



**Universidad de  
Córdoba**



**Escuela Politécnica Superior**  
INGENIERÍA EN INFORMÁTICA

**Ampliación de Ingeniería del Software**

<b>Proyecto</b>	GIGI
<b>Fase</b>	Fase 4
<b>Actividad</b>	Gestión de la Configuración
<b>Grupo</b>	12
<b>Fecha</b>	

**MANUAL TÉCNICO**

---

---

<b>Objetivo</b>	Documentación completa.
<b>Comentarios</b>	Manual Técnico.

# ÍNDICE DE CONTENIDO

<b>BASE DE DATOS .....</b>	<b>1</b>
1. BASE DE DATOS .....	2
1.1. INTRODUCCIÓN .....	2
1.2. DEFINICIÓN DEL PROBLEMA.....	3
1.3. RESTRICCIONES .....	5
1.4. RECURSOS.....	6
1.5. ESPECIFICACIÓN DE REQUISITOS.....	7
<b>1.5.1 MODELO CONCEPTUAL .....</b>	<b>7</b>
1.5.1.1 <i>Análisis de los Tipos de Entidad del “Diagrama Conceptual Básico” .....</i>	7
1.5.1.2 <i>Análisis de los tipos de interrelación del “Diagrama Conceptual Básico” .....</i>	11
1.5.1.3 <i>Análisis de los atributos del “Diagrama Conceptual Básico” .....</i>	14

1.5.1.4 Identificación de los atributos identificadores del “Diagrama Conceptual Básico” .....	25
1.5.1.5 Diagrama E-R del “Diagrama Conceptual Básico” .....	26
1.5.1.6 Análisis de los Tipos de Entidad del “Diagrama Conceptual Curriculum Vitae” .....	27
1.5.1.7 Análisis de los tipos de interrelación del “Diagrama Conceptual Curriculum Vitae” .....	30
1.5.1.8 Análisis de los atributos del “Diagrama Conceptual Curriculum Vitae” .....	34
1.5.1.9 Identificación de los atributos identificadores del “Diagrama Conceptual Curriculum Vitae” .....	40
1.5.1.10 Diagrama E-R del “Diagrama Conceptual Curriculum Vitae” .....	41
<b>1.5.2 MODELO FUNCIONAL.....</b>	<b>43</b>
<b>1.6. DISEÑO DEL SISTEMA .....</b>	<b>48</b>
<b>1.6.1 MODELO RELACIONAL BÁSICO .....</b>	<b>48</b>
1.6.1.1 Creación de las tablas .....	48
1.6.1.2 Normalización del modelo .....	53
<b>1.6.2 MODELO RELACIONAL CURRÍCULUM VITAE.....</b>	<b>55</b>
1.6.2.1 Creación de las tablas .....	56
1.6.2.2 Normalización del modelo .....	61
<b>1.6.3 DISEÑO DEL ESQUEMA DEFINITIVO.....</b>	<b>62</b>
<b>1.6.4 SENTENCIAS DE DEFINICIÓN DE LA BASE DE DATOS EN MYSQL .....</b>	<b>73</b>
<b>1.6.5 PROCEDIMIENTOS DE MANIPULACIÓN DE LA BASE DE DATOS EN MYSQL .....</b>	<b>91</b>

1.7. PRUEBAS .....	158
1.8. FICHEROS DE CONFIGURACIÓN .....	160
<b>2. INTERFAZ GENERAL.....</b>	<b>162</b>
DEFINICIÓN DEL PROBLEMA .....	163
2.1.1 DEFINICIÓN DEL PROBLEMA REAL .....	163
2.1.2 DEFINICIÓN DEL PROBLEMA TÉCNICO.....	164
2.1.2.1 FUNCIONAMIENTO.....	164
2.1.2.2 ENTORNO .....	165
2.1.2.3 VIDA ESPERADA .....	165
2.1.2.4 CICLO DE MANTENIMIENTO .....	166
2.1.2.5 COMPETENCIA.....	166
2.1.2.6 ASPECTO EXTERNO .....	167
2.1.2.7 ESTANDARIZACIÓN .....	167
2.1.2.8 CALIDAD Y FIABILIDAD .....	167
2.1.2.9 PROGRAMA DE TAREAS.....	168
2.1.2.10 PRUEBAS.....	168

<b>2.1.2.11 SEGURIDAD .....</b>	<b>169</b>
OBJETIVOS .....	170
RESTRICCIONES TÉCNICAS Y DE GESTIÓN .....	172
RECURSOS .....	174
2.4.1 RECURSOS HUMANOS .....	174
2.4.2 RECURSOS HARDWARE .....	175
2.4.3 RECURSOS SOFTWARE .....	175
ESPECIFICACIÓN DE REQUISITOS .....	176
2.5.1 IDENTIFICACIÓN DE REQUISITOS .....	177
2.5.2 DESCRIPCIÓN DE LA INFORMACIÓN .....	178
2.5.3 DESCRIPCIÓN FUNCIONAL .....	178
2.5.3.1 DETERMINACIÓN DE LOS ACTORES DEL SISTEMA.....	179
2.5.3.2 ESPECIFICACIÓN DE LOS CASOS DE USO DE TRAZO GRUESO .....	179
2.5.3.3 PRIORIZACIÓN DE LOS CASOS DE USO.....	181
2.5.3.4 MATRIZ REQUISITOS / CASOS DE USO.....	182

<b>2.5.3.5 ESPECIFICACIÓN DE LOS CASOS DE USO DE TRAZO FINO .....</b>	<b>182</b>
2.5.3.5.1 Caso de uso Conectar BD .....	183
2.5.3.5.2 Caso de uso Cambiar Usuario .....	184
2.5.3.5.3 Caso de uso Desconectar BD .....	186
<b>2.5.4 ESPECIFICACIÓN DE LA INTERFAZ .....</b>	<b>187</b>
<b>2.5.4.1 VENTANA PRINCIPAL .....</b>	<b>188</b>
2.5.4.1.1 Barra de menús .....	190
2.5.4.1.2 Barra de herramientas .....	191
2.5.4.1.3 Árbol de navegación .....	192
2.5.4.1.4 Zona de mantenimiento de información .....	192
2.5.4.1.5 Zona de visualización de información .....	192
2.5.4.1.6 Zona de ayuda .....	193
2.5.4.1.7 Barra de estado .....	193
<b>2.5.4.2 VENTANA DE CONEXIÓN .....</b>	<b>193</b>
<b>2.5.4.3 VENTANA DE CAMBIO DE USUARIO .....</b>	<b>193</b>
<b>DISEÑO DEL SISTEMA.....</b>	<b>194</b>
<b>2.6.1 DISEÑO DE DATOS .....</b>	<b>195</b>
<b>2.6.2 DISEÑO ARQUITECTÓNICO.....</b>	<b>195</b>
<b>2.6.3 DISEÑO DE CLASES.....</b>	<b>198</b>

<b>2.6.3.1 CLASES DEL SISTEMA.....</b>	<b>198</b>
2.6.3.1.1 Clase Aplicacion .....	199
2.6.3.1.2 Clase ConexiónBD .....	200
2.6.3.1.3 Clase MarcoPrincipal.....	201
2.6.3.1.4 Clase BarraEstado.....	204
2.6.3.1.5 Clase MarcoAcercaDe .....	205
2.6.3.1.6 Clase MarcoConectar .....	206
2.6.3.1.7 Clase PanelAyuda .....	208
2.6.3.1.8 Clase PanelGeneral.....	208
2.6.3.1.9 Clase PanelFrontal .....	210
2.6.3.1.10 Clase ArbolNavegacion .....	211
2.6.3.1.11 Clase NodoArbol.....	212
2.6.3.1.12 Clase RepresentadorNodosArbol .....	213
<b>2.6.3.2 DIAGRAMA DE CLASES DEL SISTEMA.....</b>	<b>214</b>
 2.6.4 DISEÑO DE LA INTERFAZ .....	215
<b>2.6.4.1 VENTANA PRINCIPAL .....</b>	<b>215</b>
2.6.4.1.1 Barra de menús .....	217
2.6.4.1.2 Barra de herramientas .....	221
2.6.4.1.3 Árbol de navegación .....	223
2.6.4.1.4 Barra de estado .....	224
<b>2.6.4.2 VENTANA DE CONEXIÓN .....</b>	<b>224</b>
<b>2.6.4.3 VENTANA DE CAMBIO DE USUARIO.....</b>	<b>225</b>

PRUEBAS .....	227
2.7.1 PRUEBAS DE LA CAJA BLANCA .....	228
2.7.2 PRUEBAS DE LA CAJA NEGRA.....	228
2.7.3 CARGA DE INFORMACIÓN .....	230
2.7.3.1 INSERCIÓN DE ARTÍCULO.....	230
2.7.3.2 INSERCIÓN DE LIBROS.....	231
2.7.3.3 INSERCIÓN DE CAPÍTULOS .....	232
2.7.3.4 INSERCIÓN DE ACTAS .....	233
2.7.3.5 INSERCIÓN DE REVISTAS .....	233
2.7.3.6 INSERCIÓN DE AUTORES .....	234
2.7.3.7 INSERCIÓN DE CONGRESOS .....	235
 <b>3. FORMULARIO DE ENTRADA .....</b>	<b>237</b>
 DEFINICIÓN DEL PROBLEMA .....	238
3.1.1 PROBLEMA REAL .....	239
3.1.2 PROBLEMA TÉCNICO.....	239



OBJETIVOS.....	240
RESTRICCIONES TÉCNICAS Y DE GESTIÓN.....	241
3.3.1 FACTORES DATO .....	241
3.3.2 FACTORES ESTRATÉGICOS .....	242
RECURSOS .....	243
3.4.1 RECURSOS HUMANOS. ....	243
3.4.2 RECURSOS HARDWARE. ....	244
3.4.3 RECURSOS SOFTWARE.....	244
ESPECIFICACIÓN DE REQUISITOS .....	245
3.5.1 IDENTIFICACIÓN DE REQUISITOS.....	245
3.5.1.1 REQUISITOS GENERALES .....	245
3.5.1.2 REQUISITOS DEL FORMULARIO MIEMBROS.....	246
3.5.1.3 REQUISITOS DEL FORMULARIO ARTÍCULOS.....	247
3.5.1.4 REQUISITOS DEL FORMULARIO REVISTAS .....	248
3.5.1.5 REQUISITOS DEL FORMULARIO AUTORES.....	249

3.5.2 DESCRIPCIÓN DE LA VISTA DE USUARIO .....	249
3.5.2.1 ACTORES .....	250
3.5.2.2 DIAGRAMA DE CASOS DE USO: CONTEXTO DEL SISTEMA.....	251
3.5.2.3 DIAGRAMA DE CASOS DE USO: GESTIÓN DE LOS FORMULARIOS DE ENTRADA ...	253
3.5.2.4 DIAGRAMA DE CASOS DE USO: FORMULARIO MIEMBRO .....	255
3.5.2.5 DIAGRAMA DE CASOS DE USO: FORMULARIO REVISTA .....	257
3.5.2.6 DIAGRAMA DE CASOS DE USO: FORMULARIO ARTÍCULO .....	259
3.5.2.7 DIAGRAMA DE CASOS DE USO: FORMULARIO LIBRO .....	261
3.5.2.8 DIAGRAMA DE CASOS DE USO: FORMULARIO CAPÍTULO .....	263
3.5.2.9 DIAGRAMA DE CASOS DE USO: FORMULARIO CONGRESO .....	265
3.5.2.10 DIAGRAMA DE CASOS DE USO: FORMULARIO ACTA.....	267
3.5.2.11 DIAGRAMA DE CASOS DE USO: FORMULARIO AUTOR.....	269
ANÁLISIS DEL SISTEMA.....	271
3.6.1 MODELO DE ANÁLISIS .....	271
3.6.1.1 ESPECIFICACIÓN DEL MODELO DE CLASES .....	273
3.6.1.1.1 Clase: GestorInformacion .....	273
3.6.1.1.2 Clase: GeneradorFE.....	274
3.6.1.1.3 Clase: Formulario .....	275

<b>3.6.1.2 ESPECIFICACIÓN DEL MODELO OBJETO DE ANÁLISIS – RELACIÓN .....</b>	<b>276</b>
3.6.1.2.1 <i>Relación: GestorInformacion – GeneradorFE .....</i>	<i>276</i>
3.6.1.2.2 <i>Relación: GeneradorFE – Formulario .....</i>	<i>277</i>
<b>3.6.1.3 DIAGRAMA DE CLASES DEL SISTEMA .....</b>	<b>277</b>
<b>3.6.1.4 ESPECIFICACIÓN DEL MODELO OBJETO DE ANÁLISIS - COMPORTAMIENTO .....</b>	<b>278</b>
3.6.1.4.1 <i>Diagrama de Colaboración: Insertar Nueva Información .....</i>	<i>279</i>
<b>DISEÑO DEL SISTEMA .....</b>	<b>281</b>
<b>3.7.1 ESPECIFICACIÓN DEL MODELO DE OBJETOS .....</b>	<b>281</b>
<b>3.7.1.1 OBJETO: FORMULARIOENTRADAACTA .....</b>	<b>282</b>
<b>3.7.1.2 OBJETO: FORMULARIOENTRADAA ARTICULO .....</b>	<b>283</b>
<b>3.7.1.3 OBJETO: FORMULARIOENTRADAAUTOR .....</b>	<b>284</b>
<b>3.7.1.4 OBJETO: FORMULARIOENTRADACAPITULO .....</b>	<b>285</b>
<b>3.7.1.5 OBJETO: FORMULARIOENTRADACONGRESO .....</b>	<b>286</b>
<b>3.7.1.6 OBJETO: FORMULARIOENTRADALIBRO .....</b>	<b>287</b>
<b>3.7.1.7 OBJETO: FORMULARIOENTRADALIBROCAPITULO .....</b>	<b>288</b>
<b>3.7.1.8 OBJETO: FORMULARIOENTRADAMIEMBRO .....</b>	<b>289</b>
<b>3.7.1.9 OBJETO: FORMULARIOENTRADAREVISTA .....</b>	<b>289</b>
<b>3.7.2 ESPECIFICACIÓN DEL MODELO OBJETO DE DISEÑO – RELACIÓN .....</b>	<b>290</b>

3.7.3 ESPECIFICACIÓN PROCEDIMENTAL .....	291
3.7.3.1 DIAGRAMA DE ACTIVIDADES: INSERTAR NUEVA INFORMACIÓN .....	292
3.7.4 MODELO DE PAQUETES .....	293
3.7.4.1 PAQUETE: SISTEMA.....	294
3.7.4.1 PAQUETE: FORMULARIOS ENTRADA .....	294
DESCRIPCIÓN DE LA INTERFAZ .....	296
3.8.1 MÓDULO DE ENTRADA .....	298
3.8.1.1 VENTANA PRINCIPAL ACTAS .....	299
3.8.1.2 VENTANA PRINCIPAL INTEGRANTES .....	300
3.8.1.3 VENTANA PRINCIPAL LIBROS.....	301
3.8.1.4 VENTANA PRINCIPAL CAPÍTULO.....	303
3.8.1.5 VENTANA PRINCIPAL ARTÍCULO.....	305
3.8.1.6 VENTANA SECUNDARIA AUTOR.....	307
3.8.1.7 VENTANA SECUNDARIA CONGRESO .....	309
3.8.1.8 VENTANA SECUNDARIA REVISTA .....	309
3.8.1.9 VENTANA SECUNDARIA LIBRO-CAPÍTULO .....	310

<b>4. FORMULARIO DE VISUALIZACIÓN .....</b>	<b>312</b>
DEFINICIÓN DEL PROBLEMA.....	313
OBJETIVOS.....	315
RESTRICCIONES .....	317
RECURSOS.....	320
ESPECIFICACIÓN DE REQUISITOS .....	322
<b>4.5.2.1 IDENTIFICACIÓN DE LOS ACTORES .....</b>	<b>324</b>
<b>4.5.2.2 CASOS DE USO.....</b>	<b>325</b>
4.5.2.2.1 Casos de uso Inicial.....	325
4.5.2.2.2 Refinamiento Casos de uso.....	326
ANÁLISIS DEL SISTEMA .....	332
DISEÑO DEL SISTEMA .....	337
<b>4.7.1.1 PAQUETE PRINCIPAL .....</b>	<b>338</b>
<b>4.7.1.2 PAQUETE PRINCIPAL.FORMULARIOS .....</b>	<b>339</b>
<b>4.7.1.3 PAQUETE PRINCIPAL.FORMULARIOS.ENTRADA.....</b>	<b>339</b>
<b>4.7.1.4 PAQUETE PRINCIPAL.FORMULARIOS.ENTRADA.RESOURCES .....</b>	<b>339</b>

<b>4.7.1.5 PAQUETE PRINCIPAL.FORMULARIOS.VISUALIZACION.....</b>	<b>340</b>
<b>4.7.1.6 PAQUETE PRINCIPAL.FORMULARIOS.VISUALIZACION.RESOURCES .....</b>	<b>341</b>
<b>4.7.2.1 PAQUETE PRINCIPAL .....</b>	<b>342</b>
4.7.2.1.1 Clase Principal.....	343
4.7.2.1.2 Clase MarcoPrincipal.....	344
<b>4.7.2.2 PAQUETE “PRINCIPAL.FORMULARIOS” .....</b>	<b>346</b>
4.7.2.2.1 Clase PanelGeneral.....	346
<b>4.7.2.3 PAQUETE “PRINCIPAL.FORMULARIOS.ENTRADA” .....</b>	<b>348</b>
4.7.2.3.1 Clase FormularioEntrada.....	348
<b>4.7.2.4 PAQUETE “PRINCIPAL.FORMULARIOS.ENTRADA.RESOURCES” .....</b>	<b>349</b>
4.7.2.4.1 Clase ElementosEntrada .....	350
4.7.2.4.2 Clase MostrarEntradaBookChapter .....	352
4.7.2.4.3 Clase MostrarEntradaPatent.....	354
<b>4.7.2.5 PAQUETE “PRINCIPAL.FORMULARIOS.VISUALIZACION” .....</b>	<b>355</b>
4.7.2.5.1 Clase FormularioVisualizacion.....	356
4.7.2.5.2 Clase MostrarInformacion.....	357
4.7.2.5.3 Clase Soporte.....	359
<b>4.7.2.6 PAQUETE “PRINCIPAL.FORMULARIOS.VISUALIZACION.RESOURCES” .....</b>	<b>360</b>
4.7.2.6.1 Clase ModeloTablas .....	362
4.7.2.6.2 Clase MostrarTablaActas.....	363
4.7.2.6.3 Clase TablaActas.....	365

<b>4.7.3.1 DIAGRAMA DE SECUENCIA .....</b>	<b>366</b>
<b>4.7.4.1 BUSCAR INFORMACIÓN .....</b>	<b>367</b>
<b>4.7.4.2 MOSTRAR DETALLE .....</b>	<b>368</b>
<b>4.7.4.3 ORDENAR TUPLAS .....</b>	<b>368</b>
 IMPLEMENTACIÓN DEL SISTEMA .....	 370
 <b>5. FORMULARIO DE IMPRESIÓN .....</b>	 <b>376</b>
 DEFINICIÓN DEL PROBLEMA .....	 377
 5.1.1 DEFINICIÓN DEL PROBLEMA REAL .....	 377
 5.1.2 DEFINICIÓN DEL PROBLEMA TÉCNICO .....	 378
5.1.2.1 FUNCIONAMIENTO.....	378
5.1.2.2 ENTORNO .....	379
5.1.2.3 VIDA ESPERADA .....	379
5.1.2.4 CICLO DE MANTENIMIENTO .....	379
5.1.2.5 COMPETENCIA.....	380
5.1.2.6 ASPECTO EXTERNO .....	380
5.1.2.7 ESTANDARIZACIÓN .....	380

<b>5.1.2.8 CALIDAD Y FIABILIDAD .....</b>	<b>380</b>
<b>5.1.2.9 PROGRAMA DE TAREAS.....</b>	<b>381</b>
<b>5.1.2.10 PRUEBAS.....</b>	<b>382</b>
<b>5.1.2.11 SEGURIDAD .....</b>	<b>382</b>
 OBJETIVOS .....	384
 RESTRICCIONES TÉCNICAS Y DE GESTIÓN .....	386
 RECURSOS .....	388
 5.4.1 RECURSOS HUMANOS.....	388
 5.4.2 RECURSOS HARDWARE.....	389
 5.4.3 RECURSOS SOFTWARE .....	389
 ESPECIFICACIÓN DE REQUISITOS .....	390
 5.5.1 IDENTIFICACIÓN DE REQUISITOS.....	390
 5.5.1.1 REQUISITO IMPRIMIR INFORMACIÓN AUTORES .....	391
 5.5.1.2 REQUISITO IMPRIMIR INFORMACIÓN INTEGRANTE .....	391
 5.5.1.3 REQUISITO IMPRIMIR INFORMACIÓN ACTA.....	391



<b>5.5.1.4 REQUISITO IMPRIMIR INFORMACIÓN ARTÍCULO .....</b>	<b>392</b>
<b>5.5.1.5 REQUISITO IMPRIMIR INFORMACIÓN CAPÍTULO.....</b>	<b>392</b>
<b>5.5.1.6 REQUISITO IMPRIMIR INFORMACIÓN CONGRESO .....</b>	<b>393</b>
<b>5.5.1.7 REQUISITO IMPRIMIR INFORMACIÓN LIBRO .....</b>	<b>393</b>
<b>5.5.1.8 REQUISITO IMPRIMIR INFORMACIÓN REVISTA.....</b>	<b>393</b>
<b>5.5.2 FORMALIZACIÓN DE REQUISITOS .....</b>	<b>394</b>
<b>5.5.2.1 IDENTIFICACIÓN DE LOS ACTORES .....</b>	<b>394</b>
<b>5.5.2.2 ESPECIFICACIÓN DE LOS CASOS DE USO .....</b>	<b>395</b>
5.5.2.2.1 Casos de uso de trazo grueso .....	395
5.5.2.2.2 Refinamiento Casos de uso.....	401
5.5.2.2.2.1 Caso de uso Imprimir Información Autor.....	401
5.5.2.2.2.2 Caso de uso Imprimir Listado Autores .....	402
<b>5.5.2.3 MATRIZ REQUISITOS / CASOS DE USO.....</b>	<b>404</b>
<b>ANÁLISIS DEL SISTEMA .....</b>	<b>405</b>
<b>5.6.1 MODELO DE ANÁLISIS .....</b>	<b>405</b>
<b>5.6.1.1 ESPECIFICACIÓN DEL MODELO DE CLASES .....</b>	<b>407</b>
5.6.1.1.1 Clase: GestorImpresionImpresión .....	408
5.6.1.1.2 Clase: GeneradorListado .....	408
5.6.1.1.3 Clase: GeneradorInforme .....	409

5.6.1.1.4 Clase: Listado .....	410
5.6.1.1.5 Clase: Informe .....	410
<b>5.6.1.2 ESPECIFICACIÓN DEL MODELO OBJETO DE ANÁLISIS – RELACIÓN .....</b>	<b>411</b>
5.6.1.2.1 Relación: GestorImpresión – GeneradorListado .....	412
5.6.1.2.2 Relación: GestorImpresión – GeneradorInforme .....	412
5.6.1.2.3 Relación: GeneradorListado – Listado .....	412
5.6.1.2.4 Relación: GeneradorInforme – Informe .....	413
<b>5.6.1.3 DIAGRAMA DE CLASES DE ANÁLISIS DEL SISTEMA .....</b>	<b>413</b>
<b>5.6.1.4 ESPECIFICACIÓN DEL MODELO OBJETO DE ANÁLISIS - COMPORTAMIENTO .....</b>	<b>414</b>
5.6.1.4.1 Diagrama de Colaboración: Imprimir Listado .....	415
5.6.1.4.1 Diagrama de Colaboración: Imprimir Informe .....	415
<b>DISEÑO DEL SISTEMA .....</b>	<b>417</b>
<b>5.7.2.1 CLASE IMPRESIÓN .....</b>	<b>419</b>
<b>5.7.2.2 CLASE IMPRESIÓNAUTOR.....</b>	<b>420</b>
<b>5.7.2.3 CLASE IMPRESIÓNCV .....</b>	<b>423</b>
<b>5.7.2.4 CLASE IMPRESIÓNLISTADOAUTORES .....</b>	<b>428</b>
<b>5.7.2.5 CLASE IMPRESIÓNLISTADOINTEGRANTES .....</b>	<b>429</b>
<b>5.7.2.6 CLASE IMPRESIÓNACTA.....</b>	<b>430</b>
<b>5.7.2.7 CLASE IMPRESIÓNARTICULO .....</b>	<b>432</b>
<b>5.7.2.8 CLASE IMPRESIÓNCAPITULO .....</b>	<b>433</b>

<b>5.7.2.9 CLASE IMPRESIÓN CONGRESO .....</b>	<b>435</b>
<b>5.7.2.10 CLASE IMPRESIÓN LIBRO .....</b>	<b>436</b>
<b>5.7.2.11 CLASE IMPRESIÓN REVISTA .....</b>	<b>438</b>
<b>5.7.3.1 DIAGRAMA DE SECUENCIA .....</b>	<b>440</b>
 IMPLEMENTACIÓN DEL SISTEMA .....	 442
 PRUEBAS DEL SISTEMA .....	 443
<b>5.9.1 PRUEBAS DE CAJA BLANCA .....</b>	<b>444</b>
<b>5.9.2 PRUEBAS DE CAJA NEGRA .....</b>	<b>444</b>
 BIBLIOGRAFÍA Y REFERENCIAS WEB .....	 446

# ÍNDICE DE FIGURAS

## I

FIGURA 1.1.1: DIAGRAMA ENTIDAD-RELACIÓN BÁSICO .....	27
FIGURA 1.1.2: DIAGRAMA ENTIDAD-RELACIÓN CURÍCULUM VITAE (VISTA 1) .....	42
FIGURA 1.1.3: DIAGRAMA ENTIDAD-RELACIÓN CURÍCULUM VITAE (VISTA 2) .....	43
FIGURA 1.1.4: DIAGRAMA ENTIDAD-RELACIÓN CURÍCULUM VITAE (VISTA 2) .....	55
FIGURA 1.1.5: DIAGRAMA ENTIDAD RELACIÓN DE CURÍCULUM VITAE.....	62
FIGURA 2.5.1: DIAGRAMA DE CASOS DE USO DE TRAZO GRUESO .....	181
FIGURA 2.5.1: CASO DE USO CONECTAR BD .....	183
FIGURA 2.5.2: CASO DE USO CAMBIAR USUARIO.....	185
FIGURA 2.5.3: CASO DE USO DESCONECTAR BD .....	186
FIGURA 2.5.4: DIAGRAMA DE LA VENTANA PRINCIPAL.....	189
FIGURA 2.5.5: DIAGRAMA DE LA ZONA DE TRABAJO .....	190
FIGURA 2.6.1: DIAGRAMA DE PAQUETES.....	197
FIGURA 2.6.2: DIAGRAMA DE PAQUETES DETALLADO .....	198
FIGURA 2.6.3: REPRESENTACIÓN UML DE LA CLASE APLICACION .....	199
FIGURA 2.6.4: REPRESENTACIÓN UML DE LA CLASE CONEXIONBD .....	201
FIGURA 2.6.5: REPRESENTACIÓN UML DE LA CLASE MARCOAPLICACION .....	204
FIGURA 2.6.6: REPRESENTACIÓN UML DE LA CLASE BARRAESTADO .....	205
FIGURA 2.6.7: REPRESENTACIÓN UML DE LA CLASE MARCOACERCADE.....	206
FIGURA 2.6.8: REPRESENTACIÓN UML DE LA CLASE MARCOCONECTAR .....	207
FIGURA 2.6.9: REPRESENTACIÓN UML DE LA CLASE PANELAYUDA.....	208

FIGURA 2.6.10: REPRESENTACIÓN UML DE LA CLASE PANELGENERAL .....	210
FIGURA 2.6.11: REPRESENTACIÓN UML DE LA CLASE PANELFRONTAL .....	211
FIGURA 2.6.12: REPRESENTACIÓN UML DE LA CLASE ARBOLNAVEGACION .....	212
FIGURA 2.6.13: REPRESENTACIÓN UML DE LA CLASE NODOARBOL .....	213
FIGURA 2.6.14: REPRESENTACIÓN UML DE LA CLASE REPRESENTADORNODOSARBOL .....	214
FIGURA 2.6.15: DIAGRAMA DE CLASES .....	215
FIGURA 2.6.16: VENTANA PRINCIPAL .....	216
FIGURA 2.6.17: VENTANA PRINCIPAL INACTIVA .....	217
FIGURA 2.6.18: BARRA DE MENÚS .....	217
FIGURA 2.6.19: MENÚ CONEXIÓN .....	218
FIGURA 2.6.20: MENÚ OPCIONES .....	219
FIGURA 2.6.21: MENÚ VER .....	220
FIGURA 2.6.22: MENÚ AYUDA .....	220
FIGURA 2.6.23: VENTANA ACERCA DE .....	221
FIGURA 2.6.24: BARRA DE HERRAMIENTAS .....	221
FIGURA 2.6.25: ÁRBOL DE NAVEGACIÓN .....	223
FIGURA 2.6.26: BARRA DE ESTADO .....	224
FIGURA 2.6.27: VENTANA DE CONEXIÓN .....	224
FIGURA 2.6.28: MENSAJE DE ERROR EN LA VENTANA DE CONEXIÓN .....	225
FIGURA 2.6.29: MENSAJE DE ERROR DURANTE LA CONEXIÓN .....	225
FIGURA 2.6.30: VENTANA DE CAMBIO DE USUARIO .....	226
FIGURA 2.7.1: INSERCIÓN DE ARTÍCULO .....	230
FIGURA 2.7.2: VER ARTÍCULOS .....	231
FIGURA 2.7.3: INSERCIÓN DE LIBRO .....	231
FIGURA 2.7.4: VER LIBROS .....	232
FIGURA 2.7.5: INSERCIÓN DE CAPÍTULO .....	232
FIGURA 2.7.6: VER CAPÍTULOS .....	232
FIGURA 2.7.7: INSERCIÓN DE ACTA .....	233
FIGURA 2.7.8: VER ACTAS (TABLA ACTA) .....	233
FIGURA 2.7.9: INSERCIÓN DE REVISTA .....	234
FIGURA 2.7.10: VER REVISTAS .....	234
FIGURA 2.7.11: INSERCIÓN DE AUTOR .....	234
FIGURA 2.7.12: VER AUTOR (TABLA PERSONALINFORMATION) .....	235

FIGURA 2.7.13: VER AUTOR (TABLA AUTHOR) .....	235
FIGURA 2.7.14: INSERCIÓN DE UN CONGRESO .....	235
FIGURA 2.7.15: VER REVISTAS .....	236
FIGURA 3.5.1: DIAGRAMA DE CASOS DE USO: CONTEXTO DEL SISTEMA .....	253
FIGURA 3.5.2: DIAGRAMA DE CASOS DE USO: GESTIÓN DE LOS FORMULARIOS DE ENTRADA.....	255
FIGURA 3.5.3: DIAGRAMA DE CASOS DE USO: FORMULARIO MIEMBRO.....	257
FIGURA 3.5.4: DIAGRAMA DE CASOS DE USO: FORMULARIO REVISTA .....	259
FIGURA 3.5.5: DIAGRAMA DE CASOS DE USO: FORMULARIO ARTÍCULO.....	261
FIGURA 3.5.6: DIAGRAMA DE CASOS DE USO: FORMULARIO LIBRO.....	263
FIGURA 3.5.7: DIAGRAMA DE CASOS DE USO: FORMULARIO CAPÍTULO .....	265
FIGURA 3.5.8: DIAGRAMA DE CASOS DE USO: FORMULARIO MIEMBRO.....	267
FIGURA 3.5.9: DIAGRAMA DE CASOS DE USO: FORMULARIO ACTA.....	269
FIGURA 3.5.10: DIAGRAMA DE CASOS DE USO: FORMULARIO AUTOR .....	270
FIGURA 3.6.1: RELACIÓN: GESTOR INFORMACION – GENERADOR FE.....	274
FIGURA 3.6.2: REPRESENTACIÓN EN UML DE LA CLASE: GENERADOR FE.....	275
FIGURA 3.6.3: REPRESENTACIÓN EN UML DE LA CLASE: FORMULARIO.....	275
FIGURA 3.6.4: RELACIÓN: GESTOR INFORMACION – GENERADOR FE.....	276
FIGURA 3.6.5: RELACIÓN: GENERADOR FE – FORMULARIO .....	277
FIGURA 3.6.6: DIAGRAMA DE CLASES DEL SUBSISTEMA.....	278
FIGURA 3.6.7: DIAGRAMA DE COLABORACIÓN: INSERTAR NUEVA INFORMACIÓN .....	280
FIGURA 3.7.1: REPRESENTACIÓN EN UML DEL OBJETO: FORMULARIO ENTRADA ACTAS .....	283
FIGURA 3.7.2: REPRESENTACIÓN EN UML DE LA CLASE: FORMULARIO ENTRADA ARTICULO.....	284
FIGURA 3.7.3: REPRESENTACIÓN EN UML DEL OBJETO: FORMULARIO ENTRADA AUTOR.....	285
FIGURA 3.7.4: REPRESENTACIÓN EN UML DEL OBJETO: FORMULARIO ENTRADA CAPITULO.....	286
FIGURA 3.7.5: REPRESENTACIÓN EN UML DEL OBJETO: FORMULARIO ENTRADA CONGRESO .....	287
FIGURA 3.7.6: REPRESENTACIÓN EN UML DEL OBJETO: FORMULARIO ENTRADA LIBRO .....	288
FIGURA 3.7.7: REPRESENTACIÓN EN UML DEL OBJETO: FORMULARIO ENTRADA LIBRO CAPITULO.....	288
FIGURA 3.7.8: REPRESENTACIÓN EN UML DEL OBJETO: FORMULARIO ENTRADA MIEMBRO .....	289
FIGURA 3.7.9: REPRESENTACIÓN EN UML DEL OBJETO: FORMULARIO ENTRADA REVISTA.....	290
FIGURA 3.7.10: DIAGRAMA DE OBJETOS DEL SISTEMA.....	291
FIGURA 3.7.11: REPRESENTACIÓN EN UML DEL OBJETO: FORMULARIO ENTRADA REVISTA.....	293
FIGURA 3.7.12: REPRESENTACIÓN EN UML DEL PAQUETE: SISTEMA .....	295
FIGURA 3.7.13: REPRESENTACIÓN EN UML DEL PAQUETE: FORMULARIOS ENTRADA.....	295

FIGURA 3.8.1: JERARQUÍA VENTANA PRINCIPAL ACTAS .....	299
FIGURA 3.8.2: VENTANA PRINCIPAL ACTA .....	299
FIGURA 3.8.3: COMBO DE EJEMPLO .....	300
FIGURA 3.8.4: VENTANA PRINCIPAL INTEGRANTE .....	301
FIGURA 3.8.5: JERARQUÍA VENTANA PRINCIPAL LIBRO .....	302
FIGURA 3.8.6: VENTANA PRINCIPAL LIBRO .....	302
FIGURA 3.8.7: DIÁLOGO SELECCIÓN DE DOCUMENTO.....	303
FIGURA 3.8.8: JERARQUÍA VENTANA PRINCIPAL CAPÍTULO .....	304
FIGURA 3.8.9: VENTANA PRINCIPAL CAPÍTULO.....	304
FIGURA 3.8.10: JERARQUÍA VENTANA PRINCIPAL ARTÍCULO .....	305
FIGURA 3.8.11: VENTANA PRINCIPAL ARTÍCULO.....	306
FIGURA 3.8.12: DIÁLOGO SELECCIÓN DE REVISTA .....	307
FIGURA 3.8.13: VENTANA SECUNDARIA AUTOR .....	308
FIGURA 3.8.14: DIÁLOGO DE SELECCIÓN DE MIEMBRO .....	308
FIGURA 3.8.15: VENTANA SECUNDARIA CONGRESO.....	309
FIGURA 3.8.16: VENTANA SECUNDARIA REVISTA .....	310
FIGURA 3.8.17: VENTANA SECUNDARIA LIBRO - CAPÍTULO .....	311
FIGURA 4.5.1: DIAGRAMA CASO DE USO DE INICIAL .....	326
FIGURA 4.5.2: DIAGRAMA CASO DE USO "DEFINIR BUSQUEDA" .....	327
FIGURA 4.5.3: DIAGRAMA CASO DE USO "FORMATEAR SALIDA" .....	329
FIGURA 4.6.1. DIAGRAMA DE CLASES DE ANÁLISIS.....	333
FIGURA 4.7.1 DIAGRAMA DE PAQUETES .....	338
FIGURA 4.7.2. DIAGRAMA DE CLASE DEL PAQUETE "PRINCIPAL" .....	343
FIGURA 4.7.3. DIAGRAMA DE CLASE DEL PAQUETE "PRINCIPA.FORMULARIOSL" .....	346
FIGURA4.7.4.DIAGRAMA DE CLASE DEL PAQUETE "PRINCIPA.FORMULARIOS.ENTRADA" .....	348
FIGURA4.7.5:DIAGRAMA DE CLASE DEL PAQUETE "PRINCIPA.FORMULARIOS.ENTRADA.RESOURCES" ....	350
FIGURA4.7.6:DIAGRAMA DE CLASE DEL PAQUETE PRINCIPAL.FORMULARIOS.VISUALIZACION .....	355
FIGURA4.7.7:DIAGRAMA DE CLASES DEL PAQUETE "PRINCIPAL.FORMULARIOS.VISUALIZACION.RESOURCES"	361
FIGURA 4.7.8: DIAGRAMA DE SECUENCIA "VISUALIZAR INFORMACIÓN".....	367
FIGURA 4.7.9: DIAGRAMA DE ACTIVIDAD "BUSCAR INFORMACIÓN".....	368
FIGURA 4.7.10: DIAGRAMA DE ACTIVIDAD "MOSTRAR DETALLE" .....	368
FIGURA 4.7.11: DIAGRAMA DE ACTIVIDAD "ORDENAR TUPLAS" .....	369

FIGURA 4.8.1 CAPTURA MARCO PRINCIPAL .....	371
FIGURA 4.8.2 CAPTURA MENÚ .....	372
FIGURA 4.8.3. EJEMPLO DE EJECUCIÓN.....	373
FIGURA 4.8.4. MATRIZ DE RESULTADOS .....	373
FIGURA 4.8.5: FORMULARIO DE VISUALIZACIÓN DE UNA REVISTA.....	374
FIGURA 4.8.6: FORMULARIO DE VISUALIZACIÓN DE UN MIEMBRO .....	375
FIGURA 5.5.1: DIAGRAMA DE CASOS DE USO DE TRAZO GRUESO. PARTE 1 .....	400
FIGURA 5.5.2: DIAGRAMA DE CASOS DE USO DE TRAZO GRUESO. PARTE 2 .....	400
FIGURA 5.5.3: DIAGRAMA CASO DE USO <i>IMPRIMIR INFORMACIÓN AUTOR</i> .....	401
FIGURA 5.5.4: DIAGRAMA CASO DE USO <i>IMPRIMIR LISTADO AUTORES</i> .....	403
FIGURA 5.6.1: CLASE ANÁLISIS <i>GESTORIMPRESIÓN</i> .....	408
FIGURA 5.6.2: CLASE ANÁLISIS <i>GENERADORLISTADO</i> .....	409
FIGURA 5.6.3: CLASE ANÁLISIS <i>GENERADORINFORME</i> .....	410
FIGURA 5.6.4: CLASE ANÁLISIS <i>LISTADO</i> .....	410
FIGURA 5.6.5: CLASE ANÁLISIS <i>INFORME</i> .....	411
FIGURA 5.6.6: RELACIÓN: <i>GESTORIMPRESIÓN – GENERADORLISTADO</i> .....	412
FIGURA 5.6.7: RELACIÓN: <i>GESTORIMPRESIÓN – GENERADORINFORME</i> .....	412
FIGURA 5.6.8: RELACIÓN: <i>GENERADORLISTADO – LISTADO</i> .....	413
FIGURA 5.6.9: RELACIÓN: <i>GENERADORINFORME – INFORME</i> .....	413
FIGURA 5.6.10: DIAGRAMA DE CLASES DE ANÁLISIS DEL SUBSISTEMA .....	414
FIGURA 5.6.11: DIAGRAMA DE COLABORACIÓN <i>IMPRIMIR LISTADO</i> .....	415
FIGURA 5.6.12: DIAGRAMA DE COLABORACIÓN <i>IMPRIMIR INFORME</i> .....	416
FIGURA 5.7.1. DIAGRAMA DE CLASES DE DISEÑO DEL SUBSISTEMA.....	419
FIGURA 5.7.2: DIAGRAMA DE SECUENCIA <i>IMPRIMIR INFORME</i> .....	440



# ÍNDICE DE TABLAS

TABLA 1.1.1: TIPO DE ENTIDADES .....	10
TABLA 1.1.2: TIPO DE INTERRELACIONES.....	14
TABLA 1.1.3: ATRIBUTOS ENTIDAD ARTICLE .....	15
TABLA 1.1.4: ATRIBUTOS ENTIDAD JOURNAL .....	16
TABLA 1.1.5: ATRIBUTOS ENTIDAD JOURNAL .....	17
TABLA 1.1.6: ATRIBUTOS ENTIDAD BOOKOFCHAPTER.....	18
TABLA 1.1.7: ATRIBUTOS ENTIDAD CHAPTER.....	19
TABLA 1.1.8: ATRIBUTOS ENTIDAD MEETING .....	19
TABLA 1.1.9: ATRIBUTOS ENTIDAD ACTA.....	21
TABLA 1.1.10: ATRIBUTOS ENTIDAD COUNTRY .....	22
TABLA 1.1.11: ATRIBUTOS ENTIDAD TYPE.....	22
TABLA 1.1.12: ATRIBUTOS ENTIDAD STATUS.....	22
TABLA 1.1.13: ATRIBUTOS ENTIDAD TREATMENT .....	23
TABLA 1.1.14: ATRIBUTOS ENTIDAD PERSONALINFORMATION .....	23
TABLA 1.1.15: ATRIBUTOS ENTIDAD AUTHOR .....	24
TABLA 1.1.16: ATRIBUTOS ENTIDAD MEMBER.....	24
TABLA 1.1.17: ATRIBUTOS IDENTIFICADORES.....	26
TABLA 1.1.18: DICCIONARIO DE INTERRELACIONES .....	33
TABLA 1.1.19: ATRIBUTOS ENTIDAD DEGREE.....	34
TABLA 1.1.20: ATRIBUTOS ENTIDAD PROFESSIONAL STATUS.....	34
TABLA 1.1.21: ATRIBUTOS ENTIDAD SPECIALIZATION .....	35

TABLA 1.1.22: ATRIBUTOS ENTIDAD ACTIVITY.....	35
TABLA 1.1.23: ATRIBUTOS ENTIDAD LANGUAGE.....	36
TABLA 1.1.24: ATRIBUTOS ENTIDAD FINANCE PROJECT .....	37
TABLA 1.1.25: ATRIBUTOS ENTIDAD RESEARCH CONTRACT .....	37
TABLA 1.1.26: ATRIBUTOS ENTIDAD PATENT .....	38
TABLA 1.1.27: ATRIBUTOS ENTIDAD FOREIGN INSTITUTION.....	38
TABLA 1.1.28: ATRIBUTOS ENTIDAD THESIS .....	39
TABLA 1.1.29: ATRIBUTOS POSITION TYPE.....	39
TABLA 1.1.30: ATRIBUTOS OTHER MERITS .....	40
TABLA 1.1.31: ATRIBUTOS IDENTIFICADORES .....	40
TABLA 1.1.32: TABLA ARTICLE .....	63
TABLA 1.1.33: TABLA JOURNAL .....	63
TABLA 1.1.34: TABLA ISI .....	63
TABLA 1.1.35: TABLA BOOK .....	64
TABLA 1.1.36: TABLA BOOK OF CHAPTERS .....	64
TABLA 1.1.37: TABLA CHAPTER.....	64
TABLA 1.1.38: TABLA MEETING .....	65
TABLA 1.1.39: TABLA ACTA.....	65
TABLA 1.1.40: TABLA COUNTRY .....	66
TABLA 1.1.41: TABLA TYPE.....	66
TABLA 1.1.42: TABLA TREATMENT .....	66
TABLA 1.1.43: TABLA STATUS.....	66
TABLA 1.1.44: TABLA PERSONAL INFORMATION .....	66
TABLA 1.1.45: TABLA AUTHOR .....	67
TABLA 1.1.46: TABLA MEMBER.....	67
TABLA 1.1.47: TABLA ARTICLE AUTHOR .....	68
TABLA 1.1.48: TABLA ACTAS AUTHOR.....	68
TABLA 1.1.49: TABLA BOOKS AUTHOR.....	68
TABLA 1.1.50: TABLA CHAPTER AUTHOR .....	68
TABLA 1.1.51: TABLA DEGREE .....	69
TABLA 1.1.52: TABLA PROFESSIONAL STATUS.....	69
TABLA 1.1.53: TABLA SPECIALIZATION .....	69
TABLA 1.1.54: TABLA ACTIVITY .....	69

TABLA 1.1.55: TABLA LANGUAGE.....	70
TABLA 1.1.56: TABLA FINANCE PROJECT .....	70
TABLA 1.1.57: TABLA FINANCEPROJECT MEMBER .....	70
TABLA 1.1.58: TABLA RESEARCH CONTRACT .....	71
TABLA 1.1.59: TABLA RESEARCHCONTRACT MEMBER .....	71
TABLA 1.1.60: TABLA PATENT .....	71
TABLA 1.1.61: TABLA PATENT MEMBER .....	72
TABLA 1.1.62: TABLA FOREIGN INSTITUTION.....	72
TABLA 1.1.63: TABLA THESIS .....	72
TABLA 1.1.64: TABLA THESIS MEMBER .....	73
TABLA 1.1.65: TABLA POSITIONTYPE .....	73
TABLA 1.1.66: TABLA OTHERMERITS .....	73
TABLA 2.5.1: MATRIZ REQUISITOS / CASOS DE USO .....	182
TABLA 2.5.1: ESPECIFICACIÓN CASO DE USO CONECTAR BD .....	184
TABLA 2.5.2: ESPECIFICACIÓN CASO DE USO CAMBIAR USUARIO .....	185
TABLA 2.5.3: ESPECIFICACIÓN CASO DE USO DESCONECTAR BD .....	187
TABLA 2.6.1: DESCRIPCIÓN DE LA CLASE APLICACIÓN .....	199
TABLA 2.6.2: DESCRIPCIÓN DE LA CLASE CONEXIONBD .....	200
TABLA 2.6.3: DESCRIPCIÓN DE LA CLASE MARCOPRINCIPAL .....	201
TABLA 2.6.4: DESCRIPCIÓN DE LA CLASE BARRAESTADO.....	204
TABLA 2.6.5: DESCRIPCIÓN DE LA CLASE MARCOACERCADE .....	205
TABLA 2.6.6: DESCRIPCIÓN DE LA CLASE MARCOCONECTAR .....	206
TABLA 2.6.7: DESCRIPCIÓN DE LA CLASE PANELAYUDA .....	208
TABLA 2.6.8: DESCRIPCIÓN DE LA CLASE PANELGENERAL .....	209
TABLA 2.6.9: DESCRIPCIÓN DE LA CLASE PANELFRONTAL.....	210
TABLA 2.6.10: DESCRIPCIÓN DE LA CLASE ARBOLNAVEGACION.....	211
TABLA 2.6.11: DESCRIPCIÓN DE LA CLASE NODOARBOL .....	212
TABLA 2.6.12: DESCRIPCIÓN DE LA CLASE REPRESENTADORNODOARBOL .....	213
TABLA 2.6.13: FUNCIONALIDAD DE LOS BOTONES DE LA BARRA DE HERRAMIENTAS .....	222
TABLA 3.5.1: DESCRIPCIÓN ACTOR 1 .....	250
TABLA 3.5.2: DESCRIPCIÓN ACTOR 2.....	251
TABLA 3.6.1: ESPECIFICACIÓN DE LA CLASE DE ANÁLISIS: GESTORINFORMACIÓN.....	273
TABLA 3.6. 2: ESPECIFICACIÓN DE LA CLASE DE ANÁLISIS: GENERADORFE .....	274

TABLA 3.6. 3: ESPECIFICACIÓN DE LA CLASE DE ANÁLISIS: FORMULARIO .....	275
TABLA 3.7. 1: OBJETO: FORMULARIOENTRADAACTAS .....	282
TABLA 3.7.2: OBJETO: FORMULARIOENTRADAARTICULO .....	283
TABLA 3.7.3: OBJETO: FORMULARIOENTRADAAUTOR.....	284
TABLA 3.7.4: OBJETO: FORMULARIOENTRADACAPITULO .....	285
TABLA 3.7.5: OBJETO: FORMULARIOENTRADACONGRESO .....	286
TABLA 3.7.6: OBJETO: FORMULARIOENTRADALIBRO .....	287
TABLA 3.7.7: OBJETO: FORMULARIOENTRADALIBROCAPITULO .....	288
TABLA 3.7.8: OBJETO: FORMULARIOENTRADAMIEMBRO .....	289
TABLA 3.7.9: OBJETO: FORMULARIOENTRADAREVISTA .....	289
TABLA4.5.1: DESCRIPCIÓN DEL ACTOR CLIENTE.....	324
TABLA 4.5.2 DESCRIPCIÓN INICIAL CASO DE USO “DEFINIR BÚSQUEDA” .....	325
TABLA 4.5.3 DESCRIPCIÓN INICIAL CASO DE USO “FORMATEAR SALIDA” .....	325
TABLA 4.5.4 DESCRIPCIÓN INICIAL CASO DE USO “MOSTRAR RESULTADO” .....	326
TABLA 4.5.5. CASO DE USO “SELECCIONAR TABLA” .....	327
TABLA 4.5.6. CASO DE USO “SELECCIONAR TABLA”.....	328
TABLA 4.5.7. CASO DE USO “SELECCIONAR TABLA” .....	328
TABLA 4.5.8. CASO DE USO “ORDENAR TUPLAS” .....	330
TABLA 4.5.9. CASO DE USO “SELECCIONAR TUPLAS” .....	330
TABLA 4.5.10. CASO DE USO “MOSTRAR DETALLE REGISTRO” .....	331
TABLA 4.6.1. ESPECIFICACIÓN DE LA CLASE PANELPRINCIPAL.....	333
TABLA 4.6.2. ESPECIFICACIÓN DE LA CLASE PANELSUPERIOR.....	333
TABLA 4.6.3. ESPECIFICACIÓN DE LA CLASE MOSTRARENTRADA .....	334
TABLA 4.6.4. ESPECIFICACIÓN DE LA CLASE PANELINFERIOR .....	334
TABLA 4.6.5 ESPECIFICACIÓN DE LA CLASE MOSTRAR SALIDA .....	335
TABLA 4.6.6 ESPECIFICACIÓN DE LA CLASE BUSCARINFORMACION .....	335
TABLA 4.6.7 ESPECIFICACIÓN DE LA CLASE MOSTRARINFORMACION.....	336
TABLA 5.5.1: DESCRIPCIÓN ACTOR <i>USUARIO</i> .....	394
TABLA 5.5.2: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR INFORMACIÓN AUTOR</i> .....	396
TABLA 5.5.3: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR LISTADO AUTORES</i> .....	396
TABLA 5.5.4: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR CURRÍCULO VITAE</i> .....	396
TABLA 5.5.5: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR LISTADO INTEGRANTES</i> .....	396
TABLA 5.5.6: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR INFORMACIÓN ACTA</i> .....	397

TABLA 5.5.7: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR LISTADO ACTAS</i> .....	397
TABLA 5.5.8: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR INFORMACIÓN ARTÍCULO</i> .....	397
TABLA 5.5.9: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR LISTADO ACTAS</i> .....	397
TABLA 5.5.10: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR INFORMACIÓN CAPÍTULO</i> .....	397
TABLA 5.5.11: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR LISTADO CAPÍTULOS</i> .....	398
TABLA 5.5.12: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR INFORMACIÓN CONGRESO</i> .....	398
TABLA 5.5.13: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR LISTADO CONGRESOS</i> .....	398
TABLA 5.5.14: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR INFORMACIÓN LIBRO</i> .....	398
TABLA 5.5.15: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR LISTADO LIBROS</i> .....	398
TABLA 5.5.16 DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR INFORMACIÓN REVISTA</i> .....	399
TABLA 5.5.17: DESCRIPCIÓN DEL CASO DE USO <i>IMPRIMIR LISTADO REVISTAS</i> .....	399
TABLA 5.5.18: DESCRIPCIÓN DEL CASO DE USO <i>CREAR, FORMATEAR Y MOSTRAR PDF</i> .....	399
TABLA 5.5.19. REFINAMIENTO CASO DE USO <i>IMPRIMIR INFORMACIÓN AUTOR</i> .....	402
TABLA 5.5.20. REFINAMIENTO CASO DE USO <i>IMPRIMIR LISTADO AUTORES</i> .....	403
TABLA 5.5.21: MATRIZ REQUISITOS / CASOS DE USO .....	404
TABLA 5.6.1: ESPECIFICACIÓN DE LA CLASE DE ANÁLISIS: GESTORINFORMACIÓN .....	408
TABLA 5.6.2: ESPECIFICACIÓN DE LA CLASE DE ANÁLISIS: GENERADORLISTADO .....	408
TABLA 5.6.3: ESPECIFICACIÓN DE LA CLASE DE ANÁLISIS: GENERADORINFORME.....	409
TABLA 5.6.4: ESPECIFICACIÓN DE LA CLASE DE ANÁLISIS: LISTADO .....	410
TABLA 5.6.5: ESPECIFICACIÓN DE LA CLASE DE ANÁLISIS: INFORME .....	410

# 1. Base de Datos

# 1.1

## BASE DE DATOS

---

### 1.1. INTRODUCCIÓN

En este documento se describirá la representación de la información del sistema Gestión de información para grupos de investigación (GIGI). Desde la creación de la base de datos, hasta la descripción y desarrollo de la funcionalidad requerida por dicho sistema de información.

