkbox"/> | [web] | Iniciar Reimplementar Reiniciar |

Figura V.104: Alojamiento de Sitio Web.

5.6.3. PRUEBAS Y FUNCIONAMIENTO.

CAPITULO VI.

6. RESULTADOS Y DISCUSIÓN.

6.1.HIPÓTESIS.

6.1.1. INTRODUCCIÓN.

La necesidad de determinar la mejor metodología para el desarrollo de aplicaciones web en un entorno de programación JAVA, que permita minimizar el tiempo en la construcción y mantenimiento de estas aplicaciones, requiere de un análisis comparativo en base a parámetros que faciliten la selección de una metodología adecuada para este fin.

Con el propósito de determinar que la metodología propuesta es adecuada en la optimización del tiempo en la construcción y mantenimiento de aplicaciones web, se busca comparar dos prototipos de aplicaciones web, que utilizan metodologías distintas, tales como son:

- Metodología Tradicional MSF para la construcción de aplicaciones web que utiliza herramientas de desarrollo como JSP y JDBC.
- La metodología Propuesta “ERCA Web”; desarrollada en el presente trabajo investigativo, que utiliza las herramientas JSF y EJB.

Para esto, se han determinado parámetros de comparación los cuales permitirán establecer el tiempo en la construcción y mantenimiento de aplicaciones web y obtener un resultado de esta comparación.

Para poder analizar y comparar las metodologías se desarrolló dos prototipos, el Prototipo 1 fue implementado siguiendo la metodología Tradicional MSF, metodología que está diseñada para proyectos de cualquier dimensión y de cualquier tecnología; se hará énfasis en el uso de herramientas como son JSP y JDBC en el uso de esta metodología.

El Prototipo 2 fue implementado siguiendo la metodología propuesta ERCA WEB, la cual usa herramientas JSF y EJB en específico.

6.1.2. DEFINICIÓN DE LOS PARÁMETROS DE COMPARACIÓN.

Los parámetros y variables que a continuación se definen para la comparación de las dos metodologías fueron seleccionados por los autores de esta tesis, en base a las referencias de la información obtenida.

Los parámetros que se consideraron para comparar las dos metodologías han sido establecidos según lo visto en libros y tesis ubicadas en la Biblioteca General²³ y Centro de Documentación de la ESPOCH:

Tabla VI.XXXVI: Lista de Parámetros.

| Nº | PARÁMETRO | CONCEPTO |
|-----------|----------------------|--|
| 1 | Líneas de código | Cantidad de líneas de código para la ejecución de una tarea específica. |
| 2 | Reusabilidad | Poder usar partes o componentes de un software determinado en otro proyecto. |
| 3 | Generación de Código | Capacidad para convertir un programa |

²³ Lawrence Pfleeger, S. Ingeniería del Software: Teoría y Práctica. Traducido del inglés por Elvira Quiroga. Primera ed. Buenos Aires, Argentina. s.e. 2002. pp. 119 – 124.

| | | |
|----|--|--|
| | | sintácticamente correcto en una serie de instrucciones a ser interpretadas por una máquina. |
| 4 | Modularidad | Capacidad que tiene un sistema de ser estudiado, visto o estudiado como la unión de varias partes que interactúan entre sí y que trabajan para alcanzar un objetivo común, realizando cada una de ellas una tarea necesaria para la consecución de dicho objetivo. |
| 5 | Compatibilidad | La aplicación desarrollada con las herramientas de estudio es compatible con sistemas operativos de versiones anteriores. |
| 6 | Facilidad del consumo de datos | Capacidad de acceder a conexiones externas y llamadas a servicios web. |
| 7 | Generación automática de documentación | Capacidad de generación automática de documentación técnica teniendo únicamente como base el código y sus respectivos comentarios. |
| 8 | Escalabilidad. | Capacidad de un sistema informático de cambiar su tamaño o configuración para adaptarse a circunstancias cambiantes. |
| 9 | Mantenibilidad. | Facilidad con la que un sistema o componente software puede ser modificado para corregir fallos, mejorar su funcionamiento u otros atributos o adaptarse a cambios en el entorno |
| 10 | Presentación. | Tiene que ver con la facilidad de realizar interfaces amigables para el usuario. |

6.1.3. DEFINICIÓN DE LOS INDICADORES QUE MIDEN EL TIEMPO EN LA CONSTRUCCIÓN Y MANTENIMIENTO DE APLICACIONES WEB.

De los parámetros establecidos para la comparación entre la metodología tradicional y la metodología propuesta, se ha seleccionado los siguientes indicadores que permiten medir la el tiempo en la construcción y mantenimiento de aplicaciones web.

Tabla VI.XXXVII: Lista de Parámetros que determinan el Tiempo.

| N° | PARÁMETRO | CONCEPTO |
|----|----------------------|---|
| 1 | Reusabilidad | Poder usar partes o componentes de un software determinado en otro proyecto. |
| 2 | Generación de Código | Capacidad para convertir un programa sintácticamente correcto en una serie de instrucciones a ser interpretadas por una máquina. |
| 3 | Modularidad | Capacidad que tiene un sistema de ser estudiado o visto como la unión de varias partes que interactúan entre sí y que trabajan para alcanzar un objetivo común, realizando cada una de ellas una tarea necesaria para la consecución de dicho objetivo. |
| 4 | Mantenibilidad. | Facilidad con la que un sistema o componente software puede ser modificado para corregir fallos, mejorar su funcionamiento u otros atributos o adaptarse a cambios en el entorno |
| 5 | Presentación. | Tiene que ver con la facilidad de realizar interfaces amigables para el usuario. |

Tabla VI.LXXXVIII: Parámetro 1 – Reusabilidad.

| ÍNDICE | DESCRIPCIÓN. |
|-------------------------------|--|
| Portabilidad. | Se define como la característica que posee un software para ejecutarse en diferentes plataformas, el código fuente del software es capaz de reutilizarse en vez de crearse un nuevo código cuando el software pasa de una plataforma a otra. |
| Independencia de componentes. | Capacidad de funcionamiento de un módulo independiente de la plataforma o de otro proyecto. |
| Facilidad de integración. | Capacidad de integrar módulos a otros proyectos fácilmente. |

Tabla VI.LXXXIX: Parámetro 2 – Generación de Código.

| ÍNDICE | DESCRIPCIÓN |
|--|--|
| Creación de capa de acceso a datos. | Es la creación automática utilizando herramientas de desarrollo de código para la generación de clases que permiten el manejo de las bases de datos. |
| Creación de capa de lógica de negocio. | Es la creación automática utilizando herramientas de desarrollo de código para la generación de clases que permiten el manejo de la lógica de datos. |
| Líneas de código. | Índice que permite medir la cantidad de líneas de código para la realización de una consulta en específico. |
| Código no utilizable. | Código generado o programado que no se utiliza. |
| Complejidad. | Índice que permite medir la facilidad o complejidad de entendimiento del código. |

Tabla VI.XL: Parámetro 3 – Modularidad.

| ÍNDICE | DESCRIPCIÓN |
|--------------------------------|--|
| Modelo Vista Controlador | Patrón para establecer un modelo de abstracción de desarrollo de software, que separa en componentes Modelo - Vista - Controlador. |
| Diseño estructurado | Indicador para establecer la estructura y organización de las diferentes objetos en la aplicación web. |
| Capacidad de descomposición | Indicar que se utiliza para cuantificar el nivel de descomposición de los diferentes componentes web. |
| Comprensión | Patrón para medir el nivel de facilidad y manejabilidad del código utilizado en la aplicación web. |

Tabla VI.XLI: Parámetro 4 – Mantenibilidad.

| ÍNDICE | DESCRIPCIÓN |
|--|--|
| Escalabilidad | Propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para reaccionar y adaptarse sin perder calidad |
| Disponibilidad de Documentación | Cantidad de información disponible en la red o en otros medios, sobre un tema específico. |
| Código Entendible. | Líneas de código entendibles y legibles por un desarrollador cualquiera. |
| Adaptabilidad a nuevos requerimientos | Facilidad con la que se adapta o acopla a nuevos requerimientos un software determinado. |

Tabla VI.XLII: Parámetro 5 – Presentación.

| ÍNDICE | DESCRIPCIÓN |
|--------------------------|--|
| Templates personalizados | Se refiere al uso de plantillas de interfaz de usuario preelaboradas, que permitan ahorrar tiempo en el desarrollo de interfaces. |
| Componentes AJAX | Permite la creación de aplicaciones web interactivas, también llamadas aplicaciones RIA. |
| Validaciones | Proceso de comprobación de si algo satisface un cierto criterio. |
| Menor uso de java script | JavaScript es un lenguaje interpretado que se embebe en una página web HTML. Un lenguaje interpretado significa que a las instrucciones las analiza y procesa el navegador en el momento que deben ser ejecutadas. Al utilizar javaScript para realizar validaciones o para ejecutar menús desplegables, datepickers, etc., esto hace que el desarrollo de la aplicación sea mayor en tiempo; en cuanto tiene que ver en el mantenimiento y desarrollo de la aplicación web. |

6.1.4. CRITERIOS DE EVALUACIÓN.

A continuación se muestran los valores cualitativos y cuantitativos que se darán a los parámetros a ser analizados en la comparación de las metodologías, para lo cual se utilizará valores del 0 al 4.

Tabla VI.XLIII: Criterios de Evaluación General.

| Cuantitativa | 0 | 1 | 2 | 3 | 4 |
|--------------|------|---------------|-------|---------------|------|
| Cualitativa | Bajo | Medio Bajo | Medio | Medio Alto | Alto |
| Porcentajes | 0% | 25% | 50% | 75% | 100% |

6.1.5. ANALISIS DE LOS PARÁMETROS DE COMPARACIÓN PARA LAS METODOLOGÍAS.

El análisis comparativo será realizado en base a la información que se obtenga de la investigación bibliográfica, también en base a encuestas realizadas a expertos en el tema y a la observación realizada por los autores de la tesis utilizando para esto los dos prototipos implementados anteriormente.

6.1.5.1. REUSABILIDAD.

6.1.5.1.1. PORTABILIDAD.

La portabilidad tiene que ver con el hecho de poder utilizar el código fuente de un software en uno totalmente independiente del anterior. También la portabilidad se refiere a que un sistema puede ser ejecutado en diferentes plataformas sin ningún inconveniente. A mayor portabilidad menor es la dependencia del software con respecto a la plataforma.

El prerrequisito para la portabilidad es la abstracción generalizada entre la aplicación lógica y las interfaces del sistema. Cuando un software se puede compilar en diversas plataformas (x86, IA64, amd64, etc.), se dice que es multiplataforma. Esta característica es importante para el desarrollo de reducción costos, cuando se quiere hacer una misma aplicación. Por lo general, todo software JAVA es multiplataforma, sin embargo, no todo software JAVA puede ser reutilizable.

El prototipo 1 desarrollado usando la metodología tradicional que utiliza las tecnologías JSP y JDBC puede ser ejecutado en diferentes plataformas, sin embargo, la reutilización del código fuente es compleja, ya que es necesario acoplar éste código al nuevo proyecto, debido a que este código no funciona de forma independiente a las interfaces de usuario.

Por otro lado el prototipo 2 desarrollado, el cual utiliza la metodología propuesta, usando las tecnologías JSF y EJB manejan este término de portabilidad completamente, ya que si bien es multiplataforma, también el código fuente puede ser reutilizado en otros proyectos, debido a que el módulo EJB, donde se maneja la lógica de negocios, actúa como un proyecto independiente de las interfaces de usuario correspondiente al módulo JSF.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.XLIV: Evaluación del Índice de Portabilidad.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 2 | 4 |
| PORCENTAJE | | 50% | 100% |

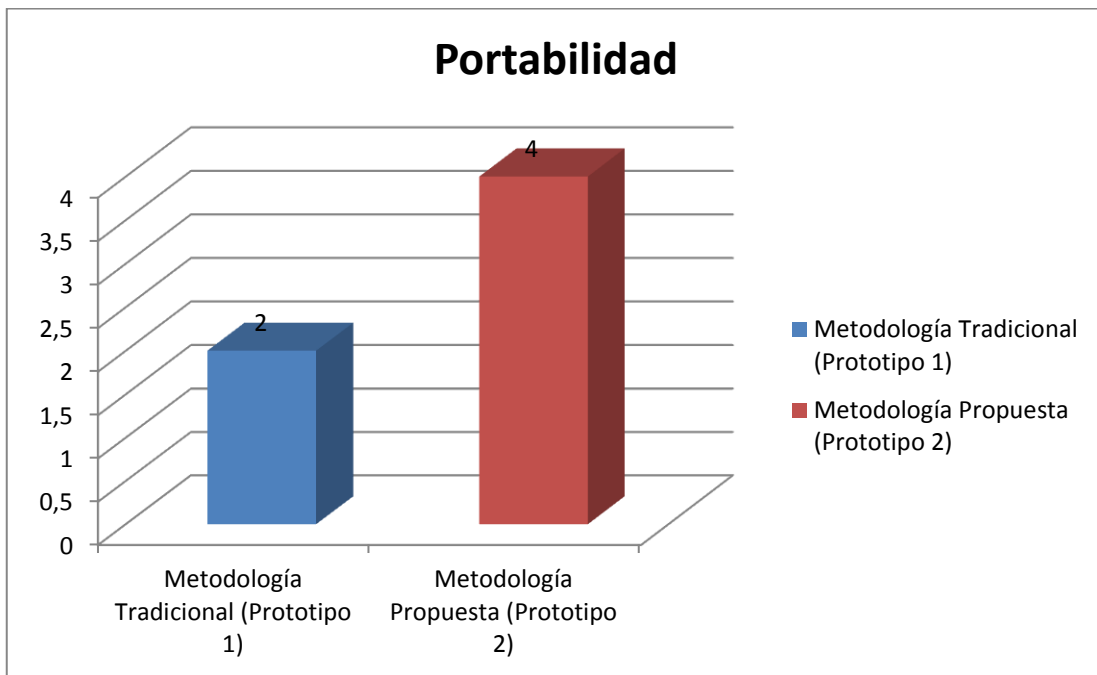


Figura VI.105: Índice de comparación Portabilidad.

En la Figura VI.105, se observa los resultados obtenidos para el índice de Portabilidad, en el cual el prototipo 2, que utiliza la metodología propuesta, con el valor de 4 toma ventaja sobre el prototipo 1, el cual utiliza la metodología tradicional, ya que en el prototipo 2 la portabilidad resulta más efectiva que en el prototipo 1.

6.1.5.1.2. INDEPENDENCIA DE COMPONENTES.

La independencia de componentes tiene que ver con el hecho de que un módulo pueda funcionar de manera independiente a otros módulos, es decir, un módulo independiente no necesariamente debe estar acoplado a otros para poderse ejecutar.

Mediante el desarrollo del prototipo 1, se establecieron dos módulos: Capa de Acceso a Datos y Lógica de Negocio y Capa de Presentación. Estos dos módulos no pueden actuar de manera independientes, es decir, la capa de acceso a datos y lógica de negocios necesita de la capa de presentación para poderse compilar, si hay un error en cualquiera de estos módulos, no se podrán compilar de forma separada.

Sin embargo, en el prototipo 2 que utiliza la metodología propuesta, se establecieron de igual forma dos módulos: Capa de Acceso a Datos y Lógica de Negocios y la Capa de Presentación. La diferencia radica en que el primer modulo el cual es gestionado por el componente EJB funciona como un proyecto independiente al segundo modulo gestionado por JSF. Es decir, en un mismo proyecto se tiene dos proyectos ejecutándose de forma conjunta pero no dependiente el uno del otro. Esto permite que el modulo EJB pueda ser reutilizado en otros proyectos y acoplarse sin ninguna dificultad a estos, así mismo el módulo JSF.

Conforme al análisis realizado se establecen los siguientes valores:

Tabla VI.XLV: Evaluación del Índice de Independencia de Componentes.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|-------------------------------|---------------------|--------------------|--------------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 1 | 3 |
| PORCENTAJE | | 25% | 75% |

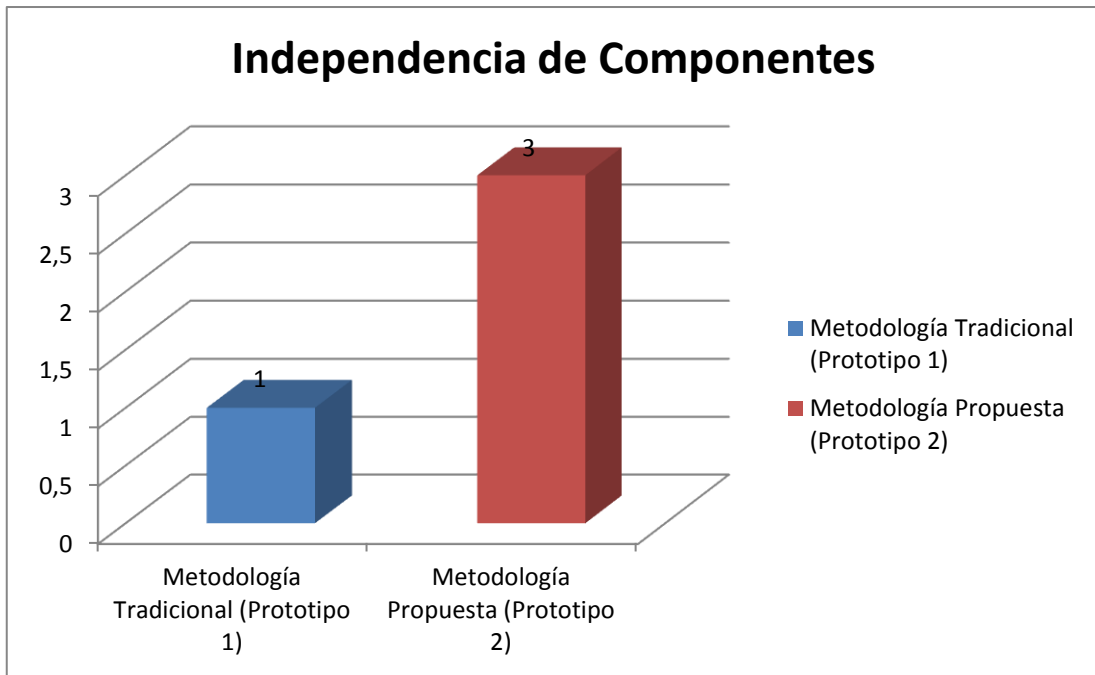


Figura VI.106: Independencia de Componentes.

En la figura VI.106 se establece los valores correspondientes al prototipo 1 que utiliza la metodología tradicional el cual posee un valor de 1 (medio bajo) frente al prototipo 2 que utiliza la metodología propuesta con un valor de 4 (muy alto), determinando así que el prototipo 2 está en ventaja frente al prototipo 1 con respecto a la independencia de sus componentes.

6.1.5.1.3. FACILIDAD DE INTEGRACIÓN.

Es muy importante poder realizar la integración de componentes ya existentes a proyectos o aplicaciones nuevas. La integración puede resultar un trabajo complejo o a la vez algo muy sencillo si se utilizan las herramientas tecnológicas adecuadas.

En el desarrollo del prototipo 1 usando la metodología tradicional se integraron algunos componentes java, para su realización, la integración no generó mayor complicación. De igual forma usando la metodología propuesta, la integración a otros proyectos no genera ninguna dificultad, ya que, como se dijo anteriormente, cada módulo se trata como componentes separados que pueden ser integrados a cualquier proyecto.

Los valores establecidos respecto al análisis hecho son:

Tabla VI.XLVI: Evaluación del Índice Facilidad de Integración.

| INDICADOR \ METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|--------------------------|-------------|-------------|
| | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | 4 | 4 |
| PORCENTAJE | 100% | 100% |

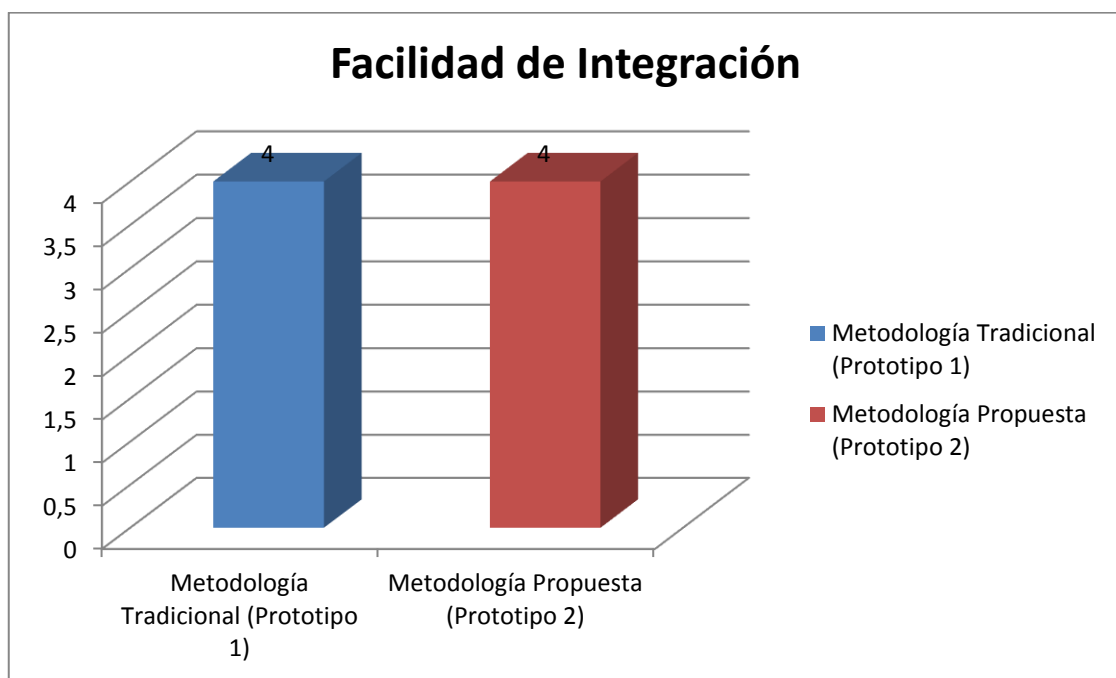


Figura VI.107: Índice Facilidad de Integración.

En la figura VI.107 se establece que existe una facilidad de integración alta utilizando la metodología tradicional así como la metodología propuesta.

De acuerdo a al análisis de cada índice se obtienen los siguientes valores para medir la reusabilidad que se obtiene desarrollando un proyecto siguiendo la metodología tradicional o siguiendo la metodología propuesta.