Además se describirá el proceso seguido para la creación del esquema de la base de datos y los procedimientos de manipulación del sistema de información GIGI, creado en MySQL.

Además se debe verificar el correcto funcionamiento del sistema de información creado sobre la base de datos.

## 1.2. DEFINICIÓN DEL PROBLEMA

GIGI es un sistema de gestión de la información para los grupos de investigación que permite gestionar los datos de los miembros del mismo y las publicaciones realizadas por los mismos.

El sistema deberá llevar a cabo la gestión de los miembros de un grupo de investigación de los que se desea mantener información personal, así como sus aportaciones en publicaciones contrastadas en diferentes medios. De esta forma también deberá mantener información sobre los autores de publicaciones relacionadas con el grupo de trabajo, y por tanto tendremos autores que sean miembros del grupo de trabajo, y autores que no lo sean.

Por otro lado se pueden tener cuatro tipos de publicaciones por las que se puede considerar autor, estos son:

- haber publicado algún artículo en alguna revista contrastada.
- haber publicado algún acta en algún congreso de interés para el grupo.
- haber participado con algún capítulo en la creación de un libro por capítulos.
- haber creado un libro, bien, de forma común o particular.

Para cada uno de estos tipos de publicaciones se deseará almacenar información en la base de datos, esta información se detallará mas adelante en el apartado 5 de este documento.

Además se desea almacenar información de los congresos de interés para el grupo de investigación, así como de los libros de actas que se generan a partir de dichos congresos o jornadas.

Por otra parte se deberá mantener una serie de tablas maestras para la gestión de la configuración del sistema, como son los diferentes tratamientos personales, académicos y profesionales que se apliquen tanto a los miembros del grupo de investigación como a los autores de los que se tenga registro en el sistema.



Otra de estas tablas maestras almacenará información de los países que puedan afectar a los diferentes elementos del sistema.

Por otro lado se deberá llevar un control sobre la incidencia de las revistas en el ámbito científico mediante el estudio y almacenamiento de diversos factores como el factor de impacto de dicha publicación.

### 1.3. RESTRICCIONES

Para el desarrollo y explotación del sistema será necesario desarrollarlo exclusivamente utilizando herramientas y tecnologías software de dominio público y libre acceso, de forma que no repercuta en la necesidad de compra de licencias o derechos a terceros.

Siguiendo con esta línea, el cliente dispone como recurso, del software mySQL como sistema gestor de base de datos (SGBD) operativo sobre una máquina del mismo, por lo que en el sistema deberá hacer uso de éste para el mantenimiento de bases de datos. Así mismo, será interesante que se permita el acceso remoto a dicho SGBD, para permitir la utilización de la aplicación desde diversos puntos de forma que el sistema esté distribuido.

## 1.4. RECURSOS

Para el desarrollo del sistema de base de datos se ha utilizado como sistema gestor de base de datos (SGBD) la aplicación de libre distribución MySQL la cuál almacena y mantiene la base de datos que resulte. Se ha elegido esta solución dado su extendido uso, abundante documentación, consigue satisfacer las necesidades que impone el problema y principalmente el coste nulo de su utilización y explotación.

## 1.5. ESPECIFICACIÓN DE REQUISITOS

A continuación pasamos a describir tanto el modelo conceptual como el funcional, para la creación de la base de datos correspondiente al sistema de información que soporta GIGI.

Se realizará una descripción detallada tanto de las entidades como las relaciones que conforman el modelo conceptual, para posteriormente detallar la funcionalidad deseada para el sistema GIGI.

### 1.5.1 Modelo Conceptual

#### 1.5.1.1 Análisis de los Tipos de Entidad del “Diagrama Conceptual Básico”

Tipo de entidad **Articles**: el cual representa al objeto del mundo real *“cada uno de las publicaciones en revistas relacionadas con el grupo de investigación”*. Para la identificación de cada artículo se utilizará un código.

Tipo de entidad **Journal**: La cual representa al objeto del mundo real *“la información de las revistas relacionadas con el grupo de investigación”*, así representará aquellas revistas en las que el grupo de investigación tenga alguna publicación. Consideramos que cada Journal tiene un código ISSN único que identifica cada revista, por lo que para su identificación utilizaremos ISSN (representa el número de serie estándar internacional, International Standard Serial Number, de la revista), permitirá la identificación unívoca de las revistas. Además de este atributo existen otros como son: name (nombre de la revista.), abbreviation (abreviación del nombre que la representa), editorial (representa la empresa editorial encargada de la publicación de dicha revista), impact (representa el índice de impacto de la revista para un determinado año), year (representa el año de publicación de la revista al que aplicar el índice de impacto).

Tipo de entidad **Books**: el cual representa al objeto del mundo real *“publicación recogida y editada por una editorial con un fin educativo o comercial”*. Cada Book (libro) tendrá un código que lo diferencia de los demás libros y por el cual lo podremos identificar mediante el atributo code. Además cada libro tendrá otros atributos como son: isbn: Campo que representa el

número de libro estándar internacional (International Standard Book Number) que se le ha asignado al libro, title, year, pages, status, editorial, city, country, document.

Tipo de entidad **Meeting**: el cual representa el objeto del mundo real “congreso o acto en el que se presentan diversos trabajos de los investigadores, que quedarán recogidos en un libro de actas”. En cada Meeting se especificarán todos los datos identificativos tanto del acto como de las publicaciones presentadas en el mismo y recogidas en el libro de actas. Cada Meeting será único dentro del sistema

Cada Meeting tendrá un código que lo diferencia de los demás libros y por el cuál lo podremos identificar mediante el atributo code. Además cada libro tendrá otros atributos como son: name, abbreviation, organization, status, country, city, type, fromDate, toDate, deadline, intention, URL, ISBN, actaTitle, editorial, editors.

Tipo de entidad **Actas**: el cual representa al objeto del mundo real “cada una de las aportaciones en los libros de actas de los congresos (proceedings) relacionados con el grupo de investigación” por lo tanto para el tipo de entidad Actas podemos considerar los siguientes atributos: code, contribution, volume, initPage, endPage, status, document. De ellos el atributo code permitirá la identificación unívoca de los elementos aún cuando el congreso no se haya celebrado o tenga un estado prematuro.

El tipo de entidad Actas será débil por existencia con respecto al tipo de entidad Meeting. Ya que no tiene sentido la presencia de un acta, sin haberse realizado el congreso o jornadas correspondiente y por consiguiente estar dicha acta incluida en el libro de actas de dicho Meeting. Por tanto para identificar cada Acta dentro del sistema se utilizará sólo su propio código.

Tipo de entidad **Book Of Chapters**: El cual representa al objeto del mundo real “libro publicado como resultado de la agregación de distintos capítulos desarrollados por varios autores diferentes”. Cada libro de capítulos estará identificado por un código (Code). Además cada Book Of Chapters presentará los siguientes atributos: ISBN, title, year, pages, editors, editorial, city, country.

Tipo de entidad **Chapter**: El cual representa al objeto del mundo real “cada uno de los capítulos publicados en un libro de capítulos (Book Of Chapter) y desarrollado por un autor”. Cada capítulo estará identificado por un código (code). Además cada Chapter presentará los siguientes atributos: title, initPage, endPage, status, document, ISBN, Este campo es una referencia a alguno de los elementos disponibles en la tabla Book Of Chapter

El tipo de entidad Chapter será débil por existencia con respecto al tipo de entidad Book Of Chapter. Ya que no tiene sentido la presencia de un capítulo, sin haberse realizado el libro de capítulos correspondiente y por consiguiente estar dicho capítulo incluido en el libro de capítulos. Por tanto para identificar cada capítulo dentro del sistema se utilizará sólo su propio código.

Tipo de entidad **Personal Information**: el cual representa al objeto del mundo real “la información correspondiente a los datos personales de las personas que aparecen en alguna de las publicaciones relacionadas con el grupo de investigación o bien son miembros del grupo de investigación”.

Esta entidad estará identificada por el agregado de dos atributos como son name y familyName (correspondientes al nombre y apellidos). Además de estos atributos presenta los siguientes: treatment, category, university, address, city, state, country, ZIP, email, phone, fax, departament,

Tipo de entidad **Author**: el cual representa al objeto del mundo real “persona que ha realizado algún tipo de publicación contrastada”. Esta entidad será un subconjunto de las personas cuya información se encuentra recogida en Personal Information.

El tipo de entidad Author será débil por identificación con respecto al tipo de entidad Personal Information. Por tanto para identificar cada autor dentro del sistema se utilizará el agregado formado por los atributos name y familyName.

Tipo de entidad **Member**: El cual representa el objeto del mundo real “cada una de las personas que pertenecen como integrantes del grupo de investigación, tanto si presentan alguna publicación como si no”.

El tipo de entidad Member será débil por identificación con respecto al tipo de entidad Personal Information. Por tanto para identificar cada autor

dentro del sistema se utilizará el agregado formado por los atributos name y familyName. Además este tipo de entidad tendrá los siguientes atributos: email\_2, phone\_2, web, fromDate, toDate, image. También, para relacionar cada miembro con su currículum será necesario incluir los siguientes atributos: dni, birthDate, sex, numOfficial, postalCode y codeSpecialization, que será una referencia a una instancia del tipo de entidad Specialization, que será descrita posteriormente.

Tipo de entidad **Type**: el cuál representa al objeto del mundo real “tipo de jornadas o congresos que se celebran”. Especificará por tanto si un Meeting es Nacional o Internacional. El identificador de este tipo de entidad será el propio tipo.

Tipo de entidad **Status**: el cuál representa al objeto del mundo real “estado en el que se puede encontrar una publicación”. Este tipo de entidad incluirá el atributo status que representará el propio estado de la publicación.

Tipo de entidad **Country**: El cuál representa el objeto del mundo real “nombre de cada uno de los países o estados soberanos reconocidos por los organismos internacionales”. El identificador de este tipo de entidad será country, es decir, el propio nombre del país representado.

Tipo de entidad **Treatment**: el cuál representa al objeto del mundo real “cada uno de los diferentes tratamientos profesionales o académicos que se pueden aplicar a los autores o miembros del grupo de investigación”. Este tipo de entidad estará identificado por el propio atributo treatment que lo identificará unívocamente.

**Tabla 1.1.1:** Tipo de entidades

Entidad	Descripción	Alias	Instancias
Journal	la información de las revistas relacionadas con el grupo de investigación	Journal	1
Article	cada uno de las publicaciones en revistas relacionadas con el grupo de investigación	Article	2
Book	publicación recogida y editada por una editorial con un fin educativo o comercial	Book	1
Meeting	congreso o acto en el que se	Meeting	1

	presentan diversos trabajos de los investigadores, que quedarán recogidos en un libro de actas		
Acta	cada una de las aportaciones en los libros de actas de los congresos (proceedings) relacionados con el grupo de investigación	Acta	2
Book Of Chapter	libro publicado como resultado de la agregación de distintos capítulos desarrollados por varios autores diferentes	BookOfChapter	1
Chapter	cada uno de los capítulos publicados en un libro de capítulos y desarrollado por un autor	Chapter	2
Personal Information	la información correspondiente a los datos personales de las personas que aparecen en alguna de las publicaciones relacionadas con el grupo de investigación o bien son miembros del grupo de investigación.	Personal Information	2
Author	persona que ha realizado algún tipo de publicación contrastada	Author	6
Member	cada una de las personas que pertenecen como integrantes del grupo de investigación, tanto si presentan alguna publicación como si no	Member	2
Type	tipo de jornadas o congresos que se celebran	Type	1
Treatment	cada uno de los diferentes tratamientos profesionales o académicos que se pueden aplicar a los autores o miembros del grupo de investigación	Treatment	1
Status	estado en el que se puede encontrar una publicación	Status	1
Country	nombre de cada uno de los países o estados soberanos reconocidos por los organismos internacionales	Country	1

#### 1.5.1.2 Análisis de los tipos de interrelación del “Diagrama Conceptual Básico”

Los tipos de entidad antes mencionados se encuentran relacionados mediante los siguientes tipos de interrelación de la siguiente forma:

Tipo de interrelación **ARTICLE/JOURNAL (AR-J)**: el cual relaciona los tipos



de entidad *ARTICLE* y *JOURNAL*. El tipo de entidad *ARTICLE* participa con cardinalidades (1,1), ya que en un artículo sólo puede aparecer publicado en una revista (*Journal*), mientras que el tipo entidad *JOURNAL* participa con cardinalidades (1,n) ya que una revista puede publicar varios artículos.

Tipo de interrelación **ARTICLE/AUTHOR (AR-A)**: el cual relaciona los tipos de entidad *ARTICLE* y *AUTHOR*. El tipo de entidad *ARTICLE* participa con cardinalidades (1,n) ya que en cada artículo puede estar escrito por uno o por varios autores. Un autor puede tener publicados varios artículos, por lo que su cardinalidad será (0,n), la cardinalidad mínima 0 se debe a que un autor puede haber realizado cualquier tipo de publicación pero distinta de la categoría artículo.

Tipo de interrelación **BOOK/AUTHOR (B-A)**: el cual relaciona los tipos de entidad *BOOK* y *AUTHOR*. El tipo de entidad *BOOK* participa con cardinalidades (1,n) ya que en cada libro puede estar escrito por uno o por varios autores. Un autor puede tener publicados varios libros, por lo que su cardinalidad será (0,n), la cardinalidad mínima 0 se debe a que un autor puede haber realizado cualquier tipo de publicación pero distinta de la categoría de libro.

Tipo de interrelación **CHAPTER/ BOOK OF CHAPTER (CH-BC)**: el cual relaciona los tipos de entidad *CHAPTER* y *BOOK OF CHAPTER*. El tipo de entidad *BOOK OF CHAPTER* participa con cardinalidades (1,n) ya que en cada libro de capítulos puede estar compuesto de uno o por varios capítulos realizados por diferentes autores. Un capítulo sólo podrá ser recogido en un único libro de capítulos, por lo que su cardinalidad será (1,1). Este tipo de relación es débil por existencia, ya que no tiene sentido la existencia de un capítulo si no existe el libro de capítulos para el que fue realizado.

Tipo de interrelación **AUTHOR/CHAPTER (A-CH)**: el cual relaciona los tipos de entidad *AUTHOR* y *CHAPTER*. El tipo de entidad *CHAPTER* participa con cardinalidades (1,n) ya que en cada libro puede estar escrito por uno o por varios autores. Un autor puede tener publicados varios capítulos de libros, por lo que su cardinalidad será (0,n), la cardinalidad mínima 0 se debe a que un autor puede haber realizado cualquier tipo de publicación pero distinta de la categoría capítulo de libro.

Tipo de interrelación **ACTA/ MEETING (AC-M)**: el cual relaciona los tipos de entidad *ACTA* y *MEETING*. El tipo de entidad *MEETING* participa con cardinalidades (1,n) ya que en cada libro de actas o congreso recogerá una o varias actas realizados por diferentes autores. Un acta sólo podrá ser recogido en un único meeting, por lo que su cardinalidad será (1,1). Este tipo de relación es débil por existencia, ya que no tiene sentido la existencia de un acta si no existe el meeting para el que fue realizado.

Tipo de interrelación **AUTHOR/ACTA (A-AC)**: el cual relaciona los tipos de entidad *AUTHOR* y *ACTA*. El tipo de entidad *ACTA* participa con cardinalidades (1,n) ya que en cada acta puede estar escrito por uno o por varios autores. Un autor puede tener publicadas varias actas de congresos o conferencias, por lo que su cardinalidad será (0,n), la cardinalidad mínima 0 se debe a que un autor puede haber realizado cualquier tipo de publicación pero distinta de la categoría acta.

Tipo de interrelación **AUTHOR/PERSONAL INFORMATION (A-PI)**: el cual relaciona los tipos de entidad *AUTHOR* y *PERSONAL INFORMATION*. El tipo de entidad *AUTHOR* participa con cardinalidades (1,1) ya que en cada autor tiene sus datos recogidos sólo una vez en el registro general de datos personales, y además la cardinalidad mínima es 1, ya que obligatoriamente dichos datos del autor deben aparecer en tal registro. A su vez, Personal Information participa con cardinalidad será (0,1), la cardinalidad mínima 0 se debe a que los datos recogidos en información personal correspondan a un miembro del grupo de investigación que aun no ha realizado ninguna publicación en ninguna de las categorías.

Tipo de interrelación **PERSONAL INFORMATION/ MEMBER (PI-M)**: el cual relaciona los tipos de entidad *PERSONAL INFORMATION* y *MEMBER*. El tipo de entidad *MEMBER* participa con cardinalidades (1,1) ya que en cada miembro del grupo de investigación tiene sus datos recogidos sólo una vez en el registro general de datos personales, y además la cardinalidad mínima es 1, ya que obligatoriamente dichos datos del autor deben aparecer en tal registro. A su vez, Personal Information participa con cardinalidad será (0,1), la cardinalidad mínima 0 se debe a que los datos recogidos en información personal correspondan a un autor del que se deseen almacenar los datos, pero que no

pertenezca al grupo de investigación como miembro.

Tipo de interrelación **AUTHOR/ MEMBER (A-M)**: el cual relaciona los tipos de entidad *AUTHOR* y *MEMBER*. El tipo de entidad *MEMBER* participa con cardinalidades (0,1) ya que en cada miembro del grupo de investigación puede ser a su vez autor (ya que dicho miembro ha realizado alguna publicación) o no, y de esta forma sólo ser miembro del grupo de investigación. A su vez, *Author* participa con cardinalidad será (0,1), por las mismas razones expuestas antes, cada autor puede ser miembro (ya que puede pertenecer al grupo de investigación) o no pertenecer, y de esta forma ser sólo autor.

Estos tipos de interrelación se encuentran recogidos y especificados en el siguiente diccionario de tipos de relación:

**Tabla 1.1.2:** Tipo de interrelaciones

Entidad	Multiplicidad	Relación	Multiplicidad	Entidad
Article	1/n	AR-J	1/1	Journal
Article	0/n	AR-A	1/n	Author
Book	0/N	B-A	1/n	Author
BookOfChapther	1/1	CH-BC	1/N	Chapter
Author	1/N	A-CH	0/N	Chapter
Acta	1/N	AC-M	1/1	Meeting
Author	1/N	A-AC	0/N	Acta
PersonallInformation	1/1	A-PI	0/1	Author
PersonallInformation	1/1	PI-M	0/1	Member
Author	0/1	A-M	0/1	Member

### 1.5.1.3 Análisis de los atributos del “Diagrama Conceptual Básico”

De acuerdo con cada uno de los tipos de entidad e interrelaciones reconocidos anteriormente y de la estructura de cada uno de estos tipos, se puede representar este problema mediante la siguiente estructura sintáctica:

## Atributos de la entidad ARTICLE:

Tabla 1.1.3: Atributos Entidad Article

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
ARTICLE	code	Código único asignado automáticamente al ser introducida una nueva entrada. Este campo permitirá la identificación unívoca de los elementos de la tabla aún cuando el artículo aún no haya sido publicado.	Numérico (0-9)	SI	1
	Title	Título del artículo. Este campo no podrá ser nulo.	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	Year	Año de publicación del artículo.	Numérico 4 (0-9)	NO	1
	number	Número de publicación de la revista a lo largo de su historia.	Numérico 4 (0-9)	NO	1
	volume	Identifica el tomo o volumen de la revista donde se encuentra el artículo.	Numérico 4 (0-9)	NO	1
	initPage	Indica el número de la página de comienzo del artículo dentro de la revista a la que pertenece.	Numérico 4 (0-9)	NO	1
	endPage	Indica el número de la página de fin del artículo dentro de la revista a la que pertenece.	Numérico 4 (0-9)	NO	1
	document	Referencia al documento del artículo.	Referencia	NO	1
	status	Estado de la publicación del libro.	Enumerado	NO	1
	Issn	Número de serie estándar internacional de la revista en que ha sido publicado.	Referencia Journal	NO	D

## Atributos de la entidad JOURNAL:

Tabla 1.1.4: Atributos Entidad Journal

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
JOURNAL	issn	Número de serie estándar internacional de la revista. Este campo permitirá la identificación unívoca de los elementos	Cadena 10 caracteres (A-Z, 0-9)	SI	1
	name	Nombre de la revista.	Cadena 128 caracteres (A-Z, 0-9)	SI	1
	abbreviation	Abreviatura del nombre de la revista.	Cadena 64 caracteres (A-Z, 0-9)	SI	1
	editorial	Empresa que edita la publicación.	Cadena 128 caracteres (A-Z, 0-9)	NO	1
	year	Año de publicación del listado de los índices.	Numérico 4 (0-9)	SI	1
	totalCites	Número total de citas que ha recibido la revista.	Numérico 4 (0-9)	NO	1
	impactFactor	Índice de impacto que se le ha asignado a una publicación concreta en un determinado año.	Numérico Real 2 con 3 decimales (0-9)	SI	1
	inmedIndex	Media del número de veces que un artículo, publicado en un año concreto y una revista concreta, es citado en ese mismo año	Numérico Real 2 con 3 decimales (0-9)	NO	1
	numArticles	Número artículos publicados en dicha revista en un año concreto.	Numérico 6 (0-9)	NO	1
	citedHalfLife	representa la vida media de las citas de dicha revista.	Numérico Real 2 con 1 decimal (0-9)	NO	1

## Atributos de la entidad BOOK:

Tabla 1.1.5: Atributos Entidad Journal

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
Book	code	Código único asignado automáticamente al ser introducida una nueva entrada. Este campo permitirá la identificación unívoca de los elementos aún cuando el libro aún no haya sido publicado.	Numérico (0-9)	SI	1
	isbn	Número de libro estándar internacional que se le ha asignado al libro.	Cadena 16 caracteres (A-Z, 0-9)	SI	1
	title	Título del libro.	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	year	Año de publicación del libro.	Numérico 4 (0-9)	NO	1
	pages	Número de páginas que contiene el libro.	Numérico 5 (0-9)	NO	1
	status	Estado de la publicación del libro.	Enumerado	NO	1
	editorial	Nombre de la empresa que ha publicado el libro.	Cadena 64 caracteres (A-Z, 0-9)	NO	1
	city	Ciudad donde ha sido publicado el libro.	Cadena 32 caracteres (A-Z, 0-9)	NO	1
	country	País donde ha sido publicado el libro.	Enumerado	NO	1
	document	Referencia al documento del libro.	Referencia a documento	NO	1

## Atributos de la entidad BOOK OF CHAPTER:

Tabla 1.1.6: Atributos Entidad BookOfChapter

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
BOOK OF CHAPTER	code	Código único asignado automáticamente al ser introducida una nueva entrada. Permitirá la identificación unívoca de los elementos de la tabla aún cuando el libro aún no haya sido publicado.	Numérico (0-9)	SI	1
	isbn	Número de libro estándar internacional con el que ha sido publicado el libro.	Cadena 16 caracteres (A-Z, 0-9)	SI	1
	title	Título del libro.	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	year	Año de publicación del libro.	Numérico 4 (0-9)	NO	1
	pages	Número de páginas que contiene el libro.	Numérico 5 (0-9)	NO	1
	status	Estado de la publicación del libro.	Enumerado	NO	1
	editorial	Nombre de la empresa que ha publicado el libro.	Cadena 64 caracteres (A-Z, 0-9)	NO	1
	city	Ciudad donde ha sido publicado el libro.	Cadena 32 caracteres (A-Z, 0-9)	NO	1
	country	País donde ha sido publicado el libro.	Enumerado	NO	1
	editors	Nombre de las personas que han realizado la edición del libro.	Cadena 64 caracteres (A-Z, 0-9)	NO	1

Atributos de la entidad CHAPTER:

Tabla 1.1.7: Atributos Entidad Chapter

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
CHAPTER	code	Código único asignado automáticamente al ser introducida una nueva entrada. Permitirá la identificación unívoca de los elementos de la tabla aún cuando el libro aún no haya sido publicado.	Numérico (0-9)	SI	1
	title	Título del capítulo del libro.	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	initPage	Página de inicio del capítulo en el libro al que pertenece.	Numérico 4 (0-9)	NO	1
	endPage	Página de fin del capítulo en el libro al que pertenece.	Numérico 4 (0-9)	NO	1
	status	Estado de la publicación del libro.	Enumerado	NO	1
	document	Referencia al documento del libro.	Referencia a Documento	NO	1
	isbn	Número de libro estándar internacional. Es una referencia a alguno de los elementos disponibles en la tabla BOOK OF CHAPTERS.	Referencia a Libro de Capítulos	SI	D

Atributos de la entidad MEETING:

Tabla 1.1.8: Atributos Entidad Meeting

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
--------	----------	-------------	---------	-------	------



MEETING	code	Código único asignado automáticamente al ser introducida una nueva entrada aun. Permitirá la identificación unívoca de los elementos de la tabla aun cuando el congreso aún no se haya celebrado o tenga un estado prematuro.	Numérico (0-9)	SI	1
	name	Nombre del congreso.	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	abbreviation	Abreviatura del nombre del congreso.	Cadena 32 caracteres (A-Z, 0-9)	SI	1
	organization	Organismo o institución que organiza el congreso.	Cadena 128 caracteres (A-Z, 0-9)	SI	1
	status	Estado de libro de actas del congreso.	Enumerado	NO	1
	country	País donde ha sido publicado el libro.	Enumerado	NO	1
	City	Ciudad donde ha sido publicado el libro.	Cadena 32 caracteres (A-Z, 0-9)	NO	1
	Type	Tipo de congreso.	Enumerado	NO	1
	fromDate	Fecha del inicio del congreso.	Fecha	NO	1
	toDate	Fecha de finalización del congreso.	Fecha	NO	1
	deadline	Fecha límite para la entrega de los trabajos que se deseen presentar en el congreso.	Cadena 64 caracteres (A-Z, 0-9)	NO	1
	intention	Motivo o temática del congreso	Cadena 32 caracteres (A-Z, 0-9)	NO	1

	url	Dirección de la página web oficial del congreso.	Cadena 255 caracteres (A-Z, 0-9)	NO	1
	isbn	Número de libro estándar internacional con el que ha sido publicado el libro de actas.	Cadena 16 caracteres (A-Z, 0-9)	NO	1
	actaTitle	Nombre asignado al libro de actas que publica el congreso.	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	editorial	Nombre de la empresa que publica el libro de actas.	Cadena 255 caracteres (A-Z, 0-9)	NO	1
	editors	Personas que han realizado la edición del libro de actas.	Cadena 64 caracteres (A-Z, 0-9)	NO	1

Atributos de la entidad ACTA:

**Tabla 1.1.9:** Atributos Entidad Acta

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
ACTA	Code	Código único asignado automáticamente al ser introducida una nueva entrada aun. Permitirá la identificación unívoca de los elementos de la tabla aun cuando el congreso aún no se haya celebrado o tenga un estado prematuro.	Numérico (0-9)	SI	1
	contribution	Nombre de la aportación que se envía al congreso.	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	volume	Tmo o volumen de la revista donde se encuentra el artículo..	Numérico 4 (0-9)	NO	1
	initPage	Página del libro de actas donde comienza la contribución aportada..	Numérico 4 (0-9)	NO	1

	endPage	Página del libro de actas donde termina la contribución aportada.	Numérico 4 (0-9)	NO	1
	status	Estado del acta.	Enumerado	NO	1
	document	Referencia al documento del acta.	Referencia a documento	NO	1
	meetingCode	Referencia al código identificador del congreso al que pertenece la contribución.	Referencia al congreso	SI	D

Atributos de la entidad COUNTRY:

**Tabla 1.1.10:** Atributos Entidad Country

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
COUNTRY	country	Nombre de los países entre la información internacional existente.	Cadena 32 caracteres (A-Z, 0-9)	SI	1

Atributos de la entidad TYPE:

**Tabla 1.1.11:** Atributos Entidad Type

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
TYPE	country	Tipo de congresos. Podrá tomar alguno de los siguientes valores: N (nacional) o I (internacional).	Cadena 1 caracteres (A-Z, 0-9)	SI	1

Atributos de la entidad STATUS:

**Tabla 1.1.12:** atributos Entidad Status

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
STATUS	status	Posibles estados de una publicación.	Cadena 4 caracteres (A-Z, 0-9)	SI	1

## Atributos de la entidad TREATMENT:

Tabla 1.1.13: atributos Entidad treatment

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
TREATMENT	treatment	Distintas formas de referencia hacia las personas	Cadena 32 caracteres (A-Z, 0-9)	SI	1

## Atributos de la entidad PERSONAL INFORMATION:

Tabla 1.1.14: Atributos Entidad PersonalInformation

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
PERSONAL INFORMATION	familyName	Apellido de la persona que se representa.	Cadena 64 caracteres (A-Z, 0-9)	SI	1
	name	Nombre de la persona que se representa.	Cadena 32 caracteres (A-Z, 0-9)	SI	1
	treatment	Forma de referirse a esta persona.	Enumerado	NO	1
	category	Categoría profesional o académica que posee esta persona.	Cadena 64 caracteres (A-Z, 0-9)	NO	1
	university	Universidad a la que pertenece esta persona.	Cadena 64 caracteres (A-Z, 0-9)	NO	1
	address	Dirección postal de esta persona.	Cadena 128 caracteres (A-Z, 0-9)	NO	1
	city	Ciudad de residencia de esta persona.	Cadena 32 caracteres (A-Z, 0-9)	NO	1
	state	Estado o provincia donde reside esta persona.	Cadena 32 caracteres (A-Z, 0-9)	NO	1
	country	País de residencia de esta persona.	Enumerado	NO	1
	zip	Código postal de la residencia de esta persona.	Cadena 16 caracteres (A-Z, 0-9)	NO	1

	email	Dirección de correo electrónico de esta persona.	Cadena 32 caracteres (A-Z, 0-9)	SI	1
	phone	Teléfono de contacto.	Cadena 16 caracteres (A-Z, 0-9)	SI	1
	fax	Número de fax.	Cadena 16 caracteres (A-Z, 0-9)	NO	1
	departament	Departamento dentro de la universidad donde trabaja.	Cadena 64 caracteres (A-Z, 0-9)	NO	1

Atributos de la entidad AUTHOR:

**Tabla 1.1.15:** Atributos Entidad Author

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
AUTHOR	familyName	Apellido de la persona que se representa.	Cadena 64 caracteres (A-Z, 0-9)	SI	D
	name	Nombre de la persona que se representa.	Cadena 32 caracteres (A-Z, 0-9)	SI	D

Atributos de la entidad MEMBER:

**Tabla 1.1.16.** Atributos Entidad Member

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
MEMBER	familyName	Apellido de la persona que se representa.	Cadena 64 caracteres (A-Z, 0-9)	SI	D
	name	Nombre de la persona que se representa.	Cadena 32 caracteres (A-Z, 0-9)	SI	D
	fromDate	Fecha en la que se dio de alta en el grupo de investigación.	Fecha	SI	1

	toDate	Fecha en la que se dio de baja en el grupo de investigación.	Fecha	NO	1
	email_2	Dirección de correo electrónico de esta persona.	Cadena 32 caracteres (A-Z, 0-9)	NO	1
	phone_2	Teléfono de contacto.	Cadena 32 caracteres (A-Z, 0-9)	NO	1
	web	dirección URL de la página web personal de esta persona.	Cadena 255 caracteres (A-Z, 0-9)	NO	1
	image	Atributo para insertar un archivo de imagen.	Referencia a una imagen	NO	1
	dni	Documento nacional de identidad del miembro.	Cadena de 9 caracteres (A-Z, 0-9)	SI	
	birthDate	Fecha de nacimiento	Fecha	SI	
	sex	Género del miembro del grupo.	Cadena de 1 carácter (A-Z)	NO	
	numOfficial	Número de funcionario.	Numérico 15 (0-9).	SI	
	postalCode	Código postal donde se ubica el miembro.	Numérico 5 (0-9).	SI	
	codeSpecialization	Especialización del miembro. Referencia al valor de una instancia de Specialization.	Cadena de 255 caracteres.		

#### 1.5.1.4 Identificación de los atributos identificadores del “Diagrama Conceptual Básico”

A continuación se detallan los atributos identificadores para cada tipo de entidad:

Tabla 1.1.17: Atributos identificadores

Entidad	Alias	Atributos
<i>Journal</i>	Journal	Issn
<i>Article</i>	Article	Code
<i>Book</i>	Book	Code
<i>Meeting</i>	Factura	Code
<i>Acta</i>	Acta	Code
<i>Book Of Chapter</i>	BookOfChapter	Code
<i>Chapter</i>	Chapter	Code
<i>Personal Information</i>	PersonallInformation	familyName
		Name
<i>Author</i>	Author	familyName
		Name
<i>Member</i>	Member	familyName
		Name
<i>Type</i>	Type	Type
<i>Treatment</i>	Treatment	treatment
<i>Status</i>	Status	status
<i>Country</i>	Country	country

#### 1.5.1.5 Diagrama E-R del “Diagrama Conceptual Básico”

A continuación se muestra el diagrama entidad relación resultante del análisis realizado:

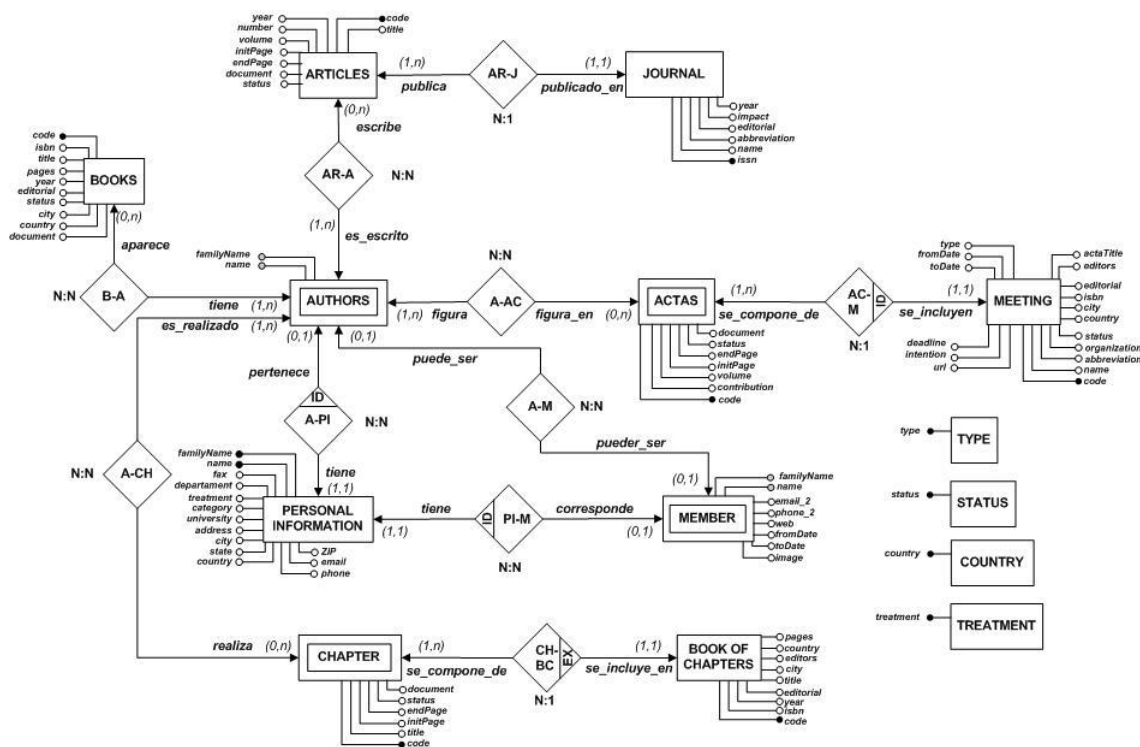


Figura 1.1.1: Diagrama Entidad-Relación Básico

### 1.5.1.6 Análisis de los Tipos de Entidad del “Diagrama Conceptual Curriculum Vitae”

Tipo de entidad **Degree**: el cual representa al objeto del mundo real “formación académica de cada uno de los miembros del grupo de investigación”. Este tipo de entidad será débil por identificación con respecto al tipo de entidad Member. Por tanto para identificar cada formación académica se utilizará el agregado formado por los atributos name y familyName, provenientes del tipo de entidad Member. Además, para la identificación de cada *ingeniería o licenciatura* propiamente dicha, se utilizará un identificador codeDegree, junto con el nombre de la ingeniería o licenciatura realizada (engineering). Esto se especifica así para contemplar la posibilidad de realizar una misma ingeniería en varios sitios debido a un cambio de domicilio u otras circunstancias. Para ello también se especifica el centro donde se cursaron los estudios (institution), junto con la fecha de inicio (fromDate), y la de fin (toDate) y finalmente el doctorado (doctorate).

Tipo de entidad **Professional Status**: el cual representa al objeto del mundo real “situación profesional actual de cada miembro del grupo de



investigación”. Este tipo de entidad será débil por identificación con respecto al tipo de entidad Member. Por tanto para identificar cada formación académica se utilizará el agregado formado por los atributos name y familyName, provenientes del tipo de entidad Member. Además se mantendrá información relativa al organismo en que se figura actualmente (organism), facultad (faculty), categoría profesional (category), fecha de inicio de la misma (fromDate), dirección postal de la organización (orgAddress) y teléfono de la misma (orgPhone).

Tipo de entidad **Specialization**: el cuál representa al objeto del mundo real “especialización del miembro del grupo de investigación”. Para llevar a cabo la identificación de cada especialización se utilizará el código asociado a cada uno de ellos (code).

Tipo de entidad **Activity**: el cual representa al objeto del mundo real “cada una de las actividades anteriormente realizadas de carácter científico o profesional”. Este tipo de entidad será débil por identificación respecto al tipo de entidad Member, por lo que los atributos familyName y name serán necesarios para dicha identificación. Además también se requiere el nombre (nameAct) de la actividad, así como un código externo (code) que permita identificar distintas actividades de un mismo miembro. Por último también tendrá una fecha de inicio (fromDate) y de fin (toDate) para poder identificar el periodo en el que se llevó a cabo la actividad. Por último, y para una mayor información de la actividad, se describirá el puesto ocupado (position) y la institución (institution) dónde se realizó la misma.

Tipo de entidad **Language**: el cual representa al objeto del mundo real “los idiomas de interés científico que un determinado miembro conoce”. Este tipo de entidad será débil por identificación respecto al tipo de entidad Member, por lo que los atributos familyName y name serán utilizados para la identificación, además del nombre del idioma (lenguaje). Será importante tener que especificar el nivel (R=regular, B=bien, C=correctamente) del idioma con respecto a como se habla (speak), se lee (reading) y se escribe (writing).

Tipo de entidad **Finance Project**: el cual representa al objeto del mundo real “participación en proyectos de investigación financiados”. Para la

identificación de cada proyecto financiado se utilizará el título del proyecto (title). Además tiene otros atributos como *fundingOrg* que se refiere a la entidad financiadora, la fecha de inicio (fromDate) y fecha fin (toDate) que especifican la duración del proyecto, y finalmente el investigador principal (researcher). Mediante los atributos familyName y name, se mantiene una referencia al miembro encargado de la dirección del proyecto.

Tipo de entidad **Research Contract**: el cual representa al objeto del mundo real “participación en contratos de investigación de especial relevancia con empresas y/o administraciones”. Para la identificación de cada contrato se utilizará el título del contrato (title). Además tiene otros atributos como *fundingOrg* que se refiere a la entidad financiadora, la fecha de inicio (fromDate) y fecha fin (toDate) que especifican la duración del proyecto, y finalmente el investigador principal (researcher). También, mediante los atributos familyName y name se mantiene una referencia al miembro responsable del contrato de investigación.

Tipo de entidad **Patent**: el cual representa al objeto del mundo real “patente y modelos de utilidad”. Para la identificación de cada contrato se utilizará el título de la patente o modelo (title). También consta de otros atributos como el número de solicitud (numberApplication), el país de prioridad (priorityCountry), fecha (date), prioridad (priority), entidad titular (firstInstitution), países en los que se ha extendido (exportCountries) y empresas que la están explotando (companyInUse).

Tipo de entidad **Foreign Institution**: el cual representa al objeto del mundo real “estancias en centros extranjeros”. Este tipo de entidad es débil por identificación respecto al tipo de entidad Member, por lo que los atributos familyName y name se requieren para su identificación. Además también son necesarios para esta identificación el nombre de cada centro extranjero (nameFI) así como un código externo (code) que permita distinguir distintas estancias de un mismo miembro en el mismo centro. También se incluye la dirección (address), el país donde se encuentra ubicado (country), el año en el que comenzó la estancia (year), y la duración de la misma (duration).

Tipo de entidad ***Thesis***: el cual representa al objeto del mundo real “tesis doctorales dirigidas”. Para la identificación de cada tesis se utilizará el título de la misma (title). También se muestra el doctorado (doctorate), la universidad donde se realiza (university), la facultad o escuela (faculty), el año (date) y la calificación obtenida (qualification). Además, para identificar al autor de la tesis, se tiene la información correspondiente, mediante los atributos name y familyName, por medio de la relación existente con el tipo de entidad Member.

Tipo de entidad ***Position Type***: el cual representa el objeto del mundo real “tipo de puesto”. Para la identificación de cada puesto existente, se utilizará el nombre del mismo (namePT).

Tipo de entidad ***Others Merits***: el cual representa al objeto del mundo real “otros méritos o aclaraciones que se deseen hacer constar”. Para la identificación de cada uno de estos méritos se utilizará el nombre en concreto de cada uno de estos méritos (merit).

#### 1.5.1.7 Análisis de los tipos de interrelación del “Diagrama Conceptual Curriculum Vitae”

Los tipos de entidad anteriormente presentados se encuentra relacionados en el sistema en estudio mediante los siguientes tipos de interrelación:

Tipo de interrelación **MEMBER/DEGREE (M-D)**: el cual relaciona los tipos de entidad *MEMBER* y *DEGREE*. Ambos tipos de entidad participan con cardinalidades (1,1), puesto que un miembro sólo tiene una formación académica, y una formación académica sólo puede pertenecer a un único miembro.

Tipo de interrelación **MEMBER/PROFESSIONAL STATUS (M-PS)**: el cual relaciona los tipos de entidad *MEMBER* y *PROFESSIONAL STATUS*. Al igual que en el caso anterior, ambos tipos de entidad participan con cardinalidades (1,1), puesto que un miembro sólo tiene una situación profesional actual, la cual sólo puede pertenecer a un único miembro.

Tipo de interrelación **MEMBER/ACTIVITY (M-A)**: el cual relaciona los tipos de entidad *MEMBER* y *ACTIVITY*. El tipo de entidad *MEMBER* participa con

cardinalidades (1,1), puesto que la actividad con la que cada miembro está relacionado es sólo suya. Por otro lado, el tipo de entidad *ACTIVITY* actúa con cardinalidades (0,n) puesto que un miembro podrá o no tener distintas actividades anteriores.

Tipo de interrelación **MEMBER/LANGUAGE (M-L)**: el cual relaciona los tipos de entidad *MEMBER* y *LANGUAGE*. El tipo de entidad *MEMBER* participa con cardinalidades (1,1), puesto que las características del idioma que cada miembro conoce son sólo propias del mismo. Por otro lado, el tipo de entidad *LANGUAGE* actúa con cardinalidades (1,n) dado que un miembro podrá conocer más de un idioma.

Tipo de interrelación **MEMBER/FOREIGN INSTITUTION (M-FI)**: el cual relaciona los tipos de entidad *MEMBER* y *FOREIGN INSTITUTION*. El tipo de entidad *MEMBER* participa con cardinalidades (1,1), puesto que la estancia con la que está relacionado cada miembro sólo pertenece a él. Por otro lado, el tipo de entidad *FOREIGN INSTITUTION* actúa con cardinalidades (0,n) puesto que un miembro puede haber tenido o no varias estancias en el extranjero.

Tipo de interrelación **MEMBER/THESIS (M-T)**: el cual relaciona los tipos de entidad *MEMBER* y *THESIS*. El tipo de entidad *MEMBER* participa con cardinalidades (1,1) denotando que una tesis sólo puede ser realizada por un único miembro. Por otro lado, el tipo de entidad *THESIS* actúa con cardinalidades (0,n) dado que un miembro podrá realizar o no varias tesis.

Tipo de interrelación **THESIS/MEMBER (T-M)**: el cual relaciona los tipos de entidad *Thesis* y *Member*. En este caso el tipo de entidad *THESIS* actúa con cardinalidades (0,n) reflejando como un miembro puede actuar como director o no de varias tesis. Por otro lado, el tipo de entidad *MEMBER* participa con cardinalidades (1,n) puesto que una tesis podrá ser dirigida por más de un director.

Tipo de interrelación **MEMBER/FINANCE PROJECT (M-FP)**: el cual relaciona los tipos de entidad *MEMBER* y *FINANCE PROJECT*. El tipo de entidad *MEMBER* participa con cardinalidades (1,n) puesto que en un proyecto financiado pueden participar más de un miembro. Por otro lado, el tipo de entidad *FINANCE*

*PROJECT* actúa con cardinalidades (0,n) reflejando que un miembro puede o no participar en distintos proyectos.

Tipo de interrelación **FINANCE PROJECT/MEMBER (FP-M)**: el cual relaciona los tipos de entidad *FINANCE PROJECT* y *MEMBER*. El tipo de entidad *FINANCE PROJECT* participa en este caso con cardinalidades (0,n) denotando que un miembro puede actuar o no como director de varios proyectos. Por otro lado, el tipo de entidad *MEMBER* actúa con cardinalidades (1,1) dado que el director de un proyecto únicamente podrá ser un miembro.

Tipo de interrelación **MEMBER/RESEARCH CONTRACT (M-RC)**: el cual relaciona los tipos de entidad *MEMBER* y *RESEARCH CONTRACT*. El tipo de entidad *MEMBER* participa con cardinalidades (1,n) puesto que en un contrato podrá participar más de un miembro. Por otro lado, el tipo de entidad *RESEARCH CONTRACT* actúa con cardinalidades (0,n) reflejando el hecho de que un miembro podrá participar o no en varios contratos de investigación.

Tipo de interrelación **RESEARCH CONTRACT/MEMBER (RC-M)**: el cual relaciona los tipos de entidad *RESEARCH CONTRACT* y *MEMBER*. En este caso el tipo de entidad *RESEARCH CONTRACT* participa con cardinalidades (0,n) denotando que un miembro del grupo podrá ser responsable o no de varios contratos. Por otro lado, el tipo de entidad *MEMBER* participa con cardinalidades (1,1) dado que el responsable de un contrato de investigación es un único miembro.

Tipo de interrelación **MEMBER/PATENT (M-P)**: el cual relaciona los tipos de entidad *MEMBER* y *PATENT*. El tipo de entidad *MEMBER* actúa con cardinalidades (1,n) dado que una patente podrá ser llevada a cabo por más de un miembro. Por otro lado, el tipo de entidad *PATENT* participa con cardinalidades (0,n) reflejando el hecho de que un miembro puede o no participar en una patente.

Tipo de interrelación **MEMBER/SPECIALIZATION (M-S)**: el cual relaciona los tipos de entidad *MEMBER* y *SPECIALIZATION*. El tipo de entidad *MEMBER* participa con cardinalidades (0,n) dado que un tipo de especialización podrá corresponder a varios miembros. Por otro lado, el tipo de entidad *SPECIALIZATION* participa con cardinalidades (1,1) puesto que un miembro del grupo sólo podrá tener un tipo de especialización.

Tipo de interrelación **MEMBER/OTHER MERITS (M-OM)**: el cual relaciona los tipos de entidad *MEMBER* y *OTHER MERITS*. El tipo de entidad *MEMBER* participa con cardinalidades (1,1) reflejando el hecho de que una ocurrencia de la entidad *OTHER MERITS* podrá estar relacionada únicamente con un miembro del grupo. Por otro lado, el tipo de entidad *OTHER MERITS* actúa con cardinalidad (0,n) puesto que un miembro podrá tener varias méritos adicionales reconocidos.

Tipo de interrelación **OTHER MERITS/POSITION TYPE (OM-PT)**: el cual relaciona los tipos de entidad *OTHER MERITS* y *POSITION TYPE*. El tipo de entidad *OTHER MERITS* participa con cardinalidades (0,n) dado que una ocurrencia del tipo de entidad *POSITION TYPE* podrá tener varias especializaciones en el primer tipo de entidad citado. Por otro lado, *POSITION TYPE* participa con cardinalidades (1,1), puesto que habrá diferentes méritos que correspondan a un único tipo de cargo.

Estos tipos de interrelación se encuentran recogidos y especificados en el siguiente diccionario de tipos de relación:

**Tabla 1.1.18:** Diccionario de interrelaciones

Entidad	Multiplicidad	Relación	Multiplicidad	Entidad
Member	1/1	M-D	1/1	Degree
Member	1/1	M-PS	1/1	Professional Status
Member	1/1	M-A	0/n	Activity
Member	1/1	M-L	1/N	Language
Member	1/1	M-FI	0/N	Foreign Institution
Member	1/1	M-T	0/N	Thesis
Thesis	0/N	T-M	1/N	Member
Member	1/N	M-FP	0/N	Finance Project
Finance Project	0/N	FP-M	1/1	Member
Member	1/N	M-RC	0/N	Research Contract
Research Contract	0/N	RC-M	1/1	Member
Member	1/N	M-P	0/N	Patent
Member	0/N	M-S	1/1	Specialization
Member	1/1	M-OM	0/N	Other Merits

Other Merits	0/N	PT-OM	1/1	Position Type
--------------	-----	-------	-----	---------------

### 1.5.1.8 Análisis de los atributos del “Diagrama Conceptual Curriculum Vitae”

Tomando como base cada uno de los tipos de entidad y de interrelación analizados anteriormente, así como la estructura de cada uno de ellos, puede llevarse a cabo la representación conceptual del problema mediante la siguiente estructura sintáctica:

Atributos de la entidad DEGREE:

**Tabla 1.1.19:** Atributos entidad Degree

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
DEGREE	familyName	Apellido de la persona que se representa.	Cadena 64 caracteres (A-Z, 0-9)	SI	D
	name	Nombre de la persona que se representa.	Cadena 32 caracteres (A-Z, 0-9)	SI	D
	codeDegree	Código único identificativo.	Numérico (0-9)	SI	1
	engineering	Ingeniería o licenciatura que ha realizado el miembro.	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	institution	Centro o institución dónde se han llevado a cabo los estudios realizados	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	fromDate	Fecha de inicio de los estudios realizados	Fecha	SI	1
	toDate	Fecha final de los estudios realizados	Fecha	SI	1
	doctorate	Doctorado realizado	Cadena 255 caracteres (A-Z, 0-9)	NO	1

Atributos de la entidad PROFESSIONAL STATUS:

**Tabla 1.1.20:** Atributos Entidad Professional Status

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
PROFESSIONAL STATUS	familyName	Apellido de la persona que se representa.	Cadena 64 caracteres (A-Z, 0-9)	SI	D

	name	Nombre de la persona que se representa.	Cadena 32 caracteres (A-Z, 0-9)	SI	D
	organism	Nombre del organismo dónde se está actualmente trabajando	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	faculty	Nombre de la facultad o escuela dónde se está actualmente trabajando	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	category	Categoría profesional que se está ocupando actualmente	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	fromDate	Fecha de inicio del trabajo.	Fecha	SI	1
	orgAddress	Dirección postal de la organización dónde se está trabajando actualmente	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	orgPhone	Teléfono de la organización dónde se está trabajando actualmente	Cadena 10 caracteres (0-9)	NO	1

Atributos de la entidad SPECIALIZATION:

**Tabla 1.1.21:** Atributos Entidad Specialization

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
SPECIALIZATION	codeSpecialization	Código de la especialización, es un atributo identificativo	Numérico (0-9)	SI	1

Atributos de la entidad ACTIVITY:

**Tabla 1.1.22:** Atributos Entidad Activity

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
activity	familyName	Apellido de la persona que se representa.	Cadena 64 caracteres (A-Z, 0-9)	SI	D



	name	Nombre de la persona que se representa.	Cadena 32 caracteres (A-Z, 0-9)	SI	D
	nameAct	Nombre de la actividad	Cadena 255 caracteres	SI	1
	code	Código único identificativo.	N Numérico (0-9)	SI	1
	fromDate	Fecha de inicio de la actividad	Fecha	SI	1
	toDate	Fecha de fin de la actividad	Fecha	SI	1
	position	Puesto que se ha tenido al realizar la actividad	Cadena 255 caracteres	SI	1
	institution	Institución dónde se ha realizado la actividad	Cadena 255 caracteres	SI	1

Atributos de la entidad LANGUAGE:

**Tabla 1.1.23:** Atributos Entidad Language

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
LANGUAGE	familyName	Apellido de la persona que se representa.	Cadena 64 caracteres (A-Z, 0-9)	SI	D
	name	Nombre de la persona que se representa.	Cadena 32 caracteres (A-Z, 0-9)	SI	D
	language	Idioma que se conoce	Cadena 50 caracteres (A-Z, 0-9)	SI	1
	speak	El nivel de idioma hablado	Cadena 1 carácter (R,B y C)	SI	1
	reading	El nivel de idioma leído	Cadena 1 carácter (R,B y C)	SI	1
	writing	El nivel de idioma escrito	Cadena 1 carácter (R,B y C)	SI	1

## Atributos de la entidad FINANCE PROJECT:

Tabla 1.1.24: Atributos Entidad Finance Project

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
FINANCE PROJECT	title	Título del proyecto	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	fundingOrg	Entidad que financia el proyecto	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	fromDate	Fecha de comienzo del proyecto	Fecha	SI	1
	toDate	Fecha de fin del proyecto	Fecha	SI	1
	researcher	Investigador principal	Cadena 255 caracteres	SI	1
	familyName	Apellido de la persona que se representa.	Cadena 64 caracteres (A-Z, 0-9)	SI	D
	name	Nombre de la persona que se representa.	Cadena 32 caracteres (A-Z, 0-9)	SI	D

## Atributos de la entidad RESEARCH CONTRACT:

Tabla 1.1.25: Atributos Entidad Research Contract

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
RESEARCH CONTRACT	title	Título del contrato de investigación. Atributo identificativo	Cadena 255 caracteres	SI	1
	fundingOrg	Empresa o administración financiadora	Cadena 255 caracteres	SI	1
	fromDate	Fecha de comienzo del contrato	Fecha	SI	1
	toDate	Fecha de finalización del contrato	Fecha	SI	1
	researcher	Responsable investigador	Cadena 255 caracteres	SI	1

## Atributos de la entidad PATENT:

Tabla 1.1.26: Atributos Entidad Patent

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
PATENT	title	Título de la patente	Cadena 255 caracteres	SI	1
	numberApplication	Número de solicitud	Numérico (0-9)	SI	1
	priorityCountry	País de prioridad	Cadena 50 caracteres	SI	1
	date	Fecha en que se realizó	Fecha	SI	1
	priority	Prioridad	Numérico (0-9)	SI	1
	firstInstitution	Entidad titular	Cadena 50 caracteres	SI	1
	exportCountries	Países a los que se ha extendido la patente	Cadena 255 caracteres	SI	1
	companyInUse	Empresas que la están explotando	Cadena 255 caracteres	SI	1

## Atributos de la entidad FOREIGN INSTITUTION:

Tabla 1.1.27: Atributos Entidad Foreign Institution

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
FOREIGN INSTITUTION	familyName	Apellido de la persona que se representa.	Cadena 64 caracteres (A-Z, 0-9)	SI	D
	name	Nombre de la persona que se representa.	Cadena 32 caracteres (A-Z, 0-9)	SI	D
	nameFI	Nombre del centro extranjero	Cadena 32 caracteres (A-Z, 0-9)	SI	1
	code	Código externo	Numérico	SI	1
	address	Dirección del centro	Cadena 32 caracteres (A-Z, 0-9)	SI	1
	country	País dónde se encuentra ubicado el centro	Cadena 32 caracteres (A-Z, 0-9)	SI	1
	year	Año en el que se estuvo en el centro	Fecha	SI	1

	duration	Duración de la estancia	Numérico	SI	1
--	----------	-------------------------	----------	----	---

Atributos de la entidad THESIS:

**Tabla 1.1.28:** Atributos Entidad Thesis

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
THESIS	title	Título de la tesis	Cadena 32 caracteres (A-Z, 0-9)	SI	1
	doctorate	Doctorado	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	university	Universidad dónde se realiza la tesis	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	faculty	Facultad donde se realiza la tesis	Cadena 255 caracteres (A-Z, 0-9)	SI	1
	year	Año en el que se realizó la tesis	Fecha	SI	1
	qualification	Calificación obtenida	Numérico (0-9)	Si	1
	familyName	Apellido de la persona que se representa.	Cadena 64 caracteres (A-Z, 0-9)	SI	D
	name	Nombre de la persona que se representa.	Cadena 32 caracteres (A-Z, 0-9)	SI	D

Atributos de la entidad POSITION TYPE:

**Tabla 1.1.29:** Atributos Position Type

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
POSITION TYPE	namePT	Nombre del tipo de puesto que se ocupa	Cadena 64 caracteres (A-Z, 0-9)	SI	1

## Atributos de la entidad OTHER MERITS:

Tabla 1.1.30: Atributos Other Merits

OBJETO	ATRIBUTO	DESCRIPCIÓN	DOMINIO	EXIST	TIPO
OTHER MERITS	familyName	Apellido de la persona que se representa.	Cadena 64 caracteres (A-Z, 0-9)	SI	D
	name	Nombre de la persona que se representa.	Cadena 32 caracteres (A-Z, 0-9)	SI	D
	namePT	Nombre del tipo de puesto que se ocupa	Cadena 64 caracteres (A-Z, 0-9)	SI	1
	merit	Méritos que se han realizado	Cadena 255 caracteres	SI	1

## 1.5.1.9 Identificación de los atributos identificadores del “Diagrama Conceptual Curriculum Vitae”

Tabla 1.1.31: Atributos Identificadores

Entidad	Alias	Atributos
<i>Degree</i>	Degree	<i>engineering</i> <i>codeDegree</i>
<i>Professional Status</i>	Status	<i>familyName</i> <i>name</i>
<i>Specialization</i>	Specialization	<i>codeSpecialization</i>
<i>Activity</i>	Activity	<i>code</i> <i>nameAct</i>
<i>Language</i>	Language	<i>language</i>
<i>Finance Project</i>	Finance Project	<i>Code</i>
<i>Research Contract</i>	Research Contrct	<i>Code</i>
<i>Patent</i>	Patent	<i>title</i>
<i>Foreign institution</i>	Foreign Institution	<i>nameFi</i> <i>code</i>
<i>Thesis</i>	Thesis	<i>title</i>
<i>Position Type</i>	Position Type	<i>namePT</i>
<i>Other merits</i>	Other Merits	<i>namePT</i>

#### **1.5.1.10 Diagrama E-R del “Diagrama Conceptual Curriculum Vitae”**

Para facilitar la comprensión y favorecer la legibilidad del esquema conceptual resultante, se ha decidido realizar dos diagramas separados, de forma que se refleje con la mayor exactitud y claridad posible el análisis realizado:

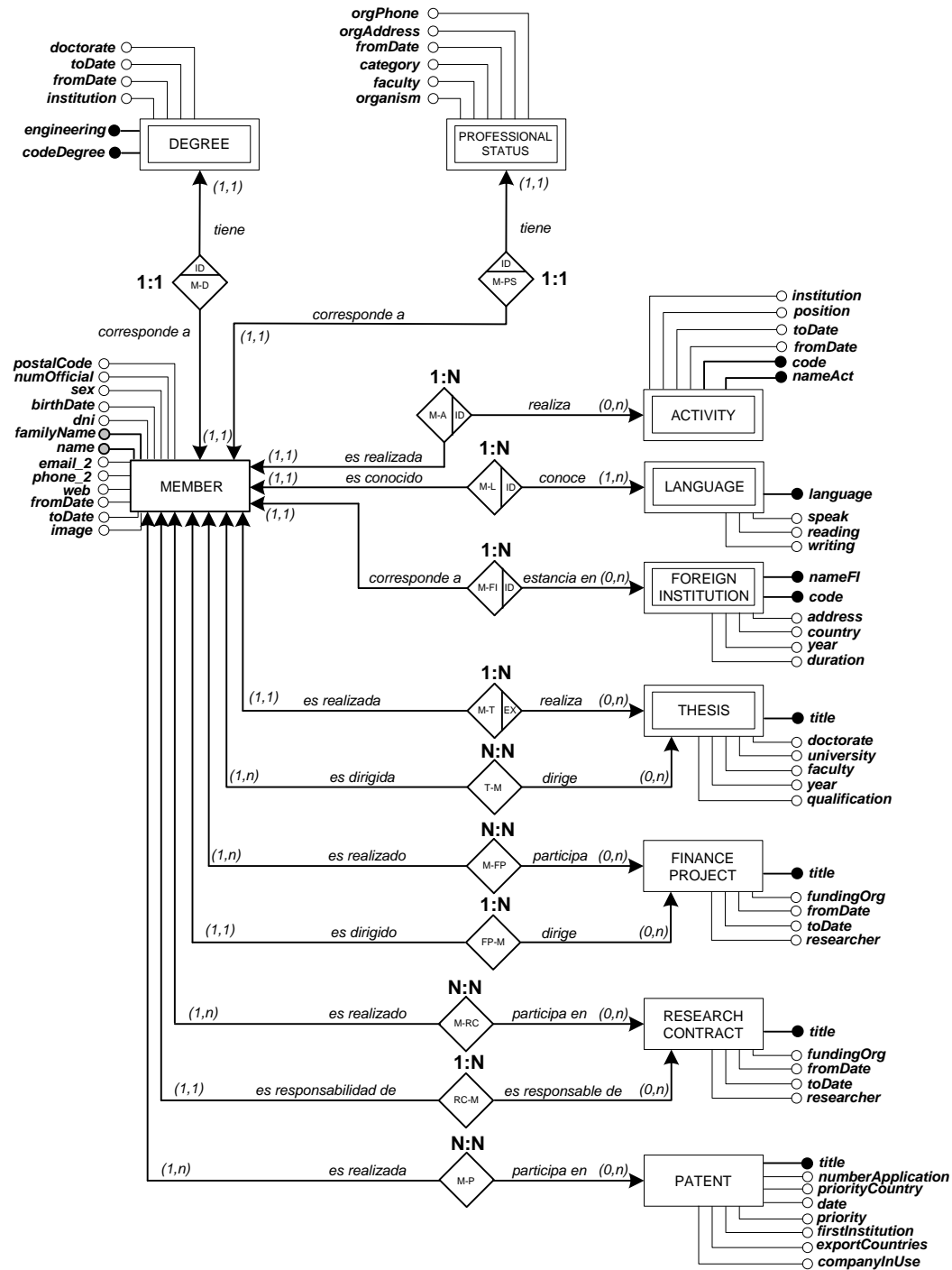


Figura 1.1.2: Diagrama Entidad-Relación Curículum Vitae (Vista 1)

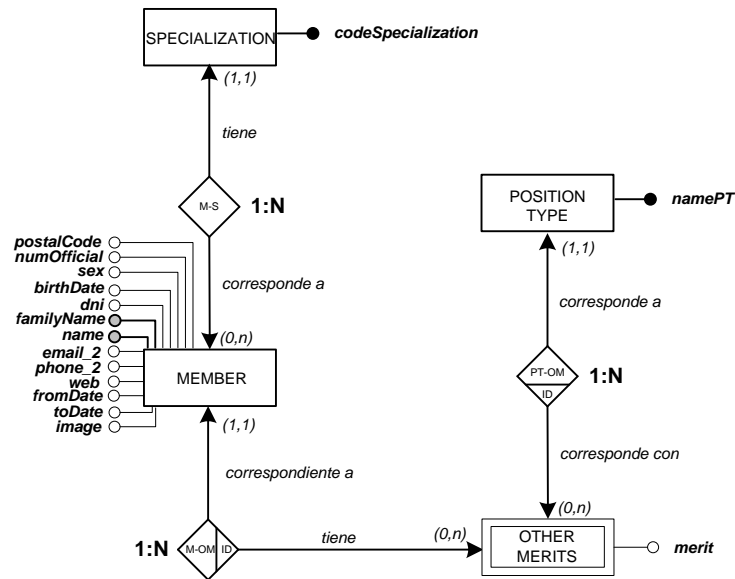


Figura 1.1.3: Diagrama Entidad-Relación Curículum Vitae (Vista 2)

### 1.5.2 Modelo funcional

En esta sección se muestra una posible arquitectura funcional para el sistema objeto del trabajo.

La funcionalidad del sistema se ha dividido en diferentes partes. Dicha división funcional se ha llevado a cabo en base a los elementos fundamentales que nos encontramos en el sistema. Los elementos funcionales relacionados con la base de datos son los siguientes:

Member	Mantenimiento (Funcionalidad F0111)
Article	Mantenimiento (Funcionalidad F0121)
Book	Mantenimiento (Funcionalidad F0131)
Chapter	Mantenimiento (Funcionalidad F0141)
Acta	Mantenimiento (Funcionalidad F0151)
Journal	Mantenimiento (Funcionalidad F0161)
Author	Mantenimiento (Funcionalidad F0171)
Meeting	Mantenimiento (Funcionalidad F0181)

A continuación pasaremos a describir cada uno de ellos:



Member: (Funcionalidad F0111)

Entre otras operaciones se debe permitir la extracción de la información de la base de datos de todos los miembros del grupo de investigación, con el nombre apellidos y fotografía.

Además de la consulta de la citada información se deberá permitir la modificación de la información correspondiente a cualquier campo que presenta un registro relacionado con un miembro del grupo de investigación.

Por último para este elemento de la base de datos se deberá permitir el borrado de los datos correspondientes a un elemento Member de la base de datos.

Los diferentes casos de excepción que se pueden presentar para este tipo de elemento son:

**CASO NUEVO** Se posibilitará la inserción de nuevos miembros al grupo de investigación, en cuyo caso se procederá de la siguiente forma, en función de la respuesta del sistema:

*Error al insertar (grabar):* La ocurrencia de un error se puede deber a:

Existe un miembro con el mismo nombre y apellidos. Se devolverá un código de error indicando que el miembro ya existe en la base de datos.

Alguno de los datos o campos obligatorios no están debidamente proporcionados. Se devolverá un código de error indicando que existen campos suministrados erróneos.

**CASO MODIFICAR** Se posibilitará la consulta de los datos de un miembro perteneciente al grupo de investigación, así como la modificación posterior de sus datos procediendo de la siguiente forma, en función de la respuesta del sistema:

*Error al insertar (grabar):* La ocurrencia de un error se puede deber a:

Se consideran los errores del CASO NUEVO

Si se ha modificado la clave. Se devolverá un código indicando que se ha modificado la información relacionada con el registro modificado existente en la base de datos

**CASO BORRAR** Se posibilitará la consulta de los datos de un miembro perteneciente al grupo de investigación, así como la eliminación posterior de sus datos procediendo de la siguiente forma, en función de la respuesta del sistema:

*Error al borrar (grabar):* La ocurrencia de error se puede deber a:

Se considerará el caso de que el registro a borrar no exista, devolviendo un código de error

Si se ha eliminado el registro. Se devolverá un código indicando que se ha modificado la información relacionada con el registro eliminado existente en la base de datos

Article: (Funcionalidad F0121)

Entre otras operaciones se debe permitir la extracción de la información de la base de datos de todos los artículos del grupo de investigación, existentes en la base de datos, indicando el estado en el que se encuentran.

Entre los casos que se pueden presentar se encuentran los mismos que los descritos para la funcionalidad F0111, cuyo tratamiento será similar

El tratamiento de casos específicos para esta funcionalidad se derivan del tratamiento de las tablas “hijas” que mantienen relación o referencia con la tabla Articles, es decir: Journals, Authors y Aut\_Articles. Se realizarán a continuación para cada funcionalidad específica.

Book: (Funcionalidad F0131)

La funcionalidad requerida para este elemento es la misma que la definida el elemento del sistema Article (Funcionalidad F0121).

Chapter: (Funcionalidad F0141)

La funcionalidad requerida para este elemento es la misma que la definida el elemento del sistema Article (Funcionalidad F0121).

Acta: (Funcionalidad F0151)

La funcionalidad requerida para este elemento es la misma que la definida el elemento del sistema Article (Funcionalidad F0121).

Journal: (Funcionalidad F0161)

El tratamiento de las revistas, está supeditado al tratamiento de la tabla principal en la que la revista es uno de sus atributos, y será invocado a través de la funcionalidad Visualizar y Editar.

*Funcionalidad Visualizar:* Permitirá la visualización de los registros asociados en las tablas hijas correspondientes a un registro seleccionado en la tabla principal. Devolviendo toda la información asociada con ese registro.

*Funcionalidad Editar:* Permitirá la edición de los registros asociados en las tablas hijas correspondientes a un registro seleccionado en la tabla principal. Modificando toda la información asociada con ese registro.

De esta forma se mantiene la coherencia de que las revistas tienen una existencia supeditada a la existencia de un artículo relacionado con el grupo de investigación, y no una existencia principal.

Authors: (Funcionalidad F0171)

El tratamiento de los autores es similar al descrito para Journals (Funcionalidad F0161), siendo invocado a través de la funcionalidad *Visualizar y Editar*.

De esta forma se mantiene la coherencia de que los autores tienen una existencia supeditada a la existencia de una publicación relacionada con el grupo de investigación, y no una existencia principal.

Meeting: (Funcionalidad F0181)

El tratamiento de los congresos es similar al descrito para Journals (Funcionalidad F0161), siendo invocado a través de la funcionalidad *Visualizar y Editar*.

De esta forma se mantiene la coherencia de que los meetings tienen una existencia supeditada a la existencia de un acta relacionada con el congreso, y no una existencia principal.

Book Of Chapter: (Funcionalidad F0191)

El tratamiento de los libros de capitulos es similar al descrito para Journals (Funcionalidad F0161), siendo invocado a través de la funcionalidad *Visualizar y Editar*.

De esta forma se mantiene la coherencia de que los libros de capítulos tienen una existencia supeditada a la existencia de un capítulo relacionada con el libro de capítulos, y no una existencia principal.

## 1.6. DISEÑO DEL SISTEMA

A continuación una vez analizado el sistema de modo conceptual, se pasará al diseño de la arquitectura y procedimental, basado en el lenguaje MySQL.

### 1.6.1 Modelo Relacional Básico

El proceso de obtención del esquema relacional consiste en la aplicación de las reglas de transformación al modelo conceptual básico descrito en el trabajo anterior. Como no se han considerado atributos compuestos ni múltiples en ninguno de los objetos existentes en el modelo conceptual, así como tampoco existen relaciones jerárquicas, nuestro proceso inicial consistirá en el paso de los tipos de entidad y relación, descritos en el trabajo 2 (Análisis EE-R) a relaciones del modelo relacional.

#### 1.6.1.1 Creación de las tablas

Una vez que se ha comprobado que no existen atributos compuestos ni múltiples en ninguno de los objetos, así como la comprobación de que el modelo conceptual no presenta relaciones de tipo jerárquico, por tanto no es necesaria la aplicación de las reglas preparatorias de transformación (PRTCAR), y por consiguiente en este proceso es conveniente comenzar aplicando estas reglas RTCAR a los tipos de entidades.

Tabla **Article**: se forma a partir del tipo de entidad *Article* incorporando todos los atributos de la misma. Además participa en una interrelación binaria 1:N con la entidad *Journal* con cardinalidad máxima muchos, por lo que el identificador del tipo de entidad *Journal*, pasa a formar parte de tabla *Article* como clave foránea, quedando el esquema de esta tabla de la forma:

**Article** (code, title, year, number, volume, initpage, endpage, document, status, issn)

Tabla **Journal**: se forma a partir del tipo de entidad *Journal* incorporando

todos los atributos de la misma. quedando el esquema de esta tabla de la forma:

**Journal** (issn, editorial, abbreviation, name, year, totalcites, impactfactor, inmedindex, numarticles, citedhalflife)

Tabla **Books**: se forma a partir del tipo de entidad *Books* incorporando todos los atributos de la misma, quedando el esquema de esta tabla de la forma:

**Books** (code, isbn, title, pages, year, status, editorial, city, country, document)

Tabla **Meeting**: se forma a partir del tipo de entidad *Meeting* incorporando todos los atributos de la misma, quedando el esquema de esta tabla de la forma:

**Meeting** (code, name, abbreviation, organization, status, country, city, isbn, editorial, editors, actatitle, type, fromdate, todate, deadline, intention, url)

Tabla **Actas**: se forma a partir del tipo de entidad *Actas* incorporando todos los atributos de la misma. Además participa en una interrelación binaria 1:N con la entidad *Meeting* con cardinalidad máxima muchos, por lo que el identificador del tipo de entidad *Meeting*, pasa a formar parte de tabla *Actas* como clave foránea, quedando el esquema de esta tabla de la forma:

**Actas** (code, contribution, volume, initpage, endpage, status, document, meetingcode)

Tabla **BookOfChapters**: se forma a partir del tipo de entidad *BookOfChapters* incorporando todos los atributos de la misma, quedando el esquema de esta tabla de la forma:

**BookOfChapters** (code, isbn, title, pages, year, status, editors, editorial, city, country)

Tabla **Chapter**: se forma a partir del tipo de entidad *Chapter* incorporando todos los atributos de la misma. Además participa en una interrelación binaria 1:N con la entidad *BookOfChapters* con cardinalidad máxima muchos, por lo que el identificador del tipo de entidad *BookOfChapters*, pasa a formar parte de tabla *Chapter* como clave foránea, quedando el esquema de esta tabla de la forma:

**Chapter** (code, title, initPage, endPage, status, document, isbn)

Tabla **PersonallInformation**: se forma a partir del tipo de entidad *PersonallInformation* incorporando todos los atributos de la misma, quedando el esquema de esta tabla de la forma:

**PersonallInformation** (familyName, name, treatment, category, university, address, city, state, country, zip, email, phone, fax, departament)

Tabla **Author**: se forma a partir del tipo de entidad *Author* incorporando todos los atributos de la misma. Además participa en una interrelación binaria 1:1 con la entidad *PersonallInformation* de forma parcial, por lo que el identificador del tipo de entidad *PersonallInformation*, pasa a formar parte de tabla *Author* como clave alterna y foránea, quedando el esquema de esta tabla de la forma:

**Author** (familyName, name)

Tabla **Member**: se forma a partir del tipo de entidad *Member* incorporando todos los atributos de la misma. Además participa en una interrelación binaria 1:1 con la entidad *PersonallInformation* de forma parcial, por lo que el identificador del tipo de entidad *PersonallInformation*, pasa a formar parte de tabla *Member* como clave alterna y foránea, quedando el esquema de esta tabla de la forma:

**Member** (familyName, name, email\_2, phone\_2, web, fromDate, todote, image, dni, birthDate, sex, numOfficial, postalCode)

Tabla **Article\_Author**: se forma por la interrelación binaria N:N entre los tipos de entidad *Article* y *Author*, de forma que las claves primarias de cada una de las tipos de entidades formarán por agregación la clave principal de esta tabla, quedando el esquema de esta tabla de la forma:

**Article\_Author** (*codeArticle*, *familyName*, *name*)

Tabla **Actas\_Author**: se forma por la interrelación binaria N:N entre los tipos de entidad *Actas* y *Author*, de forma que las claves primarias de cada una de las tipos de entidades formarán por agregación la clave principal de esta tabla, quedando el esquema de esta tabla de la forma:

**Actas\_Author** (*codeActas*, *familyName*, *name*)

Tabla **Book\_Author**: se forma por la interrelación binaria N:N entre los tipos de entidad *Books* y *Author*, de forma que las claves primarias de cada una de las tipos de entidades formarán por agregación la clave principal de esta tabla, quedando el esquema de esta tabla de la forma:

**Book\_Author** (*codeBook*, *familyName*, *name*)

Tabla **Chapter\_Author**: se forma por la interrelación binaria N:N entre los tipos de entidad *Chapter* y *Author*, de forma que las claves primarias de cada una de las tipos de entidades formarán por agregación la clave principal de esta tabla, quedando el esquema de esta tabla de la forma:

**Chapter\_Author** (*codeChapter*, *familyName*, *name*)

Una vez que se ha llevado a cabo el proceso anterior obtenemos el siguiente resultado:

**Esquema Relacional: Gestion de Información de Grupos de Investigación (Básico)**

**Article** (*code*, *title*, *year*, *number*, *volume*, *initPage*, *endPage*, *document*,



*status, issn)*

**Journal** (*issn, editorial, abbreviation, name, year, totalCites, impactFactor, inmedIndex, numArticles, citedHalflife*)

**Book** (*code, isbn, title, pages, year, status, editorial, city, country, document*)

**Meeting** (*code, name, abbreviation, organization, status, country, city, isbn, editorial, editors, actaTitle, type, fromDate, toDate, deadline, intention, url*)

**Acta** (*code, contribution, volume, initPage, endPage, status, document, meetingCode*)

**BookOfChapters** (*code, isbn, title, pages, year, status, editors, editorial, city, country*)

**Chapter** (*code, title, initPage, endPage, status, document, codeBookofChapter*)

**PersonallInformation** (*familyName, name, treatment, category, university, address, city, state, country, zip, email, phone, fax, departament*)

**Author** (*familyName, name* )

**Member** (*familyName, name, email\_2, phone\_2, web, fromDate, toDate, image*)

**Article\_Author** (*codeArticle, familyName, name\_*)

**Actas\_Author** (*codeActas, familyName, name\_*)

**Book\_Author** (*codeBook, familyName, name\_*)

**Chapter\_Author** (*codeChapter, familyName, name\_*)

### 1.6.1.2 Normalización del modelo

Tabla **Journal**: los atributos de esta tabla son atómicos, por lo que la tabla se encuentra en *FN1*, pero existen dependencias funcionales por lo que no satisface la *FNBC*.

Las dependencias funcionales existentes en la relación *Journal* son:

*Journal.issn* → *Journal.editorial*

*Journal.issn* → *Journal.abbreviation*

*Journal.issn* → *Journal.name*

*Journal.(year, issn)* → *Journal.totalCites*

*Journal.(year, issn)* → *Journal.impactFactor*

*Journal.(year, issn)* → *Journal.inmedIndex*

*Journal.(year, issn)* → *Journal.numArticles*

*Journal.(year, issn)* → *Journal.citedHalfLife*

Como se puede observar existen dependencias entre atributos que no son determinantes funcionales y que es necesario eliminar. Por tanto, la relación *Journal* debe ser descompuesta, quedando el nuevo esquema de la forma:

**Journal** (*issn*, *editorial*, *abbreviation*, *name*)

**Isi** (*year*, *issn*, *totalCites*, *impactFactor*, *inmedIndex*, *numArticles*, *citedhalfLife*)

Una vez que se ha llevado a cabo la normalización del modelo obtenemos el siguiente resultado:

**Esquema Relacional: Gestión de Información de Grupos de Investigación**

**Article** (*code*, *title*, *year*, *number*, *volume*, *initPage*, *endPage*, *document*,

*status, issn)*

**Journal** (*issn, editorial, abbreviation, name*)

**Isi** (*year, issn, totalcites, impactFactor, inmedIndex, numArticles, citedhalflife*)

**Book** (*code, isbn, title, pages, year, status, editorial, city, country, document*)

**Meeting** (*code, name, abbreviation, organization, status, country, city, isbn, editorial, editors, actaTitle, type, fromDate, toDate, deadline, intention, url*)

**Acta** (*code, contribution, volume, initPage, endPage, status, document, meetingCode*)

**BookOfChapters** (*code, isbn, title, pages, year, status, editors, editorial, city, country*)

**Chapter** (*code, title, initPage, endPage, status, document, isbn*)

**PersonalInformation** (*familyName, name, treatment, category, university, address, city, state, country, zip, email, phone, fax, departament*)

**Author** (*familyName, name*)

**Member** (*familyName, name, email\_2, phone\_2, web, fromDate, toDate, image*)

**Article\_Author** (*codeArticle, familyName, name*)

**Actas\_Author** (*codeActas, familyName, name*)

**Book\_Author** (*codeBook, familyName, name*)

**Chapter\_Author** (*codeChapter, familyName, name*)

La representación gráfica del esquema resultante para la base de datos GIGI se puede apreciar en el diagrama de la página siguiente.

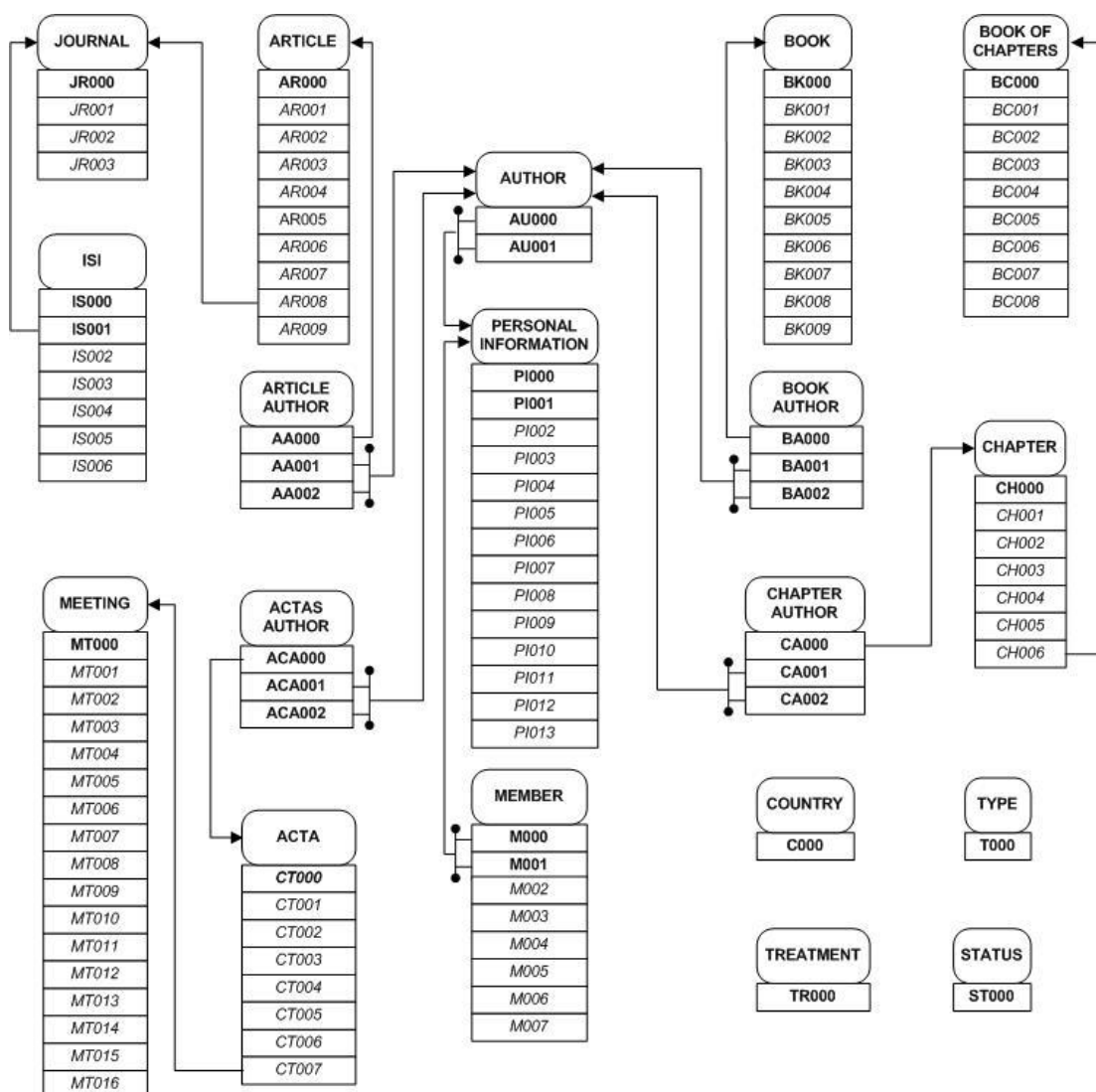


Figura 1.1.4: Diagrama Entidad-Relación Curriculm Vitae (Vista 2)

### 1.6.2 Modelo Relacional Currículm Vitae

El proceso de obtención del esquema relacional consiste en la aplicación de las reglas de transformación al modelo conceptual del currículm vitae descrito anteriormente. Como no se han considerado atributos compuestos ni múltiples en ninguno de los objetos existentes en el modelo conceptual, así como tampoco existen relaciones jerárquicas, nuestro proceso inicial consistirá en el paso de los tipos de entidad y relación a relaciones del modelo relacional.

### 1.6.2.1 Creación de las tablas

Dado que no existen atributos compuestos ni múltiples en ninguno de los objetos, y el modelo conceptual no presenta relaciones de tipo jerárquico, no es necesaria la aplicación de las reglas preparatorias de transformación, y por consiguiente, en este proceso es conveniente comenzar aplicando las propias reglas de transformación a los tipos de entidades, teniendo en consideración para ello los tipos de interrelación en los que se ven involucrados.