Tabla VI.XLVII: Tabla General del Parámetro de Reusabilidad.

| METODOLOGIAS INDICADORES | TRADICIONAL | PROPUESTA | PESO MÁXIMO |
|---|--------------------|------------------|--------------------|
| PORTABILIDAD | 2 | 4 | 4 |
| INDEPENDENCIA DE COMPONENTES | 1 | 3 | 4 |
| FACILIDAD DE INTEGRACIÓN | 4 | 4 | 4 |
| TOTAL | 7 | 11 | 12 |
| PORCENTAJE DE REUSABILIDAD | 58,33% | 91.67% | 100% |

Siguiendo la metodología tradicional se obtiene una reusabilidad del 58,33%, mientras que siguiendo la metodología propuesta se obtiene una reusabilidad del 91,67%. De modo que, está demostrado que la metodología propuesta abarca muy bien el término de reusabilidad para minimizar el tiempo en la construcción y mantenimiento de aplicaciones web. En la siguiente figura se observan los resultados obtenidos de cada uno de los índices.

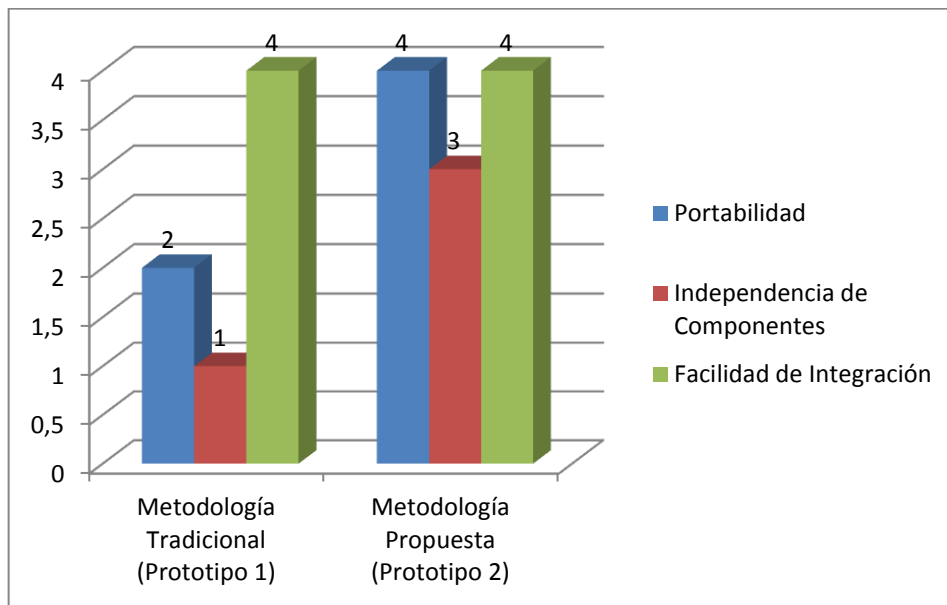


Figura VI.108: Parámetro de Reusabilidad de acuerdo a cada índice.

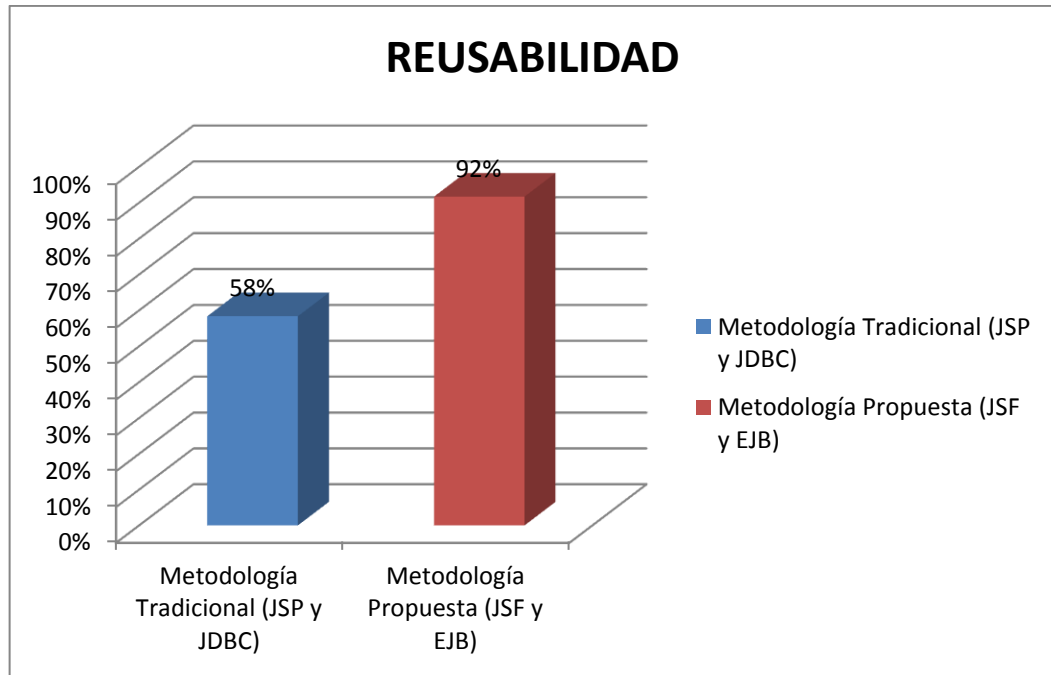


Figura VI.109: Reusabilidad en las dos metodologías.

6.1.5.1.4. INTERPRETACIÓN.

De acuerdo con la figura VI.109, al utilizar la metodología tradicional que hace uso de herramientas como Java Server Pages (JSP) y JDBC se maneja muy poco el concepto de Reusabilidad. Si se desea reutilizar componentes de una aplicación realizada en JSP con JDBC en otro proyecto se deberá acoplar cada clase creada al nuevo proyecto, de modo que esto permite que el tiempo en la construcción de un nuevo proyecto sea mayor.

Sin embargo, utilizando la metodología propuesta, es decir, haciendo uso de las tecnologías Java Server Faces (JSF) y Enterprise Java Bean (EJB), la reusabilidad es un concepto que se maneja adecuadamente, debido a que el módulo EJB trabaja independientemente y se acopla fácilmente a un nuevo proyecto.

Con todo esto, se demuestra que el prototipo desarrollado con JSP y JDBC no se acopla debidamente a nuevos proyectos, mientras que el prototipo desarrollado con EJB y JSF lo hace satisfactoriamente, ya que EJB es un proyecto independiente y puede ser exportado a otros proyectos, de la misma forma que lo puede hacer JSF.

6.1.5.2. GENERACIÓN DE CÓDIGO.

6.1.5.2.1. CREACIÓN DE CAPA DE ACCESO A DATOS.

La capa de acceso a datos sirve como puente entre la capa lógica de negocio y el proveedor de datos. Este capa pretende encapsular las especificidades del proveedor de datos tales como (SQL, Oracle, Sybase, archivos XML, texto, hojas electrónicas), a la siguiente capa. Para que si cambia el proveedor de datos solo se necesitará cambiar en una sola capa el proveedor de datos.

Generar el acceso a datos puede ser un proceso muy tedioso y complicado en algunos casos, sin embargo, si se utilizan las herramientas necesarias, se podrá generar esta capa de forma automática.

En los prototipos desarrollados con las diferentes metodologías, está realizada la capa de acceso a datos. Esta capa fue creada de manera diferente en cada prototipo.

En el prototipo 1 que utiliza la metodología tradicional la capa de acceso a datos se realizó con código programable por los desarrolladores, mas no por código generado ya que utilizando las tecnologías JSP y JDBC no permiten la generación de código de la capa de acceso a datos. En cierta manera esto podría ser una ventaja para el desarrollador ya que le permitirá familiarizarse más con el software que se está desarrollando, y como es su propio código conoce donde pueden haber posibles errores de sintaxis o de lógica, sin embargo sería una desventaja en cuanto al tiempo de desarrollo de la aplicación, ya que crear el código adecuado para esta capa es muy tedioso, peor aun si se tiene una base de datos muy grande.

En el prototipo 2, donde se utiliza la metodología propuesta, toda la capa de acceso a datos se genera automáticamente, sin necesidad que se programe, esto permite minimizar el tiempo en la construcción de aplicaciones web, ya que el desarrollador solo se centraría en algunos aspectos de la lógica de negocio y en la capa de presentación. El módulo EJB en el cual está establecida la capa de acceso a datos y la lógica de negocios, permite generar sin mayor inconveniente el acceso a datos completamente, una vez que se haya establecido la conexión a la base de datos pertinente.

De acuerdo con esta observación se establecen los siguientes valores:

Tabla VI.XLVIII: Evaluación del Índice de Creación de Capa de Acceso a Datos.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 2 | 4 |
| PORCENTAJE | | 50% | 100% |

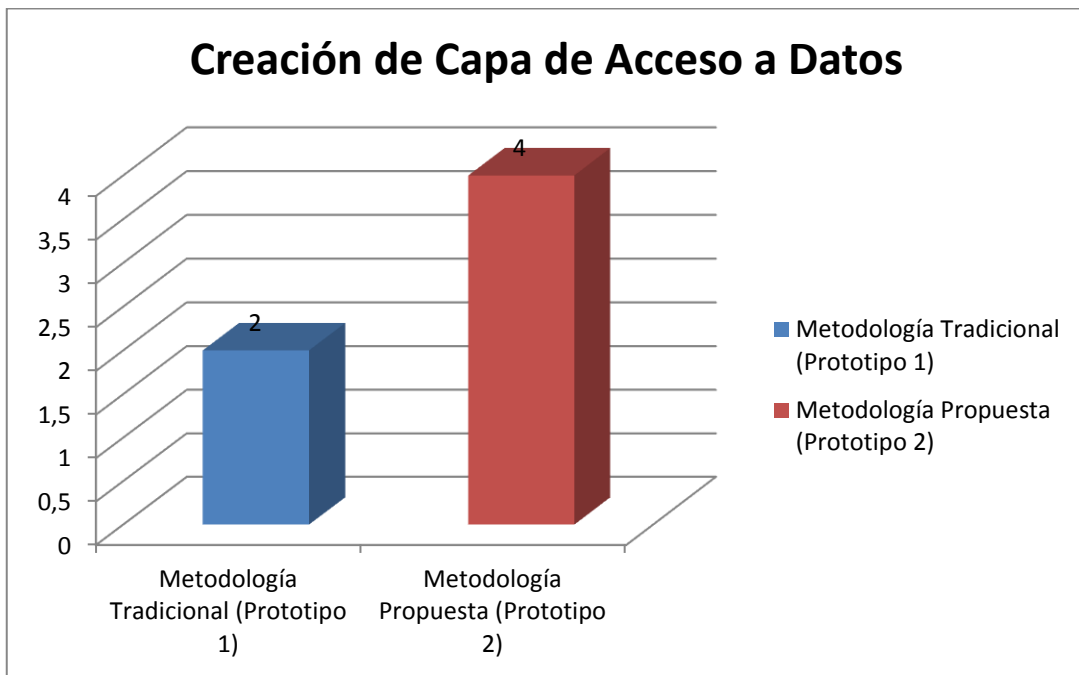


Figura VI.110: Índice Creación de Capa de Acceso a Datos.

Dada la figura VI.110 se observa claramente que siguiendo la metodología propuesta utilizando las tecnologías JSF y EJB se tiene una ventaja con la metodología tradicional que utiliza JSP y JDBC, con respecto a la creación de la capa de acceso a datos, ya que en la metodología propuesta, esta capa se genera completamente de forma automática y no hay que programarle, minimizando así el tiempo en la construcción de aplicaciones web.

6.1.5.2.2. GENERACIÓN DE CAPA DE LÓGICA DE NEGOCIO.

La capa de lógica de negocio es el corazón de la aplicación. El objetivo de esta capa es que toda la lógica de negocio esté bien localizada y no mezclada con los otros objetos de las otras capas, conllevando así a grandes ventajas. Es en esta capa donde se establecen todas las reglas de negocio que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

Utilizando la metodología tradicional se debe programar la capa de negocio totalmente conforme a los requerimientos de usuario establecidos ya que las tecnologías utilizadas en esta metodología no generan automáticamente ningún tipo de código para esta cap.

Usando la metodología propuesta, el código que genera con respecto a la lógica de negocio es muy poco. Las tecnologías utilizadas en esta metodología solo generan el CRUD (Create, Retrieve, Update, Delete) de cada entidad, por lo que las demás reglas de negocio deben ser programadas manualmente. Así como se muestra a continuación:

```
34 public List<Ubicaciontecnica> lista ()
35 {
36     Query query = em.createNamedQuery("Ubicaciontecnica.findAll");
37     return query.getResultList();
38 }
39
40 public Ubicaciontecnica actualizar (Ubicaciontecnica ubicacionTecnica)
41 {
42     return em.merge(ubicacionTecnica);
43 }
44
45 public void insertarUbicacionTecnica(Ubicaciontecnica ubicacionTecnica)
46 {
47     em.persist(ubicacionTecnica);
48 }
49
50
51 public void eliminarUbicacionTecnica(Ubicaciontecnica ubicacionTecnica)
52 {
53     em.remove(em.merge(ubicacionTecnica));
54 }
```

Figura VI.111: CRUD de Entidad Ubicación Técnica.

Dada esta observación se establecerán los siguientes valores con respecto a la generación de la capa de lógica de negocios:

Tabla VI.XLIX: Evaluación del Índice de Creación de Capa de Lógica de Negocio.

| INDICADOR \ METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|--------------------------|-------------|-------------|
| | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | 0 | 1 |
| PORCENTAJE | 0% | 25% |

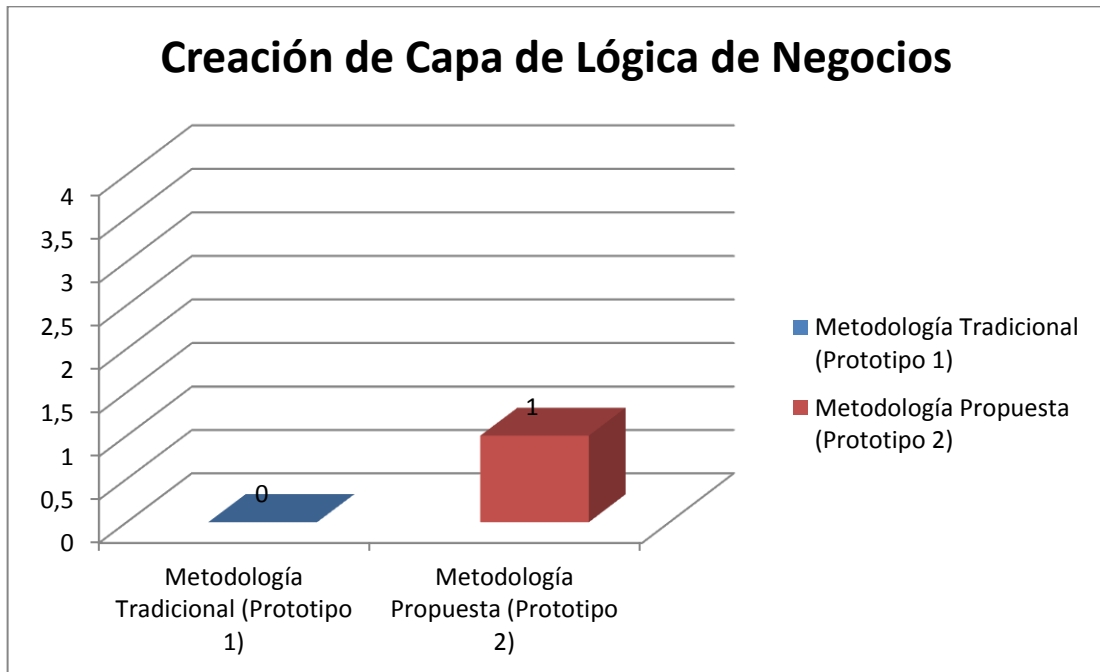


Figura VI.112: Gráfica del Índice Generación de Capa de Lógica de Negocios.

En la figura VI.112 se observa que usando la metodología propuesta se ahorra más tiempo al momento de crear la capa de lógica de negocios que usando la metodología tradicional, ya que la metodología propuesta genera poco código con respecto a la capa de negocio, sin embargo la metodología tradicional no genera ningún tipo de código para esta capa.

6.1.5.2.3. LINEAS DE CÓDIGO.

Tabla VII: Criterio de Evaluación para medir las Líneas de Código.

| Cuantitativa | 0 | 1 | 2 | 3 | 4 |
|---------------------|-----|-------|-------|-------|------|
| Cualitativa | >40 | 30-40 | 20-29 | 10-19 | <10 |
| Porcentajes | 0% | 25% | 50% | 75% | 100% |

El número líneas de código dentro de una aplicación es muy importante para poder mantener de mejor manera determinada aplicación y también minimizar el tiempo en la construcción de estas.

Para verificar el número de líneas de código en cada prototipo, se escogieron procedimientos o tareas puntuales y similares en cada uno de ellos. Se va a escoger dos tareas en cada prototipo. La primera tarea será la Inserción de una determinada Entidad y la segunda tarea será la Eliminación de esta Entidad.

En el prototipo 1 utilizando la metodología tradicional el número de líneas de código es el siguiente:

Tarea 1: Inserción de Técnicos.

```
25 public int Create() {
26     int flag = 0;
27     try {
28         conexion obj2 = new conexion();
29         obj2.conectar();
30         obj2.setPrst(obj2.getConn().prepareStatement("{call insertar_tecnico(?,?,?,?)}"));
31         obj2.getPrst().setString(1, getCedula());
32         obj2.getPrst().setString(2, getNombre());
33         obj2.getPrst().setString(3, getApellidos());
34         obj2.getPrst().setString(4, getDireccion());
35         obj2.getPrst().setString(5, getTelefono());
36         obj2.getPrst().executeQuery();
37         obj2.desconectar();
38         flag = 1;
39     } catch (ClassNotFoundException ex) {
40         Logger.getLogger(Estrategia.class.getName()).log(Level.SEVERE, null, ex);
41     } catch (InstantiationException ex) {
42         Logger.getLogger(Estrategia.class.getName()).log(Level.SEVERE, null, ex);
43     } catch (IllegalAccessException ex) {
44         Logger.getLogger(Estrategia.class.getName()).log(Level.SEVERE, null, ex);
45     } catch (SQLException exsql) {
46         flag = -1;
47     }
48     return flag;
49 }
```

Figura VI.113: Líneas de Código Método Inserción Técnicos Prototipo 1.

Número de Líneas de Código: 25 líneas de código.

Tarea 2: Eliminación de Técnicos.

```
51 public int Eliminar() {  
52     int flag = 0;  
53     try {  
54         conexion obj2 = new conexion();  
55         obj2.conectar();  
56         obj2.setPrst(obj2.getConn().prepareStatement("{call eliminar_tecnico(?)}"));  
57         obj2.getPrst().setInt(1, getId_tecnico());  
58         obj2.getPrst().executeQuery();  
59         obj2.desconectar();  
60         flag = 1;  
61     } catch (ClassNotFoundException ex) {  
62         Logger.getLogger(Estrategia.class.getName()).log(Level.SEVERE, null, ex);  
63     } catch (InstantiationException ex) {  
64         Logger.getLogger(Estrategia.class.getName()).log(Level.SEVERE, null, ex);  
65     } catch (IllegalAccessException ex) {  
66         Logger.getLogger(Estrategia.class.getName()).log(Level.SEVERE, null, ex);  
67     } catch (SQLException exsql) {  
68         flag = -1;  
69     }  
70     return flag;  
71 }
```

Figura VI.114: Líneas de Código Método Eliminación Técnicos Prototipo 1.

Número de Líneas de Código: 21 líneas de código.

En el prototipo 2 utilizando la metodología propuesta el número de líneas de código es el siguiente:

Tarea 1: Inserción de Técnicos.

```
38 public void insertarTecnico(Tecnico tecnico)  
39 {  
40     em.persist(tecnico);  
41 }
```

Figura VI.115: Líneas de Código Método Inserción Técnicos Prototipo 2.

Número de Líneas de Código: 4 líneas de código.

Tarea 2: Eliminación de Técnicos.

```
43     public void eliminarTecnico(Tecnico tecnico)
44     {
45         em.remove(em.merge(tecnico));
46     }
```

Figura VI.116: Líneas de Código Método Eliminación Técnicos Prototipo 2.

Número de Líneas de Código: 4 líneas de código.

Dada esta observación se obtienen los siguientes valores:

Tabla VI.LI: Evaluación del Índice de Líneas de Código.

| METODOLOGÍAS INDICADOR | TRADICIONAL | | PROPUESTA | |
|-------------------------------|-------------|---------|-------------|---------|
| | Prototipo 1 | | Prototipo 2 | |
| | Tarea 1 | Tarea 2 | Tarea 1 | Tarea 2 |
| LÍNEAS DE CÓDIGO | 25 | 21 | 4 | 4 |
| PROMEDIO | 23 | | 4 | |
| CRITERIO DE EVALUACIÓN | 2 | | 4 | |
| PORCENTAJE | 50% | | 100% | |

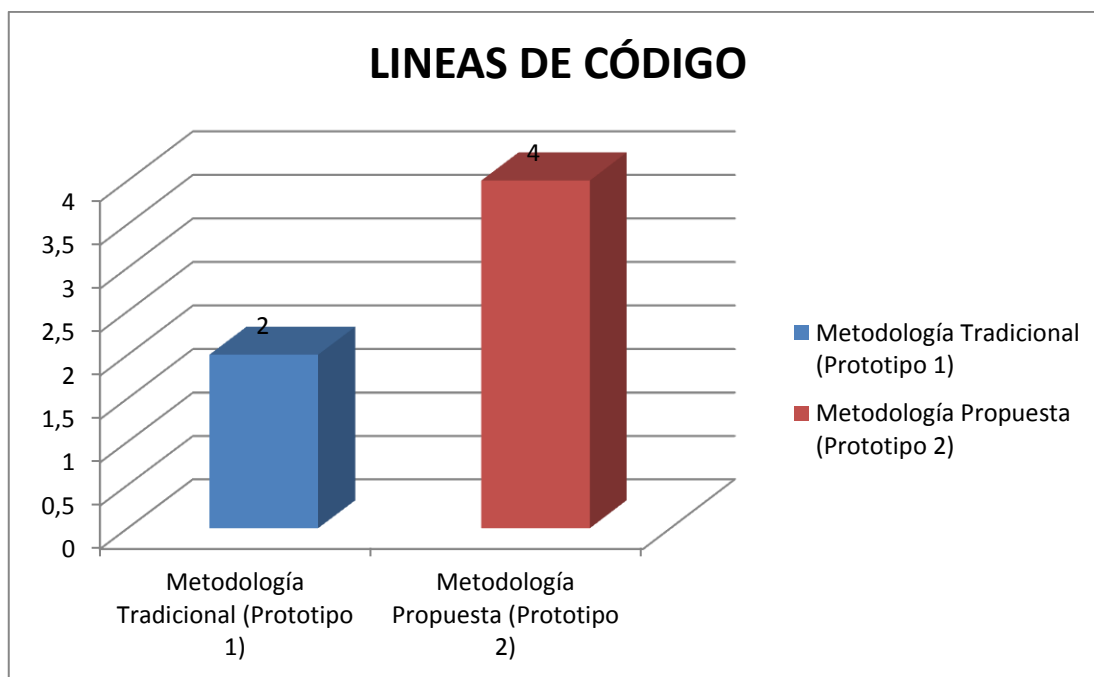


Figura VI.117: Gráfica del Índice Líneas de Código.