Tabla **Member**: se forma a partir del tipo de entidad *Member* incorporando todos los atributos de la misma. Como se vio anteriormente, esta tabla ya aparece en el esquema relacional básico, pero ahora debe modificarse dado que el tipo de entidad del que se deriva también mantiene una relación con el tipo de entidad *Specialization*, perteneciente al esquema conceptual del currículum. De esta forma, puesto que se trata de una interrelación 1:N, en la que el tipo de entidad *Member* participa con cardinalidad máxima N, y el otro con cardinalidades (1,1), el identificador del tipo de entidad *Specialization* pasa a formar parte de la tabla *Member*, actuando como clave foránea hacia la tabla *Specialization*, que posteriormente será descrita. Así, la tabla *Member* queda de la siguiente forma:

**Member** (*familyName*, *name*, *email\_2*, *phone\_2*, *web*, *fromDate*, *toDate*, *image*, *dni*, *birthDate*, *sex*, *numOfficial*, *postalCode*, ***codeSpecialization***)

Tabla **Degree**: se forma a partir del tipo de entidad *Degree*, incorporando todos los atributos del mismo. Dado que este tipo de entidad es débil por identificación respecto al tipo de entidad *Member*, los atributos identificadores de éste pasan a la tabla *Degree* como clave principal y como clave foránea, quedando finalmente como se muestra a continuación:

**Degree** (*familyName*, *name*, *codeDegree*, *engineering*, *institution*, *fromDate*, *toDate*, *doctorate*)

Tabla **ProfessionalStatus**: se forma a partir del tipo de entidad *ProfessionalStatus*, incorporando todos los atributos del mismo. Dado que este

tipo de entidad es débil por identificación respecto al tipo de entidad *Member*, los atributos identificadores de éste pasan a la tabla *ProfessionalStatus* como clave principal y como clave foránea, quedando finalmente como se muestra a continuación:

**ProfessionalStatus** (*familyName*, *name*, *organism*, *faculty*, *category*, *fromDate*, *orgAddress*, *orgPhone*)

Tabla **Activity**: se forma a partir del tipo de entidad *Activity*, incorporando todos los atributos del mismo. Puesto que este tipo de entidad es débil por identificación respecto al tipo de entidad *Member*, los atributos identificadores de éste pasan a la tabla *Activity* como clave principal y foránea, resultando la siguiente estructura de la tabla:

**Activity** (*familyName*, *name*, *nameAct*, *code*, *fromDate*, *toDate*, *position*, *institution*)

Tabla **Language**: se forma a partir del tipo de entidad *Language*, incorporando todos los atributos del mismo. Dado que este tipo de entidad es débil por identificación respecto al tipo de entidad *Member*, los atributos identificadores de éste pasan a la tabla *Language* como clave principal y foránea, quedando la tabla como se muestra a continuación:

**Language** (*familyName*, *name*, *language*, *speak*, *reading*, *writing*)

Tabla **ForeignInstitution**: se forma a partir del tipo de entidad *ForeignInstitution*, incorporando todos los atributos del mismo. Dado que este tipo de entidad es débil por identificación respecto al tipo de entidad *Member*, los atributos identificadores de éste pasan a la tabla *ForeignInstitution* como clave principal y foránea, quedando la tabla como se muestra a continuación:

**ForeignInstitution** (*familyName*, *name*, *nameFI*, *code*, *address*, *country*, *year*, *duration*)

Tabla **Thesis**: se forma a partir del tipo de entidad *Thesis*, incorporando todos los atributos del mismo. Puesto que este tipo de entidad mantiene una interrelación de cardinalidad 1:N con el tipo de entidad *Member*, actuando con cardinalidad máxima muchos, el identificador de éste último tipo de entidad pasa a formar parte de la tabla *Thesis* como clave foránea, resultando la tabla con la siguiente estructura:

**Thesis** (title, *doctorate*, *university*, *faculty*, *year*, *qualification*, *familyName*, *name*)

Tabla **Thesis\_Member**: se forma a partir del tipo de interrelación N:N que mantienen los tipos de entidad *Thesis* y *Member*, de tal forma que los atributos identificadores de ambos tipos de entidad pasan a formar parte de esta tabla como claves principales y foráneas. La tabla queda de la siguiente forma:

**Thesis\_Member** (title, familyName, name)

Tabla **FinanceProject**: se forma a partir del tipo de entidad del mismo nombre, y como tal incorpora todos los atributos del mismo. Puesto que mantiene una relación 1:N con el tipo de entidad *Member*, siendo éste último el que participa con cardinalidad máxima 1, el identificador de dicho tipo de entidad pasa a formar parte de la tabla *FinanceProject* como clave foránea, quedando la tabla de la siguiente forma:

**FinanceProject** (title, *fundingOrg*, *fromDate*, *toDate*, *researcher*, *familyname*, *name*)

Tabla **FinanceProject\_Member**: se forma a partir del tipo de interrelación N:N que mantienen los tipos de entidad *FinanceProject* y *Member*, de tal manera que los atributos identificadores de ambos tipos de entidad pasan a formar parte de esta tabla como claves principales y foráneas, resultando la siguiente tabla:

**FinanceProject\_Member** (title, familyName, name)

Tabla **ResearchContract**: se forma a partir del tipo de entidad del mismo nombre, y como tal incorpora todos los atributos del mismo. Puesto que mantiene una relación 1:N con el tipo de entidad *Member*, siendo éste último el que participa con cardinalidad máxima 1, el identificador de dicho tipo de entidad pasa a formar parte de la tabla *ResearchContract* como clave foránea, quedando la tabla de la siguiente forma:

**ResearchContract** (title, fundingOrg, fromDate, toDate, researcher, familyname, name)

Tabla **ResearchContract\_Member**: se forma a partir del tipo de interrelación N:N que mantienen los tipos de entidad *ResearchContract* y *Member*, de tal manera que los atributos identificadores de ambos tipos de entidad pasan a formar parte de esta tabla como claves principales y foráneas, resultando la siguiente tabla:

**ResearchContract\_Member** (title, familyName, name)

Tabla **Patent**: se forma a partir del tipo de entidad *Patent*, incorporando todos los atributos del mismo, y quedando la tabla de la siguiente forma:

**Patent** (title, numberApplication, priorityCountry, date, priority, firstInstitution, exportCountries, companyInUse)

Tabla **Patent\_Member**: se forma a partir del tipo de interrelación N:N que mantienen los tipos de entidad *Patent* y *Member*, de tal manera que los atributos identificadores de ambos tipos de entidad pasan a formar parte de esta tabla como claves principales y foráneas, resultando la siguiente tabla:

**Patent\_Member** (title, familyName, name)



Tabla **Specialization**: se forma a partir del tipo de entidad *Specialization*, incorporando el único atributo del que dispone, y por tanto resultando la siguiente tabla:

**Specialization** (*codeSpecialization*)

Tabla **PositionType**: se forma a partir del tipo de entidad *PositionType*, incorporando el único atributo del que dispone, y por tanto quedando la tabla de la siguiente forma:

**PositionType** (*namePT*)

Tabla **OtherMerits**: se forma a partir del tipo de entidad *OtherMerits*, incorporando todos sus atributos. Dado que este tipo de entidad es débil por identificación del tipo de entidad *Member* y *PositionType*, incorpora los atributos identificadores de ambos, actuando como claves principales y foráneas de la tabla, cuya estructura final es la siguiente:

**OtherMerits** (*familyName, name, namePT, merit*)

Una vez que se ha llevado a cabo el proceso anterior obtenemos el siguiente resultado:

**Esquema Relacional: Gestion de Información de Grupos de Investigación (Currículum Vitae)**

**Member** (*familyName, name*, *email\_2, phone\_2, web, fromDate, todate, image, dni, birthDate, sex, numOfficial, postalCode, codeSpecialization*)

**Degree** (*familyName, name, codeDegree, engineering*, *institution, fromDate, toDate, doctorate*)

**ProfessionalStatus** (*familyName, name*, *organism, faculty, category, fromDate, orgAddress, orgPhone*)

**Activity** (*familyName, name, nameAct, code*, *fromDate, toDate, position, institution*)

**Language** (familyName, name, language, speak, reading, writing)

**ForeignInstitution** (familyName, name, nameFI, code, address, country, year, duration)

**Thesis** (title, doctorate, university, faculty, year, qualification, familyName, name)

**Thesis\_Member** (title, familyName, name)

**FinanceProject** (title, fundingOrg, fromDate, toDate, researcher, familyname, name)

**FinanceProject\_Member** (title, familyName, name)

**ResearchContract** (title, fundingOrg, fromDate, toDate, researcher, familyname, name)

**ResearchContract\_Member** (title, familyName, name)

**Patent** (title, numberApplication, priorityCountry, date, priority, firstInstitution, exportCountries, companyInUse)

**Patent\_Member** (title, familyName, name)

**Specialization** (codeSpecialization)

**PositionType** (namePT)

**OtherMerits** (familyName, name, namePT, merit)

### 1.6.2.2 Normalización del modelo

Como se aprecia en las tablas resultantes del análisis, todas las relaciones se encuentran normalizadas, por lo que no es necesario realizar ninguna operación de normalización sobre las tablas definidas para esta parte del problema.

Por tanto, la representación gráfica del esquema resultante para la base de datos GIGI, en concreto para el cometido destinado a mantener información sobre el currículum de los miembros del grupo de investigación, se puede apreciar en el diagrama de la página siguiente.

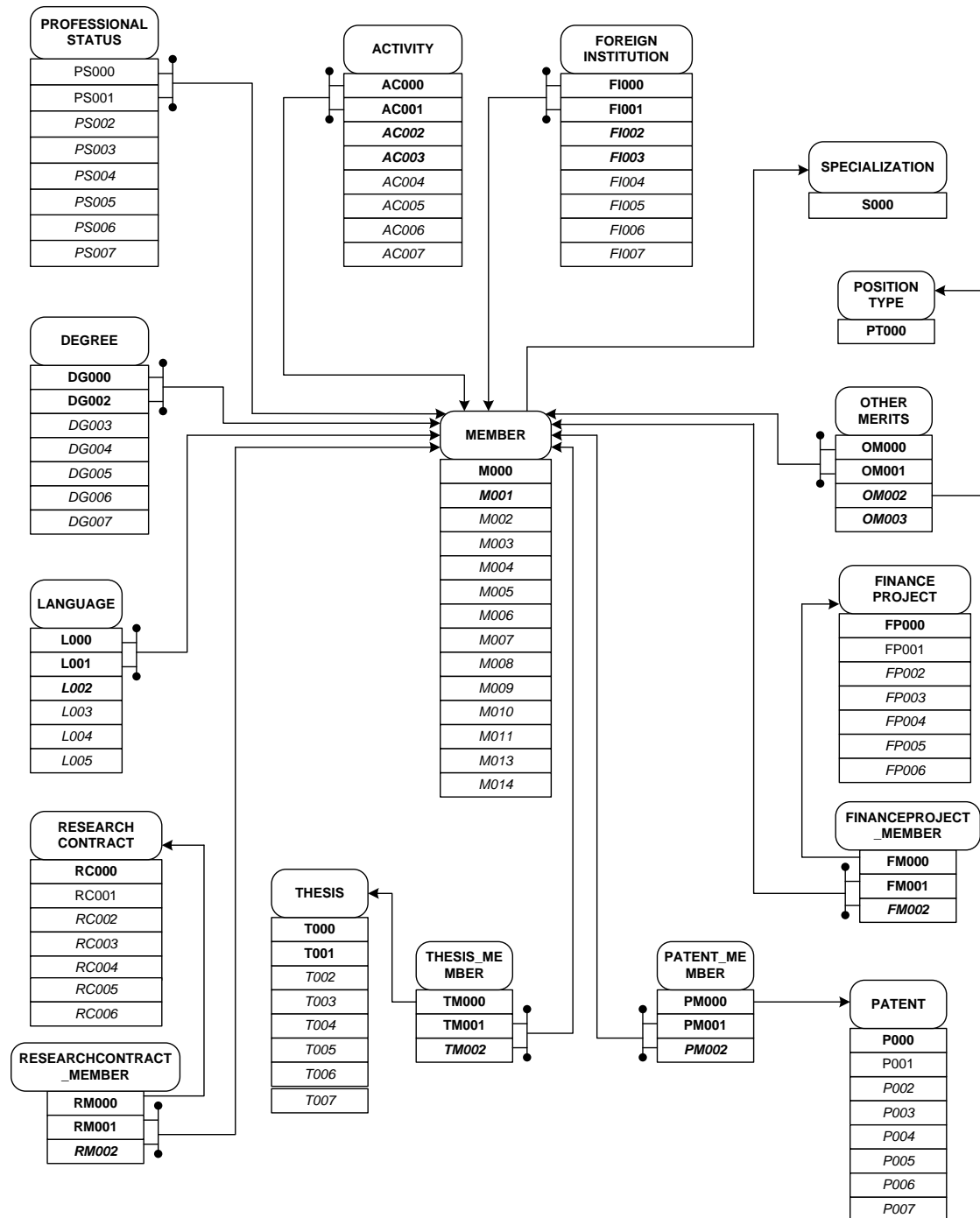


Figura 1.1.5: Diagrama Entidad Relación de Currículum Vitae

### 1.6.3 Diseño del esquema definitivo

Una vez definido el modelo relacional, se presenta el esquema final de la base de datos, que servirá para el desarrollo definitivo en MySQL. A

continuación se describen los atributos o campos de las tablas, especificándose su significado definitivo y tipos de datos y requisitos requeridos.

**Tabla 1.1.32:** Tabla Article

ARTICLE			
code	AR000	ROWID	Primary Key
title	AR001	VARCHAR(255)	NOT NULL
year	AR002	NUMBER(4)	
number	AR003	NUMBER(4)	
volume	AR004	NUMBER(4)	
initPage	AR005	NUMBER(4)	
endPage	AR006	NUMBER(4)	
document	AR007	%File	
status	AR008	%Status	Enumerated. Ref(Status)
issn	AR009	Ref(Journal)	Foreign key

**Tabla 1.1.33:** Tabla Journal

JOURNAL			
issn	JR000	VARCHAR(10)	Primary Key
name	JR001	VARCHAR(128)	NOT NULL
abbreviation	JR002	VARCHAR(64)	NOT NULL
editorial	JR003	VARCHAR(128)	

**Tabla 1.1.34:** Tabla ISI

ISI			
year	IS000	NUMBER(4)	Primary Key
issn	JR000	Ref(Journal)	Primary Key. Foreign Key
impactFactor	IS002	NUMBER(2,3)	NOT NULL
totalCites	IS003	NUMBER(6)	
inmedIndex	IS004	NUMBER(2,3)	
numArticles	IS005	NUMBER(6)	
citedHalfLife	IS006	NUMBER(2,1)	

Tabla 1.1.35: Tabla Book

BOOK			
code	BK000	ROWID	Primary Key
isbn	BK001	VARCHAR(16)	UNIQUE
title	BK002	VARCHAR(255)	NOT NULL. UNIQUE
year	BK003	NUMBER(4)	
pages	BK004	NUMBER(5)	
status	BK005	%Status	Enumerated. Ref(Status)
editorial	BK006	VARCHAR(64)	
city	BK007	VARCHAR(32)	
country	BK008	%Country	Enumerated. Ref(Country)
document	BK009	%Document	

Tabla 1.1.36: Tabla Book Of Chapters

BOOK OF CHAPTERS			
code	BC000	ROWID	Primary Key
isbn	BC001	VARCHAR(16)	UNIQUE
title	BC002	VARCHAR(255)	NOT NULL.UNIQUE
year	BC003	NUMBER(4)	
pages	BC004	NUMBER(5)	
status	BC005	%Status	Enumerated. Ref(Status)
editors	BC006	VARCHAR(64)	
editorial	BC007	VARCHAR(64)	
city	BC008	VARCHAR(32)	
country	BC009	%Country	Enumerated. Ref(Country)

Tabla 1.1.37: Tabla Chapter

CHAPTER			
code	CH000	ROWID	Primary Key
title	CH001	VARCHAR(255)	NOT NULL
initPage	CH002	NUMBER(4)	

endPage	CH003	NUMBER(4)	
status	CH004	%Status	Enumerated. Ref(Status)
document	CH005	%Document	
isbn	CH006	Ref(Book of Chapters)	Foreign key

**Tabla 1.1.38:** Tabla Meeting

MEETING			
Code	MT000	ROWID	PrimaryKey
Name	MT001	VARCHAR(255)	NOT NULL
abbreviation	MT002	VARCHAR(32)	NOT NULL
organization	MT003	VARCHAR(128)	NOT NULL
Status	MT004	%Status	Enumerated. Ref(Status)
Country	MT005	%Country	Enumerated. Ref(Country)
City	MT006	VARCHAR(32)	
Type	MT007	%Type	Enumerated. Ref(Type)
fromDate	MT008	DATE	
toDate	MT009	DATE	
Deadline	MT010	VARCHAR(64)	
Intention	MT011	VARCHAR(32)	
url	MT012	VARCHAR(255)	
isbn	MT013	VARCHAR(16)	
actaTitle	MT014	VARCHAR(255)	NOT NULL
editorial	MT015	VARCHAR(255)	
editors	MT016	VARCHAR(64)	

**Tabla 1.1.39:** Tabla Acta

ACTA			
code	CT000	ROWID	Primary Key
contribution	CT001	VARCHAR(255)	NOT NULL
volume	CT002	NUMBER(4)	
initPage	CT003	NUMBER(4)	
endPage	CT004	NUMBER(4)	
status	CT005	%Status	Enumerated. Ref(Status)
document	CT006	%Document	

meetingCode	CT007	Ref(Meeting)	Foreign Key.
-------------	-------	--------------	--------------

Tabla 1.1.40: Tabla Country

COUNTRY			
country	C000	VARCHAR(32)	NOT NULL. UNIQUE

Tabla 1.1.41: Tabla Type

TYPE			
type	T000	VARCHAR(1)	NOT NULL. UNIQUE

Tabla 1.1.42: Tabla Treatment

TREATMENT			
treatment	TR000	VARCHAR(4)	NOT NULL. UNIQUE

Tabla 1.1.43: Tabla Status

STATUS			
status	ST000	VARCHAR(32)	NOT NULL. UNIQUE

Tabla 1.1.44: Tabla Personal Information

PERSONAL INFORMATION			
familyName	PI000	VARCHAR(64)	Primary Key
name	PI001	VARCHAR(32)	Primary Key
treatment	PI002	%Treatment	Enumerated. Ref(Treatment)
category	PI003	VARCHAR(64)	
university	PI004	VARCHAR(64)	

departament	PI005	VARCHAR(64)	
address	PI006	VARCHAR(128)	
city	PI007	VARCHAR(32)	
state	PI008	VARCHAR(32)	
country	PI009	%Country	Enumerated. Ref(Country)
zip	PI010	VARCHAR(16)	
email	PI011	VARCHAR(32)	NOT NULL
phone	PI012	VARCHAR(16)	NOT NULL
fax	PI013	VARCHAR(16)	

Tabla 1.1.45: Tabla Author

AUTHOR			
familyName	AU000	Ref(Author)	Primary Key. Foreign Key
name	AU001	Ref(Author)	Primary Key. Foreign Key

Tabla 1.1.46: Tabla Member

MEMBER			
familyName	M000	Ref(Author)	Primary Key. Foreign Key
name	M001	Ref(Author)	Primary Key. Foreign Key
email_2	M002	VARCHAR(32)	
phone_2	M003	VARCHAR(16)	
web	M004	VARCHAR(255)	
fromDate	M005	DATE	NOT NULL
toDate	M006	DATE	
image	M007	%image	
dni	M008	VARCHAR(9)	NOT NULL
birthDate	M009	DATE	
sex	M010	VARCHAR(1)	
numOfficial	M011	NUMBER (15)	
codeSpecialization	M013	Ref(Specialization)	
codePostal	M014	VARCHAR(16)	



**Tabla 1.1.47.** Tabla Article Author

ARTICLE AUTHOR			
codeArticle	AA000	Ref(Article)	Primary Key. Foreign Key
familyName	AA001	Ref(Author)	Primary Key. Foreign Key
name	AA002	Ref(Author)	Primary Key. Foreign Key

**Tabla 1.1.48:** Tabla Actas Author

ACTAS AUTHOR			
codeActas	ACA000	Ref(Actas)	Primary Key. Foreign Key
familyName	ACA002	Ref(Author)	Primary Key. Foreign Key
name	ACA003	Ref(Author)	Primary Key. Foreign Key

**Tabla 1.1.49:** Tabla Books Author

BOOKS AUTHOR			
isbn	BA000	Ref(Books)	Primary Key. Foreign Key
familyName	BA001	Ref(Author)	Primary Key. Foreign Key
name	BA002	Ref(Author)	Primary Key. Foreign Key

**Tabla 1.1.50:** Tabla Chapter Author

CHAPTER AUTHOR			
codeChapter	CA000	Ref(Chapter)	Primary Key. Foreign Key
familyName	CA001	Ref(Author)	Primary Key. Foreign Key
name	CA002	Ref(Author)	Primary Key. Foreign Key

Tabla 1.1.51: Tabla Degree

DEGREE			
codeDegree	DG000	ROWID	Primary Key
engineeering	DG001	VARCHAR(255)	Primary Key
name	DG002	Ref(Member)	Primary Key. Foreign Key
familyname	DG003	Ref(Member)	Primary Key. Foreign Key
doctorate	DG004	VARCHAR(255)	
toDate	DG005	DATE	
fromDate	DG006	DATE	
institution	DG007	VARCHAR(255)	

Tabla 1.1.52: Tabla Professional Status

PROFESSIONAL STATUS			
name	PS000	Ref(Member)	Primary Key. Foreign Key
familyname	PS001	Ref(Member)	Primary Key. Foreign Key
orgPhone	PS002	VARCHAR(10)	
orgAddress	PS003	VARCHAR(255)	
fromDate	PS004	INT(4)	
category	PS005	VARCHAR(255)	
faculty	PS006	VARCHAR(255)	
organism	PS007	VARCHAR(255)	

Tabla 1.1.53: Tabla Specialization

SPECIALIZATION			
codeSpecialization	S000	ROWID	Primary Key

Tabla 1.1.54: Tabla Activity

ACTIVITY			
name	AC000	Ref(Member)	Primary Key.

			Foreign Key
familyname	AC001	Ref(Member)	Primary Key. Foreign Key
code	AC002	ROWID	Primary key
nameAct	AC003	VARCHAR(255)	Primary key
Institution	AC004	VARCHAR(255)	
position	AC005	VARCHAR(255)	
toDate	AC006	DATE	
fromDate	AC007	DATE	

Tabla 1.1.55: Tabla Language

LANGUAGE			
name	L000	Ref(Member)	Primary Key. Foreign Key
familyname	L001	Ref(Member)	Primary Key. Foreign Key
language	L002	VARCHAR(255)	Primary key
speak	L003	VARCHAR(1)	
writing	L004	VARCHAR(1)	
reading	L005	VARCHAR(1)	

Tabla 1.1.56: Tabla Finance project

FINANCE PROJECT			
title	FP000	VARCHAR(255)	Primary Key.
fundingOrg	FP001	VARCHAR(255)	
fromDate	FP002	DATE	
toDate	FP003	DATE	
researcher	FP004	VARCHAR(255)	
name	FP005	Ref(Member)	Primary Key. Foreign Key
familyname	FP006	Ref(Member)	Primary Key. Foreign Key

Tabla 1.1.57: Tabla FinanceProject Member

FINANCEPROJECT MEMBER			
title	FM000	Ref(FinanceProject)	Primary Key.
name	FM001	Ref(Member)	Primary Key. Foreign Key

familyname	FM002	Ref(Member)	Primary Key. Foreign Key
------------	-------	-------------	-----------------------------

**Tabla 1.1.58:** Tabla Research Contract

RESEARCH CONTRACT			
title	RC000	VARCHAR(255)	Primary Key.
fundingOrg	RC001	VARCHAR(255)	
fromDate	RC002	DATE	
toDate	RC003	DATE	
researcher	RC004	VARCHAR(255)	
name	RC005	Ref(Member)	Primary Key. Foreign Key
familyname	RC006	Ref(Member)	Primary Key. Foreign Key

**Tabla 1.1.59:** Tabla ResearchContract Member

RESEARCHCONTRACT MEMBER			
title	RM000	Ref(ResearchContract)	Primary Key.
name	RM001	Ref(Member)	Primary Key. Foreign Key
familyname	RM002	Ref(Member)	Primary Key. Foreign Key

**Tabla 1.1.60:** Tabla Patent

PATENT			
title	P000	VARCHAR(255)	Primary Key.
numberApplication	P001	NUMBER (4)	
priorityCountry	P002	VARCHAR(255)	
date	P003	DATE	
priority	P004	NUMBER (4)	
firstInstitution	P005	VARCHAR(255)	
exportCountries	P006	VARCHAR(255)	
companyInUse	P007	VARCHAR(255)	

Tabla 1.1.61: Tabla Patent Member

PATENT MEMBER			
title	PM000	Ref(Patent)	Primary Key.
name	PM001	Ref(Member)	Primary Key. Foreign Key
familyname	PM002	Ref(Member)	Primary Key. Foreign Key

Tabla 1.1.62: Tabla Foreign Institution

FOREIGN INSTITUTION			
name	FI000	Ref(Member)	Primary Key. Foreign Key
familyname	FI001	Ref(Member)	Primary Key. Foreign Key
nameFI	FI002	VARCHAR(255)	Primary Key.
code	FI003	ROWID	Primary Key.
address	FI004	VARCHAR(255)	
country	FI005	VARCHAR(255)	
year	FI006	DATE	
duration	FI007	NUMBER (2)	

Tabla 1.1.63: Tabla Thesis

THESIS			
title	T000	VARCHAR(255)	Primary Key.
doctorate	T001	VARCHAR(255)	
university	T002	VARCHAR(255)	
faculty	T003	VARCHAR(255)	
year	T004	INT(4)	
qualification	T005	NUMBER(2)	
name	T006	Ref(Member)	Primary Key. Foreign Key
familyname	T007	Ref(Member)	Primary Key. Foreign Key

**Tabla 1.1.64:** Tabla Thesis Member

THESIS MEMBER			
title	TM000	Ref(Thesis)	Primary Key.
familynname	TM001	Ref(Member)	Primary Key. Foreign Key
name	TM002	Ref(Member)	Primary Key. Foreign Key

**Tabla 1.1.65:** Tabla PositionType

POSITION TYPE			
namePT	PT000	VARCHAR(255)	Primary Key.

**Tabla 1.1.66:** Tabla OtherMerits

OTHER MERITS			
name	OM000	Ref(Member)	Primary Key. Foreign Key
familynname	OM001	Ref(Member)	Primary Key. Foreign Key
namePT	OM002	Ref(PositionType)	Primary Key. Foreign Key.
merit	OM003	VARCHAR(255)	Primary Key.

### 1.6.4 Sentencias de definición de la base de datos en MySQL

A continuación se muestran tanto las sentencias para crear la base de datos en su totalidad como las tablas que componen su esquema, así como la carga de las tablas maestras que utiliza dicha base de datos. Todas las sentencias están en MySQL 5.0, lenguaje que se definió en la especificación para la realización de la misma.

Las sentencias para la creación y definición de la base de datos GIGI se muestran a continuación:

```
/* mysql -u root < gigi.txt -p */

drop database if exists GIGI;

create database GIGI;

use GIGI;


DROP TABLE IF EXISTS Type;

CREATE TABLE Type(

    Type VARCHAR(1) primary Key);

LOAD DATA LOCAL INFILE "gigi-Type.txt" INTO TABLE Type;


DROP TABLE IF EXISTS Treatment;

CREATE TABLE Treatment(

    Treatment VARCHAR(16) primary Key);

LOAD DATA LOCAL INFILE "gigi-Treatment.txt" INTO TABLE
Treatment;


DROP TABLE IF EXISTS Status;

CREATE TABLE Status(

    Status VARCHAR(32) primary Key);

LOAD DATA LOCAL INFILE "gigi-Status.txt" INTO TABLE Status;


DROP TABLE IF EXISTS Country;

CREATE TABLE Country(

    Country VARCHAR(32) primary Key);

LOAD DATA LOCAL INFILE "gigi-Country.txt" INTO TABLE
Country;
```

```
DROP TABLE IF EXISTS Specialization;

CREATE TABLE Specialization(

    codeSpecialization VARCHAR(255) primary Key);


DROP TABLE IF EXISTS Journal;

CREATE TABLE Journal(

    issn VARCHAR(10) primary Key,

    editorial VARCHAR(128),

    abbreviation VARCHAR(64) NOT NULL,

    name VARCHAR(128) NOT NULL

)ENGINE=InnoDB;


DROP TABLE IF EXISTS Article;

CREATE TABLE Article(

    code INT AUTO_INCREMENT primary Key,

    title VARCHAR(255) NOT NULL,

    year INT(4),

    volume INT(4),

    initpage INT(4),

    endpage INT(4),

    document LONGBLOB,

    status VARCHAR(32),

    issn VARCHAR(10) ,

    KEY (issn),

    FOREIGN KEY (ISSN) REFERENCES Journal (issn)

    ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE=InnoDB;
```



```
DROP TABLE IF EXISTS Isi;

CREATE TABLE Isi(

    year INT(4),

    issn VARCHAR(10),

    impactfactor DECIMAL(7,3),

    totalcites INT(6),

    inmedindex DECIMAL(7,3),

    numarticles INT(6),

    citedhalflife DECIMAL(7,3),

    primary key(Year,ISSN),

    KEY (issn),

    FOREIGN KEY (ISSN) REFERENCES Journal (issn)

        ON DELETE CASCADE ON UPDATE CASCADE)

    ENGINE=InnoDB;


DROP TABLE IF EXISTS Book;

CREATE TABLE Book(

    code INT AUTO_INCREMENT primary Key,

    isbn VARCHAR(16),

    title VARCHAR(255),

    year INT(4),

    pages INT(5),

    status VARCHAR(32),

    editorial VARCHAR(64),

    city VARCHAR(32),

    country VARCHAR(32),

    document LONGBLOB)

    ENGINE=InnoDB;
```

```
DROP TABLE IF EXISTS BookOfChapters;

CREATE TABLE BookOfChapters(

    code INT AUTO_INCREMENT primary Key,

    isbn VARCHAR(16),

    title VARCHAR(255),

    year INT(4),

    pages INT(5),

    status VARCHAR(32),

    editors VARCHAR(64),

    editorial VARCHAR(64),

    city VARCHAR(32),

    country VARCHAR(32))

ENGINE=InnoDB;


DROP TABLE IF EXISTS Chapter;

CREATE TABLE Chapter(

    code INT AUTO_INCREMENT primary Key,

    title VARCHAR(255) NOT NULL,

    initPage INT(4),

    endPage INT(4),

    status VARCHAR(32),

    document LONGBLOB,

    codebookofchapter INT,

    KEY (codebookofchapter),

    FOREIGN KEY (codebookofchapter) REFERENCES

BookOfChapters (code)

    ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE=InnoDB;
```

```
DROP TABLE IF EXISTS Meeting;

CREATE TABLE Meeting(
    code INT AUTO_INCREMENT primary Key,
    name VARCHAR(255) NOT NULL,
    abbreviation VARCHAR(32) NOT NULL,
    organization VARCHAR(128) NOT NULL,
    type VARCHAR(1),
    status VARCHAR(32),
    city VARCHAR(32),
    country VARCHAR(32),
    fromdate DATE,
    todate DATE,
    deadline VARCHAR(64),
    intention VARCHAR(32),
    url VARCHAR(255),
    isbn VARCHAR(16),
    actaTitle VARCHAR(255) NOT NULL,
    editorial VARCHAR(255),
    editors VARCHAR(64)
)

ENGINE=InnoDB;

CREATE TABLE Acta(
    code INT AUTO_INCREMENT primary Key,
    contribution VARCHAR(255) NOT NULL,
    volume INT(4),
    initPage INT(4),
    endPage INT(4),
```

```
status VARCHAR(32),  
document LONGBLOB,  
meetingcode INT,  
KEY (MeetingCode),  
FOREIGN KEY (meetingcode) REFERENCES Meeting (code)  
ON DELETE CASCADE ON UPDATE CASCADE)  
ENGINE=InnoDB;
```

```
DROP TABLE IF EXISTS PersonalInformation;
```

```
CREATE TABLE PersonalInformation(  
    familyname VARCHAR(64),  
    name VARCHAR(32),  
    treatment VARCHAR(16),  
    category VARCHAR(64),  
    university VARCHAR(64),  
    departament VARCHAR(64),  
    address VARCHAR(128),  
    city VARCHAR(32),  
    state VARCHAR(32),  
    country VARCHAR(32),  
    zip VARCHAR(16),  
    email VARCHAR(32) NOT NULL,  
    phone VARCHAR(16) NOT NULL,  
    fax VARCHAR(16),  
    Primary Key (familyname,name))  
ENGINE=InnoDB ;
```

```
DROP TABLE IF EXISTS Author;

CREATE TABLE Author(

    familyname VARCHAR(64),

    name VARCHAR(32),

    Primary Key (familyname,name),

    KEY (FamilyName, Name),

    FOREIGN KEY (familyname, name) REFERENCES
PersonalInformation (familyname, name)

    ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE=InnoDB;


DROP TABLE IF EXISTS Member;

CREATE TABLE Member(

    familyname VARCHAR(64),

    name VARCHAR(32),

    email_2 VARCHAR(32),

    phone_2 VARCHAR(16),

    web VARCHAR(255),

    fromdate DATE NOT NULL,

    todate DATE,

    image MEDIUMBLOB,

    dni VARCHAR(9),

    birthDate DATE,

    sex VARCHAR(1),

    numOfficial INT(15),

    postalCode INT(5),

    codeSpecialization VARCHAR(255),

    Primary Key (familyname,name),
```

```
        KEY (familyname, name),

        FOREIGN KEY (FamilyName, Name) REFERENCES
PersonalInformation (familyname, name)

        ON DELETE CASCADE ON UPDATE CASCADE,

        FOREIGN KEY (codeSpecialization) REFERENCES
Specialization (codeSpecialization)

        ON UPDATE CASCADE)

ENGINE=MyISAM;

DROP TABLE IF EXISTS ArticleAuthor;

CREATE TABLE ArticleAuthor(

    familyname VARCHAR(64),

    name VARCHAR(32),

    codearticle INT,

    Primary Key (familyname,name, codearticle),

    KEY (familyname, name),

    FOREIGN KEY (familyname, name) REFERENCES
PersonalInformation (familyname, name)

        ON DELETE CASCADE ON UPDATE CASCADE,

    KEY (codearticle),

    FOREIGN KEY (codearticle) REFERENCES Article (code)

    ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE=InnoDB;

DROP TABLE IF EXISTS ActasAuthor;

CREATE TABLE ActasAuthor(

    familyname VARCHAR(64),

    name VARCHAR(32),

    codeactas INT,
```

```
        Primary Key (familyname,name, codeactas),
        KEY (familyname, name),
        FOREIGN KEY (familyname, name) REFERENCES
PersonalInformation (familyname, name)
        ON DELETE CASCADE ON UPDATE CASCADE,
        KEY (codeactas),
        FOREIGN KEY (codeactas) REFERENCES Acta (code)
        ON DELETE CASCADE ON UPDATE CASCADE)
ENGINE=InnoDB;

DROP TABLE IF EXISTS BookAuthor;

CREATE TABLE BookAuthor(
        familyname VARCHAR(64),
        name VARCHAR(32),
        codebook INT,
        Primary Key (familyname,name, codebook),
        KEY (familyname, name),
        FOREIGN KEY (familyname, name) REFERENCES
PersonalInformation (familyname, name)
        ON DELETE CASCADE ON UPDATE CASCADE,
        KEY (codebook),
        FOREIGN KEY (codebook) REFERENCES Book (code)
        ON DELETE CASCADE ON UPDATE CASCADE)
ENGINE=InnoDB;

DROP TABLE IF EXISTS ChapterAuthor;

CREATE TABLE ChapterAuthor(
        familyname VARCHAR(64),
```

```
name VARCHAR(32),
codechapter INT,
Primary Key (familyname,name, codechapter),
KEY (familyname, name),
FOREIGN KEY (familyname, name) REFERENCES
PersonalInformation (familyname, name)
ON DELETE CASCADE ON UPDATE CASCADE,
KEY (codechapter),
FOREIGN KEY (codechapter) REFERENCES Chapter (code)
ON DELETE CASCADE ON UPDATE CASCADE)
ENGINE=InnoDB;

DROP TABLE IF EXISTS Degree;
CREATE TABLE Degree(
familyname VARCHAR(64),
name VARCHAR(32),
codeDegree INT,
engineering VARCHAR(255),
institution VARCHAR(255),
fromDate DATE,
toDate DATE,
doctorate VARCHAR(255),
Primary Key (familyname, name, codeDegree,
engineering),
FOREIGN KEY (familyname, name) REFERENCES Member
(familyname, name)
```



```
        ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE=MyISAM;

DROP TABLE IF EXISTS ProfessionalStatus;

CREATE TABLE ProfessionalStatus(

    familyname VARCHAR(64),

    name VARCHAR(32),

    organism VARCHAR(255),

    faculty VARCHAR(255),

    category VARCHAR(255),

    fromDate DATE,

    orgAddress VARCHAR(255),

    orgPhone VARCHAR(10),

    Primary Key (familyname, name),

    FOREIGN KEY (familyname, name) REFERENCES Member
(familyname, name)

    ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE=MyISAM;

DROP TABLE IF EXISTS Activity;

CREATE TABLE Activity(

    familyname VARCHAR(64),

    name VARCHAR(32),

    nameAct VARCHAR(255),

    code INT,

    fromDate DATE,

    toDate DATE,
```

```
        position VARCHAR(255),
        institution VARCHAR(255),
        Primary Key (familyname, name, nameAct, code),
        FOREIGN KEY (familyname, name) REFERENCES Member
(familyname, name)
        ON DELETE CASCADE ON UPDATE CASCADE)
        ENGINE=MyISAM;

DROP TABLE IF EXISTS Language;

CREATE TABLE Language(
        familyname VARCHAR(64),
        name VARCHAR(32),
        language VARCHAR(50),
        speak VARCHAR(1),
        reading VARCHAR(1),
        writing VARCHAR(1),
        Primary Key (familyname, name, language),
        FOREIGN KEY (familyname, name) REFERENCES Member
(familyname, name)
        ON DELETE CASCADE ON UPDATE CASCADE)
        ENGINE=MyISAM;

DROP TABLE IF EXISTS ForeignInstitution;

CREATE TABLE ForeignInstitution(
        familyname VARCHAR(64),
        name VARCHAR(32),
```

```
nameFI VARCHAR(32),
code INT,
address VARCHAR(32),
country VARCHAR(32),
year INT(4),
duration INT,
Primary Key (familyname, name, nameFI, code),
FOREIGN KEY (familyname, name) REFERENCES Member
(familyname, name)
ON DELETE CASCADE ON UPDATE CASCADE)
ENGINE=MyISAM;

DROP TABLE IF EXISTS Thesis;

CREATE TABLE Thesis(
title VARCHAR(32),
doctorate VARCHAR(255),
university VARCHAR(255),
faculty VARCHAR(255),
year INT(4),
qualification DECIMAL(4,2),
familyname VARCHAR(64),
name VARCHAR(32),
Primary Key (title),
FOREIGN KEY (familyname, name) REFERENCES Member
(familyname, name)
ON DELETE CASCADE ON UPDATE CASCADE)
ENGINE=MyISAM;
```

```
DROP TABLE IF EXISTS Thesis_Member;

CREATE TABLE Thesis_Member(

    title VARCHAR(32),

    familyname VARCHAR(64),

    name VARCHAR(32),

    Primary Key (title, familyName, name),

    FOREIGN KEY (title) REFERENCES Thesis (title)

        ON DELETE CASCADE ON UPDATE CASCADE,

    FOREIGN KEY (familyname, name) REFERENCES Member

(familyname, name)

        ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE=MyISAM;


DROP TABLE IF EXISTS FinanceProject;

CREATE TABLE FinanceProject(

    title VARCHAR(255),

    fundingOrg VARCHAR(255),

    fromDate DATE,

    toDate DATE,

    researcher VARCHAR(255),

    familyname VARCHAR(64),

    name VARCHAR(32),

    Primary Key (title),

    FOREIGN KEY (familyname, name) REFERENCES Member

(familyname, name)

        ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE=MyISAM;
```

```
DROP TABLE IF EXISTS FinanceProject_Member;

CREATE TABLE FinanceProject_Member(

    title VARCHAR(255),

    familyname VARCHAR(64),

    name VARCHAR(32),

    Primary Key (title, familyName, name),

    FOREIGN KEY (title) REFERENCES FinanceProject (title)

        ON DELETE CASCADE ON UPDATE CASCADE,

    FOREIGN KEY (familyname, name) REFERENCES Member

(familyname, name)

        ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE=MyISAM;


DROP TABLE IF EXISTS ResearchContract;

CREATE TABLE ResearchContract(

    title VARCHAR(255),

    fundingOrg VARCHAR(255),

    fromDate DATE,

    toDate DATE,

    researcher VARCHAR(255),

    familyname VARCHAR(64),

    name VARCHAR(32),

    Primary Key (title),

    FOREIGN KEY (familyname, name) REFERENCES Member

(familyname, name)

        ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE=MyISAM;
```

```
DROP TABLE IF EXISTS ResearchContract_Member;

CREATE TABLE ResearchContract_Member(

    title VARCHAR(255),

    familyname VARCHAR(64),

    name VARCHAR(32),

    Primary Key (title, familyName, name),

    FOREIGN KEY (title) REFERENCES ResearchContract

(title)

    ON DELETE CASCADE ON UPDATE CASCADE,

    FOREIGN KEY (familyname, name) REFERENCES Member

(familyname, name)

    ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE=MyISAM;


DROP TABLE IF EXISTS Patent;

CREATE TABLE Patent(

    title VARCHAR(255),

    numberApplication INT,

    priorityCountry VARCHAR(50),

    date DATE,

    priority INT,

    firstInstitution VARCHAR(50),

    exportCountries VARCHAR(255),

    companyInUse VARCHAR(255),

    Primary Key (title))

ENGINE=InnoDB;
```

```
DROP TABLE IF EXISTS Patent_Member;

CREATE TABLE Patent_Member(
    title VARCHAR(255),
    familyname VARCHAR(64),
    name VARCHAR(32),
    Primary Key (title, familyName, name),
    FOREIGN KEY (title) REFERENCES Patent (title)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (familyname, name) REFERENCES Member
(familyname, name)
        ON DELETE CASCADE ON UPDATE CASCADE)
    ENGINE=MyISAM;

DROP TABLE IF EXISTS PositionType;

CREATE TABLE PositionType(
    namePT VARCHAR(64) primary Key);

DROP TABLE IF EXISTS OtherMerits;

CREATE TABLE OtherMerits(
    familyname VARCHAR(64),
    name VARCHAR(32),
    namePT VARCHAR(64),
    merit VARCHAR(255),
    Primary Key (familyName, name, namePT, merit),
    FOREIGN KEY (familyname, name) REFERENCES Member
(familyname, name)
```

```
ON DELETE CASCADE ON UPDATE CASCADE,  
  
    FOREIGN KEY (namePT) REFERENCES PositionType (namePT)  
  
    ON DELETE CASCADE ON UPDATE CASCADE)  
  
ENGINE=MyISAM;
```

### **1.6.5 Procedimientos de manipulación de la base de datos en MySQL**

A continuación se muestran los procedimientos básicos para la manipulación de la base de datos realizados en función de las funcionalidades requeridas para el sistema. Todas las sentencias están en MySQL 5.0, lenguaje que se definió en la especificación para la realización de la misma.