Mediante la gráfica se establece que utilizando la metodología propuesta se ahorra mucho tiempo en la programación de una aplicación web, ya que las líneas de código utilizadas para realizar tareas esenciales son muy pocas, en contraste con la metodología tradicional que utiliza de 5 a 6 veces más líneas de código que la anterior, para realizar la misma tarea. Esto permite que se optimice el tiempo en la construcción y mantenimiento de aplicaciones web usando la metodología propuesta.

6.1.5.2.4. CODIGO NO UTILIZABLE.

Tabla VI.LII: Criterio de Evaluación para medir las Líneas de Código no utilizable.

| | | | | | |
|---------------------|----------|-------|------|----------|------|
| Cuantitativa | 0 | 1 | 2 | 3 | 4 |
| Cualitativa | Excesivo | Mucho | Poco | Muy Poco | Nada |
| Porcentajes | 0% | 25% | 50% | 75% | 100% |

Tener código no utilizable en una aplicación puede generar problemas al momento de dar un mantenimiento a una determinada aplicación, además que puede volver pesada a la aplicación al momento de su ejecución.

Usando la metodología tradicional no se encontrará problemas de código innecesario o no utilizable ya que el código aquí es propiamente programado por el desarrollador, y este código es utilizado en toda la aplicación. Sin embargo, usando la metodología propuesta, como la mayoría del código es generado, se genera código que es innecesario y que no se utiliza en la aplicación, por lo cual esto sería una gran desventaja al utilizar esta metodología.

El número de líneas de código no utilizable que se genera usando la metodología propuesta es de: 6 líneas por cada Entidad.

```
27 @NamedQuery(name = "Tecnico.findByIdTecnico", query = "SELECT t FROM Tecnico t WHERE t.idTecnico = :idTecnico"),
28 @NamedQuery(name = "Tecnico.findByCedula", query = "SELECT t FROM Tecnico t WHERE t.cedula = :cedula"),
29 @NamedQuery(name = "Tecnico.findByNombre", query = "SELECT t FROM Tecnico t WHERE t.nombre = :nombre"),
30 @NamedQuery(name = "Tecnico.findByApellidos", query = "SELECT t FROM Tecnico t WHERE t.apellidos = :apellidos"),
31 @NamedQuery(name = "Tecnico.findByDireccion", query = "SELECT t FROM Tecnico t WHERE t.direccion = :direccion"),
32 @NamedQuery(name = "Tecnico.findByTelefono", query = "SELECT t FROM Tecnico t WHERE t.telefono = :telefono");
```

Figura VI.118: Código no utilizable en la Metodología Propuesto en la Capa de Acceso a Datos.

Además genera código no utilizable en la capa de negocios con un total de 15 líneas por entidad:

```
46 public List<T> findRange(int[] range) {
47     javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
48     cq.select(cq.from(entityClass));
49     javax.persistence.Query q = getEntityManager().createQuery(cq);
50     q.setMaxResults(range[1] - range[0]);
51     q.setFirstResult(range[0]);
52     return q.getResultList();
53 }
54 public int count() {
55     javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
56     javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
57     cq.select(getEntityManager().getCriteriaBuilder().count(rt));
58     javax.persistence.Query q = getEntityManager().createQuery(cq);
59     return ((Long) q.getSingleResult()).intValue();
60 }
```

Figura VI.119: Código no utilizable en la Metodología Propuesto en la Capa de Lógica de Negocios.

Realizada esta observación se establecen los siguientes valores:

Tabla VI.LIII: Evaluación del Índice de Código no utilizable.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 4 | 1 |
| PORCENTAJE | | 100% | 25% |

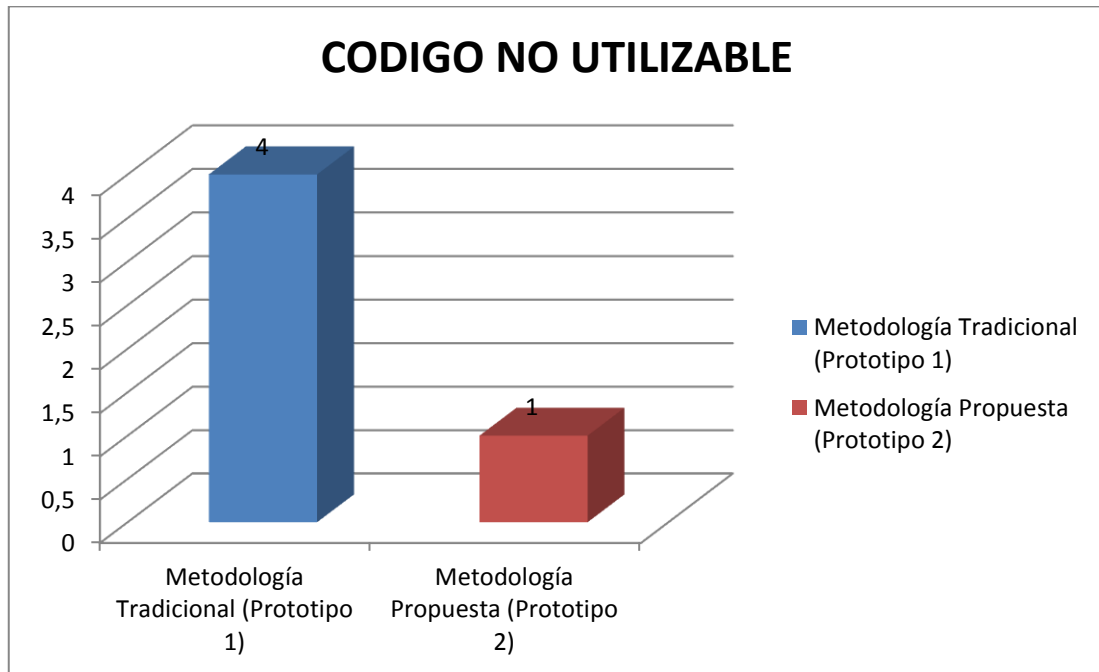


Figura VI.120: Gráfica del Índice Código no Utilizable.

De acuerdo a la figura VI.120 se establece que usando la Metodología Tradicional, el código fuente de una aplicación web va a ser aprovechado por completo y no dará lugar a código innecesario que implique dificultad al momento de mantener una aplicación o al momento de construirla. Por el contrario la metodología propuesta genera código innecesario, no utilizable, para el desarrollo de aplicaciones web.

6.1.5.2.5. COMPLEJIDAD.

Tabla VI.LIV: Criterio de Evaluación para medir la Complejidad.

| | | | | | |
|---------------------|----------|------|-------|------|----------|
| Cuantitativa | 0 | 1 | 2 | 3 | 4 |
| Cualitativa | Muy Alta | Alta | Media | Baja | Muy Baja |
| Porcentajes | 0% | 25% | 50% | 75% | 100% |

Este índice tiene que ver con cuán entendible o no es el código creado o generado en una aplicación web. Mientras más entendible sea el código menos tiempo se necesitará para construir o mantener una aplicación. Para construir aplicaciones usando la metodología tradicional el código creado es entendible por cualquier desarrollador de software, ya que las tecnologías que utiliza se basan a la programación orientada a objetos y es muy fácil de entender por cualquier desarrollador. Sin embargo usando la metodología propuesta, si se desea entender completamente el código generado o creado se debe tener un basto conocimiento sobre la programación en JAVA, de todos modos, el código aquí es un poco entendible para cualquier desarrollador de software ya que de una u otra manera también se basa en la programación orientada a objetos.

En el prototipo 1, para poder realizar el listado de todos los técnicos de una base de datos se escribe simplemente una consulta sql:

```
SQL = "SELECT Id_Tecnico,Cedula,Nombre,Apellidos,Direccion,Telefono FROM tecnico";  
bbj.consultar(SQL);
```

Este código es entendible para cualquier desarrollador.

Sin embargo, en el prototipo 2, para realizar el listado de todos los técnicos, se tiene el siguiente código:

```
public List<T> findAll() {  
    javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();  
    cq.select(cq.from(entityClass));  
    return getEntityManager().createQuery(cq).getResultList();  
}
```

Dado estos criterios se obtienen los siguientes valores:

Tabla V.LLV: Evaluación del Índice de Complejidad.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 3 | 1 |
| PORCENTAJE | | 75% | 25% |

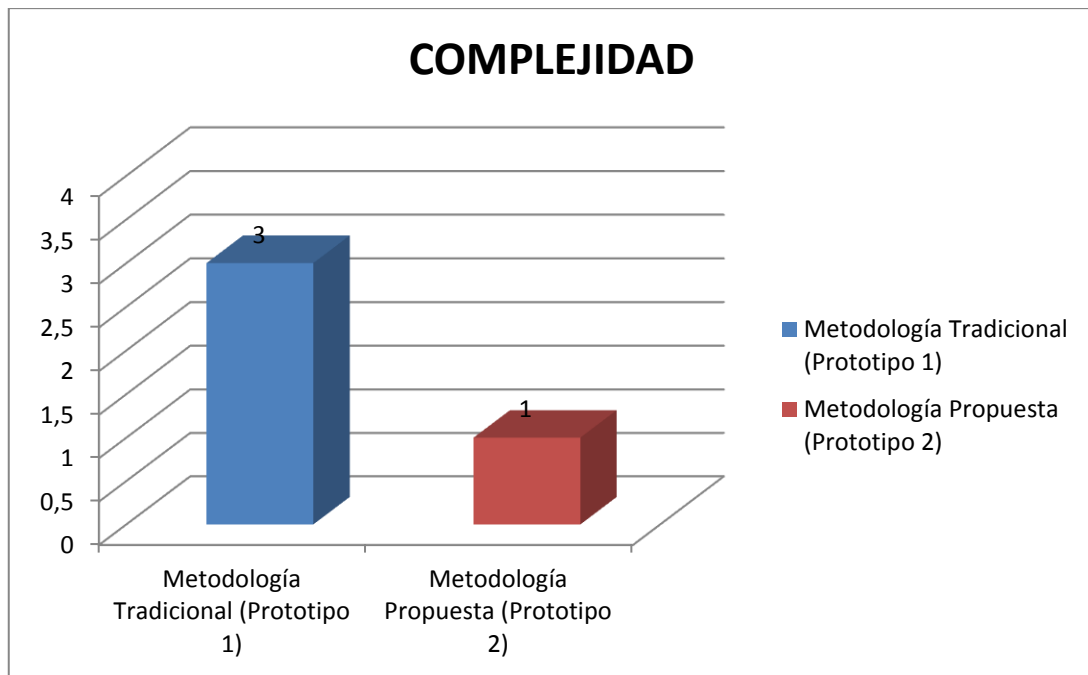


Figura VI.121: Gráfica del Índice Complejidad.

Analizada la figura VI.121 se concluye que usando la metodología tradicional el código que se genera es muy entendible y esto ayuda para un mejor mantenimiento a la aplicación minimizando así el tiempo en el mantenimiento. Por otro lado, se establece una complejidad en el código si se usa la metodología propuesta, afectando de esta manera a un mantenimiento óptimo y eficaz y a maximizar el tiempo en la construcción de nuevas aplicaciones.

De acuerdo al análisis de cada índice se obtienen los siguientes valores para medir la generación de código que se obtiene desarrollando un proyecto siguiendo la metodología tradicional o siguiendo la metodología propuesta.

Tabla VI.LVI: Valores de cada índice del Parámetro de Generación de Código.

| METODOLOGIA INDICADORES | TRADICIONAL | PROPUESTA | PESO MÁXIMO |
|---|--------------------|------------------|--------------------|
| Creación de capa de acceso a datos. | 2 | 4 | 4 |
| Creación de capa de lógica de negocio. | 0 | 1 | 4 |
| Líneas de código. | 2 | 4 | 4 |
| Código no utilizable. | 4 | 1 | 4 |
| Complejidad. | 3 | 1 | 4 |
| Total | 11 | 11 | 20 |
| Porcentaje de generación de código | 55,00% | 55,00% | 100% |

Siguiendo la metodología tradicional se obtiene una eficacia de generación de código del 55%, de la misma forma siguiendo la metodología propuesta se obtiene una eficacia de generación de código del 55%. De modo que, usando cualquiera de estas dos metodologías el código tendrá el mismo grado de eficacia y el tiempo en la construcción y mantenimiento de aplicaciones web será constante. En la siguiente figura se observan los resultados obtenidos de cada uno de los índices.

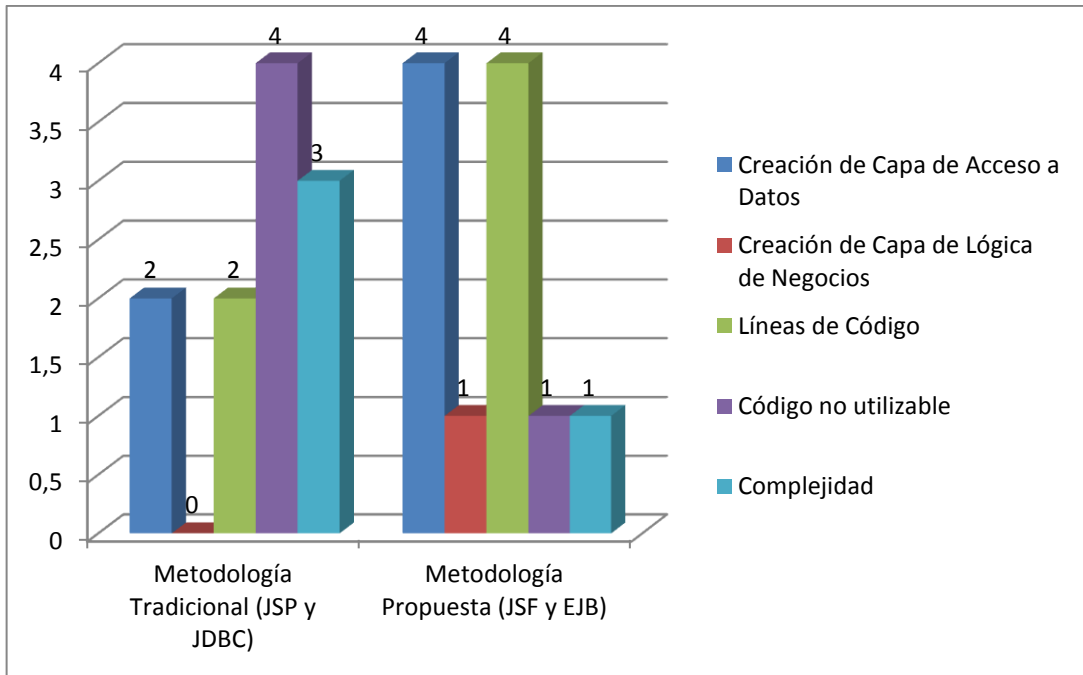


Figura VI.122: Gráfica de Valores del Parámetro Generación de Código.

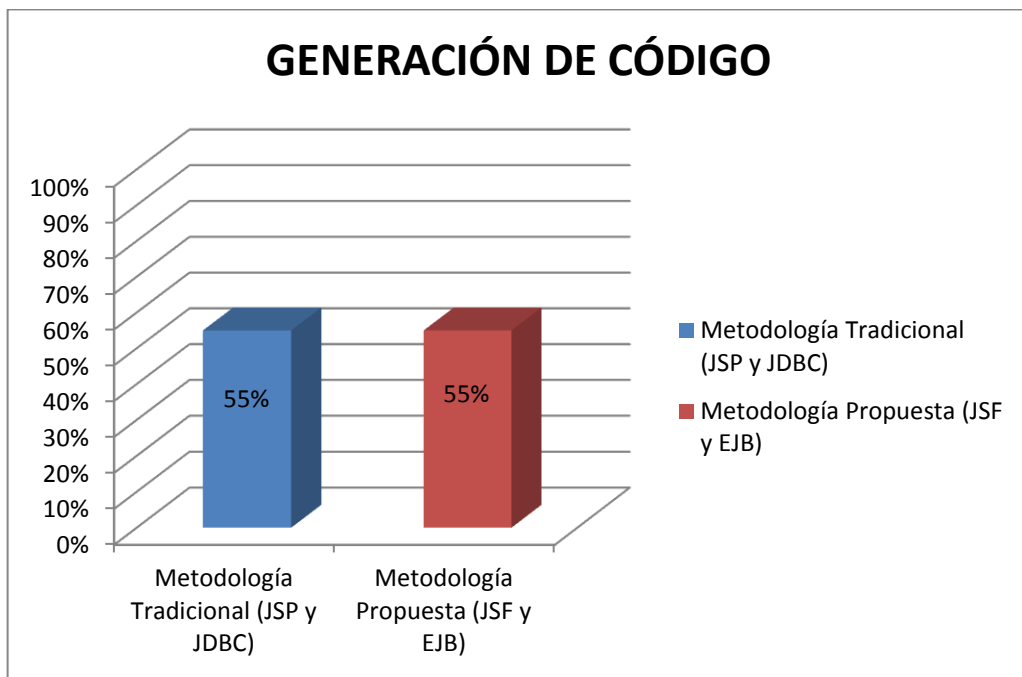


Figura VI.123: Gráfica Porcentual del Parámetro Generación de Código.

6.1.5.2.6. INTERPRETACIÓN.

En cuanto a la generación de código se puede observar en la figura VI.123 que tanto usando una metodología tradicional como usando la metodología propuesta existe un 55% de utilidad del código que se genera, por lo cual, usando una que otra metodología no se minimizará el tiempo en la construcción y/o mantenimiento de aplicaciones web.

6.1.5.3. *MODULARIDAD.*

6.1.5.3.1. MODELO VISTA CONTROLADOR

Es un patrón o modelo de abstracción de desarrollo de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. El patrón de llamada y retorno MVC (según CMU), se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página.

El prototipo realizado por la Metodología Propuesta es realizado de acuerdo a este índice, obteniendo ventajas como son:

- Sencillez para crear distintas representaciones de los mismos datos.
- Facilidad para la realización de pruebas unitarias de los componentes, así como de aplicar desarrollo guiado por pruebas (TDD).
- Reutilización de los componentes.
- Simplicidad en el mantenimiento de los sistemas.
- Facilidad para desarrollar prototipos rápidos.
- Los desarrollos suelen ser más escalables.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LVII: Evaluación del Índice de Modelo Vista Controlador.

| METODOLOGIAS | TRADICIONAL | PROPUESTA |
|-------------------------------|--------------------|--------------------|
| | Prototipo 1 | Prototipo 2 |
| INDICADOR | | |
| CRITERIO DE EVALUACIÓN | 1 | 4 |
| PORCENTAJES | 25% | 100% |

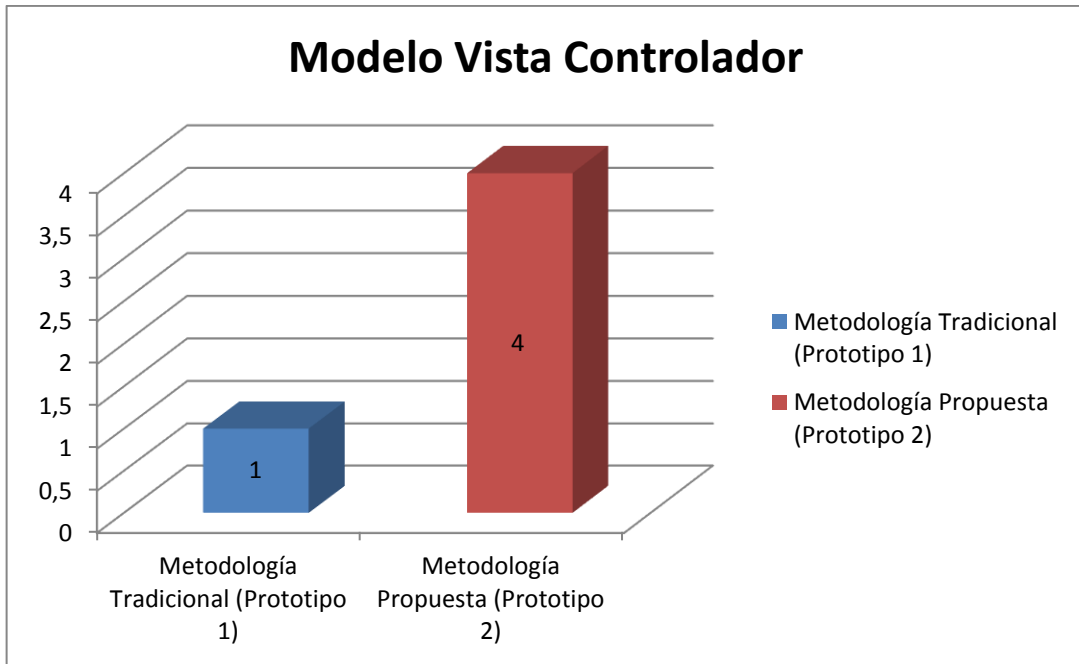


Figura VI.124: Índice de comparación Modelo Vista Controlador.

Se puede observar en la figura VI.124 que el prototipo 2 de la Metodología Propuesta tiene un valor mayor al prototipo 1, esto debido al patrón Modelo Vista Controlador que adopta la Metodología Propuesta.

6.1.5.3.2. DISEÑO ESTRUCTURADO

El hecho de que una Metodología no adopte el Modelo Vista Controlador en su desarrollo de software no implica que su diseño estructurado este mal.

Definición:

Diseño estructurado es el proceso de decidir que componentes, y la interconexión entre los mismos, para solucionar un problema bien especificado.

Frente a esto se puede encontrar los siguientes beneficios: Eficiencia, Mantenibilidad, Modificabilidad, Flexibilidad, Generalidad, Utilidad.

En el desarrollo de los prototipos se ha encontrado que tanto la Metodología Tradicional como la Metodología Propuesta siguen un diseño bien estructurado, razón por la cual tienen el mismo porcentaje de calificación en este índice.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LVIII: Evaluación del Índice de Diseño Estructurado.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 4 | 4 |
| PORCENTAJES | | 100% | 100% |

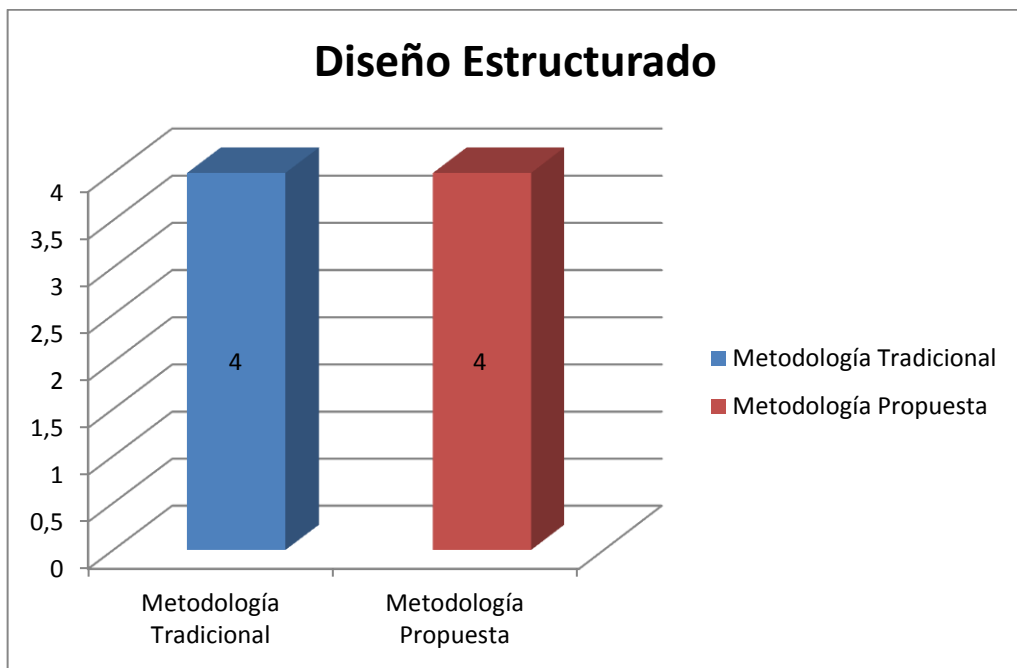


Figura VI.125: Índice de comparación Diseño Estructurado.

6.1.5.3.3. CAPACIDAD DE DESCOMPOSICIÓN

Índice que mide el tiempo que se ahorra cuando se hace una aplicación web, mediante la capacidad de descomponer el trabajo en módulos independientes, también se toma en consideración a la hora de unificar los módulos y obtener la aplicación total sin complicaciones.

La aplicación tradicional utiliza un concepto claro de descomposición, utilizado mediante paquetes, clases, páginas web; sin embargo en las páginas web se mezcla las páginas HTML y páginas JSP, haciendo esto que sea un tanto difícil a la hora de la

unificación de módulos. Por su contraparte la Metodología Propuesta tiene bien marcada la separación de los que son las paginas XHTML con la lógica de proceso de las paginas como son los BackBeans.

```
0
7 <%@page import="java.sql.SQLException"%>
8 <%@page import="general.*" %>
9 <%@page contentType="text/html" pageEncoding="UTF-8"%>
10 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
11 "http://www.w3.org/TR/html4/loose.dtd">
12
13 <html>
14 <head>
15 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
16 <title>JSP Page</title>
17 </head>
18 <body bgcolor="ECECEC" >
19 <%
20 String nombre = request.getParameter("usuario");
21 String password = request.getParameter("password");
22
23 conexion obj = new conexion();
24 obj.conectar();
25 String SQL;
26 SQL = "select count(*) from login where usuario='"+nombre+"' and clave=MDS('"+password+"') ";
27 try {
28 obj.consultar(SQL);
29 if (obj.getRs() != null) {
30 while (obj.getRs().next()) {
31 //String algo = obj.getRs().getString(1);
32 int resultado = obj.getRs().getInt(1);
33 if(resultado == 1)
34 {
35 out.print("<center><h1> Nombre de Usuario y Contrase?a <br>");
36 out.print("Correctos</h1>");
```

Figura VI.126: Página HTML prototipo 1.

```
3 <html xmlns="http://www.w3.org/1999/xhtml"
4 xmlns:ui="http://java.sun.com/jsf/facelets"
5 xmlns:h="http://java.sun.com/jsf/html">
6
7 <body>
8 <ui:composition template="/WEB-INF/templates/segundaPlantilla.xhtml">
9 <ui:define name="content">
10
11 <h3><outputText value="PROCESA LOGIN"/></h3>
12
13 <h:form rendered="#{empty backBeanLogin.estado}">
14 <h3><outputText value="CAMBIO DE LOGIN FALLO " /></h3>
15 <h4><outputText value="INGRESE CORRECTAMENTE LOS DATOS"/></h4>
16 <ul>
17 <li><commandLink value="Cambiar Contrase?a" action="/gestion/Login/cambiarPassword.xhtml?>
18
19 </li>
20 </ul>
21 </h:form>
22
23 <h:form rendered="#{not empty backBeanLogin.estado}">
24 <h3><outputText value="CAMBIO DE LOGIN FINALIZO CON EXITO " /></h3>
25 <ul>
26 <li><commandLink value="ACEPTAR" action="/Paginas_xhtml1/index.xhtml?faces-redirect=true">
27
28 </li>
29 </ul>
30 </h:form>
31
32 </ui:define>
33 </ui:composition>
34 </body>
35 </html>
```

```
5 package backBean;
6
7 import javax.ejb.EJB;
8 import javax.faces.bean.ManagedBean;
9 import javax.faces.bean.SessionScoped;
10 import sessionBean.LoginSession;
11
12 /**
13 *
14 * @author Livia
15 */
16 @ManagedBean
17 @SessionScoped
18 public class backBeanLogin {
19
20 @EJB
21 private LoginSession loginSession;
22 private String usuario = "";
23 private String clave = "";
24 private String estado = "";
25 private String claveNueva;
26 private String claveNueva1;
27
28 public String getClaveNueva() {
29 return claveNueva;
30 }
31
32 public void setClaveNueva(String claveNueva) {
33 this.claveNueva = claveNueva;
34 }
35
36 public String getClaveNueva1() {
37 return claveNueva1;
38 }
39
40 public void setClaveNueva1(String claveNueva1) {
41 this.claveNueva1 = claveNueva1;
42 }
```

Figura VI.127: Página XHTML y Backbean.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LIX: Evaluación del Índice de Capacidad de Descomposición.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 2 | 4 |
| PORCENTAJES | | 50% | 100% |

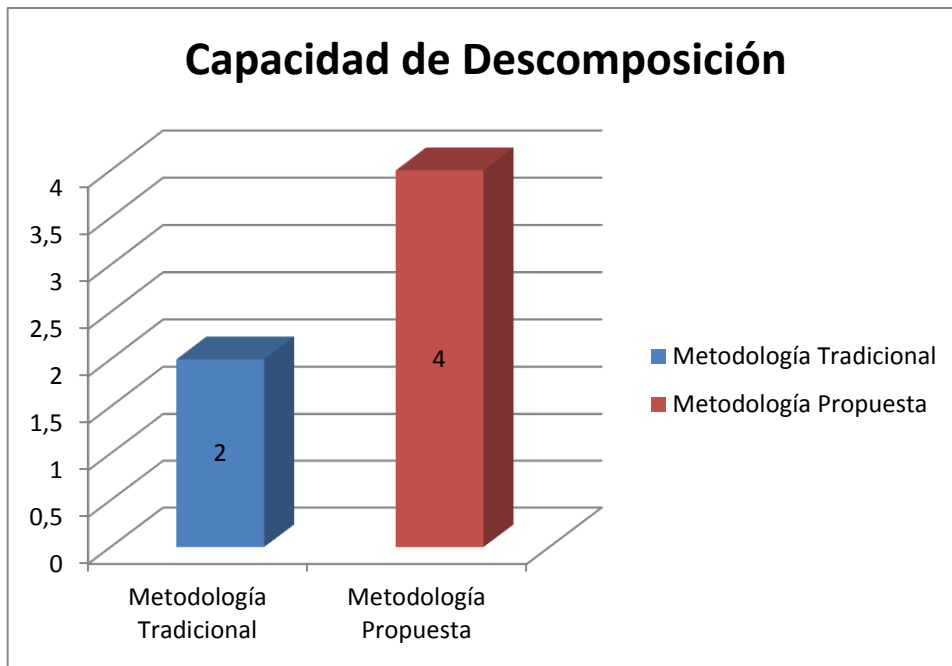


Figura VI.128: Índice de comparación Capacidad de Descomposición.

6.1.5.3.4. COMPRENSIÓN.

Es mejorar la facilidad de comprensión del código o cambiar su estructura y diseño y eliminar código muerto, para facilitar el mantenimiento en el futuro. Añadir nuevo comportamiento a un programa puede ser difícil con la estructura dada del programa, así que un desarrollador puede refactorizarlo primero para facilitar esta tarea y luego añadir el nuevo comportamiento.

Este índice sirve para medir la optimización del tiempo de desarrollo de una aplicación web especialmente a la hora de mantener la página.

En el desarrollo de la investigación, se ha notado que el prototipo 1 realizado por la Metodología Tradicional en sus páginas de presentación mezclan código HTML con

código java, esto hace que la comprensión del funcionamiento sea muy difícil, como se muestra en la figura.

```
23 | </head>
24 | <body>
25 |   <center>
26 |     <h1 style="font-family: Cambria;color: black ">
27 |       <br>
28 |       <u>PLAN DE MANTENIMIENTO</u>
29 |       <br>
30 |       <br>
31 |     </h1>
32 |   </center>
33 |   <table align="center" width="100%" border="1">
34 |     <tr>
35 |       <th>ID</th>
36 |       <th>Estrategia</th>
37 |       <th>Equipo</th>
38 |       <th>Frecuencia</th>
39 |     <%
40 |         for (int i = 1; i <= 52; i++) {
41 |             out.print("<th>" + i + "</th>");
42 |         }
43 |     %>

```

Figura VI.129: Comprensión de código prototipo 1.

Se observa en la figura VI.129 como el código java es insertado en paginas HTML, esto hace muy difícil la comprensión de la estructuración de la pagina HTML.

No así la Metodología Propuesta en el cual se facilita la comprensión de código, ya que la lógica se la realiza en el backbean, como se muestra en la figura.

```
</p:dataTable>
<br></br>
<center>
<p:panelGrid columns="2">
  <f:facet name="header">
    RESULTADOS
  </f:facet>
  <h:outputText value="TOTAL DE HORAS EN EL AÑO :" style="font-weight: bold"/>
  <h:outputText value="#{planMantenimiento.totalPromedioPlanAnual()} horas " />
  <h:outputText value="PROMEDIO DE HORAS POR SEMANA:" style="font-weight: bold"/>
  <h:outputText value="#{planMantenimiento.promedioPlanAnual()} horas / Semana " />
</p:panelGrid>
</center>

```

```
355
356
357 | public double promedioPlanAnual()
358 | {
359 |     double resultado = 0.0 ;
360 |     for(int i=0; i<53; i++)
361 |     {
362 |         if(i!=0)
363 |             resultado = resultado + sumaVector1.get(i);
364 |     }
365 |     resultado = resultado/52;
366 |     return resultado;
367 | }
368
369 | public double totalPromedioPlanAnual()
370 | {
371 |     double resultado = 0.0 ;
372 |     for(int i=0; i<53; i++)
373 |     {
374 |         if(i!=0)
375 |             resultado = resultado + sumaVector1.get(i);
376 |     }
377 |     return resultado;
378 | }
379

```

Figura VI.130: Comprensión de código prototipo 2.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LX: Evaluación del Índice de Comprensión del Código.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 2 | 4 |
| PORCENTAJES | | 50% | 100% |

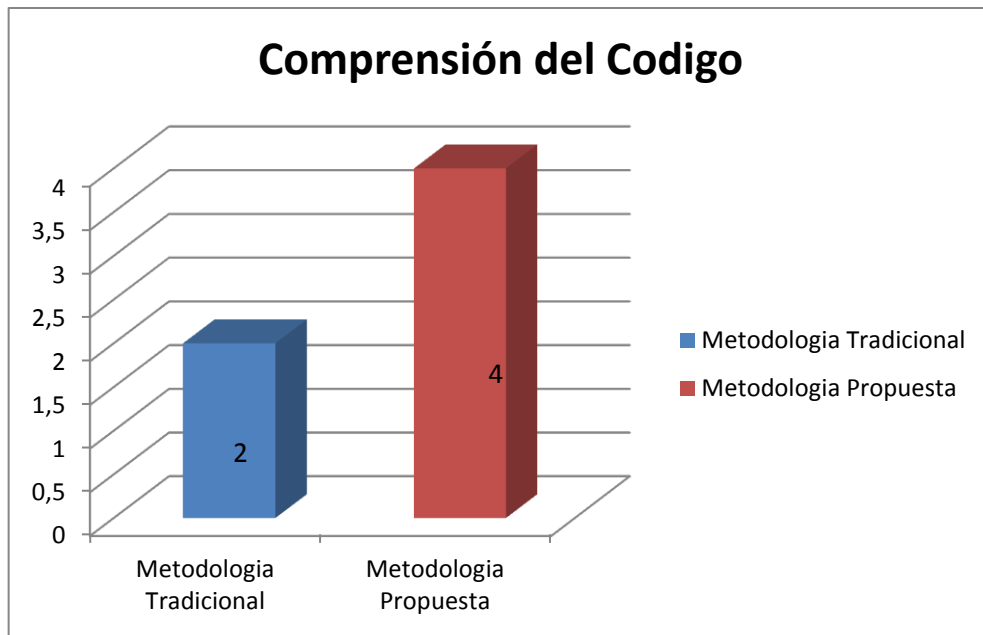


Figura VI.131: Índice de comparación – Comprensión del Código.

Como se observa en la figura VI.131 se ahorra un 50 % de tiempo más; a la hora de entender el código y realizar el mantenimiento de las aplicaciones web con el uso de la Metodología Propuesta.

Tabla VI.LXI: Índices - Parámetro Modularidad.

| INDICADORES | METODOLOGÍAS | TRADICIONAL | PROPUESTA | PESOS MÁXIMOS |
|-----------------------------|--------------|-------------|-----------|---------------|
| | | | | |
| Modelo Vista Controlador | | 1 | 4 | 4 |
| Diseño Estructurado | | 4 | 4 | 4 |
| Capacidad de Descomposición | | 2 | 4 | 4 |
| Comprensión del Código | | 2 | 4 | 4 |
| TOTAL | | 9 | 16 | 16 |
| PORCENTAJE DE PRESENTACIÓN | | 56,25% | 100% | 100% |

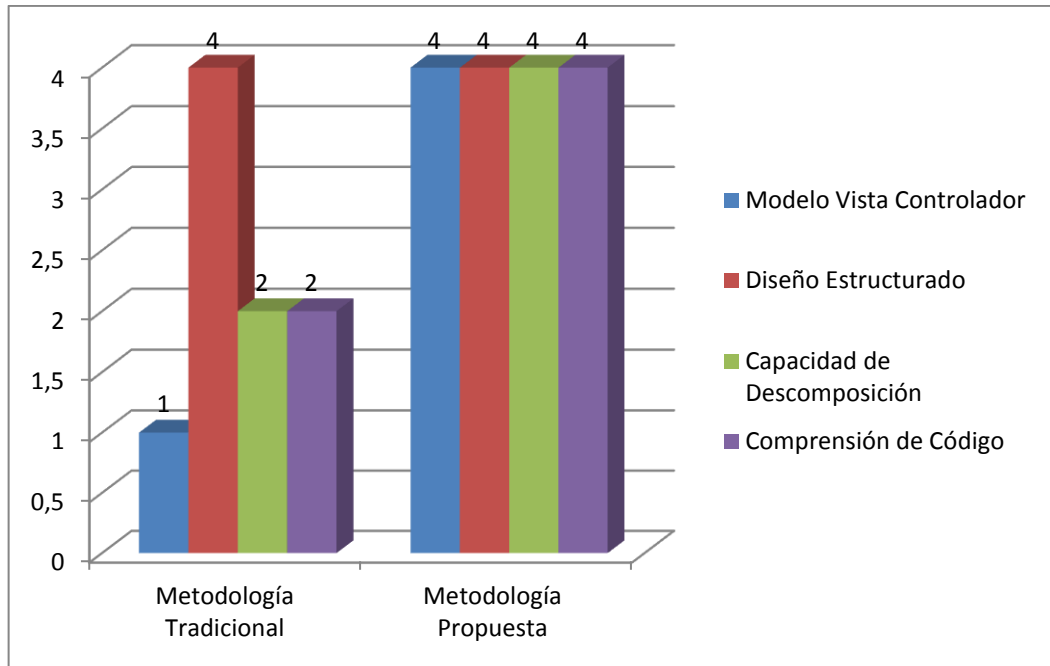


Figura VI.132: Índices de comparación - Modularidad.

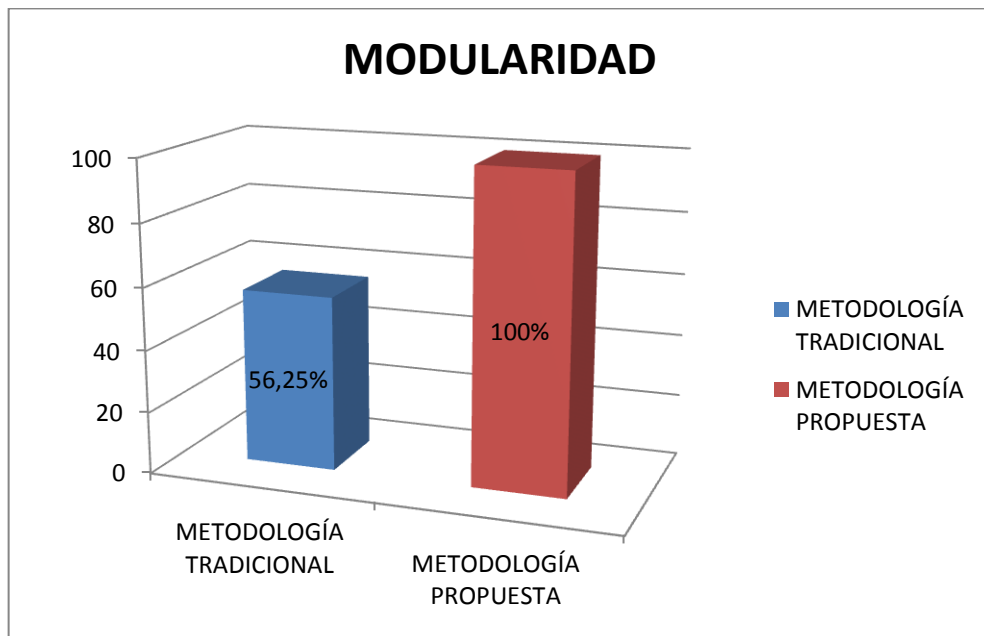


Figura VI.133: Porcentajes totales - Modularidad.

6.1.5.3.5. INTERPRETACIÓN.

Se puede observar en la figura VI.133 que el parámetro Modularidad tiene un 100% de optimización de tiempo, usando la metodología propuesta, permitiendo adaptar los beneficios que ofrece este parámetro, no así el prototipo que utiliza la Metodología Tradicional que ha calificado con la mitad aproximadamente de optimización de tiempo en el parámetro Modularidad.

6.1.5.4. *MANTENIBILIDAD.*

6.1.5.4.1. ESCALABILIDAD.

La escalabilidad es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para reaccionar y adaptarse sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

En el desarrollo de aplicaciones web es común cambios de requerimientos como también adaptación de nuevos requerimientos, es muy importante tener en cuenta este factor ya que si la aplicación es escalable se va ahorrar mucho tiempo en la construcción de los nuevos requerimientos.

Metodología Tradicional.

La metodología tradicional permite la escalabilidad de una manera no óptima ya que no tiene separado claramente la lógica de negocios con la lógica de presentación y lógica de acceso a datos. Para lo cual en la adaptación de nuevos requerimientos se debe realizar una re-estructuración de todo el código.

Metodología Propuesta.

La Metodología propuesta simplifica las cosas como ya que tiene separada claramente toda la lógica, de manera que es escalable fácilmente.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LXII: Evaluación del Índice de Escalabilidad.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 2 | 4 |
| PORCENTAJES | | 50% | 100% |

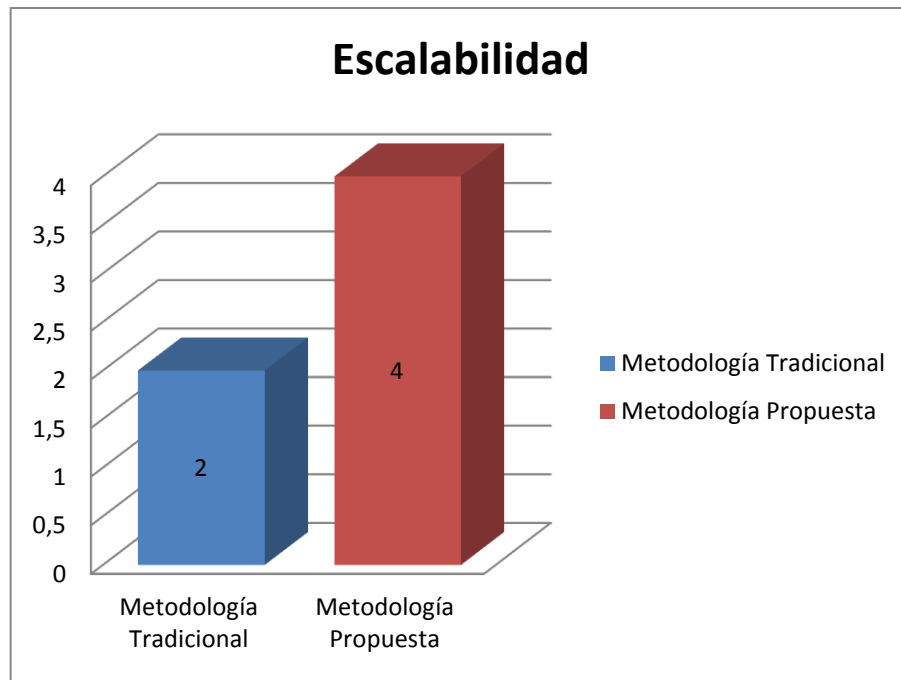


Figura VI.134: Índice de comparación Plantillas Personalizadas.

En la Figura VI.134, se observa los resultados obtenidos para el índice de Escalabilidad, en el cual el prototipo 1, tiene un porcentaje medio de escalabilidad frente al prototipo 2 que es desarrollada por la Metodología Propuesta.

6.1.5.4.2. DISPONIBILIDAD DE DOCUMENTACIÓN.

Mediante este indicador se sabrá la cantidad en porcentaje de tiempo ahorrado, a la hora de buscar información ya sea en foros, libros, wikis, etc, para la construcción o mantenibilidad de la aplicación web.