Las sentencias para la definición de procedimientos de manipulación de la base de datos GIGI se muestran a continuación, apareciendo además diversas sentencias de prueba del sistema realizadas. Como posteriormente se expondrá, en el script de carga de información, se hace uso de los procedimientos de inserción correspondientes para añadir datos de prueba a la base de datos.

```
use GIGI;  
  
/*  
 * FUNCIONES PARA INSERCIÓN  
 */  
  
/* PERSONAL INFORMATION */  
  
BEGIN;  
  
delimiter //  
  
drop procedure if exists nuevoInformacionPersonal //
```



```
create procedure nuevoInformacionPersonal (IN nombre
VARCHAR(32), IN apellidos VARCHAR(64), IN tratamiento
VARCHAR(16), IN categoria VARCHAR(64), IN universidad
VARCHAR(64), IN departamento VARCHAR(64), IN direccion
VARCHAR(128), IN ciudad VARCHAR(32), IN estado VARCHAR(32), IN
pais VARCHAR(32), IN cp VARCHAR(16), IN mail VARCHAR(32), IN
telefono VARCHAR(16), IN fax VARCHAR(16))

begin

insert into PersonalInformation values (apellidos, nombre,
tratamiento, categoria, universidad, departamento, direccion,
ciudad, estado, pais, cp, mail, telefono, fax);

end;

//

delimiter ;

COMMIT;

/* AUTHOR */

BEGIN;

delimiter //

drop procedure if exists nuevoautor //

create procedure nuevoautor (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64))

begin

insert into Author values (apellidos, nombre);

end;

//

delimiter ;

COMMIT;
```

```
/* SPECIALIZATION */

BEGIN;

delimiter //

drop procedure if exists nuevaEspecializacion//

create procedure nuevaEspecializacion (IN
codigoEspecializacion INT )
begin
insert into Specialization values (codigoEspecializacion);
end;

//

delimiter ;

COMMIT;

/* MEMBER */

BEGIN;

delimiter //

drop procedure if exists nuevomiembro//

create procedure nuevomiembro (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64), IN tratamiento VARCHAR(16), IN categoria
VARCHAR(64), IN universidad VARCHAR(64), IN departamento
VARCHAR(64), IN direccion VARCHAR(128), IN ciudad VARCHAR(32),
IN estado VARCHAR(32), IN pais VARCHAR(32), IN zip VARCHAR(16),
IN cp VARCHAR(16), IN mail VARCHAR(32), IN mail2 VARCHAR(32), IN
telefono VARCHAR(16), IN telefono2 VARCHAR(16), IN fax
VARCHAR(16), IN web VARCHAR(255), IN miembordesde DATE, IN
miembrohasta DATE, IN foto MEDIUMBLOB, IN dni VARCHAR(9), IN
fechaNacimiento DATE, IN sexo VARCHAR(1), IN numOficial INT(15),
IN codeSpecializacion INT)
```

```
begin

    insert into PersonalInformation values (apellidos, nombre,
tratamiento, categoria, universidad, departamento, direccion,
ciudad, estado, pais,zip, mail, telefono, fax);

    insert into Member values (apellidos, nombre, mail2,
telefono2, web, miembrodesde, miembrohasta, foto, dni,
fechaNacimiento, sexo, numOficial, cp, codEspecializacion);

end;

//

delimiter ;

COMMIT;

/* THESIS */

BEGIN;

delimiter //

drop procedure if exists nuevathesis//

create procedure nuevathesis (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64), IN titulo VARCHAR(255), IN doctorado
VARCHAR(255), IN universidad VARCHAR(255), IN facultad
VARCHAR(255), IN anno INT(4), IN calificacion DECIMAL(4,2) )

begin

    insert into Thesis values (titulo, doctorado, universidad,
facultad, anno, calificacion, apellidos, nombre);

end;

//

delimiter ;

COMMIT;
```

```
/* THESIS MEMBER */

BEGIN;

delimiter //

drop procedure if exists asociarMiembroThesis//

create procedure asociarMiembroThesis (IN titulo
VARCHAR(255), IN nombre VARCHAR(32), apellidos VARCHAR(64) )
begin
insert into Thesis_Member values (titulo, apellidos,
nombre);

end;

//

delimiter ;

COMMIT;

/* JOURNAL */

BEGIN;

delimiter //

drop procedure if exists nuevarevista//

create procedure nuevarevista (IN issn VARCHAR(10), IN
editorial VARCHAR(128), IN abreviatura VARCHAR(64), IN nombre
VARCHAR(128))
begin
insert into Journal values (issn, editorial, abreviatura,
nombre);

end;

//

delimiter ;
```

```
COMMIT;

/* ARTICLE */

BEGIN;

delimiter //

drop procedure if exists nuevoarticulo//

create procedure nuevoarticulo (IN codigo INTEGER, IN
titulo VARCHAR(255), IN anyo INT(4), IN volumen INT(4), IN
paginainicial INT(4), IN paginafinal INT(4), IN documento
LONGBLOB, IN estado VARCHAR(32), IN issn VARCHAR(10))

begin

insert into Article values (codigo, titulo, anyo, volumen,
paginainicial, paginafinal, documento, estado, issn);

end;

//

delimiter ;

COMMIT;

/* BOOKS */

BEGIN;

delimiter //

drop procedure if exists nuevolibro//

create procedure nuevolibro (IN codigo INTEGER, IN isbn
VARCHAR(16), IN titulo VARCHAR(255), IN anyo INT(4), IN paginas
INT(5), IN estado VARCHAR(32), IN editorial VARCHAR(64), IN
ciudad VARCHAR(32), IN pais VARCHAR(32), IN documento LONGBLOB)
```

```
begin

    insert into Book values (codigo, isbn, titulo, anyo,
paginas, estado, editorial, ciudad, pais, documento);

end;

//

delimiter ;

COMMIT;


/* ACTAS */

BEGIN;

delimiter //

drop procedure if exists nuevaacta//


create procedure nuevaacta (IN codigo INTEGER, IN
contribucion VARCHAR(255), IN volumen INT(4), IN paginainicial
INT(4), IN paginafinal INT(4), IN estado VARCHAR(32), IN
documento LONGBLOB, IN codigomeeting INTEGER)

begin

    insert into Acta values (codigo, contribucion, volumen,
paginainicial, paginafinal, estado, documento, codigomeeting);

end;

//

delimiter ;

COMMIT;


/* MEETING */

BEGIN;

delimiter //

drop procedure if exists nuevomeeting//
```

```
create procedure nuevomeeting (IN codigo INTEGER, IN nombre
VARCHAR(255), IN abreviatura VARCHAR(32), IN organizacion
VARCHAR(128), IN tipo VARCHAR(1), IN estado VARCHAR(32), IN
ciudad VARCHAR(32), IN pais VARCHAR(32), IN desde DATE, IN hasta
DATE, IN fechalimite VARCHAR(64), IN intencion VARCHAR(32), IN
url VARCHAR(255), IN isbn VARCHAR(16), IN tituloacta
VARCHAR(255), IN editorial VARCHAR(255), IN editores
VARCHAR(64))

begin

insert into Meeting values (codigo, nombre, abreviatura,
organizacion, tipo, estado, ciudad, pais, desde, hasta,
fechalimite, intencion, url, isbn, tituloacta, editorial,
editores);

end;

//

delimiter ;

COMMIT;

/* ISI */

BEGIN;

delimiter //

drop procedure if exists nuevoisi//

create procedure nuevoisi (IN anyo INT(4), IN issn
VARCHAR(10), IN factorimpacto DECIMAL(6,2), IN totalcitas
INT(6), IN indiceinmediato DECIMAL(6,2), IN numarticulos INT(6),
IN vidamediacitas DECIMAL(6,2))

begin

insert into Isi values (anyo, issn, factorimpacto,
totalcitas, indiceinmediato, numarticulos, vidamediacitas);

end;

//
```

```
delimiter ;

COMMIT;


/* BOOKOFCHAPTERS */

BEGIN;

delimiter //

drop procedure if exists nuevolibrocapitulos//


create procedure nuevolibrocapitulos (IN codigo INTEGER, IN
isbn VARCHAR(16), IN titulo VARCHAR(255), IN anyo INT(4), IN
paginas INT(5), IN estado VARCHAR(32), IN editores VARCHAR(64),
IN editorial VARCHAR(64), IN ciudad VARCHAR(32), IN pais
VARCHAR(32))

begin

insert into BookOfChapters values (codigo, isbn, titulo,
anyo, paginas, estado, editores, editorial, ciudad, pais);

end;

//

delimiter ;

COMMIT;


/* CHAPTER */

BEGIN;

delimiter //

drop procedure if exists nuevocapitulo//
```



```
create procedure nuevocapitulo (IN codigo INTEGER, IN
titulo VARCHAR(255), IN paginainicial INT(4), IN paginafinal
INT(4), IN estado VARCHAR(32), IN documento LONGBLOB, IN
codigolibrocapitulos INTEGER)

begin

insert into Chapter values (codigo, titulo, paginainicial,
paginafinal, estado, documento, codigolibrocapitulos);

end;

//

delimiter ;

COMMIT;

/* ARTICLEAUTHOR */

BEGIN;

delimiter //

drop procedure if exists asociararticuloautor//

create procedure asociararticuloautor (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN codigoarticulo INTEGER )

begin

insert into ArticleAuthor values (apellidos, nombre,
codigoarticulo);

end;

//

delimiter ;

COMMIT;

/* ACTASAUTHOR */

BEGIN;
```

```
delimiter //
```

```
drop procedure if exists asociaractasautor//
```

```
create procedure asociaractasautor (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN codigoacta INTEGER )
begin
insert into ActasAuthor values (apellidos, nombre,
codigoacta);
end;
//
delimiter ;
COMMIT;
```

```
/* CHAPTERAUTHOR */
```

```
BEGIN;
```

```
delimiter //
```

```
drop procedure if exists asociarcapituloautor//
```

```
create procedure asociarcapituloautor (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN codigocapitulo INTEGER )
begin
insert into ChapterAuthor values (apellidos, nombre,
codigocapitulo);
end;
//
delimiter ;
COMMIT;
```

```
/* BOOKAUTHOR */

BEGIN;

delimiter //

drop procedure if exists asociarlibroautor//

create procedure asociarlibroautor (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN codigolibro INTEGER )

begin

insert into BookAuthor values (apellidos, nombre,
codigolibro);

end;

//

delimiter ;

COMMIT;

/* DEGREE */

BEGIN;

delimiter //

drop procedure if exists nuevodegree//

create procedure nuevodegree (IN apellidos VARCHAR(32), IN
nombre VARCHAR(64), IN codigoDegree INT, IN ingeniero
VARCHAR(255), IN institucion VARCHAR(255), IN fechaInicio DATE,
IN fechaFin DATE, IN doctorado VARCHAR(255) )

begin

insert into Degree values (apellidos, nombre, codigoDegree,
ingeniero, institucion, fechaInicio, fechaFin, doctorado);

end;

//
```

```
delimiter ;

COMMIT;

/* PROFESSIONALSTATUS */

BEGIN;

delimiter //

drop procedure if exists nuevostatus//

create procedure nuevostatus (IN apellidos VARCHAR(64), IN
nombre VARCHAR(32), IN organismo VARCHAR(255), IN facultad
VARCHAR(255), IN categoria VARCHAR(255), IN fecha DATE, IN
direccionOrg VARCHAR(255), IN telefonoOrg VARCHAR(10) )

begin

insert into ProfessionalStatus values (apellidos, nombre,
organismo, facultad, categoria, fecha, direccionOrg,
telefonoOrg);

end;

//

delimiter ;

COMMIT;

/* ACTIVITY */

BEGIN;

delimiter //

drop procedure if exists nuevaactividad//

create procedure nuevaactividad (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64), IN codigo INT, IN nombreAct VARCHAR(255),
IN institucion VARCHAR(255), IN posicion VARCHAR(255), IN
fechaInicio DATE, IN fechaFin DATE )
```

```
begin

    insert into Activity values (apellidos, nombre, nombreAct,
codigo, fechaInicio, fechaFin, institucion, posicion);

end;

//

delimiter ;

COMMIT;


/* LANGUAGE */

BEGIN;

delimiter //

drop procedure if exists nuevolanguage//


create procedure nuevolanguage (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64), IN idioma VARCHAR(255), IN hablar
VARCHAR(1), IN leer VARCHAR(1), IN escribir VARCHAR(1) )

begin

    insert into Language values (apellidos, nombre, idioma,
hablar, leer, escribir);

end;

//

delimiter ;

COMMIT;


/* FOREIGNINSTITUTION */

BEGIN;

delimiter //

drop procedure if exists nuevoinstitution//
```

```
create procedure nuevoinstitution (IN nombre VARCHAR(32),
IN apellidos VARCHAR(64), IN nombreCentro VARCHAR(255), IN
codigo INT, IN direccion VARCHAR(255), IN pais VARCHAR(255), IN
anno INT(4), IN duracion INT (2) )

begin

insert into ForeignInstitution values (apellidos, nombre,
nombreCentro, codigo, direccion, pais, anno, duracion);

end;

//

delimiter ;

COMMIT;

/* FINANCE PROJECT */

BEGIN;

delimiter //

drop procedure if exists nuevoProyecto//

create procedure nuevoProyecto (IN titulo VARCHAR(255), IN
entidad VARCHAR(255), IN desde DATE, IN hasta DATE, IN
investigador VARCHAR(255), IN nombre VARCHAR(32), apellidos
VARCHAR(64) )

begin

insert into FinanceProject values (titulo, entidad, desde,
hasta, investigador, apellidos, nombre);

end;

//

delimiter ;

COMMIT;
```

```
/* FINANCEPROJECT MEMBER */

BEGIN;

delimiter //

drop procedure if exists asociarMiembroProyecto//

create procedure asociarMiembroProyecto (IN titulo
VARCHAR(255), IN nombre VARCHAR(32), apellidos VARCHAR(64) )
begin
insert into FinanceProject_Member values (titulo,
apellidos, nombre);

end;

//

delimiter ;

COMMIT;

/* RESEARCH CONTRACT */

BEGIN;

delimiter //

drop procedure if exists nuevoContrato//

create procedure nuevoContrato (IN titulo VARCHAR(255), IN
entidad VARCHAR(255), IN desde DATE, IN hasta DATE, IN
investigador VARCHAR(255), IN nombre VARCHAR(32), apellidos
VARCHAR(64) )
begin
insert into ResearchContract values (titulo, entidad,
desde, hasta, investigador, apellidos, nombre);

end;

//
```

```
delimiter ;

COMMIT;

/* RESEARCHCONTRACT MEMBER */

BEGIN;

delimiter //

drop procedure if exists asociarMiembroContrato//

create procedure asociarMiembroContrato (IN titulo
VARCHAR(255), IN nombre VARCHAR(32), apellidos VARCHAR(64) )

begin

insert into ResearchContract_Member values (titulo,
apellidos, nombre);

end;

//

delimiter ;

COMMIT;

/* PATENT */

BEGIN;

delimiter //

drop procedure if exists nuevaPatente//

create procedure nuevaPatente (IN titulo VARCHAR(255), IN
numeroSolicitud INT(4), IN paisPrioridad VARCHAR(255), IN fecha
DATE, IN prioridad INT(4), IN entidadTitular VARCHAR(255), IN
países VARCHAR(255), IN empresas VARCHAR(255) )

begin
```



```
        insert into Patent values (titulo, numeroSolicitud,
paisPrioridad, fecha, prioridad, entidadTitular, paises,
empresas);

    end;

    //

    delimiter ;

    COMMIT;


    /* PATENT MEMBER*/

    BEGIN;

    delimiter //

    drop procedure if exists asociarMiembroPatente//

    create procedure asociarMiembroPatente (IN titulo
VARCHAR(255), IN nombre VARCHAR(32), apellidos VARCHAR(64) )

    begin

        insert into Patent_Member values (titulo, apellidos,
nombre);

    end;

    //

    delimiter ;

    COMMIT;


    /* POSITION TYPE */

    BEGIN;

    delimiter //

    drop procedure if exists nuevoTipoCargo//
```

```
create procedure nuevoTipoCargo (IN nombreTipoCargo
VARCHAR(64) )

begin

insert into PositionType values (nombreTipoCargo);

end;

//

delimiter ;

COMMIT;


/* OTHER MERITS */

BEGIN;

delimiter //

drop procedure if exists nuevoOtrosMeritos//


create procedure nuevoOtrosMeritos (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN nombreTipoCargo
VARCHAR(64), IN merito VARCHAR(255) )

begin

insert into OtherMerits values (apellidos, nombre,
nombreTipoCargo, merito);

end;

//

delimiter ;

COMMIT;
```

```
/*  
  
* FUNCIONES PARA MODIFICACIÓN  
  
*/  
  
/* AUTHOR */  
  
BEGIN;  
  
delimiter //  
  
drop procedure if exists editarautor//  
  
create procedure editarautor (IN name VARCHAR(32), IN  
apellidos VARCHAR(64), IN tratamiento VARCHAR(16), IN categoria  
VARCHAR(64), IN universidad VARCHAR(64), IN departamento  
VARCHAR(64), IN direccion VARCHAR(128), IN ciudad VARCHAR(32),  
IN estado VARCHAR(32), IN pais VARCHAR(32), IN cp VARCHAR(16),  
IN mail VARCHAR(32), IN telefono VARCHAR(16), IN fax  
VARCHAR(16), IN apellidosnew VARCHAR(64), IN nombrenew  
VARCHAR(32))  
  
begin  
  
update PersonalInformation set familyname=apellidosnew,  
name=nombrenew, treatment=tratamiento, category=categoria,  
university=universidad,  
  
departament=departamento, address=direccion, city=ciudad,  
state=estado, country=pais, zip=cp,  
  
email=mail, phone=telefono, fax=fax  
  
where familyname=apellidos and name=name;  
  
end;  
  
//  
  
delimiter ;  
  
COMMIT;
```

```
#Prueba del metodo

#call  editarautor("David", "Martin", "Ilmo", "Ayudante",
"cordoba", "Análisis Numerico", "la calle", "cordoba",
"cordoba", "españa", "14milalgo", "i72maped@uco.es", "elmio",
"notengo");

/* MEMBER */

BEGIN;

delimiter //

drop procedure if exists editarmiembro//

create procedure editarmiembro (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64), IN tratamiento VARCHAR(16), IN categoria
VARCHAR(64), IN universidad VARCHAR(64), IN departamento
VARCHAR(64), IN direccion VARCHAR(128), IN ciudad VARCHAR(32),
IN estado VARCHAR(32), IN pais VARCHAR(32), IN cp VARCHAR(16),
IN mail VARCHAR(32), IN mail2 VARCHAR(32), IN telefono
VARCHAR(16), IN telefono2 VARCHAR(16), IN fax VARCHAR(16), IN
web VARCHAR(255), IN miembrosdesde DATE, IN miembros hasta DATE, IN
foto MEDIUMBLOB, IN dni VARCHAR(9), IN fechaNacimiento DATE, IN
sexo VARCHAR(1), IN numOficial INT(15), IN codeSpecializacion
INT)

begin

update PersonalInformation set treatment=tratamiento,
category=categoria, university=universidad,

departament=departamento, address=direccion, city=ciudad,
state=estado, country=pais, zip=cp,

email=mail, phone=telefono, fax=fax

where familyname=apellidos and name=nombre;

update Member set email_2=mail2, phone_2=telefono2,
web=web, fromdate=miembrosdesde,
```

```
todate=miembrohasta,          image=foto,          dni=dni,
birthDate=fechaNacimiento,    sex=sexo,    numOficial=numOficial,
codeSpecialization=codeSpecialization

where familyname=apellidos and name=nombre;

end;

//

delimiter ;

COMMIT;


#Prueba del metodo

#call editarmiembro("Moreno", "Manuel", "Ilmo", "Ayudante",
"cordoba",    "Análisis Numérico",    "la calle",    "cordoba",
"cordoba",    "españa",    "14milalgo",    "i02mosam@uco.es",
"otro@uco.es", "elmio", "elotromio", "notengo", "www.uco.es",
"200-10-10",  "2000-11-11",  NULL,  "75709655A",  "1983-02-22",
"F",0000000000000001,2);


/* THESIS MEMBER */

BEGIN;

delimiter //

drop procedure if exists editarThesisMiembro//

create    procedure    editarThesisMiembro    (IN    titulo
VARCHAR(255), IN nombre VARCHAR(32), apellidos VARCHAR(64))

begin

update Thesis_Member set familyname=apellidos, name=nombre
where title=titulo;

end;

//

delimiter ;
```

```
COMMIT;

#Prueba del metodo

#call      editarThesisMiembro("Ingenieria",      "Manuel",
"Moreno");

/* JOURNAL */

BEGIN;

delimiter //

drop procedure if exists editarrevista//

create procedure editarrevista (IN codigo VARCHAR(10), IN
editorial VARCHAR(128), IN abreviatura VARCHAR(64), IN nombre
VARCHAR(128))

begin

update      Journal      set      editorial=editorial,
abbreviation=abreviatura, name=nombre

where issn=codigo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editarrevista("N1", "IEEE", "IET", "IEEE Transactions
in something");
```

```
/* ARTICLE */
```

```
BEGIN;
```

```
delimiter //

drop procedure if exists editararticulo//

create procedure editararticulo (IN codigo INTEGER, IN
titulo VARCHAR(255), IN anyo INT(4), IN volumen INT(4), IN
paginainicial INT(4), IN paginafinal INT(4), IN documento
LONGBLOB, IN estado VARCHAR(32), IN issn VARCHAR(10))

begin

update Article set title=titulo, year=anyo, volume=volumen,
initpage=paginainicial,

endpage=paginafinal, document=documento, status=estado,
issn=issn

where code=codigo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editararticulo(2342, "Redes de computadores", 2000,
NULL, 100, 105, NULL, "Aceptado", "N1");

/* BOOKS */

BEGIN;

delimiter //

drop procedure if exists editarlibro//

create procedure editarlibro (IN codigo INTEGER, IN isbn
VARCHAR(16), IN titulo VARCHAR(255), IN anyo INT(4), IN paginas
```

```
INT(5), IN estado VARCHAR(32), IN editorial VARCHAR(64), IN
ciudad VARCHAR(32), IN pais VARCHAR(32), IN documento LONGBLOB)

begin

update Book set isbn=isbn, title=titulo, year=anyo,
pages=paginas, status=estado, editorial=editorial,

city=ciudad, country=pais, document=documento

where code=codigo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editarlibro(1, "ISBN-53234-232", "Redes de
computadores", 2000, 100, "Publicado", "Prentice Hall",
"Acapulco", "Honolulu", NULL);

/* ACTAS */

BEGIN;

delimiter //

drop procedure if exists editaracta//

create procedure editaracta (IN codigo INTEGER, IN
contribucion VARCHAR(255), IN volumen INT(4), IN paginainicial
INT(4), IN paginafinal INT(4), IN estado VARCHAR(32), IN
documento LONGBLOB, IN codigomeeting INTEGER)

begin

update Acta set contribution=contribucion, volume=volumen,
initpage=paginainicial,
```



```
        endpage=paginafinal,    status=estado,    document=documento,
meetingcode=codigomeeting

        where code=codigo;

        end;

        //

        delimiter ;

        COMMIT;


        #Prueba del metodo

        #call editaracta(1, "Como vuelan las mariposas", 34, 2000,
2200, "Publicado", NULL, NULL);


        /* MEETING */

        BEGIN;

        delimiter //

        drop procedure if exists editarmeeting//


        create procedure editarmeeting (IN codigo INTEGER, IN
nombre VARCHAR(255), IN abreviatura VARCHAR(32), IN organizacion
VARCHAR(128), IN tipo VARCHAR(1), IN estado VARCHAR(32), IN
ciudad VARCHAR(32), IN pais VARCHAR(32), IN desde DATE, IN hasta
DATE, IN fechalimite VARCHAR(64), IN intencion VARCHAR(32), IN
url VARCHAR(255), IN isbn VARCHAR(16), IN tituloacta
VARCHAR(255), IN editorial VARCHAR(255), IN editores
VARCHAR(64))

        begin

                update Meeting set name=nombre, abbreviation=abreviatura,
organization=organizacion, type=tipo,

                status=estado, city=ciudad, country=pais, fromdate=desde,
todate=hasta, deadline=fechalimite,
```

```
intention=intencion,          url=url,          isbn=isbn,
actatitle=tituloacta, editorial=editorial, editors=editores

where code=codigo;

end;

//

delimiter ;

COMMIT;


#Prueba del metodo

#call editarmeeting(34, "Congreso de Informatica", "CEDI",
"UG",  "N",  "Celebrado", "Granada", "España", "2006-02-23",
"2006-02-26", "Noviembre 2006", "ni idea", "http://cedi.es",
"ISBN-22342-23423", "Lo que paso en el cedi", "Alguna sera",
NULL);


/* ISI */

BEGIN;

delimiter //

drop procedure if exists editarisi//

create procedure editarisi (IN anyo INT(4), IN codigo
VARCHAR(10), IN factorimpacto DECIMAL(7,3), IN totalcitas
INT(6), IN indiceinmediato DECIMAL(7,3), IN numarticulos INT(6),
IN vidamediacitas DECIMAL(7,3))

begin

update      Isi      set      impactfactor=factorimpacto,
totalcites=totalcitas, inmedindex=indiceinmediato,

numarticles=numarticulos, citedhalfllife=vidamediacitas

where year=anyo and issn=codigo;

end;
```

```
//  
  
delimiter ;  
  
COMMIT;  
  
#Prueba del metodo  
  
#call editarisi(2000, "N1", 1.97, 32000, 0.453, 2000, 2.3);  
  
/* CHAPTER */  
  
BEGIN;  
  
delimiter //  
  
drop procedure if exists editarcapitulo//  
  
create procedure editarcapitulo (IN codigo INTEGER, IN  
titulo VARCHAR(255), IN paginainicial INT(4), IN paginafinal  
INT(4), IN estado VARCHAR(32), IN documento LONGBLOB, IN  
codigolibrocapitulos INTEGER)  
  
begin  
  
update Chapter set title=titulo, initpage=paginainicial,  
endpage=paginafinal, status=estado,  
  
document=documento, codebookofchapter=codigolibrocapitulos  
where code=codigo;  
  
end;  
  
//  
  
delimiter ;  
  
COMMIT;
```

```
#Prueba del metodo
```

```
#call editarcapitulo(1, "La subida de las mareas", 100,
115, "Aceptado", NULL, 1);

/* BOOKOFCHAPTERS */

BEGIN;

delimiter //

drop procedure if exists editarlibrocapitulos//

create procedure editarlibrocapitulos (IN codigo INTEGER,
IN isbn VARCHAR(16), IN titulo VARCHAR(255), IN anyo INT(4), IN
paginas INT(5), IN estado VARCHAR(32), IN editores VARCHAR(64),
IN editorial VARCHAR(64), IN ciudad VARCHAR(32), IN pais
VARCHAR(32))

begin

update BookOfChapters set isbn=isbn, title=titulo,
year=anyo, pages=paginas, status=estado,

editors=editores, editorial=editorial, city=ciudad,
country=pais

where code=codigo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editarlibrocapitulos(1, "ISBN-1231-1232", "El libro
de las subida de las mareas", 2002, 210, "Aceptado", "Sr. Pepito
Grillo", "Patience Bar", "Conchinchina", "EEUU");

/* DEGREE */

BEGIN;
```

```
delimiter //
```

```
drop procedure if exists editardegree//
```

```
create procedure editardegree (IN codigo INT, IN ingeniero
VARCHAR(255), IN nombre VARCHAR(32), IN apellidos VARCHAR(64),IN
doctorado VARCHAR(255), IN fechaInicio DATE, IN fechaFin DATE,
IN institucion VARCHAR(255) )

begin

update Degree set codeDegree=codigo, engineering=ingeniero,
name=nombre, familyName=apellidos,

doctorate=doctorado, toDate=fechaInicio, fromDate=fechaFin,
institution=institucion

where codeDegree=codigo and engineering=ingeniero and
name=nombre and familyName=apellidos;

end;

//

delimiter ;

COMMIT;
```

```
#Prueba del metodo

#call editardegree(1, "Maria","Manuel", "Moreno",
"doctorado Software", "2000-02-20", "2000-07-20", "cordoba");
```

```
/* PROFESSIONALSTATUS */

BEGIN;

delimiter //
```

```
drop procedure if exists editarstatus//
```

```
create procedure editarstatus (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64), IN telefonoOrg VARCHAR(10), IN
direccionOrg VARCHAR(255), IN fecha DATE, IN categoria
VARCHAR(255), IN facultad VARCHAR(255), IN organismo
VARCHAR(255) )

begin

update ProfessionalStatus set name=nombre,
familyName=apellidos, orgPhone=telefonoOrg,
orgAddress=direccionOrg, fromDate=fecha,

category=categoria, faculty=facultad, organism=organismo
where name=nombre and familyName=apellidos;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editarstatus("Manuel", "Moreno", "957562965", "calle
Santiago", "2002-03-03", "ingeniero junior", "cordoba",
"ADESLA");

/* ACTIVITY */

BEGIN;

delimiter //

drop procedure if exists editaractividad//

create procedure editaractividad (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64), IN codigo INT, IN nomActividad
VARCHAR(255), IN institucion VARCHAR(255), IN posicion
VARCHAR(255), IN fechaInicio DATE, IN fechaFin DATE )
```

```
begin

    update Activity    set  name=nombre,  familyName=apellidos,
code=codigo, nameAct=nomActividad, institution=institucion,

    position=posicion, toDate=fechaInicio, fromDate=fechaFin

    where code= codigo and nameAct=nomActividad and name=nombre
and familyName=apellidos;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editaractividad("Manuel", "Moreno", 1, "realizar Base
de Datos", "UCO", "no se", "2000-12-02", "2000-10-18");

/* LANGUAGE */

BEGIN;

delimiter //

drop procedure if exists editarlanguage//

create procedure editarlanguage (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64), IN idioma VARCHAR(255), IN hablar
VARCHAR(1), IN leer VARCHAR(1), IN escribir VARCHAR(1) )

begin

    update Language set  name=nombre,  familyName=apellidos,
language=idioma, speak=hablar, reading=leer, writing=escribir
```

```
    where      language=idioma      and      name=nombre      and
familyName=apellidos;

end;
```

```
//  
  
delimiter ;  
  
COMMIT;  
  
#Prueba del metodo  
  
#call  editarlanguage("Manuel","Moreno",  "Frances",  "C",  
"C", "C");  
  
  
/* FOREIGNINSTITUTION */  
  
BEGIN;  
  
delimiter //  
  
drop procedure if exists editarinstitution//  
  
  
create procedure editarinstitution (IN nombre VARCHAR(32),  
IN apellidos VARCHAR(64), IN nombreCentro VARCHAR(255), IN  
codigo INT, IN direccion VARCHAR(255), IN pais VARCHAR(255), IN  
anno INT(4), IN duracion INT (2) )  
  
begin  
  
update      ForeignInstitution      set      name=nombre,  
familyName=apellidos,      nameFI=nombreCentro,      code=codigo,  
address=direccion,  
  
country=pais, year=anno, duration=duracion  
  
where nameFI=nombreCentro and code=codigo and name=nombre  
and familyName=apellidos;  
  
end;  
  
//
```

```
delimiter ;  
  
COMMIT;
```



```
#Prueba del metodo

#call  editarinstitution("Manuel",  "Moreno",  "College",1,
"calle la luna", "Inglaterra",2000, 4);


/* THESIS */

BEGIN;

delimiter //

drop procedure if exists editarthesis//

create procedure editarthesis (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64), IN titulo VARCHAR(255), IN doctorado
VARCHAR(255), IN universidad VARCHAR(255), IN facultad
VARCHAR(255), IN anno INT(4), IN calificacion DECIMAL(4,2) )

begin

update Thesis set name=nombre, familyName=apellidos,
title=titulo, doctorate=doctorado, university=universidad,

faculty=facultad, year=anno, qualification=calificacion

where title=titulo and name=nombre and
familyName=apellidos;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo
```

```
#call  editarthesis("Manuel",  "Moreno","Ingenieria",
"doctorado", "cordoba","cordoba", 2000, 09);


/* FINANCE PROJECT */
```

```
BEGIN;

delimiter //

drop procedure if exists editarProyecto//

create procedure editarProyecto (IN titulo VARCHAR(255), IN
entidad VARCHAR(255), IN desde DATE, IN hasta DATE, IN
investigador VARCHAR(255), IN nombre VARCHAR(32), apellidos
VARCHAR(64))

begin

update FinanceProject set fundingOrg=entidad,
fromDate=desde, toDate=hasta,

researcher=investigador, familyname=apellidos, name=nombre
where title=titulo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editarProyecto("Titulo del proyecto", "UCO S.A.",
"2007-01-18", "2008-02-20", "Miguel Ángel Esteban Martínez",
"María", "Catalá Carbonero");

/* FINANCEPROJECT MEMBER */

BEGIN;
```

```
delimiter //

drop procedure if exists editarProyectoMiembro//
```

```
create procedure editarProyectoMiembro (IN titulo
VARCHAR(255), IN nombre VARCHAR(32), apellidos VARCHAR(64))

begin

update FinanceProject_Member set familyname=apellidos,
name=nombre

where title=titulo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editarProyectoMiembro("Titulo del proyecto", "María",
"Catalá Carbonero");

/* RESEARCH CONTRACT */

BEGIN;

delimiter //

drop procedure if exists editarContrato//

create procedure editarContrato (IN titulo VARCHAR(255), IN
entidad VARCHAR(255), IN desde DATE, IN hasta DATE, IN
investigador VARCHAR(255), IN nombre VARCHAR(32), apellidos
VARCHAR(64))

begin

update ResearchContract set fundingOrg=entidad,
fromDate=desde, toDate=hasta,
```

```
researcher=investigador, familyname=apellidos, name=nombre

where title=titulo;

end;
```

```
//

delimiter ;

COMMIT;

#Prueba del metodo

#call editarContrato("Titulo del contrato", "UCO S.A.",
"2007-05-18", "2008-07-20", "Ernesto José Figura Tiesa",
"María", "Catalá Carbonero");

/* RESEARCHCONTRACT MEMBER */

BEGIN;

delimiter //

drop procedure if exists editarContratoMiembro//

create procedure editarContratoMiembro (IN titulo
VARCHAR(255), IN nombre VARCHAR(32), apellidos VARCHAR(64))

begin

update ResearchContract_Member set familyname=apellidos,
name=nombre

where title=titulo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editarContratoMiembro("Titulo del contrato", "Javi",
```

```
"Muriel Zafra");
```

```
/* PATENT */
```

```
BEGIN;

delimiter //

drop procedure if exists editarPatente//

create procedure editarPatente (IN titulo VARCHAR(255), IN
numeroSolicitud INT(4), IN paisPrioridad VARCHAR(255), IN fecha
DATE, IN prioridad INT(4), IN entidadTitular VARCHAR(255), IN
países VARCHAR(255), IN empresas VARCHAR(255))

begin

update Patent set numberApplication=numeroSolicitud,
priorityCountry=paisPrioridad,

date=fecha, priority=prioridad,
firstInstitution=entidadTitular, exportCountries=países,

companyInUse=empresas

where title=titulo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editarPatente("Titulo de la patente", 0112,
"Nigeria", "2006-05-10", 0025, "OtraEntidadTitular",
"Afganistán, Liechestein, Namibia, Azerbaijan", "Algunas
empresas de por allí");
```

```
/* PATENT MEMBER */
```

```
BEGIN;
```

```
delimiter //
```

```
drop procedure if exists editarPatenteMiembro//

create procedure editarPatenteMiembro (IN titulo
VARCHAR(255), IN nombre VARCHAR(32), apellidos VARCHAR(64))
begin
update Patent_Member set familyname=apellidos, name=nombre
where title=titulo;
end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editarPatenteMiembro("Titulo de la patente", "Javi",
"Muriel Zafra");

/* SPECIALIZATION */

BEGIN;

delimiter //

drop procedure if exists editarEspecializacion//

create procedure editarEspecializacion (IN
codigoEspecializacionAnt INT, IN codigoEspecializacionNew INT)
begin
update Specialization set
codeSpecialization=codigoEspecializacionNew
where codeSpecialization=codigoEspecializacionAnt;
end;

//

delimiter ;
```

```
COMMIT;

#Prueba del metodo

#call editarEspecializacion(3, 30);

/* POSITION TYPE */

BEGIN;

delimiter //

drop procedure if exists editarTipoCargo//

create procedure editarTipoCargo (IN nombreTipoCargoAnt
VARCHAR(64), IN nombreTipoCargoNew VARCHAR(64))

begin

update PositionType set namePT=nombreTipoCargoNew

where namePT=nombreTipoCargoAnt;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call editarTipoCargo("Becario", "Becario/a");

/* OTHER MERITS */

BEGIN;

delimiter //

drop procedure if exists editarOtrosMeritos//

create procedure editarOtrosMeritos (IN apellidosAnt
VARCHAR(64), IN nombreAnt VARCHAR(32), IN nombreTipoCargoAnt
VARCHAR(64), IN meritoAnt VARCHAR(255),

IN apellidosNew VARCHAR(64), IN nombreNew VARCHAR(32), IN
nombreTipoCargoNew VARCHAR(64), IN meritoNew VARCHAR(255))
```

```
begin
    update      OtherMerits      set      familyname=apellidosNew,
name=nombreNew, namePT=nombreTipoCargoNew,
    merit=meritoNew
    where      familyname=apellidosAnt   and   name=nombreAnt   and
namePT=nombreTipoCargoAnt and merit=meritoAnt;
end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call      editarOtrosMeritos("Muriel      Zafra",      "Javi",
"Becario/a", "Alumno colaborador del departamento, por ejemplo",
    #"Catalá      Carbonero",      "María",      "Becario/a",      "Alumna
colaboradora del grupo de investigación");
```

```
/*

* FUNCIONES PARA BORRADO

*/
```



```
/* AUTHOR */

BEGIN;

delimiter //

drop procedure if exists borrarautor//

create procedure borrarautor (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64))

begin

delete from Author where familyname=apellidos and
name=nombre;

delete from PersonalInformation where familyname=apellidos
and name=nombre;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarautor("David", "Martin");

/* MEMBER */

BEGIN;
```

```
delimiter //

drop procedure if exists borrarmiembro//
```

```
create procedure borrar miembro (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64))

begin

delete from Member where familyname=apellidos and
name=nombre;

delete from PersonalInformation where familyname=apellidos
and name=nombre;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrar miembro("Manuel", "Moreno");


/* THESIS MEMBER*/

BEGIN;

delimiter //

drop procedure if exists borrarMiembroThesisTitulo//

create procedure borrarMiembroThesisTitulo (IN titulo
VARCHAR(255))

begin

delete from Thesis_Member where title=titulo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo
```

```
#call borrarMiembroThesisTitulo("Ingenieria");

/* THESIS MEMBER*/

BEGIN;

delimiter //

drop procedure if exists borrarMiembroThesisMiembro//

create procedure borrarMiembroThesisMiembro (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32))

begin

delete from Thesis_Member where familyname=apellidos and
name=nombre;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarMiembroThesisMiembro("Manuel", "Moreno");

/* JOURNAL */

BEGIN;

delimiter //

drop procedure if exists borrarrevista//
```

```
create procedure borrarrevista (IN codigo VARCHAR(10))

begin
```

```
delete from Journal where issn=codigo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarrevista("N1");


/* ARTICLE */

BEGIN;

delimiter //

drop procedure if exists borrararticulo//

create procedure borrararticulo (IN codigo INTEGER)

begin

delete from Article where code=codigo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrararticulo(1);
```

```
/* BOOKS */

BEGIN;
```

```
delimiter //

drop procedure if exists borrarlibro//

create procedure borrarlibro (IN codigo INTEGER)
begin
delete from Book where code=codigo;
end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarlibro(1);

/* ACTAS */

BEGIN;

delimiter //

drop procedure if exists borraracta//

create procedure borraracta (IN codigo INTEGER)
begin
delete from Acta where code=codigo;
end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borraracta(1);
```

```
/* MEETING */  
  
BEGIN;  
  
delimiter //  
  
drop procedure if exists borrarmeeting//  
  
create procedure borrarmeeting (IN codigo INTEGER)  
begin  
delete from Meeting where code=codigo;  
end;  
  
//  
  
delimiter ;  
  
COMMIT;  
  
#Prueba del metodo  
#call borrarmeeting(1);  
  
/* ISI */  
  
BEGIN;  
  
delimiter //  
  
drop procedure if exists borrarisi//  
  
create procedure borrarisi (IN anyo INT(4), IN codigo  
VARCHAR(10))  
begin
```

```
delete from Isi where year=anyo and issn=codigo;  
  
end;
```

```
//  
  
delimiter ;  
  
COMMIT;  
  
#Prueba del metodo  
#call borrarisi(2000, "N1");  
  
/* CHAPTER */  
  
BEGIN;  
  
delimiter //  
  
drop procedure if exists borrarcapitulo//  
  
create procedure borrarcapitulo (IN codigo INTEGER)  
begin  
delete from Chapter where code=codigo;  
end;  
  
//  
  
delimiter ;  
  
COMMIT;  
  
#Prueba del metodo  
#call borrarcapitulo(1);  
  
/* BOOKOFCHAPTERS */  
  
BEGIN;
```

```
delimiter //  
  
drop procedure if exists borrarlibrocapitulos//
```

```
create procedure borrarlibrocapitulos (IN codigo INTEGER)
begin
delete from BookOfChapters where code=codigo;
end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarlibrocapitulos(1);

/* DEGREE */

BEGIN;

delimiter //

drop procedure if exists borrardegree//

create procedure borrardegree (IN codigo INT, IN ingeniero
VARCHAR(255), IN nombre VARCHAR(32), IN apellidos VARCHAR(64))
begin
delete from Degree where codeDegree=codigo and
engineering=ingeniero and name=nombre and familyName=apellidos;
end;

//

delimiter ;

COMMIT;
```

```
#Prueba del metodo
```



```
#call borrardegree(1,"Maria","Manuel","Moreno");

/* PROFESSIONALSTATUS */

BEGIN;

delimiter //

drop procedure if exists borrarstatus//

create procedure borrarstatus (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64))

begin

delete from ProfessionalStatus where name=nombre and
familyName=apellidos;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarstatus("Manuel","Moreno");

/* ACTIVITY */

BEGIN;

delimiter //

drop procedure if exists borraractividad//

create procedure borraractividad (IN nombre VARCHAR(32), IN
```

```
    apellidos VARCHAR(64), IN  codigo INT,  IN  nomActividad
VARCHAR(255))

    begin

        delete    from    Activity    where    code=    codigo    and
nameAct=nomActividad and name=nombre and familyName=apellidos;

    end;

    //

    delimiter ;

    COMMIT;

    #Prueba del metodo

    #call    borraractividad(1,"realizar    Base    de    Datos",
"Manuel", "Moreno");

    /* LANGUAGE */

    BEGIN;

    delimiter //

    drop procedure if exists borrarlanguage//

    create procedure borrarlanguage (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64), IN idioma VARCHAR(255))

    begin

        delete from Language where language=idioma and name=nombre
and familyName=apellidos;

    end;

    //

    delimiter ;

    COMMIT;
```

```
#Prueba del metodo

#call borrarlanguage("Frances","Manuel","Moreno");


/* FOREIGNINSTITUTION */

BEGIN;

delimiter //

drop procedure if exists borrarinstitution//


create procedure borrarinstitution (IN nombre VARCHAR(32),
IN apellidos VARCHAR(64), IN nombreCentro VARCHAR(255), IN
codigo INT)

begin

delete from ForeignInstitution where nameFI=nombreCentro
and code=codigo and name=nombre and familyName=apellidos;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarinstitution("College",1,"Manuel","Moreno");


/* THESIS */

BEGIN;

delimiter //

drop procedure if exists borrarthesis//


create procedure borrarthesis (IN nombre VARCHAR(32), IN
apellidos VARCHAR(64), IN titulo VARCHAR(255))

begin
```

```
delete from Thesis where title=titulo and name=nombre and
familyName=apellidos;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarthesis("ingenieria", "Manuel", "Moreno");


/* FINANCE PROJECT */

BEGIN;

delimiter //

drop procedure if exists borrarProyecto//

create procedure borrarProyecto (IN titulo VARCHAR(255))
begin
delete from FinanceProject where title=titulo;
delete from FinanceProject_Member where title=titulo;
end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarProyecto("Titulo del proyecto");


/* FINANCEPROJECT MEMBER*/

BEGIN;
```

```
delimiter //
```

```
drop procedure if exists borrarMiembroProyectoTitulo//
```

```
create procedure borrarMiembroProyectoTitulo (IN titulo
VARCHAR(255))
begin
delete from FinanceProject_Member where title=titulo;
end;
//
delimiter ;
COMMIT;
#Prueba del metodo
#call borrarMiembroProyectoTitulo("Titulo del proyecto");
```

```
/* FINANCEPROJECT MEMBER*/
BEGIN;
```

```
delimiter //
```

```
drop procedure if exists borrarMiembroProyectoMiembro//
```

```
create procedure borrarMiembroProyectoMiembro (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32))
begin
delete          from          FinanceProject_Member          where
familyname=apellidos and name=nombre;
end;
//
delimiter ;
COMMIT;
#Prueba del metodo
```

```
#call    borrarMiembroProyectoMiembro("Catalá    Carbonero",
"María");

/* RESEARCH CONTRACT */

BEGIN;

delimiter //

drop procedure if exists borrarContrato//

create procedure borrarContrato (IN titulo VARCHAR(255))
begin
delete from ResearchContract where title=titulo;
delete from ResearchContract_Member where title=titulo;
end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarContrato("Titulo del contrato");

/* RESEARCHCONTRACT MEMBER*/

BEGIN;

delimiter //

drop procedure if exists borrarMiembroContratoTitulo//

create procedure borrarMiembroContratoTitulo (IN titulo
VARCHAR(255))
begin
delete from ResearchContract_Member where title=titulo;
```

```
end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarMiembroContratoTitulo("Titulo del contrato");


/* RESEARCHCONTRACT MEMBER*/

BEGIN;

delimiter //

drop procedure if exists borrarMiembroContratoMiembro//

create procedure borrarMiembroContratoMiembro (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32))

begin

delete          from          ResearchContract_Member          where
familyname=apellidos and name=nombre;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarMiembroContratoMiembro("Muriel Zafra", "Javi");


/* PATENT */

BEGIN;

delimiter //

drop procedure if exists borrarPatente//
```

```
create procedure borrarPatente (IN titulo VARCHAR(255))
begin
delete from Patent where title=titulo;
delete from Patent_Member where title=titulo;
end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarPatente("Titulo de la patente");

/* PATENT MEMBER */

BEGIN;

delimiter //

drop procedure if exists borrarMiembroPatenteTitulo//

create procedure borrarMiembroPatenteTitulo (IN titulo
VARCHAR(255))
begin
delete from Patent_Member where title=titulo;
end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarMiembroPatenteTitulo("Titulo de la patente");
```



```
/* PATENT MEMBER */

BEGIN;

delimiter //

drop procedure if exists borrarMiembroPatenteMiembro//

create procedure borrarMiembroPatenteMiembro (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32))

begin

delete from Patent_Member where familyname=apellidos and
name=nombre;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarMiembroPatenteMiembro("Muriel Zafra", "Javi");

/* SPECIALIZATION */

BEGIN;

delimiter //

drop procedure if exists borrarEspecializacion//

create procedure borrarEspecializacion (IN
codigoEspecializacion INT)

begin

delete from Specialization where
codeSpecialization=codigoEspecializacion;

end;

//
```

```
delimiter ;

COMMIT;

#Prueba del metodo

#call borrarEspecializacion(4);


/* POSITIONTYPE */

BEGIN;

delimiter //

drop procedure if exists borrarTipoCargo//

create procedure borrarTipoCargo (IN nombreTipoCargo
VARCHAR(64))

begin

delete from PositionType where namePT=nombreTipoCargo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarTipoCargo("Becario/a");


/* POSITIONTYPE */

BEGIN;

delimiter //

drop procedure if exists borrarOtrosMeritosMiembro//

create procedure borrarOtrosMeritosMiembro (IN apellidos
VARCHAR(64), nombre VARCHAR(32))
```

```
begin
    delete from OtherMerits where familyname=apellidos and
name=nombre;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call borrarOtrosMeritosMiembro("Catalá Carbonero",
"María");

/* ARTICLEAUTHOR */

BEGIN;

delimiter //

drop procedure if exists desasociararticuloautor//

create procedure desasociararticuloautor (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN codigoarticulo INTEGER )

begin

delete from ArticleAuthor where familyname=apellidos and
name=nombre and codearticle=codigoarticulo;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call desasociararticuloautor("Martin", "David", 1);
```

```
/* ACTASAUTHOR */

BEGIN;

delimiter //

drop procedure if exists desasociaractasautor//

create procedure desasociaractasautor (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN codigoacta INTEGER )

begin

delete from ActasAuthor where familyname=apellidos and
name=nombre and codeactas=codigoacta;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call desasociaractasautor("Martin", "David", 1);

/* CHAPTERAUTHOR */

BEGIN;

delimiter //

drop procedure if exists desasociarcapituloautor//

create procedure desasociarcapituloautor (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN codigocapitulo INTEGER )

begin

delete from ChapterAuthor where familyname=apellidos and
name=nombre and codechapter=codigocapitulo;

end;

//
```

```
delimiter ;

COMMIT;

#Prueba del metodo

#call desasociarcapituloautor("Martin", "David", 1);

/* BOOKAUTHOR */

BEGIN;

delimiter //

drop procedure if exists desasociarlibroautor//

create procedure desasociarlibroautor (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN codigolibro INTEGER )
begin
delete from BookAuthor where familyname=apellidos and
name=nombre and codebook=codigolibro;

end;

//

delimiter ;

COMMIT;

#Prueba del metodo

#call desasociarlibroautor("Martin", "David", 1);

/* PATENTMEMBER */

BEGIN;

delimiter //

drop procedure if exists desasociarPatenteMiembro//
```

```
create procedure desasociarPatenteMiembro (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN titulo VARCHAR(255) )

begin

delete from Patent_Member where familyname=apellidos and
name=nombre and title=titulo;

end;

//

delimiter ;

COMMIT;

/* RESEARCHCONTACTMEMBER */

BEGIN;

delimiter //

drop procedure if exists desasociarMiembroContrato//

create procedure desasociarMiembroContrato (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN titulo VARCHAR(255) )

begin

delete      from      ResearchContract_Member      where
familyname=apellidos and name=nombre and title=titulo;

end;

//

delimiter ;

COMMIT;

/* FINANCEPROJECTMEMBER */

BEGIN;

delimiter //
```

```
drop procedure if exists desasociarMiembroProyecto//

create procedure desasociarMiembroProyecto (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN titulo VARCHAR(255) )

begin

delete          from          FinanceProject_Member          where
familyname=apellidos and name=nombre and title=titulo;

end;

//

delimiter ;

COMMIT;

/* THESISMEMBER */

BEGIN;

delimiter //

drop procedure if exists desasociarMiembroTesis//

create procedure desasociarMiembroTesis (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN titulo VARCHAR(255) )

begin

delete from Thesis_Member where familyname=apellidos and
name=nombre and title=titulo;

end;

//

delimiter ;

COMMIT;

BEGIN;

delimiter //
```

```
drop procedure if exists desasociarMiembroArticulo//

create procedure desasociarMiembroArticulo (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32), IN titulo VARCHAR(255) )

begin

delete from BookAuthor where familyname=apellidos and
name=nombre and title=titulo;

end;

//

delimiter ;

COMMIT;

/* INFORMACION AUTOR */

BEGIN;

delimiter //

drop procedure if exists informacionautor//

create procedure informacionautor (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32) )

begin

select *

from personalInformation

where familyname=apellidos and name=nombre;

end;

//

delimiter ;

COMMIT;
```



```
#call informacionautor("Martin", "David");

/* INFORMACION MIEMBRO */

BEGIN;

delimiter //

drop procedure if exists informacionmiembro//

create procedure informacionmiembro (IN apellidos
VARCHAR(64), IN nombre VARCHAR(32) )

begin

select                                personalInformation.treatment,
personalInformation.category, personalInformation.university,
personalInformation.departament,
personalInformation.address, personalInformation.city,
personalInformation.state,    personalInformation.country,
personalInformation.zip, personalInformation.email,
member.email_2, personalInformation.phone,
member.phone_2,    personalInformation.fax,    member.web,
member.fromdate,

member.todate,member.image, member.dni, member.birthDate,
member.sex, member.numOfficial, member.postalCode,
member.codeSpecialization

from member, personalInformation

where member.familyname=apellidos and member.name=nombre
and    personalInformation.familyname=apellidos    and
personalInformation.name=nombre;

end;

//
```

```
delimiter ;

COMMIT;

#call informacionmiembro("Martin", "David");

/* AUTHOR NUMERO DE PUBLICACIONES*/

/*Devuelve el numero de publicaciones*/

BEGIN;

delimiter //

CREATE FUNCTION autorpublicaciones (apellidos VARCHAR(32),
nombre VARCHAR(32)) RETURNS INT

NO SQL

READS SQL DATA

BEGIN

    DECLARE num1 INT;

    DECLARE num2 INT;

    DECLARE num3 INT;

    DECLARE num4 INT;

    SET num1 = (Select count(*) from articleauthor where
familynome=apellidos and name=nombre);

    SET num2 = (Select count(*) from actasauthor where
familynome=apellidos and name=nombre);

    SET num3 = (Select count(*) from bookauthor where
familynome=apellidos and name=nombre);
```

```
        SET num4 = (Select count(*) from chapterauthor where  
familyname=apellidos and name=nombre);  
  
        RETURN (num1+num2+num3+num4);  
  
    END  
  
    //  
  
    delimiter ;  
  
    COMMIT;
```

## 1.7. PRUEBAS

En la realización del esquema de la base de datos se han realizado pruebas unitarias para la creación de cada una de las tablas, así como se ha establecido la creación de las mismas en orden acorde con las referencias establecidas entre las mismas.

De este modo se ha verificado la correcta creación de la misma, tanto con la existencia de la base de datos sobre el sistema como sin ella, ya que previo a la creación de la misma se procede a las comprobaciones que subsanan cualquier tipo de error, obteniéndose pruebas satisfactorias para las sentencias de creación de la base de datos GIGI en MySQL 5.0.

Por otra parte también se han realizado pruebas unitarias sobre cada uno de los procedimientos. Las pruebas realizadas han consistido en la correcta definición y borrado de los mismos del sistema, como para su utilización y visualización posterior de su correcto funcionamiento.

La verificación de la creación del esquema y de los procedimientos así como la carga de tablas maestras y comprobación de funcionalidad, se ha realizado mediante la herramienta libre MySQL Administrador y MySQL Query Browser.