Se puede decir que utilizando la Metodología Tradicional se ahorra mucho tiempo a la hora de buscar información para la construcción de aplicaciones web, ya que esta utiliza técnicas estables y conocidas, lo cual hace que haya mucha disponibilidad de

documentación, no así en la Metodología Propuesta ya que utiliza técnicas nuevas y no se tendrá mucha documentación.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LXIII: Evaluación del Índice de Disponibilidad de Documentación.

| METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|-------------|-------------|
| | Prototipo 1 | Prototipo 2 |
| INDICADOR | | |
| CRITERIO DE EVALUACIÓN | 4 | 1 |
| PORCENTAJES | 100% | 25% |

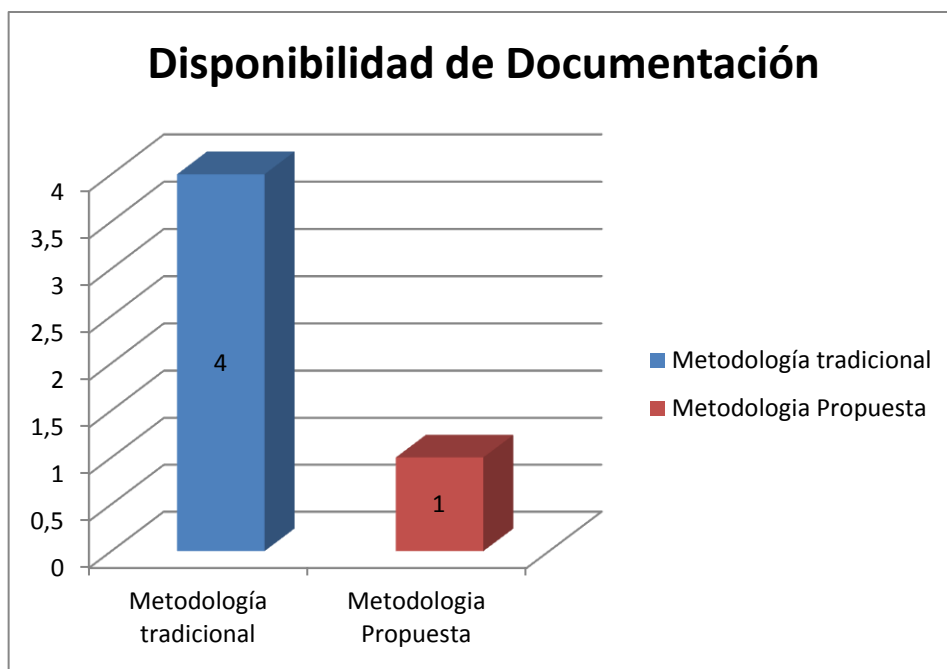


Figura VI.135: Índice de comparación Disponibilidad de Documentación.

En la figura VI.135 se observa que la Metodología Propuesta no ahorra tiempo a la hora de buscar documentación, a lo contrario de la Metodología Tradicional que ahorra un 100% de tiempo a la hora de buscar documentación para el desarrollo de aplicaciones web.

6.1.5.4.3. CÓDIGO ENTENDIBLE.

Es un índice muy importante que se toma en consideración principalmente a la hora de la mantenibilidad de las aplicaciones web. Mide que tan entendible es el código que se ha utilizado en la construcción de las aplicaciones web.

Utilizando la Metodología Tradicional los desarrolladores de software están obligados a escribir todo el código razón por la cual es más entendible a la hora de la mantenibilidad. Como se muestra en la figura de un ejemplo para llamar un procedimiento almacenado.

```
28
29 public int Create() {
30     int flag = 0;
31     try {
32         conexion obj2 = new conexion();
33         obj2.conectar();
34         obj2.setPrst(obj2.getConn().prepareStatement("{call insertar_equipo(?,?,?,?,?, ?, ?, ?, ?)}");
35         obj2.getPrst().setString(1, getDescripcion());
36         obj2.getPrst().setString(2, getCodigo());
37         obj2.getPrst().setString(3, getMarca());
38         obj2.getPrst().setString(4, getModelo());
39         obj2.getPrst().setString(5, getFabricante());
40         obj2.getPrst().setString(6, getDistribuidor());
41         obj2.getPrst().setInt(7, getAniofabricacion());
42         obj2.getPrst().setInt(8, getId_ubicacion());
43         obj2.getPrst().setString(9, getFicha());
44         obj2.getPrst().executeQuery();
45         obj2.desconectar();
46         flag = 1;
47     } catch (ClassNotFoundException ex) {
48         Logger.getLogger(Estrategia.class.getName()).log(Level.SEVERE, null, ex);
49     } catch (InstantiationException ex) {
50         Logger.getLogger(Estrategia.class.getName()).log(Level.SEVERE, null, ex);
51     } catch (IllegalAccessException ex) {
52         Logger.getLogger(Estrategia.class.getName()).log(Level.SEVERE, null, ex);
53     } catch (SQLException exsql) {
54         flag = -1;
55     }
56     return flag;
57 }
```

Figura VI.136: Código de Método Insertar Prototipo 1.

Utilizando la Metodología Propuesta la mayor parte de código es generado por frameworks, razón por la cual existe mayor código no entendible a la hora de la construcción y mantenibilidad de aplicaciones web.

Ejemplo en el Grafico se muestra código generado por frameworks.


```

*/
@Entity
@Table(name = "equipo")
@NamedQueries({
    @NamedQuery(name = "Equipo.findAll", query = "SELECT e FROM Equipo e"),
    @NamedQuery(name = "Equipo.findById", query = "SELECT e FROM Equipo e WHERE e.id = :id"),
    @NamedQuery(name = "Equipo.findByName", query = "SELECT e FROM Equipo e WHERE e.nombre = :nombre"),
    @NamedQuery(name = "Equipo.findByCodigo", query = "SELECT e FROM Equipo e WHERE e.codigo = :codigo"),
    @NamedQuery(name = "Equipo.findByUbicacion tecnicaId", query = "SELECT e FROM Equipo e WHERE e.ubicacion tecnicaId = :ubicacion tecnicaId")
})
public class Equipo implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "ID")
    private Long id;
    @Column(name = "NOMBRE")
    private String nombre;
    @Column(name = "CODIGO")
    private String codigo;
    @Column(name = "UBICACION TECNICA_ID")
    private BigInteger ubicacion tecnicaId;
}
    
```

Figura VI.137: Código Generado por Framework.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LXIV: Evaluación del Índice de Código Entendible.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 4 | 3 |
| PORCENTAJES | | 100% | 75% |

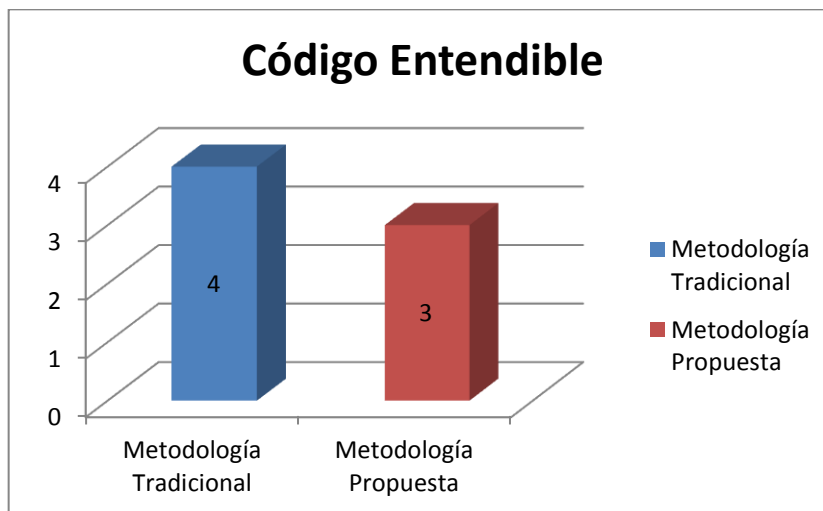


Figura VI.138: Índice de comparación Código Entendible.

En la figura VI.138 se observa que el prototipo realizado con la Metodología Tradicional tiene como porcentaje 100% , esto quiere decir que todo su código es entendible y favorece a la hora de re-estructurar algunos requerimientos, también se observa que el prototipo que utiliza la Metodología Propuesta tiene 75% pretendiendo decir que hace un poco difícil la comprensión del código.

6.1.5.4.4. ADAPTABILIDAD DE NUEVOS REQUERIMIENTOS.

Los requerimientos pueden dividirse en requerimientos funcionales y requerimientos no funcionales. Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas.

Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.

La capacidad de introducir nuevos requerimientos sin la dificultad a la hora de programar, por ende ganar tiempo y recursos.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LXV: Evaluación del Índice de Adaptabilidad de Nuevos Requerimientos.

| METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|-------------------------------|--------------------|--------------------|
| INDICADOR | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | 2 | 4 |
| PORCENTAJES | 50% | 100% |

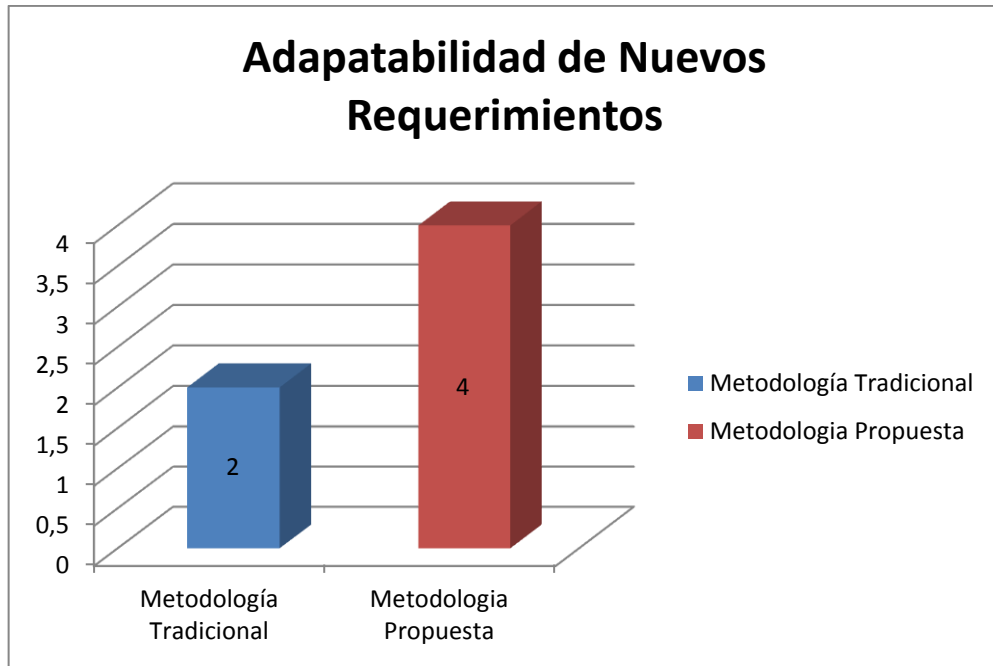


Figura VI.139: Índice de comparación Adaptabilidad de Nuevos Requerimientos.

Como se observa en la figura VI.139 el prototipo realizado con la Metodología propuesta ofrece el doble de productividad en lo que tiene que ver con el tiempo, en la adaptabilidad de Nuevos Requerimientos frente al prototipo realizado con la Metodología Tradicional.

Tabla VI.LXVI: Tabla general de índices - parámetro Mantenibilidad.

| METODOLOGÍAS | TRADICIONAL | PROPUESTA | PESOS MÁXIMOS |
|--|--------------------|------------------|----------------------|
| INDICADORES | | | |
| Escalabilidad | 2 | 4 | 4 |
| Disponibilidad de Documentación | 4 | 1 | 4 |
| Código Entendible | 4 | 3 | 4 |
| Adaptabilidad de nuevos requerimientos | 2 | 4 | 4 |
| TOTAL | 12 | 12 | 16 |
| PORCENTAJE DE PRESENTACIÓN | 75% | 75% | 100% |

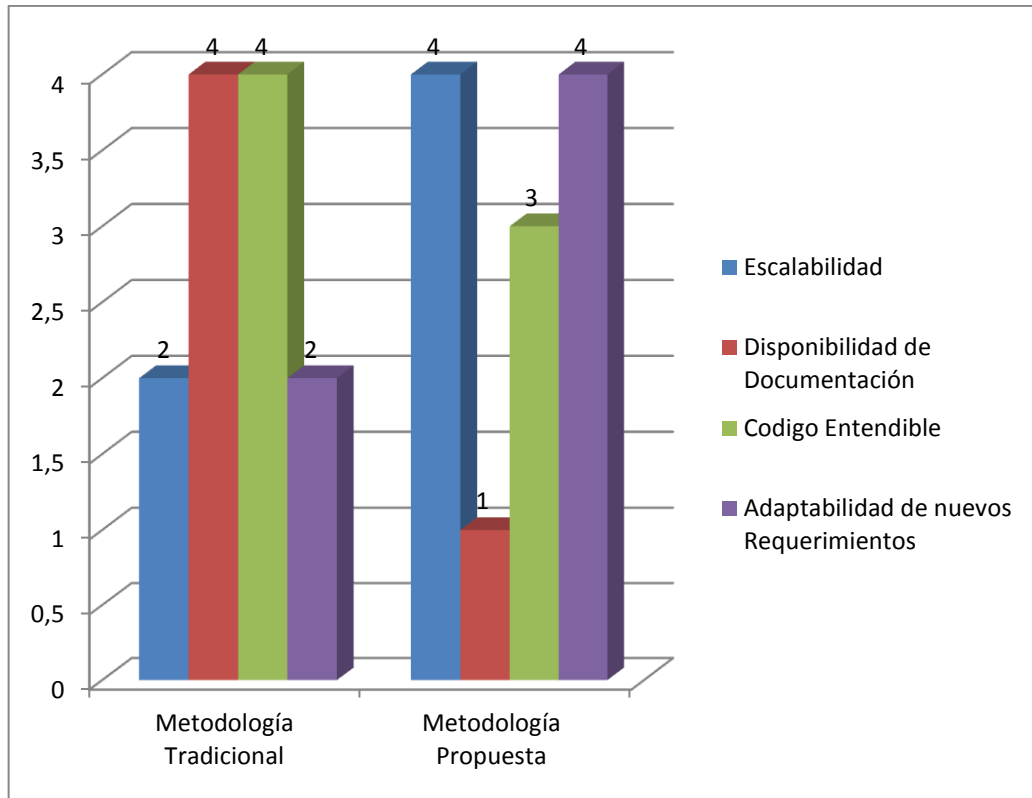


Figura VI.140: Índices de comparación - Mantenibilidad

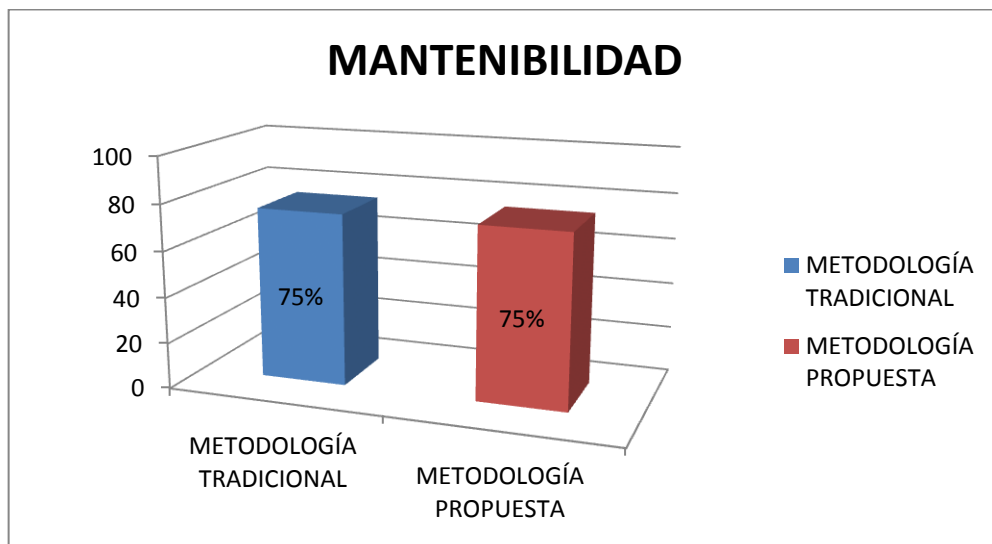


Figura VI.141: Porcentajes totales - Mantenibilidad.

6.1.5.4.5. INTERPRETACIÓN.

En este parámetro como es el de Mantenibilidad se observa en la figura VI.141 que tanto la Metodología Tradicional como la Metodología Propuesta tienen un mismo porcentaje de ahorro de tiempo en cuanto tiene que ver con la construcción y adaptación de nuevos requerimientos en una aplicación web.

Debido a los indicadores de: “Documentación Disponible” y “Código entendible”, la Metodología Tradicional ofrece varias ventajas frente a la Metodología Propuesta, sin embargo los indicadores de: “Escalabilidad” y “Adaptación de Nuevos Requerimientos” trae desventajas frente a la Metodología Propuesta.

6.1.5.5. *PRESENTACIÓN.*

6.1.5.5.1. TEMPLATES PERSONALIZADOS.

El uso de templating trae consigo las ventajas de tener un código ordenado (sobre todo en aplicaciones grandes), fácil de desarrollar y reusar. Se puede crear componentes o templates que abarcan desde simplemente mostrar texto hasta crear componentes que muestren un checkbox, un input o un dropdown dependiendo de la data del backing bean.

En la metodología tradicional se desarrollan los Templates mediante hojas de estilo, no así en la metodología propuesta que ofrece Templates personalizados que benefician en el tiempo a la hora de la creación de los Templates.

Templates en Metodología Tradicional

La metodología tradicional para el uso de de Templates hace uso de un sin número de técnicas tales como son jquery, javascript, frames, etc.

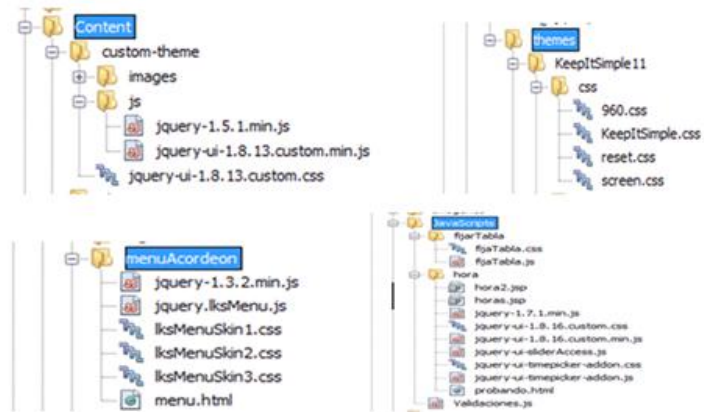


Figura VI.142: Hojas de Estilo prototipo 1.

```
6
7 <!--page import="java.sql.SQLException" -->
8 <!--page contentType="text/html" pageEncoding="UTF-8" -->
9 <!--page import="general.conexion" -->
10 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
11 "http://www.w3.org/TR/html4/loose.dtd">
12
13 <script src="../JavaScripts/Validaciones.js" type="text/javascript"></script>
14
15 <link href="../Content/custom-theme/jquery-ui-1.8.13.custom.css" rel="stylesheet" type="text/css" />
16 <script src="../Content/custom-theme/js/jquery-1.5.1.min.js" type="text/javascript"></script>
17 <script src="../Content/custom-theme/js/jquery-ui-1.8.13.custom.min.js" type="text/javascript"></script>
18 <link href="../styles/itunesTabla.css" rel="stylesheet" type="text/css" />
19
20 <link href="../JavaScripts/hora/jquery-ui-1.8.16.custom.css" rel="stylesheet" type="text/css" media="all" />
21 <script src="../JavaScripts/hora/jquery-1.7.1.min.js" type="text/javascript"></script>
22 <script src="../JavaScripts/hora/jquery-ui-1.8.16.custom.min.js" type="text/javascript"></script>
23 <script src="../JavaScripts/hora/jquery-ui-timepicker-addon.js" type="text/javascript"></script>
24 <script src="../JavaScripts/hora/jquery-ui-sliderAccess.js" type="text/javascript"></script>
25 <link href="../JavaScripts/hora/jquery-ui-timepicker-addon.css" rel="stylesheet" type="text/css" media="all" />
26
27
143
144
145 <html>
146 <head>
147 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
148 <link href="../styles/grid.css" rel="stylesheet" type="text/css" />
149 <link rel="stylesheet" type="text/css" href="../styles/style3.css" media="screen" />
150 <link href="../styles/body.css" rel="stylesheet" type="text/css" />
151 <link href="../styles/itunesTabla.css" rel="stylesheet" type="text/css" />
152 <title>Ingresar Equipo</title>
153 </head>
```

Figura VI.143: Referencias - Hojas de Estilo prototipo 1.

Templates en Metodología Propuesta.

La Metodología propuesta simplifica las cosas como se muestra a continuación.

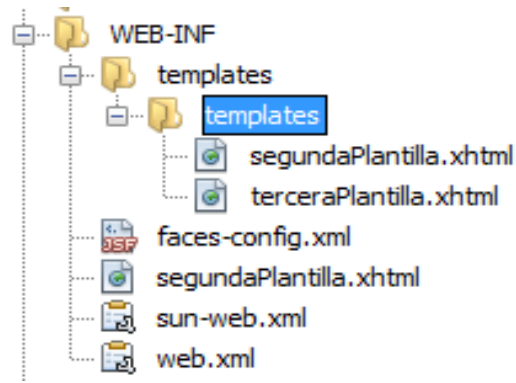


Figura VI.144: Templates prototipo 2.

```
17 <h:body>  
18 <ui:composition template="/WEB-INF/templates/templates/terceraPlantilla.xhtml">  
19 <ui:define name="content">  
20 <f:view>  
21 <h:form>
```

Figura VI.145: Referencias - Templates prototipo 2.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LXVII: Evaluación del Índice de Templates Personalizados.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 3 | 4 |
| PORCENTAJES | | 75% | 100% |

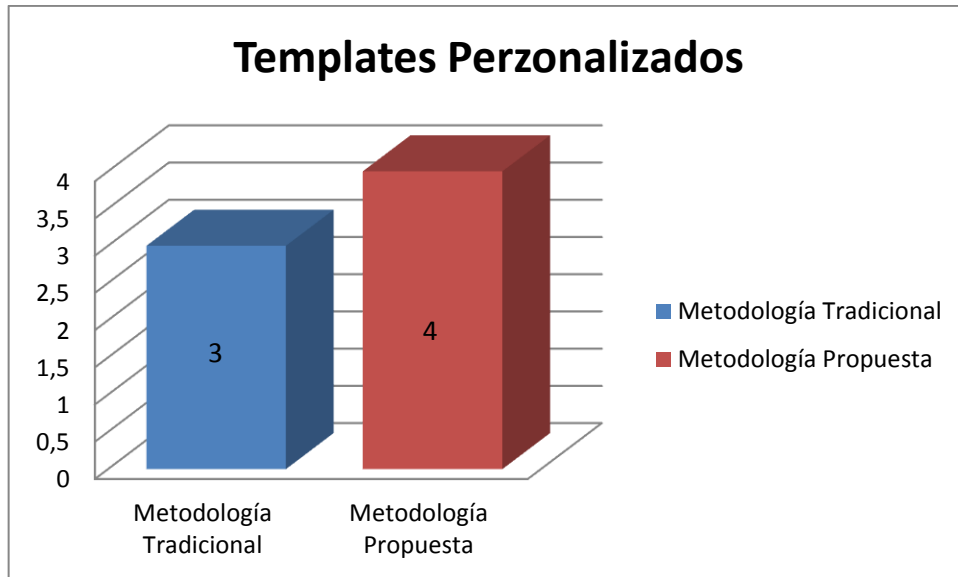


Figura VI.146: Índice de comparación Templates Personalizados.