Los ejemplos utilizados en la verificación y validación de la base de datos y su funcionalidad se encuentran incluidos en el script correspondiente a

la carga de información, así como también se incluyen ciertas pruebas de modificación y borrado de información apareciendo como comentarios en la definición de los procedimientos de manipulación de la base de datos (apartado 6.5 de este documento).

## 1.8. FICHEROS DE CONFIGURACIÓN

Se han definido básicamente cuatro tipos de ficheros de configuración, cada uno con una misión diferente:

**Fichero de creación del esquema de la base de datos.** En el fichero “*gigi.txt*” se encuentra el código indicado en el apartado 6.1 de este documento, para la creación de la base de datos.

**Ficheros de carga.** Se han incluido 4 ficheros de carga de datos en la base de datos, para la creación por parte del administrador de la misma de las tablas maestras de las que hará uso esta base de datos GIGI. Los ficheros con su correspondiente utilidad se describe a continuación:

Fichero “*gigi-Country.txt*” contiene la carga de los países que formarán el contenido de la tabla COUNTRY.

Fichero “*gigi-Status.txt*” contiene la carga de los estados posibles en los que se encuentra un documento que formarán el contenido de la tabla STATUS.

Fichero “*gigi-Treatment.txt*” contiene la carga de diferentes tratamientos académicos y profesionales que formarán el contenido de la tabla TREATMENT.

Fichero “*gigi-Type.txt*” contiene la carga de los tipos de congresos posibles en los que formarán el contenido de la tabla TYPE.

**Fichero de creación y prueba de procedimientos de manipulación.** El fichero “*gigi-procedures.txt*” contiene un script para la carga de los procedimientos, así como los comentarios de las pruebas realizadas sobre los mismos.

**Fichero de carga de información en la base de datos.** El fichero “*gigi-CargaInformacion.txt*” se trata un script cuyo cometido es la carga de información en la base de datos, utilizando los procedimientos previamente cargados mediante el script correspondiente, de tal manera que es útil tanto para validar la integridad del esquema propuesto así como para comprobar la exactitud de los procedimientos de inserción codificados.

La forma de ejecutar dichos ficheros de configuración y carga viene indicada como comentario en la primera línea del contenido de los mismos.

## **2. Interfaz General**

# 2.1

## DEFINICIÓN DEL PROBLEMA

---

En este apartado se tratarán de identificar las necesidades que pretendemos cubrir así como de definir el principal objetivo a alcanzar con el desarrollo de la práctica, mostrando las alternativas y posibilidades mediante las cuales será posible conseguir el resultado que deseamos.

La enumeración de las necesidades y la definición del problema podrán llevarse a cabo desde dos puntos de vista diferentes. Por un lado, se intentará identificar el problema desde la perspectiva del cliente (problema real) y, por otro lado, se intentará presentar la forma de dar solución a dichas necesidades (problema técnico).

### 2.1.1 DEFINICIÓN DEL PROBLEMA REAL

Desde el punto de vista real, el problema que se plantea es simple, ya que es la realización de una interfaz general para el sistema. Se entiende por interfaz general aquella con la que el usuario puede interaccionar y mediante la cual puede acceder a todas las funciones y opciones que proporcione el sistema.

Esta interfaz debe ser fácil de usar e intuitiva, ya que el objetivo primordial buscado como se ha comentado anteriormente es que los miembros del grupo que utilicen el sistema no tengan dificultad a la hora de utilizar toda la funcionalidad ofrecida por el sistema, ya que estos no tienen porque estar relacionados con el mundo de la informática.



Por tanto, se deberá desarrollar un módulo que sea capaz de realizar estas funcionalidades, es decir, permitir al usuario acceder de forma intuitiva a toda la funcionalidad del sistema.

## **2.1.2 DEFINICIÓN DEL PROBLEMA TÉCNICO**

El desarrollo de la solución consistirá en la construcción de aplicación visual, ya que lo que buscamos es facilidad de uso. Esta aplicación se compondrá de una serie de ventanas que nos permitirán la interacción con las distintas partes de nuestro sistema. Esta parte del proyecto es la base, ya que a ella se enlazarán los distintos tipos de formularios para la manipulación de la información que se dispone en el sistema. Por este motivo, la interfaz general debe presentar facilidad de uso, permitiendo que usuarios no relacionados con la informática no encuentren ningún problema en su utilización.

Una vez identificado el problema real a resolver, será necesario realizar la definición del mismo desde un punto de vista técnico. Para ello, se utilizará la técnica de Ingeniería conocida con el nombre de PDS (Product Design Specification) [Pressman, 2002]. Esta metodología propone realizar un análisis de los principales condicionantes técnicos del problema a resolver, mediante la respuesta a una serie de cuestiones básicas, las cuales se exponen a lo largo de los siguientes subapartados.

### **2.1.2.1 Funcionamiento**

La interfaz general será el módulo encargado de unir las demás partes del sistema, mostrando en cada momento la parte de información que el usuario haya solicitado.

Se pone a disposición del usuario una serie de menús y vías de acceso a los distintos elementos del sistema, de forma que el usuario pueda elegir fácilmente que información desea visualizar o manipular en un determinado momento.

Esta parte del sistema sólo es la encargada de poner a disposición del usuario dichos menús y facilidades, así como de llamar a los formularios, creados por otros grupos dentro del proyecto.

Un detalle a tener en cuenta es que es ésta parte del proyecto la encargada de facilitar la información al usuario, pero sólo cuándo éste tenga acceso a la misma, por lo que será implementado aquí la conexión a la base de datos donde se encuentra almacenada la información del sistema, no mostrando funcionalidad el sistema, hasta que el usuario se haya autenticado dentro del sistema.

Debido a que es muy probable que el sistema se use en ordenadores de uso exclusivo por parte de integrantes del grupo de investigación, se ha implementado la funcionalidad de permitir a un usuario almacenar su usuario e incluso su contraseña de forma que no tenga que introducirla cada vez que desee utilizar el sistema, ya que suponemos que su Sistema Operativo será el encargado de restringir el acceso al sistema.

### **2.1.2.2 Entorno**

La interfaz general estará formada por una única ventana, de tamaño variable, la cual puede ser movida como cualquier ventana estándar del sistema operativo. La interfaz en cuestión se encuentra dividida en varias zonas: zona de menús, barras de herramientas, árbol de la base de datos, panel de inserción de información, panel de visualización de información y panel de ayuda.

Dichas partes de la interfaz serán actualizadas de forma dinámica respondiendo a las solicitudes por parte del usuario.

### **2.1.2.3 Vida esperada**

La estimación de la vida de un software es un parámetro muy importante pues puede que no merezca la pena realizar un gran esfuerzo en el diseño de un software si después va a tener una vida muy corta. Sin embargo, este

parámetro es difícil de calcular pues depende de varios factores, como pueden ser su mantenimiento o nuevas necesidades por parte de los usuarios.

Lo primero que debemos tener en cuenta es que este sistema se sustenta sobre una base de datos dirigida a un grupo de investigación, por lo cual, si el diseño de dicha base de datos es correcto, su estructura no debe ser modificada en un largo periodo de tiempo. Como es lógico esto no supone que no se modifique la información contenida en dicha base de datos, ya que la idea de la aplicación es la manipulación de dicha información.

Otra idea que debemos tener en cuenta es que este módulo, la interfaz general, no tiene sentido como aplicación por sí misma, sino que debe ser integrada junto con los distintos módulos que forman la aplicación, por lo que si se invierte tanto esfuerzo en el desarrollo es de suponer que su tiempo de vida será elevado.

#### **2.1.2.4 Ciclo de mantenimiento**

El software está preparado para cualquier modificación que se deba realizar pues está diseñado de tal forma que es fácil añadir tanto una nueva opción como una nueva funcionalidad. Es más, en esta segunda fase se parte del modelo de la fase inicial, perfilando algunos detalles y añadiendo nueva funcionalidad al sistema.

#### **2.1.2.5 Competencia**

En el mundo del desarrollo de sistemas de información existe mucha competencia, aunque es probable que en cuanto al desarrollo de una aplicación tan específica para manipular toda la información requerida por un grupo de investigación la competencia sea bastante reducida.

### **2.1.2.6 Aspecto externo**

El aspecto externo de una aplicación hace referencia no solamente al impacto visual que recibe el usuario al realizar la ejecución de la misma, sino también a la propia presentación física del sistema.

La interfaz general deberá proporcionar al usuario una interfaz de comunicación lo más sencilla e intuitiva posible, de manera que se facilite el acceso rápido a las diferentes funcionalidades ofrecidas por la aplicación, sin excesivas dificultades.

### **2.1.2.7 Estandarización**

El diseño de la aplicación se ha realizado de forma tal que la aplicación usuario pueda ejecutarse en cualquier Sistema Operativo, ya sea de tipo Windows, Linux o de cualquier otro. Además, se ha procurado utilizar los elementos de la forma más estándar posible, ya sea el paradigma de programación, los botones, los menús, etc.

### **2.1.2.8 Calidad y fiabilidad**

La calidad y la fiabilidad son campos que cada vez van tomando mayor importancia en la sociedad. Por ello, ambos han sido muy tenidos en cuenta a la hora de diseñar el módulo para la aplicación global.

De esta manera, este módulo ha sido desarrollado identificando los puntos o elementos de riesgo o de mayor probabilidad de fallo e intentando minimizar esta probabilidad.

Ha sido necesario llevar a cabo gran cantidad de pruebas ya que esta es la parte del proyecto que mayor contacto con el usuario tiene, y como es bien sabido, es en la interacción con el usuario donde aparecen mayor número de errores en el software.

### 2.1.2.9 Programa de tareas

A continuación se va a desarrollar el programa detallado de la realización del proyecto a lo largo del tiempo: Se conoce como estilo *Normal* aquel estilo cuyo uso es el predominante, esto es, se trata del estilo principal que se sigue para documentar los distintos aspectos de un proyecto o trabajo. En nuestro caso se ha definido con las siguientes características:

- **Fase de estudio.** En esta primera fase se estudia como se ha realizado la interfaz general en la primera fase del proyecto. Se estudiarán también las herramientas necesarias para el desarrollo de la nueva funcionalidad que se incorporará en la misma.
- **Análisis y diseño** de la interfaz con las mejoras pertinentes. En esta fase se realizará el diseño y la implementación del módulo en cuestión. Para la implementación se utilizará el lenguaje Java, utilizando fundamentalmente la librería swing para la implementación gráfica.
- **Realización de pruebas.** Para asegurar el correcto comportamiento del módulo así como su integración dentro de la aplicación global se deberán realizar una serie de pruebas y un análisis exhaustivo de los resultados.
- **Documentación de la memoria.** En esta fase se ha elaborado la documentación técnica, es decir, la memoria. La elaboración de la memoria se ha realizado conjuntamente con el diseño del módulo de la aplicación para dotar al documento de una mayor coherencia.

### 2.1.2.10 Pruebas

Las pruebas que se llevan a cabo en esta etapa serán tanto de caja negro o pruebas funcionales como de caja blanca. Las pruebas de caja negra son de especial interés en este módulo del proyecto ya que tratan fundamentalmente las entradas y salidas del sistema, y al tratarse de la interfaz

general de la aplicación es normal que este tipo de pruebas tengan el mayor peso posible.

Se realizarán también pruebas de caja blanca en aquellas partes donde haya un funcionamiento complicado, respecto a implementación propiamente dicha, es decir, que tengan una algoritmia compleja.

Por último, se realizará la prueba de aceptación, en donde intervendrán una o varias personas distintas al autor, y la prueba del sistema, para comprobar que se cumplen todos los requisitos y que el propio sistema responde a situaciones límite de sobrecarga.

El módulo desarrollado para la aplicación se dará por válido cuando se hayan corregido todos los errores encontrados en las pruebas anteriores y se cumplan con los objetivos para los que se diseñó.

#### **2.1.2.11 Seguridad**

El módulo se ha diseñado de forma que el usuario no pueda modificar el diseño de la misma. Como ya se ha comentado, el usuario podrá interactuar con la información recogida en la base de datos que sustenta el sistema, pero nunca podrá acceder a la base de datos como tal.

Además antes de poder realizar cualquier operación que afecte a la base de datos es necesario que el usuario se identifique introduciendo su nombre de usuario y su contraseña. Estos datos se utilizarán para conectarse a la base de datos, sirviendo esta conexión a su vez de autenticación del usuario.

# 2.2

## OBJETIVOS

---

El objeto principal de este módulo es el diseño e implementación de la interfaz general de la aplicación, permitiendo al usuario acceder a toda la funcionalidad de la aplicación.

Podemos puntualizar los siguientes objetivos principales:

- Se debe dar toda la funcionalidad necesaria para que el sistema cree una conexión con la base de datos del sistema.
- Se debe permitir acceder a cada una de las entidades que forman el modelo entidad relación, de forma que se puedan insertar nuevos datos, mantener la información existente, o simplemente visualizar dicha información.
- Se debe realizar la interfaz de forma que personas no expertas en informática puedan utilizarla de forma fácil, intuitiva y cómoda.
- Se debe llevar a cabo la implementación de forma que pueda ser entendida fácilmente por los demás miembros del proyecto, para lo cual se utilizarán comentarios cada vez que sea necesario.
- Se debe realizar una documentación de forma correcta para que en las siguientes fases del proyecto los grupos que lo necesiten puedan consultarla de forma cómoda, y ésta les sea útil.

- Al tratarse de una interfaz, se tendrá especial cuidado a la hora de entradas al sistema, controlando los posibles errores que pueda cometer el usuario, avisándole siempre que sea posible.



# 2.3

## RESTRICCIONES TÉCNICAS Y DE GESTIÓN

---

Las limitaciones a las que se ha visto sometido el desarrollo de esta aplicación son:

- El sistema debe presentar una interfaz de usuario homogénea e intuitiva. Desde la interfaz será accesible toda la funcionalidad del sistema.
- El sistema deberá ser capaz de manejar toda la información correspondiente al grupo de investigación.
- Debido a la naturaleza multiplataforma del sistema, y a la modularidad exigida, el desarrollo del sistema debe realizarse en el lenguaje de programación Java. Este lenguaje proporciona un paradigma de programación orientado a objetos. Tiene un API que proporciona una fuente de recursos lo suficientemente rica y en continua expansión para cubrir todas las necesidades de la fase de codificación del sistema.
- Para el desarrollo del entorno gráfico se utilizará la librería SWING de JAVA. Esto se ha decidido entre los grupos de desarrollo de las distintas partes de la aplicación, por tratarse de una librería estándar de JAVA y por lo tanto la cantidad de documentación existente de la misma es muy útil a la hora de desarrollar.

- La utilización del lenguaje de programación Java para la codificación hará necesaria la aplicación de una perspectiva orientada a objetos tanto en el proceso de especificación de requisitos como en el proceso de diseño.
- Se utilizará la tecnología que MySql 5.0 proporciona para el desarrollo de la aplicación. A su vez, para lograr la interacción entre el sistema y la base de datos se utilizará JDBC, como puente de conexión entre ambos.

# 2.4

## RECURSOS

---

Los recursos utilizados para el desarrollo de este proyecto podemos dividirlos en recursos humanos y recursos materiales, ya sean hardware o software.

### 2.4.1 RECURSOS HUMANOS

Los recursos humanos para la realización de la interfaz general consisten en los autores del proyecto y los gestores del mismo.

Los autores del proyecto son los integrantes de los grupos 12 y 13 de desarrollo, ingenieros técnicos y estudiantes de 2º curso de Ingeniería Informática:

- Francisco Javier Muriel Zafra.
- María Catalá Carbonero.
- Antonio Cid García.
- Jorge Fernández Marín.

Por otro lado, los gestores del proyecto, cuyas pautas deben seguir los autores mencionados anteriormente son:

- Dña. Irene Luque Ruiz, Titular de Universidad.

- Miguel Ángel Gómez Nieto, Catedrático E.U.

## 2.4.2 RECURSOS HARDWARE

Se han utilizado tres máquinas principalmente para el desarrollo del proyecto, las cuáles se describen a continuación:

- Intel Pentium IV 2,4 GHz, 1 GB RAM.
- AMD Athlon 64 Bits 3500+, 1 GB RAM.
- Intel Centrino 1.6 GHz, 512 MB RAM.

## 2.4.3 RECURSOS SOFTWARE

Entre los recursos software se encuentran cada uno de los programas, sistemas operativos, herramientas de programación y de documentación que han sido utilizadas a lo largo del desarrollo y documentación del prototipo.

- **Sistema Operativo**
  - Windows XP Professional.
  - Distribución de Linux basada en Debian: Ubuntu.
- **Herramientas de programación y librerías**
  - Máquina virtual de Java JDK 6.
  - Entorno de programación de dominio público Eclipse.
  - Librería de Java JRE 5.0.
  - Servidor de Base de Datos *MySQL 5.0*, ejecutado de forma local y *MySQLClient*.
- **Herramientas de documentación**
  - Procesador de textos Microsoft Office Word 2003.
  - Microsoft Office Visio 2003.
  - Adobe Acrobat Professional 6.0.

# 2.5

## ESPECIFICACIÓN DE REQUISITOS

---

En los apartados anteriores se ha llevado a cabo una descripción detallada del problema, y cómo se plantea solucionarlo mediante este proyecto. En la sección de especificación de requisitos nos centraremos en analizar los requisitos del sistema para obtener una idea clara de qué debe hacer el sistema, sin tener en cuenta por el momento el cómo debe hacerlo.

Para la especificación de requisitos se utilizará UML ya que éste se basa en una metodología orientada a objetos, ideal para el proyecto [Booch et al., 1999].

Esta sección dentro de la documentación se encuentre dividida en cuatro fases, que se enumeran a continuación y se expondrán en los siguientes apartados:

- Identificación de requisitos.
- Descripción de la información.
- Descripción funcional.
- Descripción de la interfaz.

## 2.5.1 IDENTIFICACIÓN DE REQUISITOS

En esta fase como se ha comentado anteriormente se pretende desarrollar la interfaz general de la aplicación, la cual debe cumplir una serie de requisitos que se enumeran a continuación:

1. Dar soporte para los formularios de mantenimiento y de visualización de la información de la base de datos de nuestro sistema.
2. Permitir al usuario conectar al sistema, lo cual supondrá la conexión a la base de datos creada en MySQL.
3. Desconectar el sistema, lo cual supondrá desconectar con la base de datos de MySQL.
4. Deshabilitar en cada momento las funcionalidades del sistema que no puedan ser utilizadas debido al estado actual del sistema, es decir, una vez conectado, el usuario no podrá volver a llevar a cabo la acción de conectar, por ejemplo.
5. Ofrecer al usuario la posibilidad de almacenar sus datos de conexión al sistema, de forma que éste no tenga que introducirlos cada vez que lo inicie. Esto se hace ya que suponemos que la aplicación irá destinada a los integrantes de un grupo de investigación, y lo más común es que el ordenador que usen sea personal.
6. Cuando el sistema tenga los datos de conexión almacenados, brindar al usuario la posibilidad de cambiar el usuario, ya que puede ser que se necesita entrar con otro usuario por algún motivo.
7. Mostrar en todo momento la ayuda, referida al estado actual del sistema.
8. Permitir al usuario cambiar entre las distintas partes del sistema (artículos, autores, etc.) de forma rápida y sencilla, a través del árbol de navegación, del menú o de la barra de herramientas.

## 2.5.2 DESCRIPCIÓN DE LA INFORMACIÓN

En este apartado se presta especial atención a la información utilizada por el módulo dentro de la aplicación y como ésta debe ser manipulada por el sistema.

Como se ha comentado a lo largo del documento, este módulo es básicamente la interfaz general de la aplicación, por lo que la información manejada por éste es nula, ya que la funcionalidad de este módulo es el de llamar a las diferentes partes del sistema que serán las encargadas de manipular la información de la base de datos del sistema.

La única información que maneja la interfaz de forma directa es el usuario y la contraseña, campos que serán utilizados para acceder a la base de datos, ya que si el usuario del sistema no se autentifica propiamente la aplicación no podrá acceder a la base de datos. La aplicación también maneja la cadena de conexión que será necesaria para poder conectar a la base de datos.

## 2.5.3 DESCRIPCIÓN FUNCIONAL

En este apartado se va a realizar la descripción funcional del sistema que se pretende desarrollar. Dicha descripción se realizará mediante el uso de la metodología UML como ya se comentó anteriormente [Booch et al., 1999].

Dentro de la metodología UML utilizaremos los diagramas que mejor describe funcionalmente el sistema, dichos diagramas son los Diagramas de casos de uso, el cual representa la interacción de entidades externas con el sistema en estudio.

Para la realización correcta de los casos de uso seguiremos unos pasos determinados, que son:

- Determinación de los actores del sistema.
- Especificación de los casos de uso de trazo grueso.

- Priorización de los casos de uso.
- Matriz requisitos / casos de uso.
- Especificación de los casos de uso de trazo fino.

### **2.5.3.1 Determinación de los actores del sistema**

Cabe destacar que en nuestro sistema aparece un único actor representando el rol del usuario. Este actor puede representar tanto al usuario de la aplicación que será un integrante del grupo de investigación interactuando con la interfaz general del sistema, así como un módulo de la aplicación realizado por otros integrantes del proyecto que interactúa con nuestra interfaz accediendo a funcionalidad del sistema.

### **2.5.3.2 Especificación de los casos de uso de trazo grueso**

Los casos de uso de trazo grueso identificados en nuestro sistema son:

1. Conectar a la base de datos.
2. Desconectar de la base de datos.
3. Cambiar de usuario.
4. Visualizar de la información.
5. Mantener la información.
6. Ayuda.

A continuación se describen los casos de uso de trazo grueso identificados de forma más detallada.

1. **Conectar a la base de datos:** este caso muestra como el usuario como actor del sistema utiliza el menú de la aplicación para realizar la conexión a la base de datos. Para ello tendrá que introducir el usuario y la contraseña, que serán utilizados para conectar con MySQL.



Un requisito impuesto simplemente por lógica, será que el sistema no podrá poner a disposición del usuario el acceso a la base de datos, hasta que éste se haya autenticado de forma correcta contra la base de datos de MySQL, por tanto se debe restringir el uso de estos comandos hasta la autenticación.

Cuando se realiza la conexión se debe permitir poder elegir almacenar la información del usuario para que en futuras conexiones no sea necesario introducir dicha información. Por lo tanto, si hay almacenado un usuario cuando se realice el proceso de conexión no se le pedirán esos datos al usuario y la conexión se realizará de forma automática.

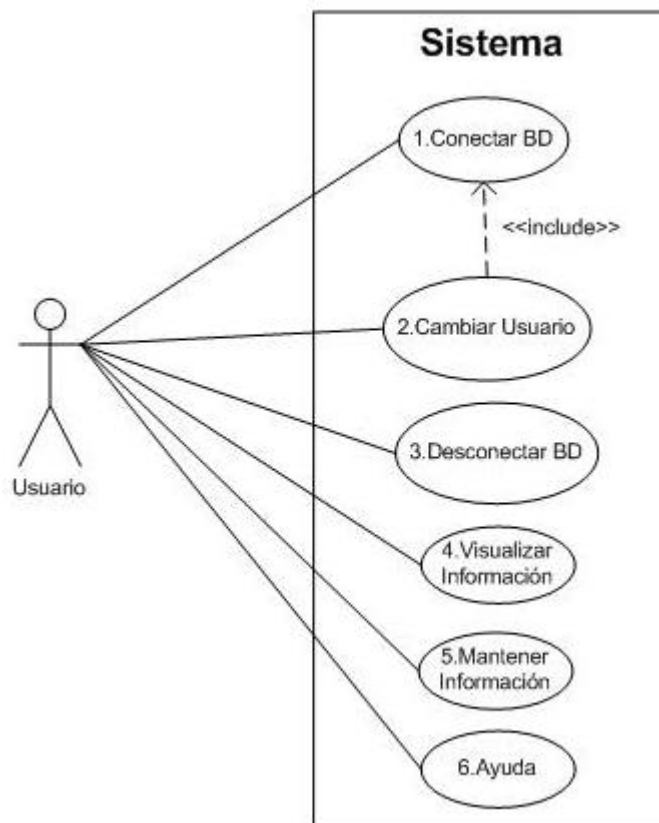
2. **Cambiar de usuario:** como hemos dicho se ha posibilitado al usuario del sistema la opción de almacenar sus datos de forma constante para que no tenga que introducir continuamente sus datos para conectarse a la base de datos a través del sistema.

Este caso de uso muestra las acciones que lleva a cabo el usuario para cambiar el usuario actual del sistema, cambiando por tanto el usuario de la base de datos.

3. **Desconectar de la base de datos:** este caso muestra las acciones que el usuario lleva a cabo para desconectarse de la base de datos.
4. **Visualizar la información:** este caso muestra como el usuario interactúa con el sistema para obtener la información almacenada en la base de datos a través del módulo del proyecto referente a formularios de visualización.
5. **Mantener la información:** este caso muestra las acciones del usuario para poder acceder a los formularios de entrada de información realizados en otro módulo del proyecto. Estos formularios permitirán al usuario introducir nueva información en la base de datos, modificar información existente y borrar información almacenada no deseada.

6. **Ayuda:** este caso de uso muestra las opciones que tiene que realizar el usuario del sistema para que la ayuda sea mostrada.

En la Figura 2.5.1 se muestra el diagrama de caso de uso de trazo grueso mostrando la interacción del usuario del sistema con éstos.



**Figura 2.5.1:** Diagrama de Casos de uso de trazo grueso

### 2.5.3.3 Priorización de los casos de uso

Observando los casos de uso de trazo grueso que aparecen la priorización es bastante simple, ya que podemos entender que el caso de uso más prioritarios es el caso *Conectar BD*, ya que de él depende que la aplicación ofrezca toda su funcionalidad. De hecho un requisito fundamental de la interfaz general es que no se pueda acceder a las funciones de visualización y mantenimiento de la información sin previamente haberse conectado a la base de datos.

También tienen una gran importancia los casos de uso de visualización y entrada de información, ya que permiten manipular la información almacenada en la base de datos.

Por tanto como secundarios quedan los casos de uso de acceso a la ayuda, el de desconexión y el de cambio de usuario, que no soportan la funcionalidad básica del sistema, si no que son complementos del mismo.

#### **2.5.3.4 Matriz requisitos / casos de uso**

En la Tabla 2.5.1 se muestra la matriz que resalta que requisitos son cubiertos por qué casos de uso, dejando claro que todos los requisitos impuestos han sido cumplido mediante el planteamiento que se ha seguido con los casos de uso detallados.

**Tabla 2.5.1:** Matriz Requisitos / Casos de Uso

	<b>C1</b>	<b>C2</b>	<b>C3</b>	<b>C4</b>	<b>C5</b>	<b>C6</b>
<b>R1</b>				X	X	
<b>R2</b>	X					
<b>R3</b>			X			
<b>R4</b>	X	X	X			
<b>R5</b>	X	X				
<b>R6</b>		X				
<b>R7</b>						X
<b>R8</b>				X	X	

#### **2.5.3.5 Especificación de los casos de uso de trazo fino**

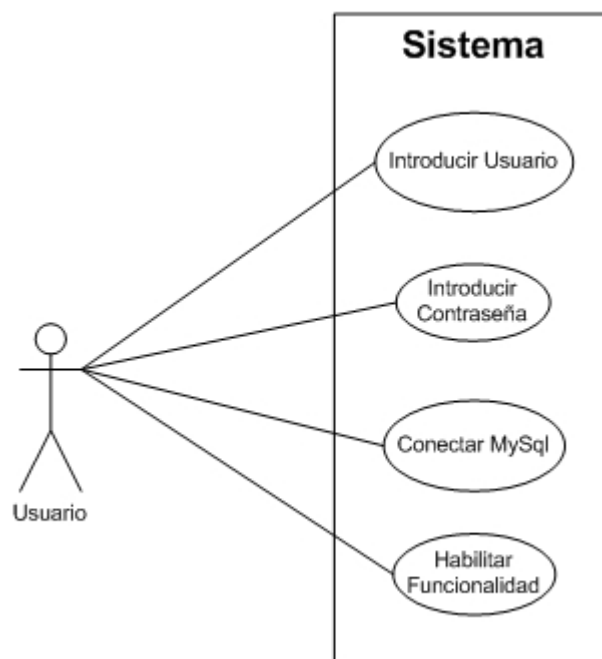
En cuanto a la realización de los casos de uso de trazo fino, podemos decir que no corresponde al desarrollo de este módulo del proyecto nada más que los de conexión a la base de datos, desconexión y cambio de usuario, ya

que tanto el de ayuda, como los de visualización y mantenimiento serán realizados en otras fases o por otros componentes del proyecto.

También cabe destacar que debido a que el objeto de este módulo es el desarrollo de la interfaz general, y que la información que manipulará es muy reducida, el realizar diagramas de secuencia haría más complicado el correcto entendimiento del documento, y no aportaría información debido a su simplicidad.

#### 2.5.3.5.1 Caso de uso Conectar BD

El caso de uso de conectar a la base de datos se muestra en la Figura 2.5.2.



**Figura 2.5.1:** Caso de uso Conectar BD

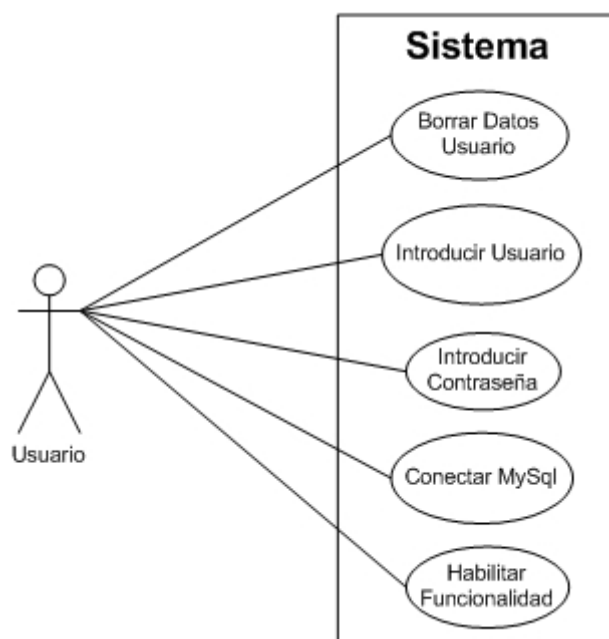
La especificación detallada del caso de uso *Conectar BD* se observa en la Tabla 2.5.2.

**Tabla 2.5.1:** Especificación Caso de uso Conectar BD

<b>Caso de Uso</b>	<b>Conectar BD</b>
<b>Ámbito</b>	Módulo de Interfaz General
<b>Nivel</b>	Usuario
<b>Contexto de Utilización</b>	El usuario desea conectarse a la BD
<b>Actor Principal</b>	Usuario
<b>Escenario de Éxito Principal</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona la opción de conexión, ya sea desde el menú o desde la botonera.</li> <li>2. El sistema pide al usuario los datos de conexión, es decir, usuario y contraseña.</li> <li>3. El sistema conecta con la base de datos MySql.</li> <li>4. El sistema habilita toda la funcionalidad.</li> </ol>
<b>Extensiones</b>	<ol style="list-style-type: none"> <li>2a. El usuario selecciona la opción de recordar los datos, por lo que el sistema guarda en las propiedades los datos para utilizarlos en próximas conexiones.</li> <li>2b. Cuando el usuario conecta se encuentran datos almacenados para la conexión por lo que no se pide que se inserten, si no que se utilizan los almacenados.</li> <li>2c. El usuario cancela la conexión, debiendo el sistema volver al estado en el que estaba anteriormente.</li> </ol>

#### 2.5.3.5.2 Caso de uso Cambiar Usuario

En cuanto al caso de uso de cambio de usuario se muestra en la Figura 2.5.3.



**Figura 2.5.2:** Caso de uso Cambiar Usuario

La especificación detallada del caso de uso *Cambiar Usuario* se observa en la Tabla 2.5.3.

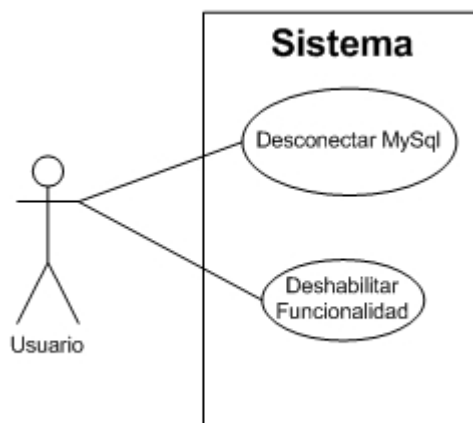
**Tabla 2.5.2:** Especificación Caso de uso Cambiar Usuario

<b>Caso de Uso</b>	<b><i>Cambiar Usuario</i></b>
<b>Ámbito</b>	Módulo Interfaz General
<b>Nivel</b>	Usuario
<b>Contexto de Utilización</b>	El usuario desea cambiar de usuario
<b>Actor Principal</b>	Usuario
<b>Escenario de Éxito Principal</b>	1. El usuario selecciona la opción de cambio de usuario, ya sea desde el menú o desde la botonera.

	<p>2. El sistema borra los datos del usuario anterior.</p> <p>3. El sistema pide al usuario los datos de conexión, es decir, usuario y contraseña.</p> <p>4. El sistema conecta con la base de datos MySQL.</p> <p>5. El sistema habilita toda la funcionalidad.</p>
<b>Extensiones</b>	<p>1a. El usuario sólo tendrá disponible esta opción en el caso de que tenga almacenada la información de conexión en el sistema.</p> <p>3a. El usuario selecciona la opción de recordar los datos, por lo que el sistema guarda en las propiedades los datos para utilizarlos en próximas conexiones.</p> <p>3b. El usuario cancela el proceso de cambio de usuario, teniendo el sistema que volver al estado en que se encontraba anteriormente.</p>

#### 2.5.3.5.3 Caso de uso Desconectar BD

El caso de uso de desconexión de la base de datos se muestra en la Figura 2.5.4.



**Figura 2.5.3:** Caso de uso Desconectar BD

La especificación detallada del caso de uso *Desconectar BD* se observa en la Tabla 2.5.4.

Tabla 2.5.3: Especificación Caso de uso Desconectar BD

<b>Caso de Uso</b>	<b>Desconectar BD</b>
<b>Ámbito</b>	Módulo Interfaz General
<b>Nivel</b>	Usuario
<b>Contexto de Utilización</b>	El usuario desea desconectarse de la BD.
<b>Actor Principal</b>	Usuario
<b>Escenario de Éxito Principal</b>	1. El usuario marca desconectar de la base de datos, ya sea a través del menú o del botón facilitado para tal acción.
<b>Escenario de Éxito Principal</b>	2. El sistema se desconecta de la base de datos MySql.  3. El sistema deshabilita todos la funcionalidad del sistema que no deba ser accedida cuando éste se encuentra desconectado.

## 2.5.4 ESPECIFICACIÓN DE LA INTERFAZ

En este apartado se presenta la descripción de la interfaz a desarrollar. Toda interfaz de un sistema es la unión entre el usuario y el propio sistema, por lo cual esta debe permitir llevar a cabo todo el trabajo que el usuario requiere del sistema y por tanto se deben identificar todas las pantallas necesarias, y la relación que existirá entre ellas. Es decir, desde que pantalla se puede acceder a otra, y que pantallas pueden acceder a una dada, especificando en cada pantalla la funcionalidad ofrecida.

Por norma general toda interfaz debe cumplir una serie de características ergonómicas de forma que facilite la interacción del usuario con el sistema, haciendo que esta sea lo más cómoda, sencilla e intuitiva posible.



Para esto se deben cumplir una serie de objetivos principales en la realización de una interfaz:

- Ubicar toda la funcionalidad del sistema en el mínimo número de pantallas posible, evitando continuos cambios.
- Disminuir en la medida de lo posible la frecuencia de cambio entre los distintos dispositivos de entrada del sistema (ratón, teclado, etc.)
- Los elementos de la interfaz deben acomodarse de manera que su posición en la pantalla facilite la transición entre el pensamiento del usuario y la acción a realizar.

Estos objetivos básicos están englobados en lo que se denomina “reglas de oro” para el diseño de la interfaz [Pressman, 2002].

A continuación iremos detallando cada una de las zonas y ventanas que deben formar parte de la interfaz general, indicando qué funcionalidad deben prestar al usuario cada una de ellas.

#### **2.5.4.1 Ventana principal**

Esta será la pantalla que se cargue al iniciar la aplicación, y tendrá que brindar al usuario toda la funcionalidad que se ha comentado en secciones anteriores. En verdad lo que tendrá que dar al usuario son los medios para acceder a toda la funcionalidad del sistema. Para ello constará de los siguientes formularios en su interior, así como barras de herramientas y demás:

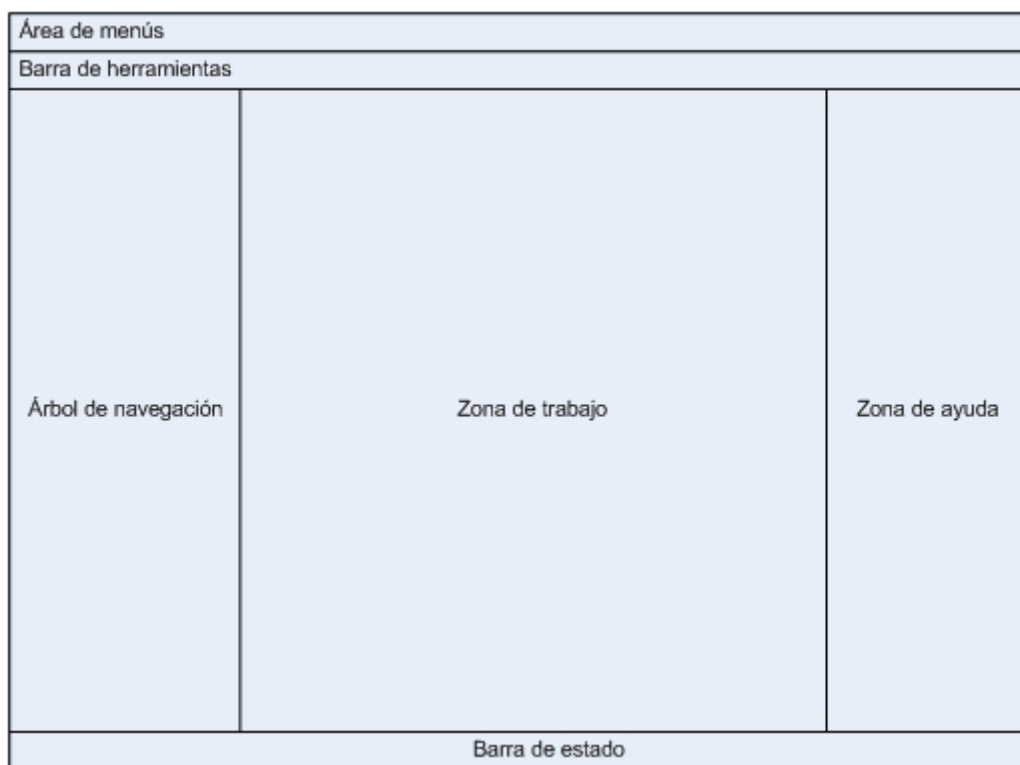
- Área de menús.
- Barra de herramientas.
- Área del árbol de navegación.
- Zona de mantenimiento de información.
- Zona de visualización de información.
- Zona de ayuda.

- Barra de estado.

La zona de mantenimiento de información y la zona de visualización de información constituyen el la zona de trabajo de la aplicación.

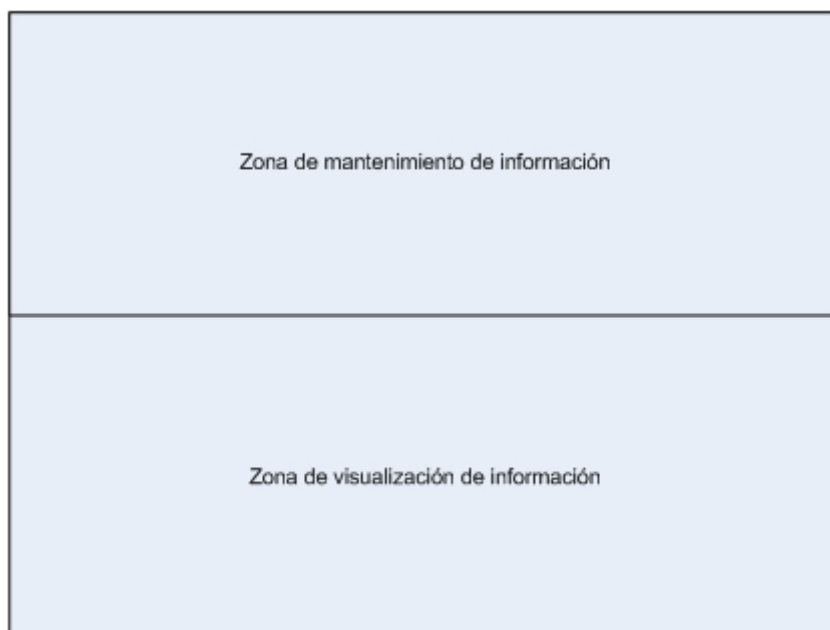
El interfaz general deberá controlar que cuando el usuario no esté conectado a la base de datos desactive las acciones y botones que accedan a la funcionalidad de mantenimiento y visualización de la información de la base de datos.

En la Figura 2.5.5 se muestra un diagrama simple de cómo será esta pantalla.



**Figura 2.5.4:** Diagrama de la ventana principal

En la Figura 2.5.6 se muestra la zona de trabajo que aparece en la figura 5.5 una vez que se haya cargado alguna información de la base datos.



**Figura 2.5.5:** Diagrama de la zona de trabajo

#### 2.5.4.1.1 Barra de menús

La barra de menús abarca todo el conjunto de las operaciones que el sistema es capaz de realizar, aportando un fácil acceso a las mismas. Se puede dividir en distintos menús, cada uno asociado a una misma funcionalidad, y dispuestos para ejecutar las acciones asociadas a cada opción presente. En función de la situación en que se encuentre la ejecución del sistema, ciertas opciones serán deshabilitadas, impidiendo así la ejecución de una acción inoportuna en un instante concreto.

La barra de menús de nuestra aplicación estará compuesta por cuatro menús diferentes:

- **Menú Conexión:** En este menú se proporciona la funcionalidad necesaria para la conexión, desconexión de la base de datos. Además permite el cambio de usuario y la salida del sistema.

No se puede seleccionar la desconexión de la base de datos si previamente no se ha realizado la conexión. Tampoco se podrá seleccionar el cambio de usuario si previamente no hay un usuario almacenado.

La opción de conexión llamará a la ventana de conexión, siempre que no haya un usuario por defecto, y la opción de cambio de usuario a la ventana de cambio de usuario.

- **Menú Opciones:** Este menú permite el acceso a los formularios de visualización y manipulación de la información de las distintas entidades que tienen relevancia en el sistema. Habrá una opción por cada una de las entidades existentes. Este menú no debe estar activo hasta que no se realice la conexión a la base de datos.
- **Menú Ver:** este menú ofrece distintas opciones de configuración de la interfaz del sistema. Permite activar y desactivar la barra de herramientas, el árbol de navegación y la ayuda del sistema.
- **Menú Ayuda:** En este menú de ayuda permitirá el acceso a la funcionalidad de ayuda que proporciona la aplicación.

#### 2.5.4.1.2 Barra de herramientas

La barra de herramientas va a estar formada por un conjunto de botones que nos van a permitir acceder de formas más rápida y eficaz a alguna de la funcionalidad suministrada por el área de menús. Es decir, se podría considerar que se tratan de botones de acceso directo a la mayor parte de las opciones disponibles en la barra de menús. Hay botones para la conexión y desconexión del sistema, para el acceso a la información de las entidades y para el acceso a la ayuda de la aplicación.

Los botones de acceso a la información y de desconexión deben estar inactivos cuando no se esta conectado a la base de datos. Igualmente el botón de cambio de usuario estará inactivo cuando no haya establecido un usuario por defecto.

Como comentamos antes, el botón de conexión llamará a la ventana de conexión, siempre que no haya un usuario por defecto, y el botón de cambio de usuario a la ventana de cambio de usuario.

#### **2.5.4.1.3 Árbol de navegación**

El árbol de navegación esta formado por una serie de nodos, los cuales nos permitirán acceder a los formularios de entrada y visualización de cada una de las entidades que forman parte de la base de datos.

Este menú no debe estar activo hasta que no se realice la conexión a la base de datos.

#### **2.5.4.1.4 Zona de mantenimiento de información**

En esta zona se cargaran el formulario de entrada correspondiente a la opción que se seleccione. Estos formularios brindan al usuario la posibilidad de insertar, modificar y borrar la información almacenada en la base de datos. Estos formularios serán descritos en el apartado correspondiente (cuando se junta la documentación completa).

Esta zona estará vacía cuando la aplicación no esté conectada a la base de datos o esté conectada y no se haya pulsado ninguna opción.

#### **2.5.4.1.5 Zona de visualización de información**

En esta zona se cargará el formulario de visualización correspondiente a la opción que se seleccione. Estos formularios brindan al usuario la posibilidad de observar la información recogida en la base de datos de la aplicación, permitiendo al usuario ordenarla de acuerdo a sus necesidades en cada momento. Estos formularios, al igual que los formularios de entrada serán descritos en el apartado correspondiente (cuando se junte la documentación completa).

Esta zona estará vacía cuando la aplicación no esté conectada a la base de datos o esté conectada y no se haya pulsado ninguna opción.

#### **2.5.4.1.6 Zona de ayuda**

En esta zona se debe insertar el formulario de ayuda, cuya misión es la de servir de utilidad a modo de ayuda contextual, con el objeto de ofrecer soporte, aconsejar e informar al usuario acerca de las distintas operaciones y acciones que en cada momento están disponibles en el sistema en ejecución.

La realización de este formulario, y por tanto su fase de especificación y diseño corresponden a una fase posterior.

#### **2.5.4.1.7 Barra de estado**

La barra de estado nos mostrará en todo momento la hora del sistema y la fecha.

### **2.5.4.2 Ventana de conexión**

Esta ventana se deberá activar cuando se pulse la opción de conectar a la base de datos en el caso de que la aplicación no tenga almacenada la información de ningún usuario, ya que en ese caso la conexión será automática.

Esta ventana debe permitir introducir el nombre del usuario y la contraseña de acceso a la base de datos. También dispondrá de dos botones, uno que realice el proceso de conexión y otro que cancele el establecimiento de la conexión. Ambos botones devuelven el control a la ventana principal.

Además la ventana de conexión deberá tener un marcador para indicar si se quiere almacenar los datos del usuario para futuras conexiones.

### **2.5.4.3 Ventana de cambio de usuario**

Esta ventana se deberá activar cuando se pulse la opción de cambio de usuario. Esta ventana es idéntica a la ventana de conexión y tiene los mismos componentes.

# 2.6

## DISEÑO DEL SISTEMA

---

En el capítulo anterior se ha realizado un análisis, desde el punto de vista técnico, de las principales características de la aplicación a desarrollar, así como del conjunto de requisitos que la misma deberá reunir para garantizar la correcta resolución del problema propuesto.

Dicho análisis se ha realizado considerando especialmente aquellos aspectos relativos a la funcionalidad que debe presentar el sistema, sin plantear la manera en la que ésta debe implementarse.

En este capítulo se proporcionará una descripción de la manera en que debe realizarse el Diseño del Sistema para garantizar que el mismo satisfaga la especificación de requisitos anteriormente desarrollada. El Diseño del Sistema se realizará utilizando la metodología UML, al igual que sucedió en la etapa dedicada a la Especificación de Requisitos [Booch et al., 1999].