En la Figura VI.146, se observa los resultados obtenidos para el índice de Templates Personalizados, en el cual el prototipo 2, que utiliza la metodología propuesta, con el valor de 4 toma ventaja sobre el prototipo 1, el cual utiliza la metodología tradicional, esto significa que utilizando los Templates personalizados que ofrece la metodología propuesta se obtiene mayor ventaja en el ahorro de tiempo en la construcción de los mismos.

6.1.5.5.2. COMPONENTES AJAX.

Es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax permite cargar la página en segundo plano sin actualizar la página completa y la presentación. Ajax es un script simple JavaScript que trabajar con XML petición HTTP.

Ajax en Metodología Tradicional (JSP).

Para realizar componentes ajax con jsp, se debe implementar un sin número de clases y métodos, haciendo así a la aplicación más demorosa, como se muestra en las figuras siguientes.

Index.jsp

```
1 <%@page contentType="text/html" pageEncoding="UTF-8" language="java" %>
2 <%@page import="java.util.*" %>
3 <%@page import="ajax.logic.productoBo" %>
4 <%@page import="ajax.bean.productoBean" %>
5 <html>
6 <head><title>AJAX DEMO</title></head>
7 <body>
8 <table border="0" align="center">
9 <tr>
10 <td>PRODUCTOS: </td>
11 <td>
12 <select name="productos">
13 <%
14     int id_pro;
15     String desc_pro = null;
16     ArrayList lst_pro = new ArrayList();
17     productoBo pro_bo = new productoBo();
18     lst_pro = pro_bo.getProductos();
19     if(lst_pro.size() > 0){
20         for(int i = 0; i<lst_pro.size(); i++){
21             id_pro = ((productoBean)lst_pro.get(i)).getId_pro();
22             desc_pro = ((productoBean)lst_pro.get(i)).getDesc_pro();
23             out.write("<option value='" + id_pro + "'> " + desc_pro + "</option>");
24         }
25     }
26 <%>
27 </select>
28 </td>
29 </tr>
```

Figura VI.147: Ajax en JSP.

```
1 var xmlhttp
2
3 function getDetalleProducto(id_pro){
4     xmlhttp=new XMLHttpRequest();
5     if (xmlhttp==null){
6         alert ("Tu navegador no soporta AJAX!");
7         return;
8     }
9     today = new Date();
10    id= Math.abs(Math.sin(today.getTime()));
11    var url = "ajax_detalleProducto.jsp";
12    url = url + "?id_pro=" + id_pro;
13    url = url + "&id=" + id;
14    xmlhttp.onreadystatechange = resultado_detalle;
15    xmlhttp.open("GET",url,true);
16    xmlhttp.send(null);
17 }
18
19 function resultado_detalle(){
20     if (xmlhttp.readyState==4){
21         document.getElementById("result_detalle").innerHTML=xmlhttp.responseText;
22     }
23 }
```

Figura VI.148: Función AJAX.

```
1 <!--
2 Document : ajax_detalleProducto
3 Created on : 31/03/2009, 11:28:51 AM
4 Author : nachxs
5 -->
6
7 <@page contentType="text/html" pageEncoding="UTF-8" language="java" %>
8 <@page import="java.util.*" %>
9 <@page import="ajax.logic.productoBo"%>
10 <@page import="ajax.bean.detalleProductoBean"%>
11 <%
12 String id_pro = null;
13 int id_det_pro;
14 String detalle = null;
15 ArrayList lst_detpro = new ArrayList();
16 productoBo pro_bo = new productoBo();
17 detalleProductoBean detpro_bean = new detalleProductoBean();
18 id_pro = request.getParameter("id_pro");
19 lst_detpro = pro_bo.getDetalleProducto(Integer.parseInt(id_pro));
20 if(lst_detpro.size()>0){
21 out.write("<select name='detalle_producto'>");
22 for(int i=0; i<lst_detpro.size(); i++){
23 id_det_pro = ((detalleProductoBean)lst_detpro.get(i)).getId_det_pro();
24 detalle = ((detalleProductoBean)lst_detpro.get(i)).getDetalle();
25 out.write("<option value=" + id_det_pro + ">" + detalle + "</option>");
26 }
27 out.write("</select>");
28 }
29 %>
```

Figura VI.149: AJAX HTML.

AJAX en Metodología Propuesta (JSF con PrimeFaces)

Para implementar AJAX con JSF, se hace más fácil simplemente se utiliza pocas líneas de código, como se muestra en las siguientes figuras.


Técnico.xhtml

```
<h3> Inicio de Actividad: </h3>
<p:calendar value="#{equipoEstrategia.fecha}" pattern="yyyy-MM-dd" showOn="button">
<p:ajax event="click" update="anio, inicioSemana" listener="#{equipoEstrategia.calcular}>
</p:calendar>
```

Figura VI.150: AJAX XHTML prototipo 2.

Backbean – equipoEstrategia.calcular

```
public void calcular()  
{  
    Calendar nuevoCalendario = Calendar.getInstance();  
    nuevoCalendario.setTime(fecha);  
    valor = nuevoCalendario.get(Calendar.WEEK_OF_YEAR);  
    anio = nuevoCalendario.get(Calendar.YEAR);  
}
```



Inicio de Actividad: 2012-10-05
Año: 2012
Inicio de Semana: 40
ASIGNAR CANCELAR

Figura VI.151: AJAX PRIMEFACES.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LXVIII: Evaluación del Índice de Componentes AJAX.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 1 | 3 |
| PORCENTAJE | | 25% | 75% |

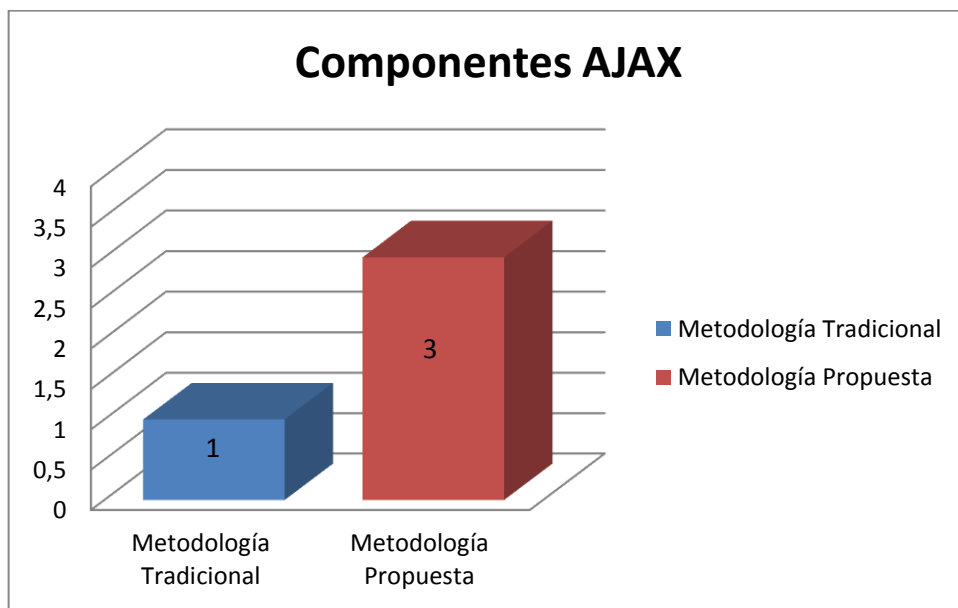


Figura VI.152: Índice de comparación Componentes AJAX.

En la Figura VI.152, se observa los resultados obtenidos para el índice de Componentes AJAX, se puede concluir diciendo que ambas metodologías utilizan componentes AJAX, sin embargo, la metodología tradicional hace se uso de forma más extensa y difícil, no así la metodología propuesta en la cual ahorra mucho tiempo para el desarrollo de componentes AJAX.

6.1.5.5.3. VALIDACIONES.

La validación se define como: El proceso de comprobación de si algo satisface un cierto criterio. Los ejemplos incluyen la comprobación de si una afirmación es verdadera (la validez), si un aparato funciona como está previsto, si un sistema es seguro, o si los datos informáticos son compatibles con un estándar abierto. La validación implica un documento, puede que la solución o el proceso es correcto o es adecuado para su uso.

La validación de HTML y CSS es un proceso que no es impuesto sobre un diseñador / desarrollador. Sin embargo, se realiza con el fin de asegurar la compatibilidad y la estabilidad de una página web a través de los distintos navegadores. El defensor número de la validación es el W3C (World Wide Web Consortium). La validación es la mejor manera de evitar diversas irregularidades en la prestación de una página web.

Validaciones Metodología Tradicional.

Se utiliza validaciones usando javascript, no tiene mensajes de error personalizados.

```
79 <input type="text" value="" on="procesaIngresarEstrategia.jsp" method="get" onsubmit="return validate()" enctype="MULTIPART/FORM-DATA">
```

```
37 function SoloDecimal1(texto)
38 {
39     var nums = "0123456789.";
40     cadena = texto.value;
41     numCars = cadena.length;
42     ultimo = cadena.charAt(numCars-1);
43     if (nums.indexOf(ultimo) < 0)
44     {
45         texto.value = cadena.substr(0,numCars-1);
46     }
47 }
```



Figura VI.153: Validaciones prototipo 1.

Validaciones Metodología Propuesta.

Posee validaciones personalizadas y mensajes de error personalizados.

```
<h:outputLabel value="Actividad * :." for="actividad" />
<p:inputText id="actividad" value="#{backBeanEstrategia1.tablaestrategia.actividad}" title="Actividad"
size="30" required="true" requiredMessage="El campo actividad es obligatorio para continuar."/>
```

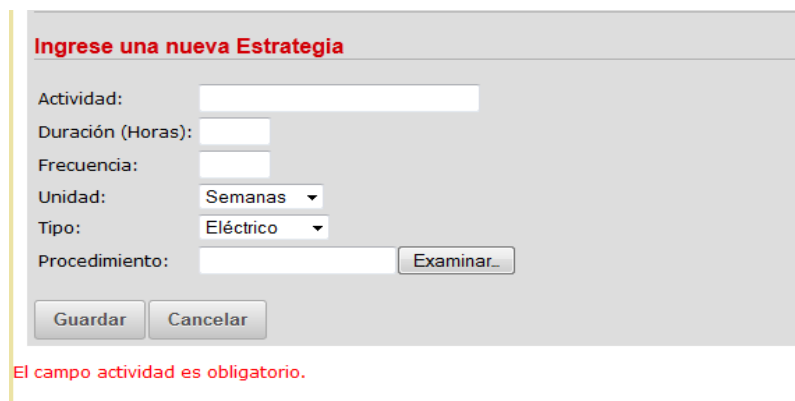


Figura VI.154: Validaciones prototipo 2.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LXIX: Evaluación del Índice de Validaciones.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 3 | 4 |
| PORCENTAJE | | 75% | 100% |

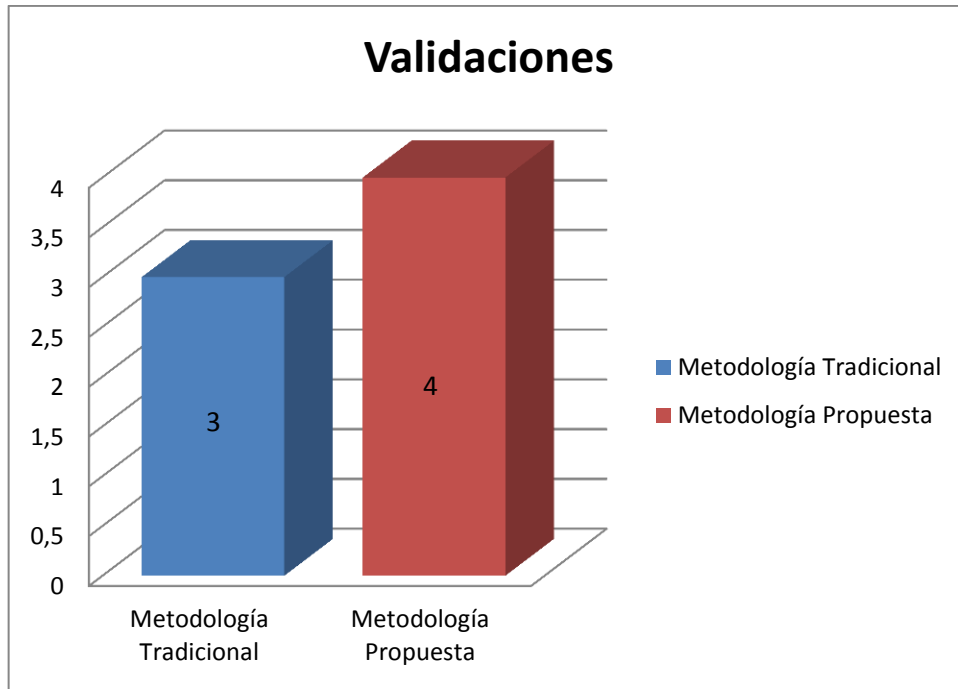


Figura VI.155: Índice de comparación Validaciones.

En la Figura VI.155, se observa los resultados obtenidos para el índice de Validaciones, se puede concluir diciendo que tanto la metodología tradicional como la metodología propuesta utilizan validaciones, pero aquí la metodología propuesta supera por poco a la metodología tradicional, ya que trae incluidos mensajes personalizados de error.

6.1.5.5.4. MENOR USO DE JAVA SCRIPT.

JavaScript, al igual que Flash, Visual Basic Script, es una de las múltiples maneras que han surgido para extender las capacidades del lenguaje HTML (lenguaje para el diseño de páginas de Internet). Al ser la más sencilla, es por el momento la más extendida. JavaScript no es un lenguaje de programación propiamente. Es un lenguaje script u orientado a documento, como pueden ser los lenguajes de macros que tienen muchos procesadores de texto y planillas de cálculo. No se puede desarrollar un programa con JavaScript que se ejecute fuera de un Navegador.

JavaScript es un lenguaje interpretado que se embebe en una página web HTML. Un lenguaje interpretado significa que a las instrucciones las analiza y procesa el navegador en el momento que deben ser ejecutadas.

Al utilizar JavaScript para realizar validaciones o para ejecutar menús desplegables, datepickers hace que el desarrollo de la aplicación sea mayor en tiempo; en cuanto tiene que ver en el mantenimiento y desarrollo de la aplicación web.

Es decir al utilizar menos java script en el desarrollo de una aplicación web, será más beneficioso en tiempo de desarrollo y construcción.

Java Script en Metodología Tradicional

```
29 <script type="text/javascript">
30 $(function () {
31     // Datepicker
32     $('#datepicker').datepicker({
33         inline: true
34     });
35 });
36
37
38 $(function () {
39     // Datepicker
40     $('#datepicker2').datepicker({
41         inline: true
42     });
43 });
44
45 $(function () {
46     $('.example-container > pre').each(function(i) {
47         eval($(this).text());
48     });
49 });
50
51 function validacion()
52 {
53     var horaInicio = document.getElementById("example3").value;
54     var fecha = document.getElementById("datepicker").value;
55
56     if ((horaInicio.toString().length == 0) || (fecha.toString().length == 0))
57     {
58         alert("debe ingresar la fecha y hora Inicial");
59         return false;
60     }
61     return true;
```

The screenshot shows a web form titled "INGRESAR FALLAS" on a blue textured background. The form has a light green sidebar on the left with labels for "Descripcion Equipo:", "Novedad:", "Fecha Inicio (*):", "Hora Inicio (*):", "Fecha Fin:", "Hora Fin:", and "Tiempo de Falla:". The "Descripcion Equipo:" field contains a dropdown menu with "nuevo equipo - 11" selected. The "Fecha Inicio (*):" field is empty. A calendar widget is displayed over the form, showing "October 2012" with the date "4" highlighted in yellow. The calendar grid shows days from Sunday to Saturday, with the 31st also highlighted.

Figura VI.156: JavaScript prototipo 1.

Java Script en Metodología Propuesta.

```
<h3> Inicio de Actividad: </h3>  
<p:calendar value="#{equipoEstrategia.fecha}" pattern="yyyy-MM-dd" showOn="button">
```

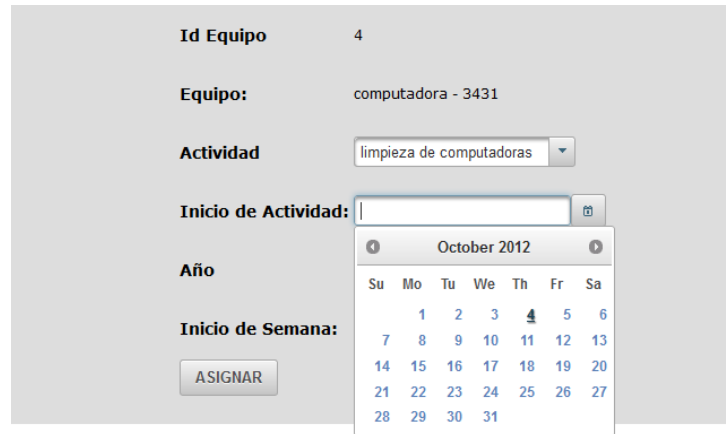


Figura VI.157: JavaScript prototipo 2.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla VI.LXX: Evaluación del Índice de Menor Uso de JavaScript.

| INDICADOR | METODOLOGÍAS | TRADICIONAL | PROPUESTA |
|------------------------|--------------|-------------|-------------|
| | | Prototipo 1 | Prototipo 2 |
| CRITERIO DE EVALUACIÓN | | 2 | 4 |
| PORCENTAJE | | 50% | 100% |

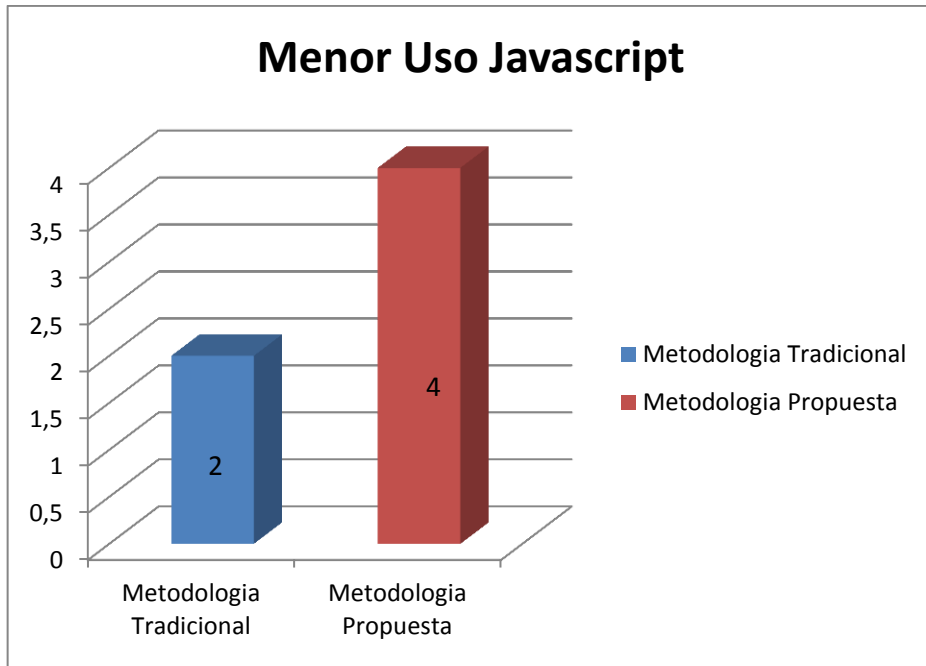


Figura VI.158: Índice de comparación Menor Uso de Javascript.

En la figura VI.158, se observa los resultados obtenidos para el índice de Menor JavaScript, se puede notar que la Metodología Propuesta es más óptima en cuanto tiene que ver con este índice ya que hace un uso casi nulo de Javascript y no así la Metodología tradicional que usa Javascript para realizar muchas de sus acciones.

Tabla VI.LXXI: Tabla general de índices - parámetro Presentación

| METODOLOGÍAS | TRADICIONAL | PROPUESTA | PESOS MÁXIMOS |
|--------------------------|--------------------|------------------|----------------------|
| INDICADORES | | | |
| Templates personalizados | 3 | 4 | 4 |
| Componentes AJAX | 1 | 3 | 4 |
| Validaciones | 3 | 4 | 4 |
| Menor Uso De Java Script | 2 | 4 | 4 |
| TOTAL | 9 | 15 | 16 |
| PORCENTAJE | 56,25% | 93,75% | 100% |

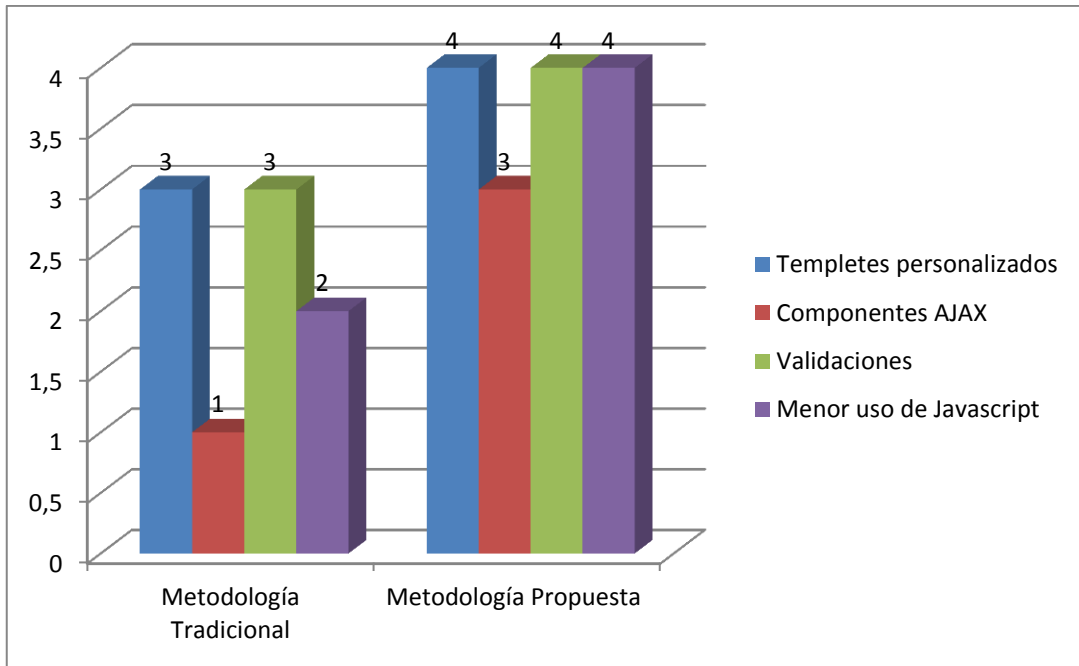


Figura VI.159: Índices de comparación -Presentación.