Esta sección dentro de la documentación se encuentre dividida en cuatro fases, que se enumeran a continuación y se expondrán en los siguientes apartados:

- Diseño de datos.
- Diseño arquitectónico.
- Diseño de clases.
- Diseño de la interfaz

## 2.6.1 DISEÑO DE DATOS

Este apartado se encuentra dedicado al análisis de la estructura que presentará la información manipulada por el sistema.

Como hemos comentado, la interfaz general sólo maneja la información relacionada con el usuario, contraseña y cadena de conexión que utiliza el programa para poder conectar con la base de datos. Esta información consiste en el nombre del usuario y su contraseña, campos que se utilizarán para acceder a la base de datos. Ambos están formados como es lógico por cadenas, las cuales serán concatenación simple de letras o números, o caracteres dentro de la codificación ASCII.

Esta información no será almacenada en una clase como tal del sistema, ni en la base de datos correspondiente, ya que se trata de información permanente, y que se encuentra dentro de lo que se ha denominado dentro del sistema, las propiedades de configuración, por lo que se encuentran en un fichero del sistema, llamado *config.properties*, utilizado para almacenar éstos y otros datos necesarios del sistema relacionados con su configuración.

Dentro de la clase *MarcoPrincipal* que se detallará a continuación, se encuentran métodos que manipulan y acceden a la información contenida en este fichero.

## 2.6.2 DISEÑO ARQUITECTÓNICO

En este apartado se planteará la estructura global que debe presentar el sistema. Para ello, será necesario realizar la descomposición del mismo en una serie de componentes individuales, denominados subsistemas, que se establecerán en base a las diferentes funcionalidades desarrolladas por los mismos con respecto al comportamiento del sistema global.

Cada uno de estos subsistemas estará constituido por una serie de clases que presentan características en común, generalmente relativas a la funcionalidad o a la ubicación física de sus componentes.



Estas clases se organizarán, para facilitar la estructuración del sistema software, en una serie de módulos, establecidos fundamentalmente en base a la funcionalidad desarrollada por las clases pertenecientes a los mismos con respecto al funcionamiento del sistema de edición de fórmulas en su conjunto.

En función de lo anteriormente expuesto, la interfaz general puede organizarse en una serie de módulos claramente diferenciados, los cuales se especifican a continuación.

### **Módulo aplicacion**

El módulo aplicacion constituye la raíz de la jerarquía. En el mismo se incluye el componente que se encarga de definir, configurar y mostrar la interfaz general. Por lo tanto, este modulo es el que pone en marcha la aplicación.

### **Módulo aplicación.baseDatos**

El módulo aplicación.baseDatos constituye un submódulo de aplicación, y en el se incorpora el componente que se encarga de realizar la conexión a la base de datos, manteniendo el conector entre la aplicación y la base de datos.

### **Módulo aplicacion.interfaz**

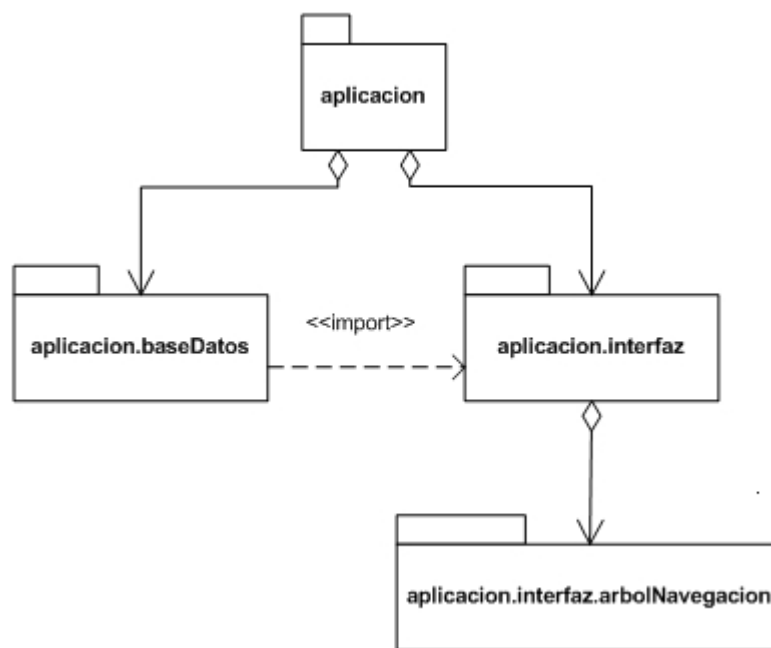
El módulo aplicacion.interfaz constituye un submódulo de aplicacion, y en el mismo se realiza la integración de aquellas clases del sistema que representan los diferentes componentes de los que está compuesta la interfaz general.

### **Módulo aplicacion.interfaz.arbolNavegacion**

El módulo aplicacion.interfaz.arbolNavegacion constituye un submódulo de aplicacion.interfaz, y en el mismo se incorporan de

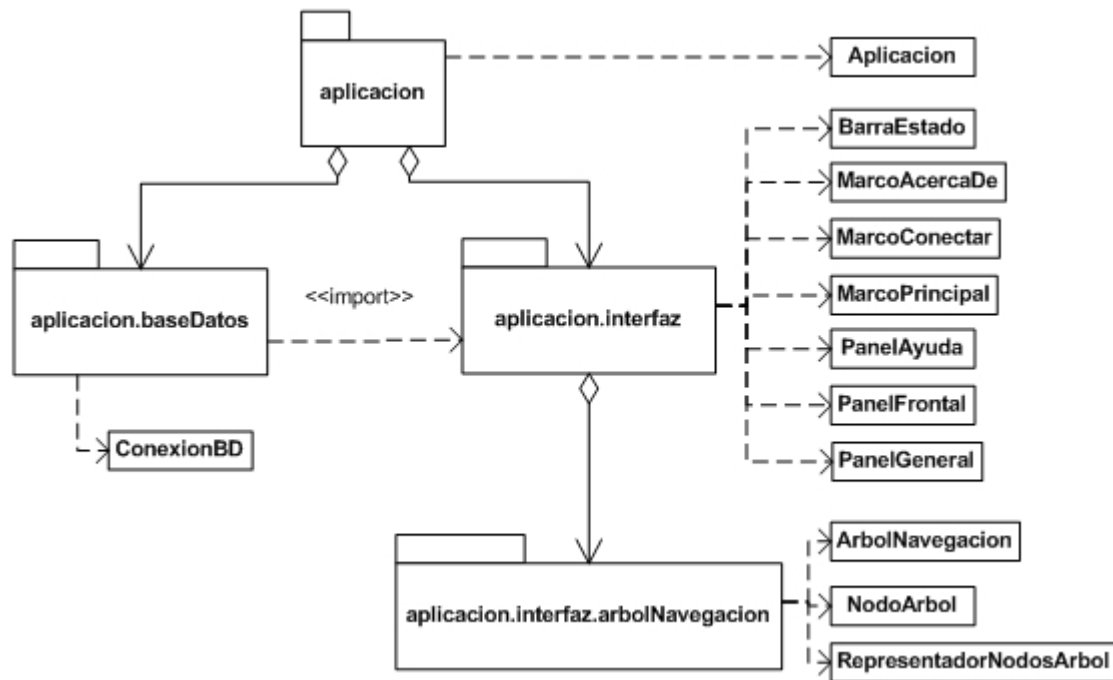
manera específica aquellos componentes que se encargan de crear y darle funcionalidad al árbol de navegación que forma parte de la de la interfaz general del sistema.

En la Figura 2.6.1 puede observarse el diagrama de paquetes utilizado para la representación de la arquitectura de la interfaz general.



**Figura 2.6.1:** Diagrama de paquetes

El diagrama general anteriormente representado puede ser completado con el conjunto de clases pertenecientes a cada uno de los diferentes módulos establecidos, de manera que se proporcione una estructura general sobre la composición del sistema. El diagrama de paquetes detallado se representa en la Figura 2.6.2.



**Figura 2.6.2:** Diagrama de paquetes detallado

El conjunto de clases pertenecientes a cada uno de estos módulos generales especificados será analizado posteriormente, en el apartado dedicado al Diseño de las Clases del sistema.

## 2.6.3 DISEÑO DE CLASES

Una vez analizada la estructura general que deberá presentar el sistema, constituida básicamente por la organización en módulos que puede establecerse para ordenar el contenido del mismo de una manera lógica desde el punto de vista funcional, se procederá a realizar a continuación el estudio de las diferentes clases que componen dichos módulos.

### 2.6.3.1 Clases del sistema

En este apartado se realizará una descripción de cada una de las clases del sistema especificando para cada una de ellas sus características más significativas mediante el uso de una plantilla. También se mostrará la

representación gráfica de cada clase, utilizando para ello la notación de UML [Booch et al., 1999].

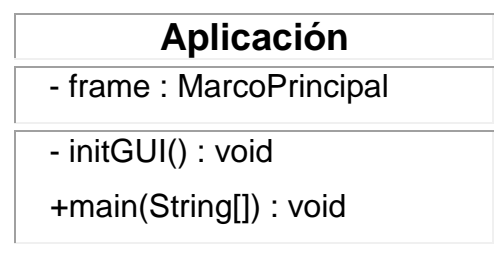
### 2.6.3.1.1 Clase Aplicacion

En la Tabla 2.6.1 podemos ver la descripción de la clase *Aplicacion*.

**Tabla 2.6.1:** Descripción de la clase Aplicación

NOMBRE	
<b><i>Aplicacion</i></b>	
DESCRIPCIÓN	
Esta es la clase principal del sistema y es la encargada de configurar y poner en marcha la ventana principal del sistema	
ATRIBUTOS	
<b><i>frame</i></b> . Instancia de la clase MarcoPrincipal, la cual representa el marco principal de la aplicación.	
OPERACIONES	
<b><i>initGUI()</i></b> . Este método permite crear y mostrar la interfaz gráfica. Establece la dimensión y características de la ventana. <b><i>main(String[])</i></b> . Este método es el encargado de permitir la ejecución de la aplicación y de seleccionar el aspecto de la interfaz.	

En la Figura 2.6.3 se puede ver la representación en UML esta clase.



**Figura 2.6.3:** Representación UML de la Clase Aplicacion

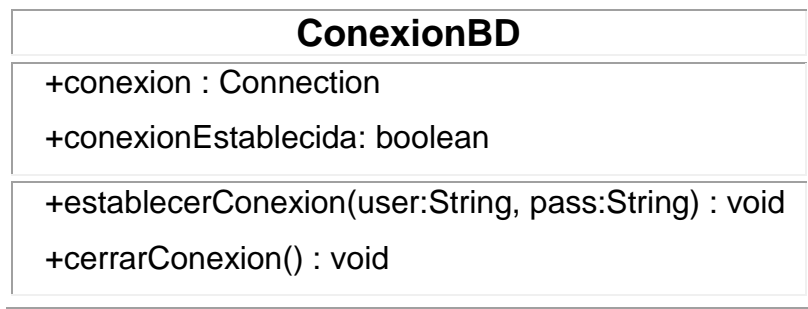
### 2.6.3.1.2 Clase ConexiónBD

En la Tabla 2.6.2 podemos ver la descripción de la clase ConexionBD.

**Tabla 2.6.2:** Descripción de la clase ConexionBD

NOMBRE	<b><i>ConexionBD</i></b>
DESCRIPCIÓN	Clase que maneja la conexión a la base de datos y proporciona los métodos para conectarse y desconectarse de la base de datos.
ATRIBUTOS	<p><b><i>conexion</i></b>. Conector de la aplicación con la base de datos. Es una variable estática (global) que puede ser accedida sin necesidad de tener que crear un objeto de la clase.</p> <p><b><i>conexionEstablecida</i></b>. Variable booleana estática que nos indica si la aplicación está o no está conectada a la base de datos.</p>
OPERACIONES	<p><b><i>establecerConexion(String user, String pass)</i></b>. Este método realiza la conexión con la base de datos utilizando el usuario y la contraseña pasada como parámetros. Es un método estático (global) que puede ser llamado sin necesidad de tener que crear un objeto de la clase.</p> <p><b><i>cerrarConexion()</i></b>. Este método es el encargado de cerrar la conexión con la base de datos. Es un método estático (global) que puede ser llamado sin necesidad de tener que crear un objeto de la clase.</p>

En la Figura 2.6.4 se puede ver la representación en UML esta clase.



**Figura 2.6.4:** Representación UML de la Clase ConexionBD

### 2.6.3.1.3 Clase MarcoPrincipal

En la Tabla 2.6.3 podemos ver la descripción de la clase *MarcoPrincipal*.

**Tabla 2.6.3:** Descripción de la clase MarcoPrincipal

NOMBRE	
<b><i>MarcoPrincipal</i></b>	
DESCRIPCIÓN	
	Representa el marco principal donde se van a insertar los componentes gráficos que constituyen la interfaz gráfica de usuario del sistema. La definición de esta clase se realizará en base a la derivación de la clase javax.swing.JFrame definida por el lenguaje de programación Java.
ATRIBUTOS	
	<p><b><i>contenPane</i></b>. Panel donde se añaden los componentes de la interfaz general.</p> <p><b><i>panelGeneral</i></b>. Instancia de la clase PanelGeneral donde se insertan el árbol navegación y los distintos formularios.</p> <p><b><i>panelAyuda</i></b>. Instancia de la clase PanelAyuda donde se insertara el formulario de ayuda.</p>

***statusBar***. Instancia de la clase BarraEstado que conforma la barra de estado de la interfaz general.

***barraMenu***. Variable que representa la barra de menús.

***barraHerramientas***. Variable que representa la barra de herramientas.

## OPERACIONES

***jblnit()***. Este método configura el marco principal añadiéndole los componentes y la funcionalidad deseada.

***cargarPropiedades()***. Función que carga las propiedades del sistema del fichero de configuración (config.properties).

***modificarPropiedad(String item, String valor)***. Modifica la propiedad *item* del sistema con *valor*.

***getPanelGeneral()* : PanelGeneral**. Devuelve el panel general que está incluido dentro del marco.

***getPanelAyuda()* : PanelAyuda**. Devuelve el panel de ayuda que está incluido dentro del marco.

***reiniciarPanelFrontal()***. Se encarga de recargar el panel frontal con los nuevos componentes.

***setEnabledBotones(boolean estado)***. Habilita o deshabilita botones de la barra de herramientas según el valor de la variable *estado*.

***setEnabledArbol(boolean estado)***. Habilita o deshabilita el árbol de navegación según el valor de la variable *estado*.

***setEnabledMenus(boolean estado)***. Habilita o deshabilita algunos menús según el valor de la variable *estado*.

***setEnabledCambiarUsuario(boolean estado)***. Habilita o deshabilita la funcionalidad de cambio de usuario según el valor de la variable *estado*.

***menuSalirAccion()***. Finaliza la aplicación.

***menuAcercaDeAccion()***. Llama a mensaje Acerca De de la

aplicación.

***menuConectarAccion()***. Llama a la ventana de conexión del sistema.

***menuCambiarUsuarioAccion()***. Llama a la ventana de cambio de usuario.

***menuDesconectarAccion()***. Realiza la desconexión del sistema.

En la Figura 2.6.5 se puede ver la representación en UML esta clase.





```

- jblnit() : void
+cargarPropiedades() : void
+modificarPropiedades(item:String, valor:String) : void
+getPanelGeneral() : PanelGeneral
+getPanelAyuda() : PanelAyuda
- reiniciarPanelFrontal() : void
- setEnabledBotones(estado:boolean) : void
- setEnabledArbol(estado:boolean) : void
- setEnabledMenus(estado:boolean) : void
- setEnabledCambiarUsuario(estado:boolean) : void
- menuSalirAccion() : void
- menuAcercaDeAccion() : void
- menuConectarAccion() :void
- menuCambiarUsuarioAccion() : void
- menuDesconectarAccion() : void

```

**Figura 2.6.5:** Representación UML de la Clase MarcoAplicacion

#### 2.6.3.1.4 Clase BarraEstado

En la Tabla 2.6.4 podemos ver la descripción de la clase *BarraEstado*.

**Tabla 2.6.4:** Descripción de la clase BarraEstado

NOMBRE	<b><i>BarraEstado</i></b>
DESCRIPCIÓN	Representa la barra de estado de la aplicación. La definición de esta clase se realizará en base a la derivación de la clase javax.swing.JPanel definida por el lenguaje de programación Java.
ATRIBUTOS:	<b><i>cprTxt1</i></b> . Campo de texto para introducir la fecha.

***cprTxt2***. Campo de texto para introducir la hora.

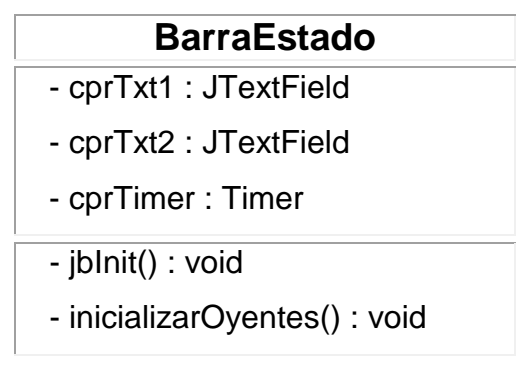
***cprTimer***. Contador temporal para la visualización de la hora.

#### OPERACIONES

***jblnit()***. Este método configura los componentes de la barra de estado y los añade a la misma.

***inicializarOyentes()***. Obtiene la hora y fecha del sistema de forma continua mediante eventos.

En la Figura 2.6.6 se puede ver la representación en UML esta clase.



**Figura 2.6.6:** Representación UML de la Clase BarraEstado

#### 2.6.3.1.5 Clase MarcoAcercaDe

En la Tabla 2.6.5 podemos ver la descripción de la clase *MarcoAcercaDe*.

**Tabla 2.6.5:** Descripción de la clase MarcoAcercaDe

NOMBRE	<b><i>MarcoAcercaDe</i></b>
DESCRIPCIÓN	Representa el marco que se mostrará cuando se acceda al menú

*Acerca De*. La definición de esta clase se realizará en base a la derivación de la clase `javax.swing. JDialog` definida por el lenguaje de programación Java.

#### ATRIBUTOS:

***botonAceptar***. Botón *Aceptar* del marco.

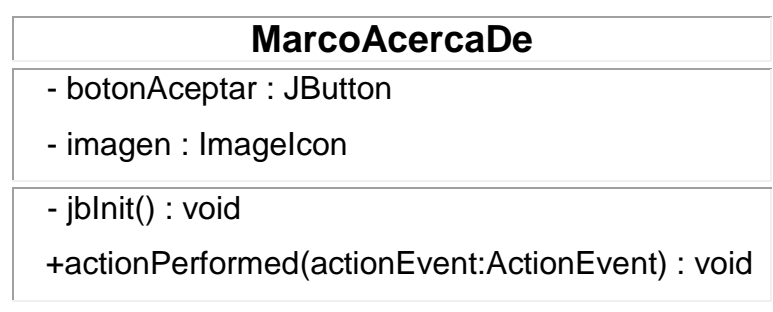
***imagen***. Imagen que se muestra en el marco.

#### OPERACIONES

***jblnit()***. Este método configura los componentes del marco y los añade al mismo.

***actionPerformed(ActionEvent actionEvent)***. Funcionalidad del botón *Aceptar* que consiste en cerrar el marco.

En la Figura 2.6.7 se puede ver la representación en UML esta clase.



**Figura 2.6.7:** Representación UML de la Clase *MarcoAcercaDe*

#### 2.6.3.1.6 Clase *MarcoConectar*

En la Tabla 2.6.6 podemos ver la descripción de la clase *MarcoConectar*.

**Tabla 2.6.6:** Descripción de la clase *MarcoConectar*

NOMBRE	
<b><i>MarcoConectar</i></b>	
DESCRIPCIÓN	

Representa el marco que se mostrará cuando se pulse la opción de conexión y no haya almacenado ningún usuario. La definición de esta clase se realizará en base a la derivación de la clase `javax.swing. JDialog` definida por el lenguaje de programación Java.

#### ATRIBUTOS:

***padre.*** Instancia de la clase `MarcoPrincipal`, que almacena el marco padre, que es el marco principal de la aplicación.

***user.*** Campo de texto donde se va a introducir el nombre de usuario.

***pass.*** Campo de texto encriptado para introducir la contraseña.

***botonConectar.*** Botón *Conectar* del marco.

***botonCancelar.*** Botón *Cancelar* del marco.

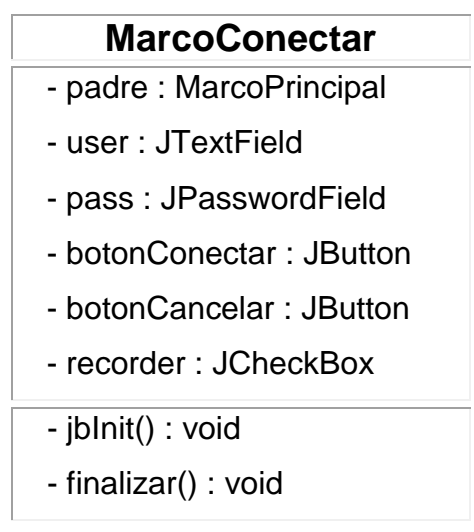
***recordar.*** Marcador para recordar o no los datos de conexión.

#### OPERACIONES

***jblnit().*** Este método configura los componentes del marco y los añade al mismo.

***conectar().*** Realiza la conexión a la base de datos.

En la Figura 2.6.8 se puede ver la representación en UML esta clase.



**Figura 2.6.8:** Representación UML de la Clase `MarcoConectar`

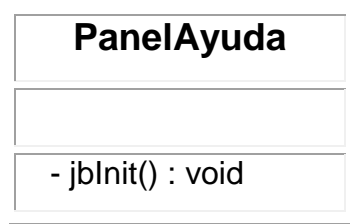
### 2.6.3.1.7 Clase PanelAyuda

En la Tabla 2.6.7 podemos ver la descripción de la clase *PanelAyuda*.

**Tabla 2.6.7:** Descripción de la clase PanelAyuda

NOMBRE	
<b><i>PanelAyuda</i></b>	
DESCRIPCIÓN	
	Representa el panel donde se insertará el formulario de ayuda. La definición de esta clase se realizará en base a la derivación de la clase javax.swing.JPanel definida por el lenguaje de programación Java.
ATRIBUTOS:	
	No tiene.
OPERACIONES	
	<b><i>jblnit()</i></b> . Este método configura los componentes del panel y los añade al mismo.

En la Figura 2.6.9 se puede ver la representación en UML esta clase.



**Figura 2.6.9:** Representación UML de la Clase PanelAyuda

### 2.6.3.1.8 Clase PanelGeneral

En la Tabla 2.6.8 podemos ver la descripción de la clase *PanelGeneral*.

Tabla 2.6.8: Descripción de la clase PanelGeneral

NOMBRE	<b><i>PanelGeneral</i></b>
DESCRIPCIÓN	Representa el panel donde se insertarán el árbol de navegación y los distintos formularios de la aplicación. Esta clase deriva de la clase javax.swing.JSplitPane definida por el lenguaje de programación Java.
ATRIBUTOS:	<p><b><i>arbolNavegacion</i></b>. Instancia de la clase Arbolnavegacion que representa el árbol de navegación del interfaz general.</p> <p><b><i>panelFrontal</i></b>. Instancia de la clase PanelFrontal que representa el panel donde se insertaran los formularios de entrada y visualización.</p>
OPERACIONES	<p><b><i>jblnit()</i></b>. Este método configura los componentes del panel y los añade al mismo.</p> <p><b><i>getArbolNavegacion()</i> : ArbolNavegacion</b>. Devuelve el árbol de navegación.</p> <p><b><i>getPanelFrontal()</i> : PanelFrontal</b>. Devuelve el panel frontal que está incluido dentro de este panel general.</p>

En la Figura 2.6.10 se puede ver la representación en UML esta clase.

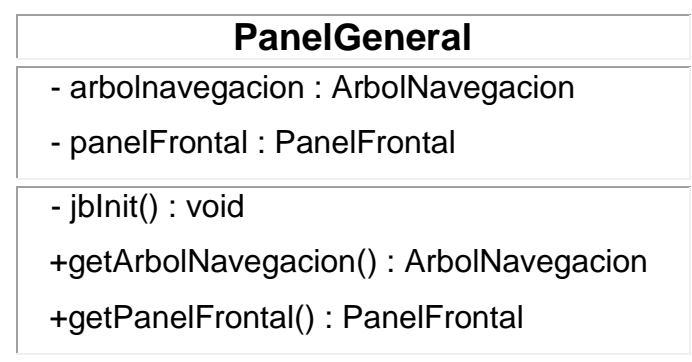


Figura 2.6.10: Representación UML de la Clase PanelGeneral

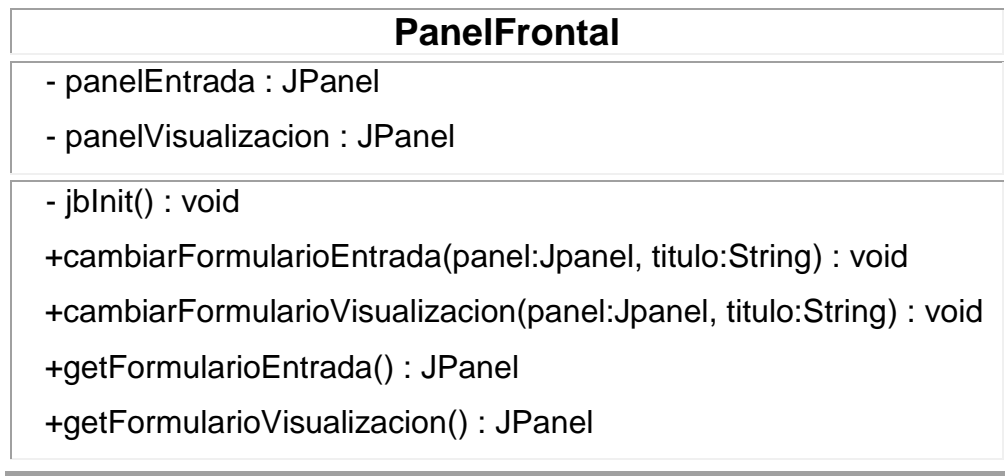
### 2.6.3.1.9 Clase PanelFrontal

En la Tabla 2.6.9 podemos ver la descripción de la clase *PanelFrontal*.

Tabla 2.6.9: Descripción de la clase PanelFrontal

NOMBRE	<b><i>PanelFrontal</i></b>
DESCRIPCIÓN	Representa el panel donde se insertarán los formularios de entrada y visualización. La definición de esta clase se realizará en base a la derivación de la clase <code>javax.swing.JSplitPane</code> definida por el lenguaje de programación Java.
ATRIBUTOS:	<p><b><i>panelEntrada</i></b>. Panel donde se irán cargando los diferentes formularios de entrada.</p> <p><b><i>panelVisualizacion</i></b>. Panel donde se irán cargando los diferentes formularios de visualización.</p>
OPERACIONES	<p><b><i>jblnit()</i></b>. Configura los componentes del panel y los añade al mismo.</p> <p><b><i>cambiarFormularioEntrada(Jpanel panel, String titulo)</i></b>. Sustituye el formulario de entrada actual por el indicado en los parámetros.</p> <p><b><i>cambiarFormularioVisualizacion(Jpanel panel, String titulo)</i></b>. Sustituye el formulario de visualización actual por el indicado en los parámetros.</p> <p><b><i>getFormularioEntrada()</i> : JPanel</b>. Devuelve el panel del formulario de entrada.</p> <p><b><i>getFormularioVisualizacion()</i> : JPanel</b>. Devuelve el panel del formulario de visualización.</p>

En la Figura 2.6.11 se puede ver la representación en UML esta clase.



**Figura 2.6.11:** Representación UML de la Clase PanelFrontal

#### 2.6.3.1.10 Clase ArbolNavegacion

En la Tabla 2.6.10 podemos ver la descripción de la clase *ArbolNavegacion*.

**Tabla 2.6.10:** Descripción de la clase ArbolNavegacion

NOMBRE	<b><i>ArbolNavegacion</i></b>
DESCRIPCIÓN	Representa el árbol de navegación de la aplicación. Esta clase deriva de la clase javax.swing.JTree definida por el lenguaje de programación Java.
ATRIBUTOS:	<p><b><i>padre</i></b>. Instancia de la clase PanelGeneral, que almacena el panel padre, que es el panel general de la aplicación.</p> <p><b><i>nodoSeleccionado</i></b>. Almacena el nodo del árbol que está seleccionado actualmente.</p> <p><b><i>popupMenuNodos</i></b>. Menú asociado a cada nodo. Se desplegará al pulsar el botón derecho del ratón sobre cualquier nodo.</p>



**activo.** Variable booleana que indica si el árbol está actualmente activo o deshabilitado.

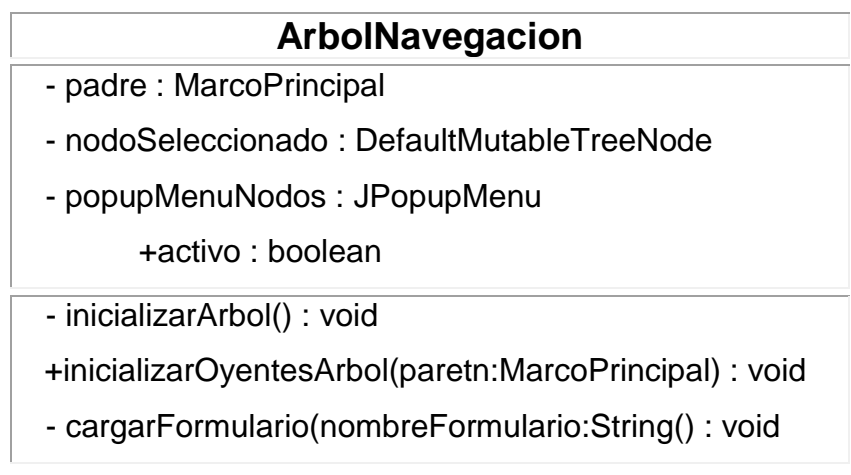
#### OPERACIONES

**inicializarArbol().** Este método crea y configura el árbol.

**inicializarOyentesArbol(*PanelGeneral parent*).** Inicializa los oyentes de eventos de los nodos del árbol.

**cargarFormulario(*String nombreFormulario*).** Carga los formularios de entrada y visualización correspondientes a la cadena *nombreFormulario* en los paneles correspondientes.

En la Figura 2.6.12 se puede ver la representación en UML esta clase.



**Figura 2.6.12:** Representación UML de la Clase ArbolNavegacion

#### 2.6.3.1.11 Clase NodoArbol

En la Tabla 2.6.11 podemos ver la descripción de la clase *NodoArbol*.

**Tabla 2.6.11:** Descripción de la clase NodoArbol

NOMBRE	
<b><i>NodoArbol</i></b>	
DESCRIPCIÓN	

Representa un nodo del árbol de navegación de la aplicación. La definición de esta clase se realizará en base a la derivación de la clase `javax.swing.tree.DefaultMutableTreeNode` definida por el lenguaje de programación Java.

#### ATRIBUTOS:

***nombreNodo***. Cadena que almacena el nombre del nodo.

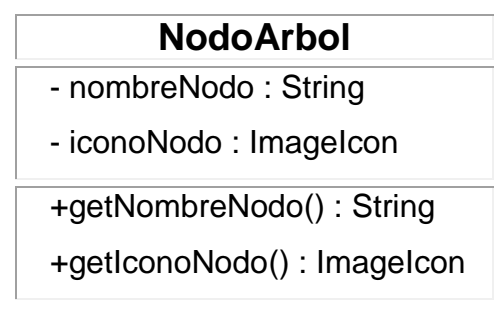
***iconoNodo***. Icono asociado al nodo.

#### OPERACIONES

***getNombreNodo()* : *String()***. Devuelve el nombre del nodo.

***getIconoNodo()* : *ImageIcon()***. Devuelve el icono del nodo.

En la Figura 2.6.13 se puede ver la representación en UML esta clase.



**Figura 2.6.13:** Representación UML de la Clase *NodoArbol*

#### 2.6.3.1.12 Clase RepresentadorNodosArbol

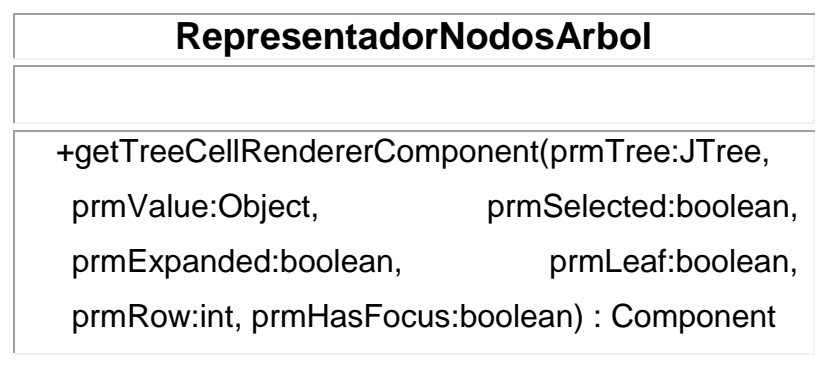
En la Tabla 2.6.12 podemos ver la descripción de la clase *RepresentadorNodosArbol*.

**Tabla 2.6.12:** Descripción de la clase *RepresentadorNodoArbol*

NOMBRE	
	<b><i>RepresentadorNodosArbol</i></b>

DESCRIPCIÓN	
	Representa a un gestor de la representación visual de cada nodo del árbol de navegación. Esta clase deriva de la clase javax.swing.tree.DefaultTreeCellRenderer definida por el lenguaje Java.
ATRIBUTOS:	
	No tiene.
OPERACIONES	
	<b><i>getTreeCellRendererComponent(JTree prmTree, Object prmValue, boolean prmSelected, boolean prmExpanded, boolean prmLeaf, int prmRow, boolean prmHasFocus) : Component.</i></b> Establece los parámetros de representación de los nodos del árbol.

En la Figura 2.6.14 se puede ver la representación en UML esta clase.



**Figura 2.6.14:** Representación UML de la Clase RepresentadorNodosArbol

### 2.6.3.2 Diagrama de clases del sistema

En la Figura 2.6.15 se muestra el diagrama de clases, el cual pretende mostrar los aspectos estructurales acerca de las clases consideradas dentro del dominio del problema y de las relaciones principales necesarias entre dichas clases para cumplir los requisitos funcionales del sistema.

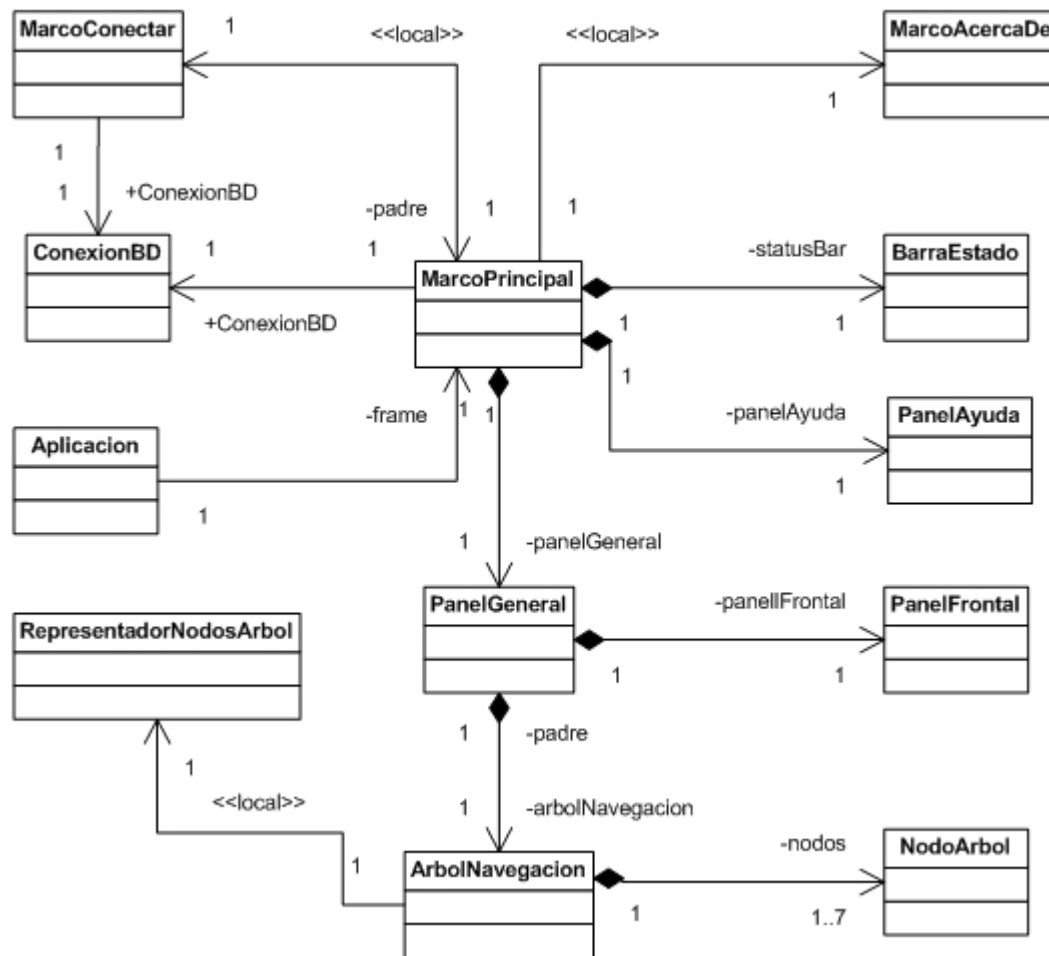


Figura 2.6.15: Diagrama de Clases

## 2.6.4 DISEÑO DE LA INTERFAZ

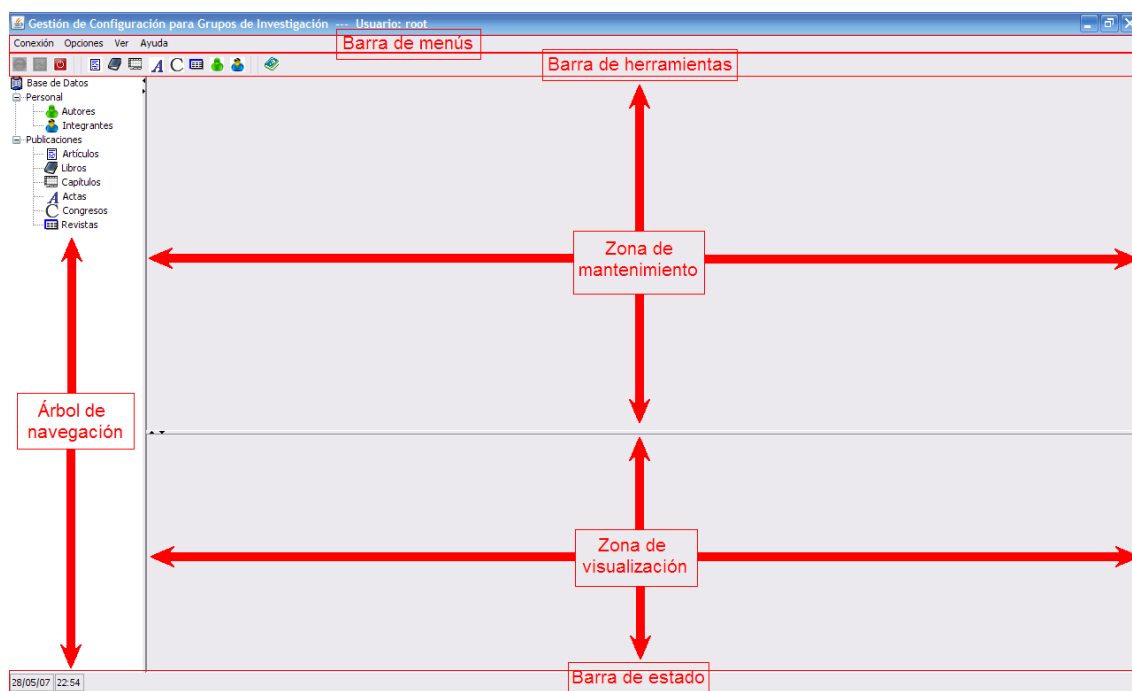
En este capítulo se procederá a describir el aspecto final de la interfaz de usuario de este módulo que previamente fue especificada en el apartado 2.5.4 del presente documento. Dicha descripción atenderá principalmente al diseño final de cada una de las ventanas.

### 2.6.4.1 Ventana Principal

La ventana principal es el componente situado en el nivel superior, sobre el que se apoyan todos los elementos que constituyen la interfaz del sistema, cumpliendo así como gestor de todos los procesos a realizar. Mediante este marco, se puede acceder a toda la funcionalidad que se desarrolla. La ventana

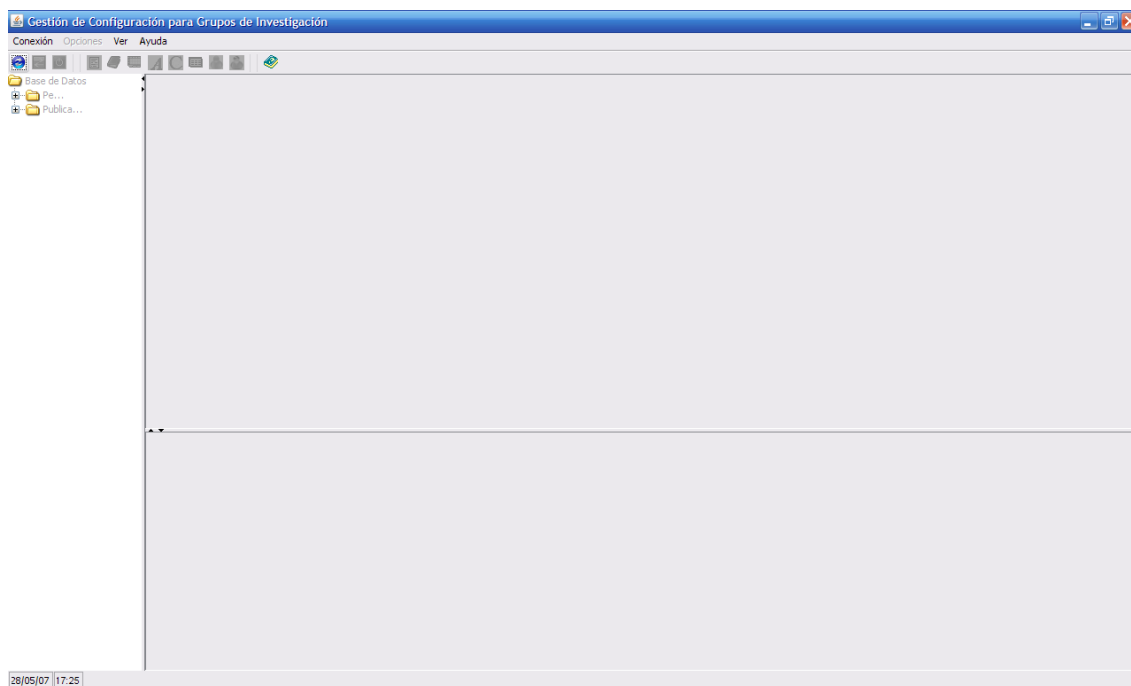
principal está dividida en los componentes que fueron especificados en el apartado 2.5.4.1.

A continuación en la Figura 2.6.16 se muestra una captura de la ventana principal cuando la aplicación está conectada a la base de datos.



**Figura 2.6.16:** Ventana Principal

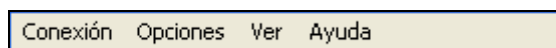
En el caso de que la aplicación no esté conectada a la base de datos, el aspecto de la ventana principal es el que se puede ver en la Figura 2.6.17.



**Figura 2.6.17:** Ventana Principal Inactiva

#### 2.6.4.1.1 Barra de menús

La barra de menús de la ventana principal puede observarse en la Figura 2.6.18.

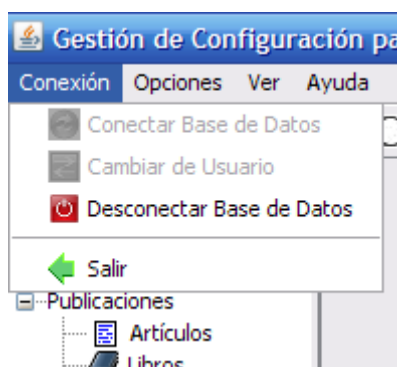


**Figura 2.6.18:** Barra de menús

En la Figura 2.6.17 se puede observar el aspecto de la barra de menús cuando la aplicación está desconectada.

A continuación se describirán y mostrarán las distintas opciones disponibles en cada sección de menú integrante de este componente:

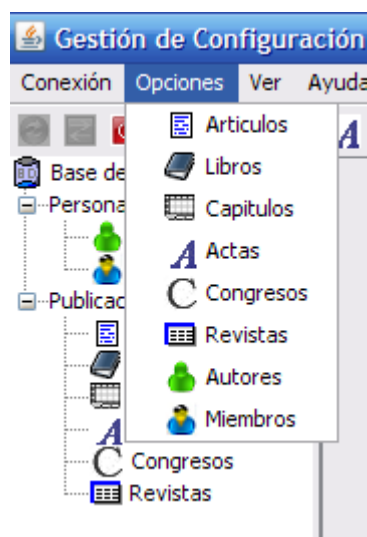
**Menú *Conexión*:** En este menú se proporciona la funcionalidad necesaria para la gestión de la base de datos. A continuación se describen las siguientes opciones de las que consta, las cuales son mostradas en la Figura 2.6.19:



**Figura 2.6.19:** Menú Conexión

- Opción *Conectar Base de Datos*: esta opción permite establecer la conexión con la base de datos, para ello llama al formulario de conexión.
- Opción *Desconectar Base de Datos*: a través de esta opción se permite la desconexión del sistema de la base de datos.
- Opción *Cambiar de Usuario*: muestra un formulario para el cambio de usuario en el caso de que haya establecido un usuario por defecto.
- Opción *Salir*: permite la salida de la aplicación mediante la correspondiente opción del menú.

**Menú *Opciones*:** Este menú permite la gestión y tratamiento de las distintas entidades que tienen relevancia en el sistema. Este menú estará deshabilitado cuando la aplicación esté desconectada de la base de datos tal y como se puede ver en la Figura 2.6.17. A continuación se especifican las diferentes opciones, las cuales se muestran en la Figura 2.6.20:

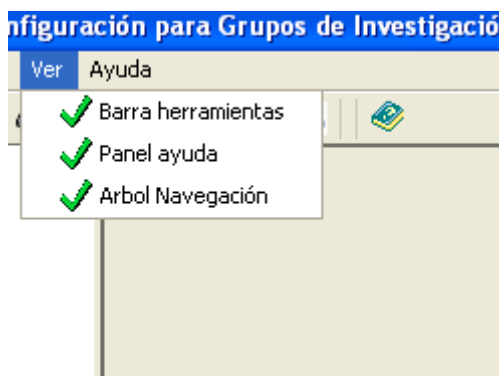


**Figura 2.6.20:** Menú Opciones

- Opción *Artículos*: mediante esta opción se accede a los formularios asociados al tratamiento de diversos artículos.
- Opción *Libros*: a través de esta opción se permite el acceso a los formularios asociados al tratamiento de diferentes libros.
- Opción *Capítulos*: esta opción permite el acceso a la gestión de los capítulos presentes en los distintos libros existentes en el sistema.
- Opción *Actas*: mediante esta opción se accede a los formularios asociados al tratamiento de diversas actas.
- Opción *Congresos*: esta opción permite acceder a los formularios relativos a los congresos de interés para el grupo de investigación.
- Opción *Revistas*: a través de esta opción se permite el acceso a los formularios asociados al tratamiento de diferentes revistas.
- Opción *Autores*: con esta opción se permite la manipulación de diferentes operaciones asociadas a la gestión de distintos autores asociados al grupo de investigación.
- Opción *Miembros*: mediante esta opción se accede a la gestión de los miembros relacionados con el grupo de investigación.