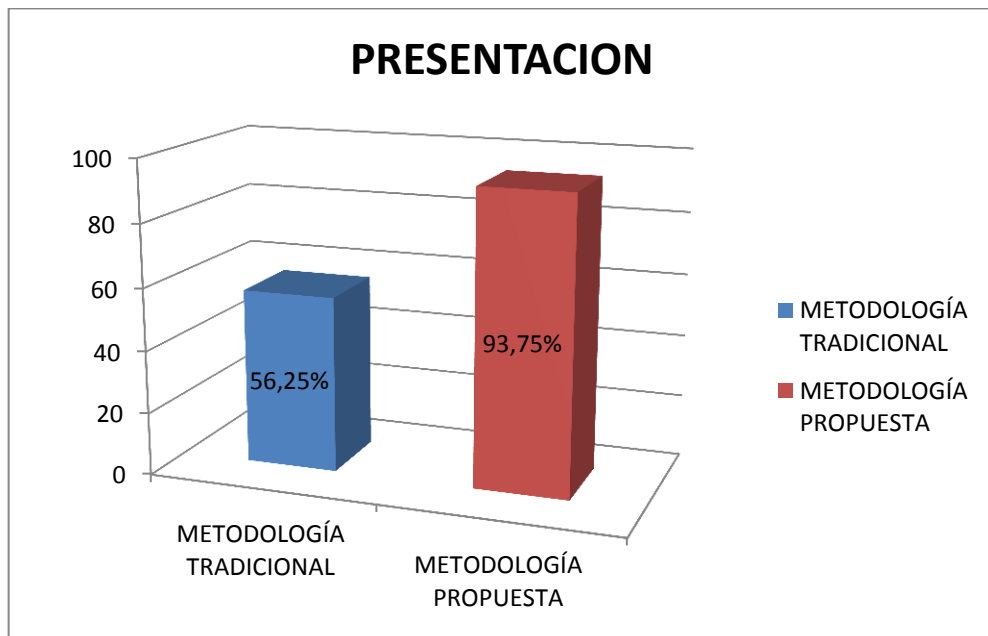


Figura VI.160: Porcentajes Totales - Presentación.

6.1.5.5. INTERPRETACIÓN.

En consecuencia de la valoración de los diferentes índices para este parámetro se tiene como resultado que el Prototipo que utiliza la Metodología Tradicional ha alcanzado un porcentaje de 56,25% y que el Prototipo que utiliza la Metodología Propuesta ha alcanzado un porcentaje del 93,75%, esto quiere decir que utilizando la Metodología Propuesta se optimiza tiempo en cuanto tiene que ver con el parámetro de Presentación para el desarrollo y tiempo de mantenibilidad de páginas web.

6.1.6. ANÁLISIS GENERAL DE COMPROBACIÓN.

6.1.6.1. HIPÓTESIS.

Mediante el desarrollo de una metodología para la construcción de una aplicación web a través de la integración de las tecnologías JSF y EJB se minimizará el tiempo en la construcción y mantenimiento de aplicaciones web.

6.1.6.1.1. DETERMINACIÓN DE VARIABLES.

Variable Independiente. Metodología para la construcción de una aplicación web a través de la integración de las tecnologías JSF y EJB.

Variable Dependiente. Tiempo en la construcción y mantenimiento de aplicaciones web.

6.1.6.1.2. OPERACIONALIZACIÓN DE LAS VARIABLES.

Operacionalización Conceptual.

Tabla VI.LXXII: Operacionalización Conceptual.

| VARIABLE | TIPO | CONCEPTO |
|---|---------------|--|
| Metodología para la construcción de una aplicación web a través de la integración de las tecnologías JSF y EJB. | Independiente | Conjunto de etapas formalmente estructuradas, de manera que brinden a los interesados parámetros de acción en el desarrollo de sus proyectos: plan general |

| | | |
|--|-------------|---|
| | | y detallado, tareas y acciones, tiempos, aseguramiento de la calidad, involucrados, etapas, revisiones de avance, responsables, recursos requeridos, entre otros. |
| Tiempo en la construcción y mantenimiento de aplicaciones web. | Dependiente | Es el periodo o lapso de tiempo utilizado desde que se inicia el desarrollo de la aplicación hasta su fin. |

Operacionalización Metodológica.

Tabla VI.LXXIII: Operacionalización Metodológica.

| VARIABLE | CATEGORÍA | INDICADORES | TÉCNICAS | FUENTES DE VERIFICACIÓN |
|---|---------------|--|------------------------|--|
| Metodología para la construcción de una aplicación web a través de la integración de las tecnologías JSF y EJB. | Investigación | Metodología para el desarrollo de software | Revisión de Documentos | <ul style="list-style-type: none"> • Internet. • Libros. • Manuales. • Tutoriales. |
| Tiempo en la construcción y mantenimiento de aplicaciones web. | Tiempo | Reusabilidad. | Observación | <ul style="list-style-type: none"> • Prototipo 1 • Prototipo 2 |
| | | Generación de Código. | Observación | <ul style="list-style-type: none"> • Prototipo 1 • Prototipo 2 |
| | | Modularidad. | Observación | <ul style="list-style-type: none"> • Prototipo 1 • Prototipo 2 |
| | | Mantenibilidad. | Observación | <ul style="list-style-type: none"> • Prototipo 1 |

| | | | | |
|--|--|---------------|-------------|--|
| | | | | <ul style="list-style-type: none"> • Prototipo 2 |
| | | Presentación. | Observación | <ul style="list-style-type: none"> • Prototipo 1 • Prototipo 2 |

6.1.6.2. COMPARACIÓN DE LOS RESULTADOS OBTENIDOS.

Para la comprobación de hipótesis se utilizará el Método Estadístico Chi cuadrado, se trabajará con los porcentajes, valores observados:

De acuerdo a la evaluación realizada por cada parámetro se establecieron valores que permiten determinar la minimización del tiempo en la construcción y mantenimiento de aplicaciones web. En la tabla siguiente se muestra los valores obtenidos.

Tabla VI.LXXIV: Paramaros de Optimización de tiempo y cuantificación

| | METODOLOGÍAS | |
|-------------------------------|---------------------|------------------|
| OPTIMIZACIÓN DE TIEMPO | TRADICIONAL | PROPUESTA |
| REUSABILIDAD | 58,33 | 91,67 |
| GENERACIÓN DE CÓDIGO | 55 | 55 |
| MODULARIDAD | 56,25 | 100 |
| MANTENIBILIDAD | 75 | 75 |
| PRESENTACIÓN | 56,25 | 93,75 |

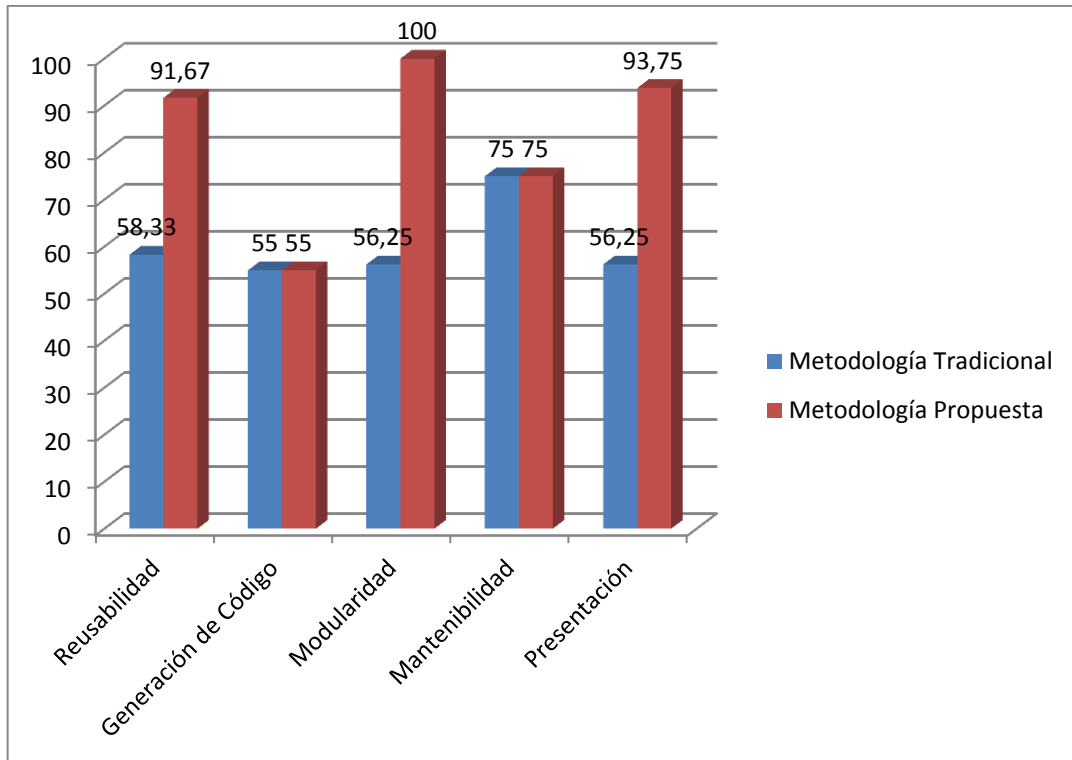


Figura VI.161: Tabla General de Valores.

La figura VI.161 representa los valores obtenidos en cada parámetro de cada metodología. Mientras más alto sea el valor del parámetro de la metodología, se establece que es más óptima para minimizar el tiempo de desarrollo y construcción de aplicaciones web.

Usando la Metodología Propuesta se obtiene mayor Reusabilidad, Modularidad y Presentación de interfaces, permitiendo de esta manera minimizar el tiempo en la construcción y mantenimiento de aplicaciones web.

Entonces, con respecto a la metodología tradicional, la metodología propuesta permite minimizar el tiempo en la construcción y mantenimiento de aplicaciones web en un 21,25% más, así como se observa en la figura:

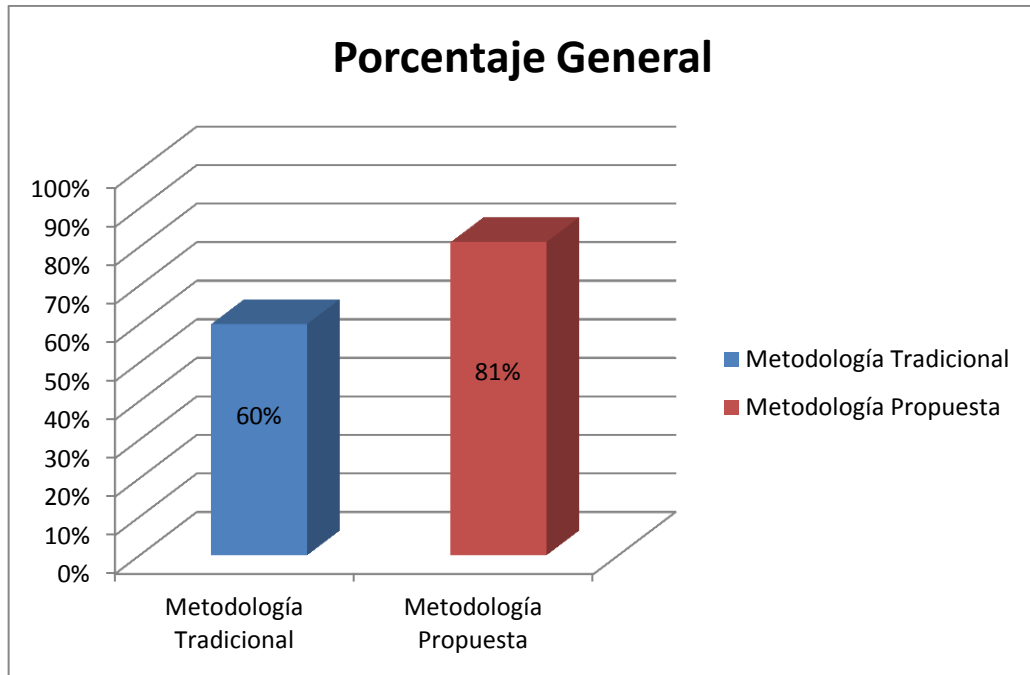


Figura VI.162: Porcentaje General.

La figura VI.162 refleja que se optimiza el tiempo en un 21% utilizando la Metodología Propuesta ERCA Web que usa tecnologías como JSF y EJB, con respecto a la Metodología Tradicional MSF que utiliza tecnologías como JSP y JDBC.

6.1.6.2.1. APLICACIÓN DE CHI CUADRADO.

Planteamiento de la hipótesis:

H₀: Las Metodología Propuesta NO optimizara el tiempo en la construcción de aplicaciones web mediante la Integración JSF y EJB.

H₁: Las Metodología Propuesta optimizara el tiempo en la construcción de aplicaciones web mediante la Integración JSF y EJB.

Encontrar los totales por cada fila y por cada columna:

Tabla VI.LXXV: Valores Observados y Totales.

| | METODOLOGÍAS | TOTAL |
|--|--------------|-------|
|--|--------------|-------|

| OPTIMIZACIÓN DE TIEMPO | DE | TRADICIONAL | PROPUESTA | FILAS |
|-------------------------------|-----------|--------------------|------------------|---------------|
| REUSABILIDAD | | 58,33 | 91,67 | 150 |
| GENERACIÓN DE CÓDIGO | | 55 | 55 | 110 |
| MODULARIDAD | | 56,25 | 100 | 156,25 |
| MANTENIBILIDAD | | 75 | 75 | 150 |
| PRESENTACIÓN | | 56,25 | 93,75 | 150 |
| TOTAL COLUMNAS | | 300,83 | 415,42 | 716,25 |

Valores esperados

De la tabla VI.LXXV, que son los valores observados, se encuentra los valores esperados, con la siguiente formula

$$Valor Esperado = \frac{Total Columna(para dicha celda) \times Total fila(para dicha fila)}{Suma Total}$$

Ejemplo:

$$Valor Esperado(1,1) = \frac{300,83 \times 150}{716,25}$$

$$Valor Esperado (1,1) = 63,00$$

Obteniendo como resultado los siguientes valores esperados:

Tabla VI.LXXVI: Valores Esperados y Totales.

| OPTIMIZACIÓN DE TIEMPO | METODOLOGÍAS | | TOTAL FILAS |
|-------------------------------|---------------------|------------------|--------------------|
| | TRADICIONAL | PROPUESTA | |
| REUSABILIDAD | 63,00 | 87,00 | 150 |
| GENERACIÓN DE CÓDIGO | 46,20 | 63,80 | 110 |
| MODULARIDAD | 65,63 | 90,62 | 156,25 |

| | | | |
|-----------------------|--------|--------|--------|
| MANTENIBILIDAD | 63,00 | 87,00 | 150 |
| PRESENTACIÓN | 63,00 | 87,00 | 150 |
| TOTAL COLUMNAS | 300,83 | 415,42 | 716,25 |

Se obtiene el valor de Chi cuadrado mediante la fórmula:

$$X^2_{calc} = \sum \frac{(f_o - f_e)^2}{f_e}$$

f_o = Frecuencia del valor observado.

f_e = Frecuencia del valor esperado.

Tabla VI.LXXVII: Resumen Valores Observados y valores Esperados.

| | Valores Observados | | Valores Esperados | |
|----------------------|--------------------|-----------------|-------------------|-----------------|
| | M. Tradicional | M. Propuesta | M. Tradicional | M. Propuesta |
| Reusabilidad | 58,33 | 91,67 | 63,00 | 87,00 |
| Generación de Código | 55 | 55 | 46,20 | 63,80 |
| Modularidad | 56,25 | 100 | 65,63 | 90,62 |
| Mantenibilidad | 75 | 75 | 63,00 | 87,00 |
| Presentación | 56,25 | 93,75 | 63,00 | 87,00 |

$$X^2_{calc} = \frac{(58,33-63,00)^2}{63,00} + \frac{(91,67-87,00)^2}{87,00} + \frac{(55-46,20)^2}{46,20} + \frac{(55-63,80)^2}{63,80} + \frac{(56,25-65,63)^2}{65,63} + \frac{(100-90,62)^2}{90,62} + \frac{(75-63,00)^2}{63,00} + \frac{(75-87,00)^2}{87,00} + \frac{(56,25-63,00)^2}{63,00} + \frac{(93,75-87,00)^2}{87,00}$$

$$X^2_{calc} = 0,35 + 0,25 + 1,68 + 1,21 + 1,34 + 0,97 + 2,29 + 1,66 + 0,72 + 0,52$$

$$X^2_{calc} = 10,99$$

Chi cuadrado experimental = 10,99

Grados De Libertad

Para calcular los grados de libertad se realiza con la fórmula:

$$V = (\text{Cantidad de filas} - 1) (\text{Cantidad de columnas} - 1)$$

$$\text{Grados de libertad} = (5-1) (2-1) = 4$$

Nivel de confianza.

Por lo general se trabaja con un porcentaje de 95% (0,95), es decir el nivel de error es del 5%(0,05).

DISTRIBUCION DE χ^2

| Grados de libertad | Probabilidad | | | | | | | | | | | |
|--------------------|------------------|------|------|------|------|-------|-------|-------|---------------|-------|-------|--|
| | 0,95 | 0,90 | 0,80 | 0,70 | 0,50 | 0,30 | 0,20 | 0,10 | 0,05 | 0,01 | 0,001 | |
| 1 | 0,004 | 0,02 | 0,06 | 0,15 | 0,46 | 1,07 | 1,64 | 2,71 | 3,84 | 6,64 | 10,83 | |
| 2 | 0,10 | 0,21 | 0,45 | 0,71 | 1,39 | 2,41 | 3,22 | 4,60 | 5,99 | 9,21 | 13,82 | |
| 3 | 0,35 | 0,58 | 1,01 | 1,42 | 2,37 | 3,66 | 4,64 | 6,25 | 7,82 | 11,34 | 16,27 | |
| 4 | 0,71 | 1,06 | 1,65 | 2,20 | 3,36 | 4,88 | 5,99 | 7,78 | 9,49 | 13,28 | 18,47 | |
| 5 | 1,14 | 1,61 | 2,34 | 3,00 | 4,35 | 6,06 | 7,29 | 9,24 | 11,07 | 15,09 | 20,52 | |
| 6 | 1,63 | 2,20 | 3,07 | 3,83 | 5,35 | 7,23 | 8,56 | 10,64 | 12,59 | 16,81 | 22,46 | |
| 7 | 2,17 | 2,83 | 3,82 | 4,67 | 6,35 | 8,38 | 9,80 | 12,02 | 14,07 | 18,48 | 24,32 | |
| 8 | 2,73 | 3,49 | 4,59 | 5,53 | 7,34 | 9,52 | 11,03 | 13,36 | 15,51 | 20,09 | 26,12 | |
| 9 | 3,32 | 4,17 | 5,38 | 6,39 | 8,34 | 10,66 | 12,24 | 14,68 | 16,92 | 21,67 | 27,88 | |
| 10 | 3,94 | 4,86 | 6,18 | 7,27 | 9,34 | 11,78 | 13,44 | 15,99 | 18,31 | 23,21 | 29,59 | |
| | No significativo | | | | | | | | Significativo | | | |

Figura VI.163: Tabla de Chi-Cuadrado.

Obteniendo el valor critico con 4 grados de libertad y valor de error de 0,05 = **9,49**

Conclusión:

Como el chi cuadrado experimental es mayor al valor critico, cae en la zona de rechazo de la Hipótesis Nula (Ho), aceptando la Hipótesis Alternativa.

$$10,99 > 9,49$$

6.1.6.2.2. CÁLCULOS OBTENIDOS CON SOFTWARE ESTADÍSTICO SPSS

Cálculos: para la obtención de los datos que se muestran en la tabla, se utilizó el software SPSS el cual es un programa estadístico informático.

Ingreso de datos al programa.

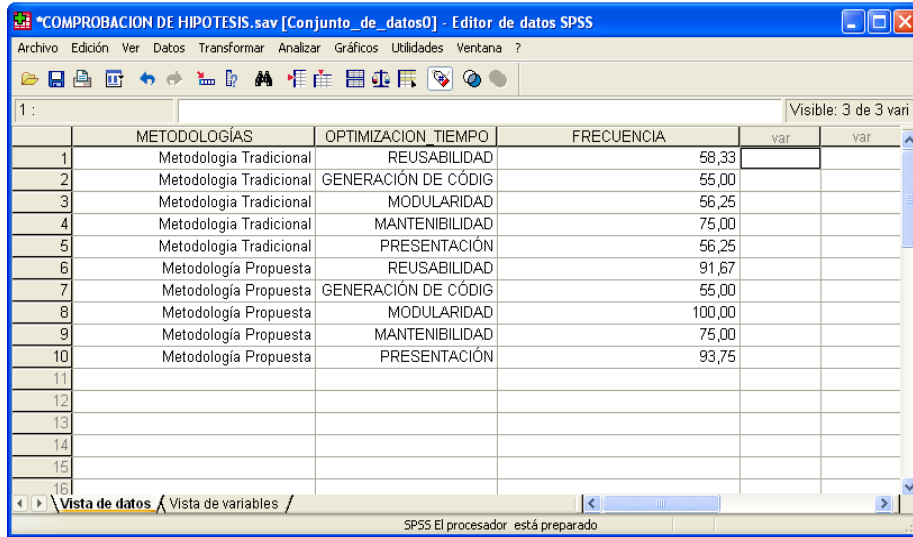


Figura VI.164: Ingreso de datos al Spss.

Resultados.

Valores esperados.

Tabla de contingencia OPTIMIZACIÓN DEL TIEMPO * METODOLOGÍAS

| | | | METODOLOGÍAS | | Total |
|-------------------------|----------------------|---------------------|-------------------------|-----------------------|--------|
| | | | Metodología Tradicional | Metodología Propuesta | |
| OPTIMIZACIÓN DEL TIEMPO | REUSABILIDAD | Recuento | 58 | 92 | 150 |
| | | Frecuencia esperada | 62,8 | 87,2 | 150,0 |
| | | % del total | 8,1% | 12,8% | 20,9% |
| | GENERACIÓN DE CÓDIGO | Recuento | 55 | 55 | 110 |
| | | Frecuencia esperada | 46,1 | 63,9 | 110,0 |
| | | % del total | 7,7% | 7,7% | 15,4% |
| | MODULARIDAD | Recuento | 56 | 100 | 156 |
| | | Frecuencia esperada | 65,4 | 90,6 | 156,0 |
| | | % del total | 7,8% | 14,0% | 21,8% |
| MANTENIBILIDAD | Recuento | 75 | 75 | 150 | |
| | Frecuencia esperada | 62,8 | 87,2 | 150,0 | |
| | % del total | 10,5% | 10,5% | 20,9% | |
| PRESENTACIÓN | Recuento | 56 | 94 | 150 | |
| | Frecuencia esperada | 62,8 | 87,2 | 150,0 | |
| | % del total | 7,8% | 13,1% | 20,9% | |
| Total | | Recuento | 300 | 416 | 716 |
| | | Frecuencia esperada | 300,0 | 416,0 | 716,0 |
| | | % del total | 41,9% | 58,1% | 100,0% |

Figura VI.165: Optimización del tiempo – Metodologías.

Valor del chi cuadrado

| Pruebas de chi-cuadrado | | | |
|------------------------------|---------------------|----|-----------------------------|
| | Valor | gl | Sig. asintótica (bilateral) |
| Chi-cuadrado de Pearson | 11,245 ^a | 4 | ,024 |
| Razón de verosimilitudes | 11,210 | 4 | ,024 |
| Asociación lineal por lineal | ,002 | 1 | ,968 |
| N de casos válidos | 716 | | |

a. 0 casillas (.0%) tienen una frecuencia esperada inferior a 5. La frecuencia mínima esperada es 46,09.

Figura VI.166: Resultado Chi cuadrado SPSS.

Valores obtenidos con SPSS

Chi cuadrado experimental = 11,245

El valor crítico con 4 grados de libertad y valor de error de 0,05 = **9,49**

11,245 > 9,49 se rechaza hipótesis nula.

Aceptación de Hipótesis Alternativa

H1: Las Metodología Propuesta optimizara el tiempo en la construcción de aplicaciones web mediante la Integración JSF y EJB.

CONCLUSIONES.

Una vez realizado los análisis pertinentes se ha llegado a las siguientes conclusiones:

- Se desarrolló una Metodología para la construcción de aplicaciones web mediante la integración de JSF y EJB la cual fue aplicada para la construcción del software de gestión SGM Pro, que permitió minimizar el tiempo de desarrollo de la aplicación.
- Java Server Faces permite realizar interfaces de usuario de una manera más fácil, rápida y de forma profesional, con la posibilidad de incorporar código para complementar la lógica de negocio.
- Enterprise Java Bean es una herramienta que permite generar automáticamente la capa de acceso datos y parte de la lógica de negocio sin esfuerzo alguno.
- La integración de EJB y JSF permite realizar aplicaciones web de tipo empresarial, de forma estructurada, ordenada y escalable; además de ahorrar tiempo en la construcción de estas.
- La utilización de las herramientas tecnológicas JSF y EJB permitió desarrollar el módulo de Plan de Mantenimiento de una manera fácil y rápida.
- La Metodología Propuesta ERCA Web que utiliza tecnologías de desarrollo como JSF y EJB permite minimizar el tiempo en un 21% más, con respecto a la Metodología Tradicional MSF que usa tecnologías como JSP y JDBC.

RECOMENDACIONES.

Una vez realizado el trabajo investigativo, se han establecido las siguientes recomendaciones:

- Investigar más a fondo sobre las tecnologías utilizadas en la metodología ERCA Web, de modo que en un futuro se pueda mejorar esta metodología.
- Utilizar librerías compatibles con JSF para crear interfaces de usuarios de una forma más rápida, amigable e intuitiva; además de no ingresar código de negocio si no es realmente necesario.
- Dedicar un tiempo considerable en el diseño y construcción de la base de datos de la aplicación, de modo que no haya inconvenientes al momento de que se genere la capa de acceso a datos y de lógica de negocios por parte del framework EJB.
- Estudiar y analizar profundamente las tecnologías EJB, JSF y los beneficios de su integración, para hacer uso de toda la potencialidad que estas tecnologías ofrecen.
- Para la construcción de aplicaciones web pequeñas se recomienda usar solamente el framework JSF, ya que este maneja internamente el concepto de EJB.

RESUMEN.

La formulación, desarrollo e implementación de la aplicación web de gestión de mantenimiento industrial utilizando la metodología ERCA Web fue implantada en el Hospital General Provincial Docente Chimborazo en la Ciudad de Riobamba, en la Facultad de Informática y Electrónica en la Escuela Superior Politécnica de Chimborazo.

Se aplicó un método experimental e inductivo, utilizando herramientas de software libre para el desarrollo de la aplicación. Se realizó dos prototipos de software; el primer prototipo se desarrolló utilizando la metodología tradicional de desarrollo de aplicaciones web, la cual usa herramientas como JSP y JDBC, mientras que el segundo prototipo se desarrolló utilizando la metodología propuesta (ERCA Web) de desarrollo de aplicaciones web, que usa tecnologías como JSF y EJB.

La metodología que permite minimizar el tiempo en la construcción y mantenimiento de aplicaciones web es la Metodología ERCA Web, obteniendo una Reusabilidad del 92%, Generación de Código del 55%, Modularidad del 100%, Mantenibilidad del 75% y Presentación del 93,75%; logrando así minimizar el tiempo de construcción y mantenimiento de aplicaciones web en un 21% más que la Metodología Tradicional.

De esta manera se concluye que utilizando la Metodología ERCA Web en el desarrollo y mantenimiento de aplicaciones web se reduce el tiempo en la construcción de éstas.

Se recomienda investigar más a fondo de las tecnologías usadas en la metodología ERCA Web, de modo que se pueda mejorar esta metodología en un futuro.

ABSTRACT.

The present thesis is the formulation, development and implementation of an industrial maintenance management using the ERCA Web Methodology in the Hospital Provincial General Docente Riobamba, through the Informatics and Electronics Faculty from Higher Polytechnic School of Chimborazo.

The current web applications nowadays have influenced hugely in the enterprise world in consequence is important to investigate the usage of new tools to simplify the construction of these ones.

The main problems that are pretended to solve are the construction time and web application maintenance in order to use different technologies. It was made two prototypes, each one to determine if the proposed research solves the problem.

Among the main objectives of this work are: to analyze the JSF and API Framework, also to study and develop the JSF integration through EJB, and finally to develop the methodology for a web application construction through integration.

It was applied an experimental and inductive method, using free software tools for the development of the application. It was made two kinds of software, the first one it was developed using the traditional method of web applications development, which uses tools such as: JSP and JDBC, while the second prototype was made using the proposed methodology (ERCA Web) of web development of applications that uses technologies like JSF and EJB.

The methodology allows to minimize the construction time and web application maintenance which is the ERCA Web Methodology, obtaining a Reusability of 92%; Code Generation of 55%, Modularity of 100%, Maintenance of 75% and Presentation of 93,75%; managing to minimize the construction time and web applications maintenance by 21% more than the Traditional Methodology.

On this way it can conclude that using ERCA Web Methodology in the development and the web applications, the time is reduced in their construction.

It is recommended to investigate deeper about the used technologies in the ERCA Web Methodology, on this form this methodology could be used in the future.

agregarUbicacion.xhtml

NUEVA UBICACION TECNICA

Ingrese una nueva Ubicacion Tecnica

Codigo *:

Descripcion:

Guardar Cancelar

editarUbicacion.xhtml

EDITAR UBICACION TECNICA

IdUbicacion: 1

Codigo: L01

Descripcion: LAVANDERIA

Actualizar Regresar

listaEquipos.xhtml

EQUIPOS

Gestión Asignar Actividades Generar Reportes

Agregar Equipo Asignar Actividad

| ID EQUIPO | UBICACION TECNICA | EQUIPO | CODIGO | MARCA | MODELO | FABRICANTE | DISTRIBUIDOR | AÑO FABRICA | EDITAR | ELIMINAR | ACTIVIDADE! | DETALLE! |
|-----------|-------------------|----------|--------|----------|--------|------------|--------------|-------------|--------|----------|-------------|----------|
| 1 | LAVANDERIA | Lavadora | LA/1 | Hagspiel | LG | Nadie | Otro | 2010 | Editar | Eliminar | ✓ | |
| 2 | LAVANDERIA | Lavadora | LA/2 | Hagspiel | | | | | Editar | Eliminar | ✓ | |

agregarEquipo.xhtml



Información General

Inicio

Introducción

Gestión

Reportes

Soporte Técnico

admin

NUEVO EQUIPO

Ingrese un nuevo Equipo

Descripción *:

Código *:

Marca:

Modelo:

Fabricante:

Distribuidor:

Año de fabricación:

Id Ubicación:

equiposEstrategia.xhtml

ASIGNACION EQUIPOS ACTIVIDADES AÑO 2012

| ID EQUIPO | UBICACION TECNICA | EQUIPO | CODIGO | MARCA | ASIGNAR ACTIVIDAD | MIS ACTIVIDADES |
|-----------|-------------------|----------|--------|----------|----------------------------------|----------------------------------|
| 1 | LAVANDERIA | Lavadora | LAV1 | Hagspiel | <input type="button" value="✖"/> | <input type="button" value="✎"/> |
| 2 | LAVANDERIA | Lavadora | LAV2 | Hagspiel | <input type="button" value="✖"/> | <input type="button" value="✎"/> |

asignarEquipoEstrategia.xhtml

ASIGNACIÓN DE ACTIVIDADES A EQUIPO - 2012



Plan:

Fecha Asignación:

Equipo: Lavadora - LAV1

Actividad:

listaEstrategias.xhtml

| ID ESTRATEGIA | ACTIVIDAD | DURACIÓN | FRECUENCIA | UNIDAD | TIPO | | | |
|---------------|--|----------|------------|---------|------|--------|----------|--|
| 1 | Cambio de rodamientos | 1.0 | 4 | Semanas | M | Editar | Eliminar | |
| 2 | Cambio de valvulas | 3.0 | 6 | Semanas | M | Editar | Eliminar | |
| 3 | Mantenimiento de inyectores electricos | 2.0 | 10 | Semanas | E | Editar | Eliminar | |
| 4 | Cambio de aceite de motor | 1.0 | 1000 | Horas | M | Editar | Eliminar | |

agregarEstrategias.xhtml

NUEVA ESTRATEGIA

Ingrese una nueva Estrategia

Los campos con asterisco (*) son campos requeridos.

Actividad * :

Duración (Horas) * :

Frecuencia * :

Unidad:

Tipo:

editarEstrategias.xhtml

Editar Estrategia

Los campos con asterisco (*) son campos requeridos.

IdEstrategia:

Actividad * :

Duración * :

Frecuencia * :

Unidad:

Tipo:

listaTecnicos.xhtml

TÉCNICOS

Gestión Generar Reportes

Agregar Técnico

| ID TÉCNICO | CEDULA | NOMBRE | APELLIDO | DIRECCION | TELEFONO | | | |
|------------|------------|---------|-----------------|--------------------------------|------------|--------|----------|--|
| 1 | 0803110451 | Alex | Erazo Vivero | Cdla. Juan Montalvo | 0986008061 | Editar | Eliminar | |
| 2 | 0803880756 | Jessica | Sanchez Verduga | Cdla. Nueva Concordia Sector 3 | 0969686485 | Editar | Eliminar | |

listaRepuestos.xhtml

REPUESTOS

Gestión Generar Reportes

Agregar Repuesto

| ID REPUESTO | DESCRIPCION | STOCK MINIMO | STOCK REAL | COSTO | UNIDAD | | | |
|-------------|-------------|--------------|------------|-------|----------|--------|----------|---------|
| 1 | BANDAS | 3 | 10 | 2.5 | Unidades | Editar | Eliminar | Comprar |
| 2 | TORVILLOS | 10 | 100 | 0.25 | Unidades | Editar | Eliminar | Comprar |

historialMantenimiento.xhtml

HISTORIAL PLAN DE MANTENIMIENTO

Plan Mantenimiento:

| ANIO | ACCION |
|-------------------------|--------|
| Plan Mantenimiento 2012 | Ver |

Nuevo Plan

Generar Plan Mmatenimiento Año 2013

planMantenimiento.xhtml

The screenshot shows the 'planMantenimiento AÑO 2012' interface. At the top, there is a header with the logo of the 'Ministerio de Salud Pública' and 'INGENIERÍA MECÁNICA'. Below the header is a navigation menu on the left with options like 'Información General', 'Inicio', 'Introducción', 'Gestión', 'Reportes', 'Soporte Técnico', and 'admin'. The main content area displays 'PLAN MANTENIMIENTO AÑO 2012' with a sub-section 'OPCIONES' containing navigation icons. Below this is a table with columns for 'ESTRATEGIA', 'EQUIPO', 'FRECUENCIA', 'INICIO SEMANA', and 17 days of the week. The table shows 'NO EXISTE ESTRATEGIAS ASIGNADAS' and a 'TOTAL HORAS' of 0.0 for each day. A 'RESULTADOS' box at the bottom right indicates 'TOTAL DE HORAS EN EL AÑO: 0.0 horas' and 'PROMEDIO DE HORAS POR SEMANA: 0.0 horas / Semana'.

buscarSemanaPlan.xhtml

The screenshot shows the 'PLAN DE ACTIVIDADES POR SEMANA' interface. It features the same header and navigation menu as the previous page. The main content area prompts the user to 'Seleccione una semana para generar plan de Actividades de la Semana:'. It includes a calendar widget for the year 2012, with the 1st of the month highlighted. Below the calendar are input fields for 'Año: 2012' and 'Semana: Seleccione una Semana', along with 'Aceptar' and 'Cancelar' buttons.

planMantenimientoSemana.xhtml

The screenshot shows the 'PLAN MANTENIMIENTO AÑO 2012' interface for a specific week. It includes the header, navigation menu, and 'OPCIONES' section. Below this is a 'Reporte' button and a table titled 'PLAN ACTIVIDAD SEMANA 1'. The table has columns for 'EQUIPO', 'ESTRATEGIA', 'DURACION', 'FRECUENCIA', and 'INICIO DE SEMANA'. The table content shows 'NO EXISTE ACTIVIDADES PARA ESTA SEMANA'.

planMantenimientoSemanaAsignar.xhtml

Ministerio de Salud Pública
INGENIERÍA MECÁNICA

SOFTWARE DE GESTIÓN
SGM Pro

PLAN MANTENIMIENTO AÑO 2012

OPCIONES Reporte

PLAN ACTIVIDAD SEMANA 1 - ASIGNACIÓN DE TÉCNICOS

| EQUIPO | ESTRATEGIA | DURACION | FRECUENCIA | INICIO SEMANA | A SIGNAR |
|--|------------|----------|------------|---------------|----------|
| NO EXISTE MAS ESTRATEGIAS PARA ASIGNAR | | | | | |

Información General
Gestión
Reportes
Plan de Actividades
Reporte Fallas
Actividades Asignadas
Técnicos - Actividades
Soporte Técnico
admin

listadoFallas.xhtml

Ministerio de Salud Pública
INGENIERÍA MECÁNICA

SOFTWARE DE GESTIÓN
SGM Pro

REPORTE DE FALLAS

Gestión Generar Reportes
Reportar Falla

| ID FALLA | EQUIPO | NOVEDAD | HORA INICIO | HORA FIN | FECHA INICIO | FECHA FIN | TIEMPO FALLA | EDITAR | ELIMINAR | REPUESTOS | MIS RE |
|-------------------|--------|---------|-------------|----------|--------------|-----------|--------------|--------|----------|-----------|--------|
| No records found. | | | | | | | | | | | |

Inicio
Introducción
Gestión
Reportes
Soporte Técnico
admin

agregarFalla.xhtml

Ministerio de Salud Pública
INGENIERÍA MECÁNICA

SOFTWARE DE GESTIÓN
SGM Pro

NUEVA FALLA

Reportar nueva Falla

Novedad:

Fecha Inicial:

Hora Inicial:

Fecha Final:

Hora Final:

Tiempo Falla: 0.0 Horas

Equipos: LAV1 - Lavadora

Guardar Cancelar

Información General
Inicio
Introducción
Gestión
Reportes
Soporte Técnico
admin

listaTecnicos1.xhtml

LISTA DE TECNICOS

| ID TECNICO | CEDULA | NOMBRE | APELLIDO | |
|------------|------------|---------|-----------------|---------------------------------|
| 1 | 0803110451 | Alex | Erazo Vivero | Ver Actividades |
| 2 | 0803880756 | Jessica | Sanchez Verduga | Ver Actividades |

listaActividad.xhtml

LISTA DE ACTIVIDADES

[Gestión](#)
[Aceptar](#)

| ACTIVIDAD | EQUIPO | FRECUENCIA | DURACION |
|-------------------|--------|------------|----------|
| No records found. | | | |

ENCUESTA REALIZADA A EXPERTOS.

Proyecto de Tesis – Escuela de Ingeniería en Sistemas

Objetivo: determinar la optimización de tiempo en el desarrollo de aplicaciones web mediante la utilización de diferentes herramientas de programación.

Nota: Encuesta dirigida para programadores que utilicen herramientas de programación como son JSP, JDBC, JSF Y EJB.

1. Seleccione el nivel de portabilidad de cada uno de las tecnologías.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

2. Selección el nivel de que herramienta considera que tiene independencia de componentes.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

3. Selección el nivel de que herramienta considera que tiene facilidad de integración con nuevas herramientas de programación.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

4. Selección el nivel de que herramienta le facilita la creación de la capa de acceso a datos.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

5. Selección el nivel que tiene cada herramienta considerando la facilidad de la creación de la capa de lógica.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

6. Seleccione el nivel de que herramienta se la hace más complejo utilizar.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

7. Defina el nivel de la capacidad de descomposición de las herramientas.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

8. Defina el nivel de escalabilidad.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

9. Mediante que herramientas se le hace más fácil para la adaptación de nuevos requerimientos.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

10. Con que herramienta se le hace más fácil en la construcción de templates personalizados.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

11. Establezca la facilidad del uso de componentes AJAX.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

12. Establezca la facilidad del uso de validaciones en las páginas web.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

13. Establezca la facilidad de uso de java script.

| | Bajo - 0% | Medio Bajo- 25% | Medio - 50% | Medio Alto - 75% | Alto - 100% |
|------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| JDBC - JSP | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| EJB - JSF | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

GLOSARIO.

| | |
|---------------|--|
| Administrador | Persona responsable del manejo del sistema. |
| Aplicación | Tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajo. |
| Arquitectura | Nivel de diseño que hace foco en aspectos "más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema". |
| Atributo | Indica una o más características de un objeto. |
| Bean | Componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno. |
| Cliente | Persona que usa el sistema. |
| Concurrencia | Es el grado en que las fases, etapas o actividades pueden ser superpuestas a pesar de que no tengan relación entre sí. |
| Cookies | Fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador |
| Diseño | Proceso de definición de la arquitectura, componentes, interfaces y otras características de un sistema o componente que resulta de este proceso |
| Framework | Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado. |
| Hardware | Corresponde a todas las partes físicas y tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos. |

| | |
|----------------|---|
| Implementación | Realización de una aplicación, o la ejecución de un plan, idea, modelo científico, diseño, especificación, estándar, algoritmo o política. |
| Ingeniería | Conjunto de conocimientos y técnicas científicas aplicadas a la invención, perfeccionamiento y utilización de la técnica industrial en todos sus diversos aspectos incluyendo la resolución u optimización de problemas que afectan directamente a los seres humanos en su actividad. |
| Interfaz | Lo que es visible para el usuario. |
| Internet | Conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas que la componen funcionen como una red lógica única, de alcance mundial. |
| Intranet | Es una red de ordenadores privados que utiliza tecnología Internet para compartir dentro de una organización parte de sus sistemas de información y sistemas operacionales. |
| Java | Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. |
| Javascript | Lenguaje de scripting basado en objetos no tipeado y liviano, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. |
| Middleware | Software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. |
| Persistencia | Se refiere a la propiedad de los datos para que estos sobrevivan de alguna manera. |

| | |
|------------|---|
| Protocolo | Conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red. |
| Proxy | Referencia a un programa o dispositivo que realiza una acción en representación de otro. |
| Requisitos | Necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. |
| Servidor | Computadora que, formando parte de una red, provee servicios a otras computadoras denominadas clientes. |
| Servlets | Son objetos que corren dentro del contexto de un contenedor de servlets (ej: Tomcat) y extienden su funcionalidad. |
| Software | Conjunto de programas, métodos y procedimientos relacionados con la explotación, funcionamiento y manejo de un sistema de proceso de datos. |
| Threads | Hilo de ejecución o subprocesso es una característica que permite a una aplicación realizar varias tareas a la vez. |
| Web | Red informática, especialmente para referirse a internet. |

BIBLIOGRAFÍA.

1. **CORONEL, E.**, Desarrollando soluciones con Java y MySQL., Editorial macro., 2009., Pp. 428.
2. **KEOGH, J.**, J2EE: Manual de Referencia., Editorial McGraw-Hill Interamericana., 2002., Pp. 803.
3. **ROZANSKI, U.**, Enterprise Java Bean 3.0 con Eclipse y JBoss., Barcelona, Editorial Alfaomega, MARCOMBO S.A., 2009., Pp. 563.
4. **SIERRA, M.**, Ajax en J2EE., Editorial Alfaomega., 2008., Pp. 264.

TESIS.

5. **VALDIVIEZO, P., GUACHO, M.**, ANÁLISIS COMPARATIVO DE TECNOLOGÍAS DE APLICACIONES WEB EN EL ENTORNO JSF Y ADF. CASO PRÁCTICO: IESS DE RIOBAMBA: CHIMBORAZO.

BIBILOGRAFÍA DE INTERNET.

6. **ENTERPRISE JAVA BEAN.**
<http://osgg.net/omarsite/resources/proceedings/EJB.pdf>
2011/12/21.

7. ESTUDIO, COMPARATIVA Y APLICACIÓN PRÁCTICA DE METODOLOGÍAS DE DESARROLLO DE APLICACIONES WEB EN JAVA.

<http://openaccess.uoc.edu/webapps/o2/bitstream/10609/619/1/00804ffc.pdf>

2012/04/18

8. JAVA SERVER FACES.

<http://www.sicuma.uma.es/sicuma/Formacion/documentacion/JSF.pdf>

2011/12/21.

9. JAVA SERVER FACES.

<http://ccia.ei.uvigo.es/docencia/SCS/1011/transparencias/Tema5-3.JSF.pdf>

2012/03/20.

10. JAVA SERVER FACES: RICHFACES Y ICEFACES.

<http://www.slideshare.net/israelalcazar/introduccion-jsf-richfaces-e-icefaces-2267358>

2012/03/15.

11. PRIME FACES.

<http://www.primefaces.org/showcase/ui/home.jsf>

2012/05/12.

12. PROGRAMACIÓN DE APLICACIONES WEB.

<http://gplsi.dlsi.ua.es/~slujan/materiales/pi-cliente2-muestra.pdf>

2012/01/02.

13. RECURSOS EJB.

http://www.dsc.ufcg.edu.br/~jacques/cursos/map/recursos/ejb/white_per.pdf

2012/12/28.

14. TUTORIAL JSF 2.0 Y PRIMEFACES.

<http://www.javahispano.org/storage/1.%20Parte%20I%20-%20NetBeans.pdf>

2012/06/02.