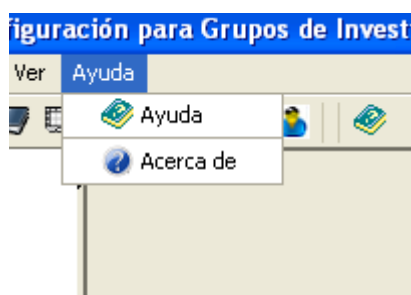
**Menú Ver:** este menú ofrece distintas opciones de configuración de la interfaz del sistema, entre las que se pueden destacar las mostradas en la Figura 2.6.21. Además se puede observar como en este caso, los tres componentes están activos.



**Figura 2.6.21:** Menú Ver

- Opción *Barra Herramientas*: esta opción permite activar o desactivar la barra de herramientas.
- Opción *Panel Ayuda*: a través de esta opción se permite tener activo o no el panel de ayuda.
- Opción *Árbol Navegación*: con esta opción permite tener visible u oculto el árbol de navegación.

**Menú Ayuda:** En este menú de ayuda se presentan algunas opciones sobre la ayuda de la aplicación, las cuales pueden visualizarse en la Figura 2.6.22:



**Figura 2.6.22:** Menú Ayuda

- Opción *Ayuda*: mediante esta opción de menú se permite al usuario acceder a la ayuda del sistema.
- Opción *Acerca de*: Esta opción muestra una ventana con información relativa a la identificación de la aplicación. El resultado de pulsar esta opción se puede observar en la Figura 2.6.23.



**Figura 2.6.23:** Ventana Acerca De

#### 2.6.4.1.2 Barra de herramientas

La interfaz dispone de una barra de herramientas, situada justo debajo de la barra de menús, y constituida por un conjunto de botones cuya única misión es proporcionar un acceso más rápido a la funcionalidad que el sistema es capaz de ofrecer. Es por ello por lo que estos botones estarán habilitados siempre que el estado del sistema lo permita, al igual que ocurría en la barra de menús.

La barra de herramientas puede observarse en la Figura 2.6.24.















**Figura 2.6.24:** Barra de herramientas

En la Figura 2.6.17 se puede observar el aspecto de la barra de herramientas cuando la aplicación está desconectada.

En la Tabla 2.6.13 se representa una descripción de la funcionalidad general implementada por cada botón de la barra de herramientas, y que como puede apreciarse, es similar a la ofrecida en la barra de menús.

**Tabla 2.6.13:** Funcionalidad de los botones de la Barra de Herramientas

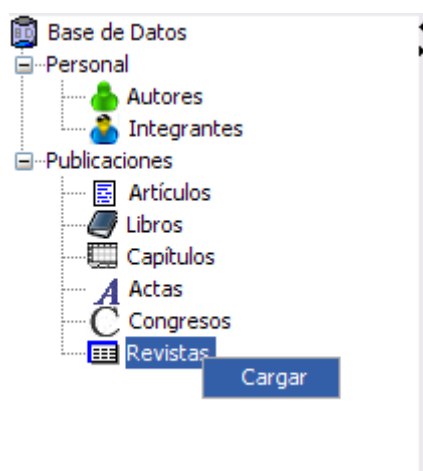
Botón	Funcionalidad
	<b>Conectar</b> con la Base de Datos del sistema.
	<b>Cambiar</b> el <b>usuario</b> de la aplicación y por lo tanto de la base de datos.
	<b>Desconectar</b> el sistema de la Base de Datos.
	Opciones de gestión de la información de <b>Artículos</b> del sistema.
	Gestión de <b>Libros</b> en el sistema.
	Opciones de tratamiento de <b>Capítulos</b> pertenecientes a libros que desean ser mantenidos por el grupo de investigación.
	Opciones de gestión de <b>Actas</b> relacionadas con el grupo de investigación.
	Tratamiento de <b>Congresos</b> relacionados, de algún modo, con el grupo de investigación.
	Opciones de gestión de <b>Revistas</b> a considerar en el sistema.
	Tratamiento de operaciones asociadas a <b>Autores</b> relacionados con el sistema.
	Operaciones relacionadas con los <b>Miembros</b> del grupo de investigación.

	Acceso a la <b>Ayuda</b> del sistema.
---	---------------------------------------

### 2.6.4.1.3 Árbol de navegación

La interfaz cuenta con un árbol de exploración de la información presente en el proyecto. Se encuentra ubicado en la parte izquierda de la aplicación y tiene como utilidad básica proporcionar un acceso más sencillo a toda la información disponible en la base de datos, mostrado de forma esquemática, mediante los nodos dispuestos para ello.

El árbol, cuya estructura puede apreciarse en la Figura 2.6.25, cuenta con una serie de nodos básicos, cuya funcionalidad es la misma que la descrita para cada botón en la Tabla 2.6.13.

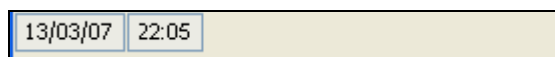


**Figura 2.6.25:** Árbol de Navegación

Como se puede observar en la figura anterior, al seleccionar un nodo del árbol de navegación y pulsar el botón derecho del ratón se permite cargar los formularios de entrada y visualización de la opción seleccionada. También se cargarán haciendo doble clic con el botón izquierdo del ratón en la opción deseada o presionando la tecla *enter* en la opción seleccionada.

#### 2.6.4.1.4 Barra de estado

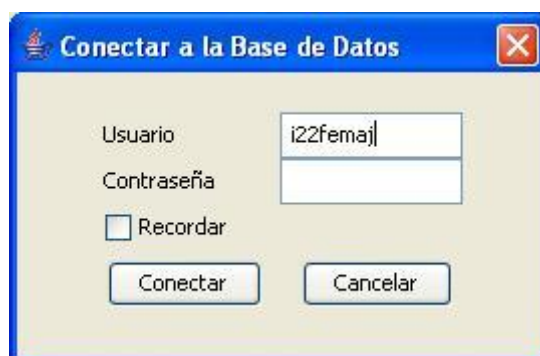
Este componente se encuentra en la parte inferior de la aplicación. En él se puede visualizar la fecha y hora actual. El formato utilizado para especificar la fecha es *día/mes/año*, mientras que la hora se especifica siguiendo el formato siguiente *hora:minutos*. Su apariencia puede verse en la Figura 2.6.26.



**Figura 2.6.26:** Barra de estado

#### 2.6.4.2 Ventana de conexión

En la Figura 2.6.27 podemos ver la ventana de conexión. Esta ventana se mostrará cuando se solicite la conexión a la base de datos y el sistema no tenga ningún usuario almacenado.



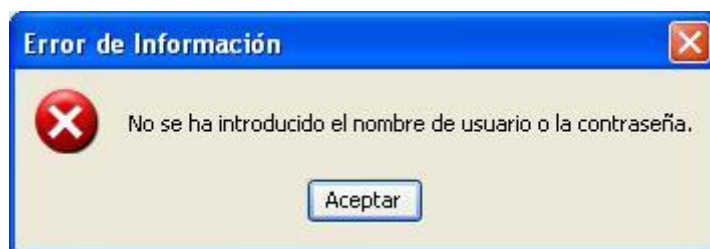
**Figura 2.6.27:** Ventana de Conexión

Como vemos, la ventana posee dos cuadros de texto donde se deben introducir el nombre del usuario y la contraseña del mismo en la base de datos. Cuando se introduce la contraseña en la pantalla se muestra una cadena de asteriscos para aumentar la seguridad del sistema.

Dicha ventana también está compuesta por un recuadro marcador, el cual será tachado si se desea que el sistema almacene la información del usuario para próximas conexiones.

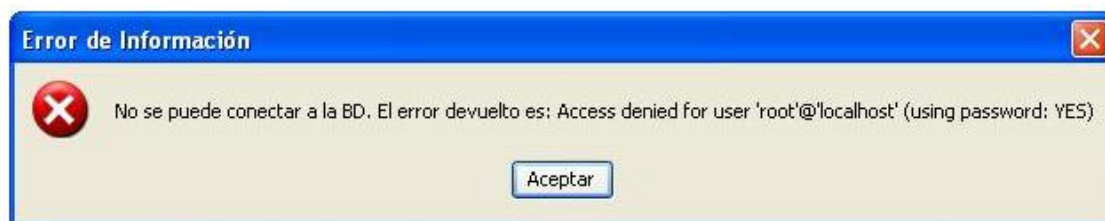
Por último, la ventana posee dos botones: *Aceptar* y *Cancelar*. El botón *Cancelar* cierra esta ventana y devuelve el control a la ventana principal sin

realizar ningún proceso. El botón *Aceptar* cierra la ventana realizando el proceso de conexión a la base de datos en el caso de que los datos estén correctamente introducidos. Si al pulsar el botón *Aceptar* no se ha introducido texto en alguno de los dos campos se muestra el mensaje de error que podemos ver en la Figura 2.6.28.



**Figura 2.6.28:** Mensaje de Error en la Ventana de Conexión

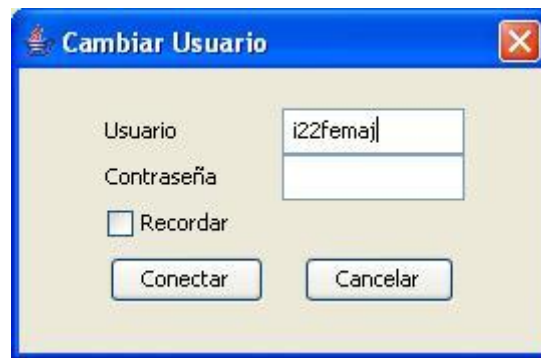
También se mostrará un mensaje de error cuando no se pueda realizar la conexión a la base de datos. En ese mensaje se mostrará el motivo por el cual no se ha podido establecer la conexión. Un ejemplo de este tipo de mensaje lo podemos ver en la Figura 2.6.29.



**Figura 2.6.29:** Mensaje de Error durante la Conexión

#### **2.6.4.3 Ventana de cambio de usuario**

En la Figura 2.6.30 podemos ver la ventana de cambio de usuario. Esta ventana se mostrará cuando se solicite el cambio de usuario.



**Figura 2.6.30:** Ventana de Cambio de Usuario

Como vemos, la ventana posee dos cuadros de texto donde se deben introducir el nombre del usuario y la contraseña del mismo en la base de datos. Cuando se introduce la contraseña en la pantalla se muestra una cadena de asteriscos para aumentar la seguridad del sistema.

Dicha ventana también está compuesta por un recuadro marcador, el cual será tachado si se desea que el sistema almacene la información del usuario para próximas conexiones.

Por último, la ventana posee dos botones: *Aceptar* y *Cancelar*. El botón *Cancelar* cierra esta ventana y devuelve el control a la ventana principal sin realizar ningún proceso. El botón *Aceptar* cierra la ventana realizando el proceso de cambio de usuario y conexión a la base de datos en el caso de que los datos estén correctamente introducidos. Si al pulsar el botón *Aceptar* no se ha introducido texto en alguno de los dos campos se muestra el mismo mensaje de error que en la ventana anterior, el cual podemos ver en la figura 6.28.

Igualmente, si se produce algún problema en el establecimiento de la conexión se mostrará un mensaje como el que podemos ver en la Figura 2.6.29.

# 2.7

## PRUEBAS

---

A continuación se detalla la fase de pruebas a realizar sobre el sistema. Esta fase intentará provocar el mayor número de errores en nuestra aplicación, ya que su objetivo primordial es el de detectar posibles fallos en los requisitos, diseño e incluso implementación del sistema.

Para esto lo que intentaremos es realizar un número determinado de pruebas sobre el sistema, que abarque el mayor número posible de caminos dentro del código, o de requisitos. Es decir, si en el código aparecen dos caminos claramente diferenciados por una estructura condicional, debemos intentar que las pruebas analicen ambos caminos, para comprobar el correcto funcionamiento de ambos. Otro tipo de pruebas que se utilizarán son aquellas en las que se busca comprobar el correcto funcionamiento de las interfaces entre los distintos módulos de la aplicación, asegurando una correcta interacción entre los mismos.

El tipo de pruebas que llevaremos a cabo son:

- Pruebas de caja blanca.
- Pruebas de caja negra.

También se ha incluido en esta parte de pruebas, la relacionada con las pruebas en cuanto a la inserción de información en la base de datos haciendo



uso de la aplicación, es decir, de los formularios de entrada. También debería haberse probado que los datos fueran mostrados de forma correcta, sin embargo al no estar disponibles los formularios de visualización no se ha podido llevar a cabo estas pruebas.

### **2.7.1 PRUEBAS DE LA CAJA BLANCA**

En este tipo de pruebas se busca recorrer el mayor número de caminos dentro de la implementación. Como se comentó anteriormente, ante determinados tipos de estructuras, es conveniente probar todas las combinaciones posibles. Por ejemplo, las estructuras condicionales simples, o anidadas, en las que el número de posibilidades a probar sube exponencialmente.

Otro aspecto a tener en cuenta normalmente es el de los bucles, que suelen acaparar gran parte del tiempo de cómputo de las aplicaciones, y sobre los que tendremos que probar principalmente los puntos límite.

Debido a que este módulo en concreto trata sobre una interfaz, la cual está continuamente interaccionando con el usuario, la posibilidad de fallos es muy elevada, ya que es imposible comprobar todas las combinaciones posibles que el usuario podría llevar a cabo al usar el sistema. Por ejemplo, en el caso en el que se pide al usuario que introduzca el usuario y la contraseña, no podemos probar todas las cadenas que dicho usuario podría introducir, por lo que no podemos confirmar que no fallará esta parte del código, sin embargo si podemos restringir esto realizando ciertas comprobaciones en el código.

Estas pruebas han sido llevadas a cabo a la par que se implementaban los distintos módulos que forman la aplicación.

### **2.7.2 PRUEBAS DE LA CAJA NEGRA**

Estas pruebas son las encargadas de comprobar el correcto funcionamiento de las distintas interfaces de los distintos módulos que forma la

aplicación, probando como cada módulo responde a través de sus interfaces, sin fijarnos en como internamente el código genera los valores.

Las pruebas que se han llevado a cabo son:

- Comprobación del correcto funcionamiento de la iniciación de la interfaz.
- Generación correcta de las distintas partes de la interfaz.
- Actuación correcta de los controladores de eventos de la interfaz
  - Menú de herramientas.
  - Árbol de navegación.
  - Menú de botones.
- Actuación correcta del sistema al almacenar como propiedades los datos de conexión de un usuario.
- Actuación correcta del sistema al desconectar de la base de datos, deshabilitando la posibilidad de acceso a las partes del sistema que necesitan de la base de datos.
- Control del correcto funcionamiento de las llamadas a los distintos formularios a través de las funciones que son controlados por los eventos de la interfaz, es decir, formularios de mantenimiento y visualización.
- Actuación correcta de la aplicación al cerrarse, prestando especial atención en que las variables del sistema sean almacenadas para recordar las preferencias del sistema cuando éste se vuelva a arrancar.

Podemos concluir que las pruebas llevadas a cabo han dado como resultado un funcionamiento adecuado del sistema, consiguiendo cumplir todos los requisitos impuestos para el sistema.

## 2.7.3 CARGA DE INFORMACIÓN

En cuanto a la carga de información se ha llevado a cabo una serie de pruebas utilizando la aplicación, para ver si a través de la interfaz se accedía de forma correcta a la base de datos insertando datos.

Lo que se ha realizado es una serie de inserciones, y se ha comprobado. A continuación se muestran las capturas de las distintas inserciones y sus posteriores comprobaciones.

### 2.7.3.1 Inserción de Artículo

En la Figura 2.7.1 se muestra un ejemplo de inserción de un nuevo artículo a través del formulario correspondiente.

The screenshot shows a web-based form for inserting a new article. The form is organized into three main panels. The top panel, 'Datos del Artículo', includes input fields for 'Código', 'Título', 'Año', 'Volumen', 'Página Inicial', 'Página Final', and a dropdown for 'Estado'. The middle panel, 'Referencias del Artículo', contains a 'Revista' section with 'ISSN' and 'Nombre' fields, and a 'Documento' section with a 'Documento Seleccionado' field and a 'Seleccionar Documento' button. The bottom panel, 'Autores', features a table with 'Apellidos' and 'Nombre' columns, and 'Ver' and 'Eliminar' buttons.

**Figura 2.7.1:** Inserción de artículo

La captura correspondiente que muestra que se ha insertado de forma correcta en la base de datos se observa en la Figura 2.7.2.

Ver Detalle    Imprimir Listado

Artículos

**Información de Artículos**

#	Título	Año	Estado	Revista	ISSN
1	Nuevo método evolutivo en desarrollo en la UCO	2000	Aceptado	Computación Evolutiva en SPAIN	ISSN_1
10	Nuevo siste consultas pediátricas	2006	Aceptado	Pediatrics	ISSN_2
11	Chip multiproceso	2006	Sometido	Anales Informáticos	ISSN_3
12	Versión 5 de Java	2005	Aceptado	Anales Informáticos	ISSN_3
2	Conferencia en marzo sobre Genética	2003	Aceptado	Computación Evolutiva en SPAIN	ISSN_1
3	La informática en la radiología	2006	Publicado	Pediatrics	ISSN_2
4	Nuevo siste consultas pediátricas	2006	Aceptado	Pediatrics	ISSN_2
5	Chip multiproceso	2006	Sometido	Anales Informáticos	ISSN_3

Figura 2.7.2: Ver artículos

### 2.7.3.2 Inserción de Libros

En la Figura 2.7.3 se muestra un ejemplo de inserción de un nuevo libro artículo a través del formulario correspondiente.

**Libro** (Campos obligatorios marcados en negrita)

**Datos del Libro**

Código  ISBN  Título

Año  Páginas  Estado  Editorial

Ciudad  País

**Información Adicional del Libro**

**Autores**

Apellidos	Nombre
<input type="text"/>	

Ver Eliminar

**Documento**

Documento Seleccionado

Seleccionar

Insertar    Modificar    Borrar    Cancelar

Figura 2.7.3: Inserción de libro

La captura correspondiente que muestra que se ha insertado de forma correcta en la base de datos se observa en la Figura 2.7.4.

Ver DetalleImprimir Listado

Libros

Información de Libros

#	ISBN	Título	Año	Estado	Editorial	País
1	ISBN-53234-232	Redes de computadores	2000	Publicado	Prentice Hall	FRANCE
10	ISBN-23456-789	La Ingeniería del Software	2005	Publicado	SM	FRANCE
11	ISBN-23456-788	La Ingeniería del Software	2005	Publicado	SM	FRANCE
12	ISBN-23456-799	La Ingeniería del Software	2005	Publicado	SM	FRANCE
2	ISBN-12345-678	Bases de Datos	2003	Publicado	Ra-Ma	SPAIN
3	ISBN-11234-567	UML. Lenguaje Unificado de Modelado	2004	Publicado	ANAYA	SPAIN

Figura 2.7.4: Ver libros

2.7.3.3 Inserción de Capítulos

En la Figura 2.7.5 se muestra un ejemplo de inserción de un nuevo capítulo a través del formulario correspondiente.

Capítulo (Campos obligatorios marcados en negrita)

Datos del Capítulo

Código4

TítuloCapítulo 1 Libro Capítulos 2

Página Inicial220

Página Final228

EstadoAceptado

Referencias del Capítulo

Libro de Capítulos

ISBN2

TítuloLibro de capitulos 2

Ver

Editar

Autores

Apellidos	Nombre

Ver

Eliminar

Documento

Documento Seleccionado

Seleccionar Documento

Figura 2.7.5: Inserción de capítulo

La captura correspondiente que muestra que se ha insertado de forma correcta en la base de datos se observa en la Figura 2.7.6.

#	Título	Estado	Libro
1	Capítulo 1 Libro Capítulos 1	Aceptado	Libro de capitulos 1
4	Capítulo 1 Libro Capítulos 2	Aceptado	Libro de capitulos 2
6	Capítulo 1 Libro Capítulos 3	Aceptado	Libro de capitulos 3
2	Capítulo 2 Libro Capítulos 1	Aceptado	Libro de capitulos 1
5	Capítulo 2 Libro Capítulos 2	Aceptado	Libro de capitulos 2
7	Capítulo 2 Libro Capítulos 3	Aceptado	Libro de capitulos 3
3	Capítulo 3 Libro Capítulos 1	Aceptado	Libro de capitulos 1
9	hola	En revisión	Libro de capitulos 2

Figura 2.7.6: Ver capítulos

232

### 2.7.3.4 Inserción de Actas

En la Figura 2.7.7 se muestra un ejemplo de inserción de una nueva acta a través del formulario correspondiente.

**Actas** (Campos obligatorios marcados en negrita)

**Datos del Acta**

**Código** 3 **Contribución** Contribucion 1 al congreso 2

Volumen 5 **Página Inicial** 456 **Página Final** 500 Estado Publicado

**Referencias del Acta**

**Congresos**

**Código** 2 **Título** Congreso 2 [Ver](#) [Editar](#)

**Autores**

Apellidos	Nombre

[Ver](#) [Eliminar](#)

**Documento**

Documento

[Seleccionar Documento](#)

**Figura 2.7.7:** Inserción de acta

La captura correspondiente que muestra que se ha insertado de forma correcta en la base de datos se observa en las Figuras 2.7.8.

[Ver Detalle](#) [Imprimir Listado](#)

Actas

**Información de Actas**

#	Contribución	Volu...	P. i...	P. Fi...	Estado	Congreso
1	Contribucion 1 al congreso 1	2	200	220	Publicado	Congreso 1
2	Contribucion 2 al congreso 1	3	150	200	Publicado	Congreso 1
3	Contribucion 1 al congreso 2	5	456	500	Publicado	Congreso 2
4	Contribucion 2 al congreso 2	7	190	210	Publicado	Congreso 2
7	hola	3	3	4	Publicado	Congreso 1

**Figura 2.7.8:** Ver actas (Tabla Acta)

### 2.7.3.5 Inserción de Revistas

En la Figura 2.7.9 se muestra un ejemplo de inserción de una nueva revista a través del formulario correspondiente.

**Revista** (Campos obligatorios marcados en negrita)

**Datos de la Revista**

**ISSN** ISSN\_1 **Editorial** Editorial Cúspide **Abreviatura** ECCEE

**Nombre** Computación Evolutiva en España

---

**ISI (Institute for Scientific Information)**

Año 2000 Factor de Impacto 1.77 Total citas 32000

Índice inmediato 0.453 Número de Artículos 2000 Vida media 2.3

InsertarISI VerISI BorrarISI ModificarISI

Figura 2.7.9: Inserción de revista

La captura correspondiente que muestra que se ha insertado de forma correcta en la base de datos se observa en la Figura 2.7.10.

Issn	Editorial	Abreviatura	Nombre
ISSN_1	Editorial Cúspide	ECCEE	Computación Evolutiva en España
ISSN_3	Grupo POSIT	GPAI	Anales Informáticos
54654	hola	hola	hgolaaa
ISSN_2	Publicaciones META	PMP	Pediatrics

Figura 2.7.10: Ver revistas

### 2.7.3.6 Inserción de Autores

En la Figura 2.7.11 se muestra un ejemplo de inserción de un nuevo autor a través del formulario correspondiente.

**Datos Personales**

**Apellidos** Martínez Pérez **Nombre** Fernando **Tratamiento** Ldo.

---

**Información Laboral**

**Categoría** Jefe **Universidad** Córdoba **Departamento** Análisis Numerico

---

**Ubicación**

**Dirección** Calle Fernando **Ciudad** Córdoba

**Provincia** Córdoba **País** ---- **Código Postal** 705

---

**Contacto**

**E-mail** i12femape@uco.es **Telefono** 957777777 **Fax** 9957777777

Figura 2.7.4: Inserción de autor

La captura correspondiente que muestra que se ha insertado de forma correcta en la base de datos se observa en la Figuras 2.7.12 y 2.7.13.

Apellido	Nombre	Categoría	Universidad	Departamento	Email
Catalá Carbonero	María	Ayudante	Córdoba	Análisis Numerico	i12cacam@uco.es
Hidalgo Serrano	Cristobal	Ayudante	Córdoba	Matemáticas y Matemática	i12crhise@uco.es
Muriel Zafra	Javi	Ayudante	Córdoba	Análisis Numerico	i12muzaf@uco.es
Lozano Rodríguez	Esteban	Director	Córdoba	Análisis Numerico	i12esloro@uco.es
Martínez Pérez	Fernando	Jefe	Córdoba	Análisis Numerico	i12femape@uco.es
Pascual Ruíz	Gema	Titulada	Córdoba	Historia de la Informática	i12geparu@uco.es

**Figura 2.7.5:** Ver autor (Tabla personalInformation)

Apellidos	Nombre
Catalá Carbonero	María
Lozano Rodríguez	Esteban
Martínez Pérez	Fernando
Mejías Real	Javier
Muriel Zafra	Javi
Pascual Ruíz	Gema

Aceptar

**Figura 2.7.6:** Ver autor (Tabla author)

### 2.7.3.7 Inserción de Congresos

En la Figura 2.7.14 se muestra un ejemplo de inserción de un nuevo congreso a través del formulario correspondiente.

**Congreso** (Campos obligatorios marcados en negrita)

**Datos del Congreso**

**Código** 2 **Nombre** Congreso 2

Abreviatura C2 Organización Organización 2 Tipo I Estado ----

Ciudad Granada País ---- Fecha Inicio 2006-02-20

Fecha Final 2006-02-23 Deadline Marzo 2006 Intención intencion 2

**Información Adicional del Congreso**

ISBN ISBN-77777-77777 Título Acta Congreso Extraordinario 2

URL http://www.congreso2.es Editorial Editorial 2 Editores Editores 2

**Figura 2.7.14:** Inserción de un congreso

La captura correspondiente que muestra que se ha insertado de forma correcta en la base de datos se observa en la Figura 2.7.15.



#	Abreviatura	Organización	Tipo	País	Fecha límite	Título acta
2	C2	Organizacion 2	I	España	Marzo 2006	Congreso Extraordinario 2
1	C1	Organizacion 1	N	España	Octubre 2006	Congreso Extraordinario 1

**Figura 2.7.15: Ver revistas**

# 3. Formulario de Entrada

# 3.1

## DEFINICIÓN DEL PROBLEMA

---

El problema que se plantea consiste en la construcción de una herramienta sencilla que permita gestionar la información perteneciente a un grupo de investigación.

El objetivo es la creación de una herramienta haciendo uso de software libre y que conlleve un mantenimiento que pueda ser llevado por personal no especializado

El sistema deberá manejar y mantener toda la información correspondiente a: información general del grupo de investigación, sus miembros o integrantes, publicaciones del grupo (artículos, libros, capítulos de libros), así como la información relacionada con la misma (revistas científicas, congresos, etc.).

Esta información deberá incluir, al menos, la información requerida a los investigadores por los organismos oficiales, en los procesos de remisión de información científica.

El sistema deberá permitir de forma simple la inserción y manipulación (visualización, modificación y borrado) de la información gestionada.

El sistema en cuestión será desarrollado en 4 fases:

- Fase 1: Construcción del prototipo.
- Fase2: Revisión y prueba del prototipo.

- Fase 3: Completitud del sistema.
- Fase 4: Prueba del sistema y depuración.
- Fase 5: Documentación y entrega del sistema.

Cada fase será realizada por 5 grupos. Cada uno de ellos realizara una tarea distinta. En el presente documento se describe la tarea de modificación de los formularios de entrada correspondiente a la Fase2.

### **3.1.1 PROBLEMA REAL.**

Desde el punto de vista real, el problema que se plantea es simple, ya que es la realización de una interfaz con el usuario para la inserción y manipulación de la información correspondiente a grupos de investigación. Esta interfaz debe ser fácil de usar e intuitiva, ya que el objetivo primordial buscado como se ha comentado anteriormente es que los miembros del grupo que utilicen el sistema no tengan dificultad a la hora de utilizar el sistema, ya que estos no tienen porque estar relacionados con el mundo de la informática.

### **3.1.2 PROBLEMA TÉCNICO.**

Desde el punto de vista técnico, el problema engloba la realización de una estructura de formularios que se utilizarán a modo de interfaz con el usuario para la inserción y manipulación de datos. Para su realización se tendrán que tener en cuenta las restricciones que aparecen en la base de datos, así como restricciones impuestas por el cliente.

El desarrollo de la solución consistirá en la construcción de aplicación visual, ya que lo que buscamos es facilidad de uso. Esta aplicación se compondrá de una serie de ventanas que nos permitirán la inserción, modificación y borrado de información en las distintas partes de nuestro sistema, de forma que se distingan claramente las partes a las que se puede acceder de forma directa, y aquéllas que tienen una existencia dependiente de otras. Este caso se explicará más adelante.

# 3.2

## OBJETIVOS

---

El objetivo de la tarea es la modificación de los formularios de entrada que se desarrollaron en la *Fase 1*. Para conseguir dicho propósito, se han de cubrir las siguientes metas u objetivos particulares:

- Se modificarán los métodos necesarios para el acceso a la base de datos.
- Modificación de las opciones de la interfaz gráfica para que se adapte a los requerimientos especificados.
- Añadir todas aquellas opciones nuevas que se hayan considerado y que no están incluidas en la fase anterior.

# 3.3

## RESTRICCIONES TÉCNICAS Y DE GESTIÓN

A continuación se detallan en primer lugar los factores dato que vienen impuestos por el contexto de desarrollo del sistema y por sus capacidades. En segundo lugar se especifican los factores estratégicos cuya elección está justificada y motivada en función del beneficio que puedan aportar al desarrollo del proyecto, y que, por supuesto, condicional el resultado obtenido.

### 3.3.1 FACTORES DATO

Se consideran los siguientes:

- El sistema debe presentar una interfaz de usuario homogénea e intuitiva. Desde la interfaz será accesible toda la funcionalidad del sistema.
- El sistema deberá ser capaz de manejar toda la información correspondiente al grupo de investigación.
- Debido a la naturaleza multiplataforma del sistema, y a la modularidad exigida, el desarrollo del sistema debe realizarse en el lenguaje de programación Java. Este lenguaje proporciona un paradigma de programación orientado a objetos. Tiene un API que proporciona una fuente de recursos lo suficientemente rica y

en continua expansión para cubrir todas las necesidades de la fase de codificación del sistema.

### 3.3.2 FACTORES ESTRATÉGICOS

Se consideran los siguientes:

- Se utilizará la tecnología que MySQL 5.0 proporciona para el desarrollo de la aplicación. A su vez, para lograr la interacción entre el sistema y la base de datos se utilizará JDBC, como puente de conexión entre ambos.
- Para optimizar la fase de codificación se utilizará el entorno gráfico de desarrollo Eclipse-SDK 3.2.1 para Java. Un entorno integral de este tipo mejora notablemente la codificación de los distintos componentes del sistema, ya que proporciona: herramientas de gestión de la configuración, diseñadores de entornos gráficos de usuario, plantillas de componentes reutilizables, configuración del acceso a librerías de clases de terceros y sistemas de depuración de código, entre otras muchas utilidades.
- Durante la fase de especificación de requisitos las técnicas gráficas de especificación y modelado se realizarán en el lenguaje UML. Este lenguaje proporciona un métodos unificado de modelado del sistema a todos los niveles necesarios: estructural, funcional, de flujo de control de actividades, de implementación, de despliegue etc. UML no impone un método rígido de análisis y diseño sino que aporta un conjunto de símbolos (sintaxis) y un conjunto de reglas (semántica) que permiten representar un sistema desde diferentes perspectivas.

# 3.4

## RECURSOS

---

En este apartado enumeramos los distintos recursos necesarios para la realización del proyecto agrupándolos en las categorías especificadas en los puntos siguientes:

### 3.4.1 RECURSOS HUMANOS.

La tarea de *Modificación del los Formularios de Entrada*, correspondiente a la *Fase 2* de desarrollo del sistema, será realizada por los alumnos, pertenecientes al *Grupo 14*:

- Francisco Manuel Borrego Jaraba.
- M<sup>a</sup> Ángeles Redondo Pachón.

Bajos las pautas marcadas por los jefes de proyecto pertenecientes al grupo de investigación de Ingeniería del Software, Conocimiento y Bases de Datos (**ISCBD**) de la Universidad de Córdoba:

- Dña. Irene Luque Ruiz, Titular de Universidad.
- Miguel Ángel Gómez-Nieto, Catedrático E.U.



### 3.4.2 RECURSOS HARDWARE.

Los recursos hardware utilizados para el desarrollo del módulo del sistema descrito en el presente documento son los siguientes:

- Un PC Pentium IV (2.4GHz, 512 Mb RAM y 60Gb de Hd).
- Un Equipo portátil Pentium Centrino Dual Core (1.7GHz, 1GB RAM y 120GB de disco duro).
- Un Equipo portátil Pentium Dual Core (1.6 GHz, 1 GB de RAM y 120 GB de disco duro).

### 3.4.3 RECURSOS SOFTWARE.

Para la realización de los diversos apartados del sistema se dispondrá de los siguientes recursos software:

- **Documentación:** Para la documentación del sistema se empleará el paquete ofimático **Microsoft Office 2003**, que incluye el procesador de textos **Word 2003**. También haremos uso de otras herramientas como puede ser **Microsoft Visio 2000 SR1**, que nos facilitará la elaboración de representaciones gráficas diagramáticas.
- **Desarrollo:** Para el desarrollo del proyecto se contará con el siguiente software:
  - **Eclipse SDK-3.2.1:** Entorno de programación de código Java. Utilizador para generar la interfaz y las principales aplicaciones del proyecto.
  - **JDK (1.4.2):** Kit de desarrollo de Java.
  - Un servidor de Base de Datos **MySQL 5.0**, ejecutado de forma local y **MySQLClient**.

# 3.5

## ESPECIFICACIÓN DE REQUISITOS

---

### 3.5.1 IDENTIFICACIÓN DE REQUISITOS

En este apartado se especifican los requisitos que ha de cumplir el sistema.

#### *3.5.1.1 Requisitos generales*

A continuación se detallarán los requisitos principales que tendrá que cumplir el proyecto.

Los campos que sean de obligado cumplimiento deben ser señalados explícitamente en el formulario, para que el usuario se percate de su obligatoriedad, para ello se les pondrá a los cuadro de inserción de datos de dichos atributos un borde negro de mayor grosor.

Los valores de los atributos que sean una lista enumerada deben aparecer como una lista elegible. Es el caso de los campos Estado o País, por ejemplo.

En aquellos campos que consistan en un documento o una imagen, se utilizará un botón **Seleccionar** que permita elegir entre los ficheros del sistema.

### 3.5.1.2 Requisitos del Formulario Miembros

En el formulario de entrada de la información de los “miembros” deben aparecer los siguientes botones: **Grabar**, cuya función será grabar los datos en la base de datos, **Cancelar**, cuya función es anular el trabajo realizado limpiando los campos, y **Borrar**, cuya función será borrar los datos de la tupla correspondiente en la base de datos.

Dentro del mantenimiento de la información de los miembros se presentan tres casos distintos. El primero que se quiera insertar un miembro nuevo, el segundo que se quiera modificar uno existente y el tercero que se quiera borrar un miembro.

Se ha decidido juntar el primer y segundo caso, de forma que cuando se inserte un miembro ya existente se le pregunte al usuario si lo que quiere es modificar los datos almacenados.

Se ha decidido realizar de esta forma porque que el usuario final lo hará de forma intuitiva, ya que cuando quiera modificar un miembro seleccionará en el formulario de visualización de miembros uno existente y se cargarán de forma automática los datos del miembro en el formulario correspondiente de entrada, sólo teniendo que modificar dichos datos.

Cuando se presione el botón de insertar se pueden presentar los siguientes casos:

- El miembro a insertar ya existe por lo cual como se ha comentado anteriormente se pregunta al usuario si quiere modificar los datos.
- Faltan datos obligatorios, por lo cual debe avisarse al usuario de que deben introducirse dichos datos, resaltándolos en color rojo los bordes de los cuadros de dichos campos.
- Se inserta de forma correcta el miembro.

Cuando se presiona el botón cancelar simplemente se deben borrar los valores existentes en los cuadros de texto de todos los atributos.

Cuando se presione el botón borrar se presentan dos casos simples:

- Si se encontraba algún dato señalado en el formulario de visualización, y por tanto cargados los datos en el formulario de entrada de datos, se procederá a su borrado de la base de datos en el caso de que exista.
- Si no hay ningún elemento seleccionado no se llevará a cabo ninguna acción.

Siempre que no se describan las actuaciones de los botones en los siguientes casos, se supondrá que su tratamiento es el mismo al citado en este apartado.

### 3.5.1.3 Requisitos del Formulario Artículos

En el formulario de mantenimiento de información de los “artículos” deben los siguientes botones: **Grabar**, cuya función será grabar los datos en la base de datos, **Cancelar**, cuya función es anular el trabajo realizado mediante el borrado de los campos, y **Borrar**, cuya función será borrar los datos de la tupla correspondiente en la base de datos.

En todos aquellos en los que estén presentes atributos que hagan referencia a otras tablas de información, estarán presentes dos botones acompañando al campo correspondiente que haga referencia a dicha tabla. En el formulario de los “artículos” habrá dos campos de este tipo: “autores”, que referencia a los autores que han realizado el artículo y “ISSN”.

Los botones asociados a estos campos serán **Visualizar** y **Editar**:

- **Visualizar**: presentará un formulario en el que se presentarán las distintas tuplas de la tabla referenciada permitiendo al usuario seleccionar alguna de las tuplas, cuyo valor pasará a actualizar el valor del campo en edición. Este formulario contendrá algunos de los valores de los atributos de la tabla referenciada.
- **Editar**: se presentará una página con la funcionalidad de inserción de un nuevo elemento para la tabla que contiene el

campo en edición. Este nuevo formulario tendrá la estructura igual a los formularios de inserción de miembros, por ejemplo.

La funcionalidad definida para el formulario “artículos” es la misma que para “libros”, “capítulos” y “actas”, ya que todos ellos tienen campos que referencian a otras tablas. Las ventanas en cada caso tendrán la misma apariencia, funcionalidad y significado.

#### **3.5.1.4 Requisitos del Formulario Revistas**

El tratamiento de las revistas, como cualquier “tabla hija”, está supeditado al tratamiento de la tabla principal en la que la revista es uno de sus atributos, y será invocado a través de la funcionalidad **Visualizar** y **Editar**.

De esta forma se mantiene la coherencia de que las revistas tienen una existencia supeditada a la existencia de un artículo relacionado con el grupo de investigación, y no una existencia principal.

En el caso de la acción de **Visualizar**, se presentará formulario “hijo” con la relación de las revistas existentes. El usuario podrá seleccionar alguna de estas revistas y el valor de la misma actualizará el formulario principal por el cual esta funcionalidad fue invocada.

Una vez seleccionada la revista, de forma automática, la ventana correspondiente será cerrada. Esta ventana, no permitirá navegación alguna.

En el caso de **Editar**, se presentará una ventana similar a la del tratamiento de las tablas principales, aunque con menor funcionalidad. Esta ventana, no permitirá navegación y será cerrada por el usuario o una vez el usuario haya transferido la información de la revista al formulario principal por el cual fue invocada esta ventana.

La funcionalidad definida para el formulario “revistas” es la misma que para “libros de capítulos” y “conferencias”, ya que todos ellos corresponden a “tablas hijas”. Las ventanas en cada caso tendrán la misma apariencia, funcionalidad y significado.

### 3.5.1.5 Requisitos del Formulario Autores

El tratamiento de los autores es similar al descrito para “**revistas**”, siendo invocado a través de la funcionalidad **Visualizar** y **Editar** desde los formularios principales.

De esta forma se mantiene la coherencia de que los autores tienen una existencia supeditada a la existencia de una publicación relacionada con el grupo de investigación, y no una existencia principal.

En el caso de la acción de **Visualizar**, se presentará un formulario “hijo” con la relación de los autores existentes. El usuario podrá seleccionar alguno de estos autores y su valor actualizará el formulario principal por el cual esta funcionalidad fue invocada.

Dado que una publicación puede tener más de un autor, esta ventana no se cerrará de forma automática una vez el autor sea seleccionado. Se cerrará cuando: a) a petición del usuario, b) cuando se cierre el formulario por el cual fue invocada. Esta ventana, no permitirá navegación alguna.

En el caso de **Editar**, el tratamiento es también el mismo que el descrito para “revistas”, aunque en el área de visualización e incluirá una funcionalidad añadida: **Visualizar Miembros**. Esta funcionalidad dará lugar a una nueva ventana con la misma característica que Visualizar-Autores mediante la cual se podrá seleccionar a un miembro del grupo de investigación de forma que sus datos sean automáticamente transferidos al formulario de edición de los autores.

La ventana de visualización de los miembros del grupo de investigación será cerrada una vez se haya seleccionado algún elemento o por decisión del usuario.

## 3.5.2 DESCRIPCIÓN DE LA VISTA DE USUARIO

La vista del usuario modela el sistema desde la perspectiva de los usuarios finales del mismo (actores). Por ello, el objetivo de esta fase de especificación es dotar al sistema de la funcionalidad necesaria para llevar a

cabo los requisitos propuestos. La especificación de este modelo funcional se realizará mediante el uso de los diagramas de casos de uso; una de las técnicas de modelado de UML. Un caso de uso representa un requisito básico esencial (comportamiento deseado) del sistema sin especificar su implementación. Los casos de uso pueden desarrollarse de forma jerárquica, aumentando el nivel de refinamiento a medida que se desciende por la jerarquía. Un caso de uso se compone de una descripción de un conjunto de secuencias de acciones que ejecuta el sistema para producir un resultado observable de valor para un actor. Un escenario de un caso de uso determina una secuencia específica de acciones (flujo de control) que ilustran un comportamiento del caso de uso. Cada escenario es una instancia del caso de uso. Los objetos del modelo de clase y del modelo funcional colaborarán para llevar a cabo el comportamiento del caso de uso. (de Booch, 1999)

Un diagrama de casos de uso muestra un conjunto de casos de uso, actores y sus relaciones. Se emplean para modelar la vista estática de casos de uso del sistema (los servicios visibles externamente que proporciona el sistema en el contexto de su entorno). (de Booch, 1999)

### 3.5.2.1 Actores

Un actor representa un rol (papel) que es jugado por un usuario final, un dispositivo hardware o incluso otro sistema al interactuar con el sistema en desarrollo. Un actor solo se puede conectar a un caso de uso a través de una asociación, que indica que el actor y el caso de uso se comunican entre sí a través del envío y recepción de mensajes.

Dentro del dominio del problema tratado se identifican los siguientes actores:

**Tabla 3.5.1:** Descripción Actor 1

ACTOR: Usuario	
Identificador	1
Id – Padre	
Descripción	Representa a una persona que interactúa con el sistema a través de su

	interfaz gráfica de usuario.
--	------------------------------

Tabla 3.5.2: Descripción Actor 2

ACTOR: Sistema	
Identificador	2
Id – Padre	
Descripción	Representa al propio sistema software en desarrollo que interactúa con todos aquellos objetos de procesamiento interno que dan soporte a los requisitos funcionales y que no son accesibles por el resto de actores.

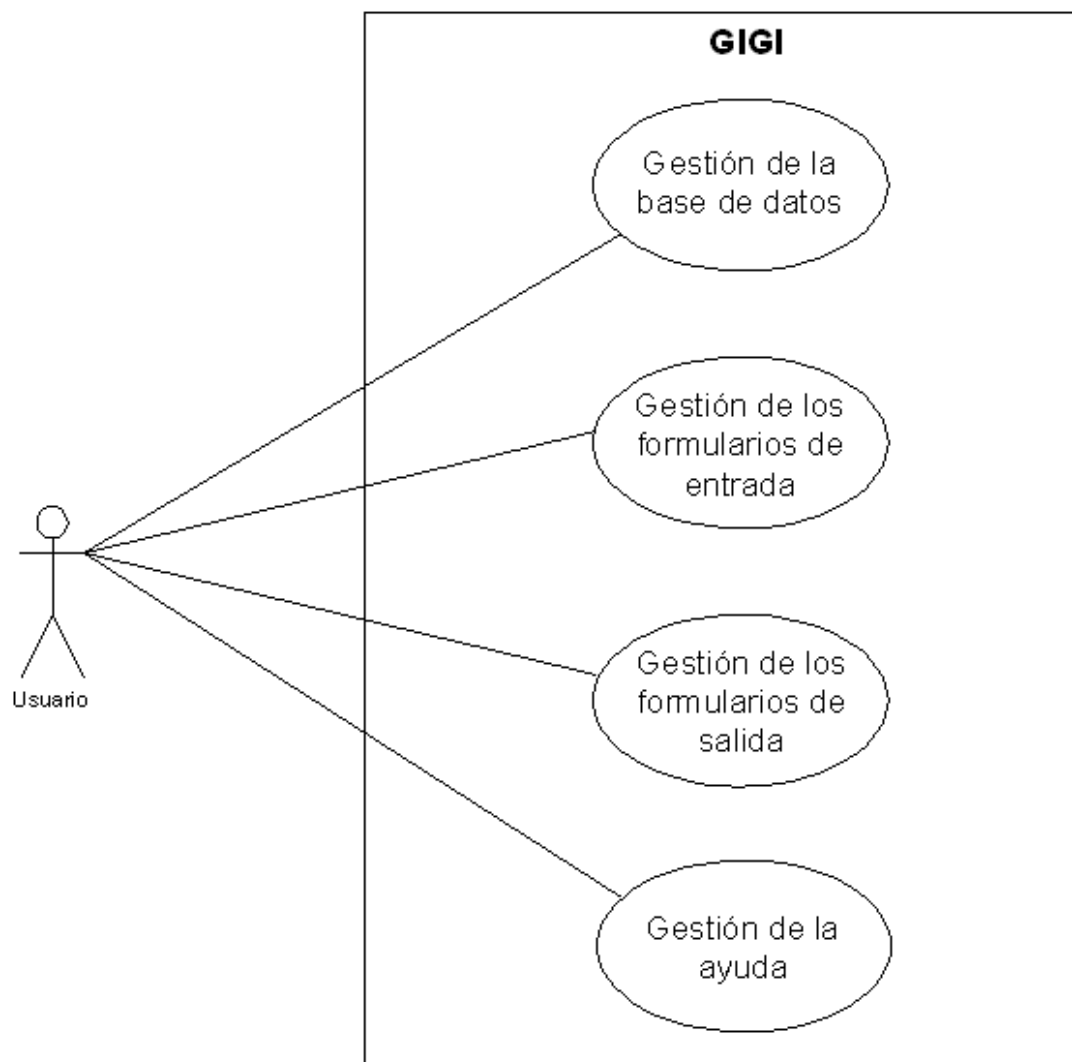
### 3.5.2.2 Diagrama de casos de uso: Contexto del sistema

NOMBRE	<b>Contexto del sistema</b>
DESCRIPCIÓN	La funcionalidad del sistema queda representada por sus elementos internos responsables de llevar a cabo el comportamiento que esperan los elementos externos. Estos elementos externos que interactúan con el sistema constituyen su contexto. El entorno en el que reside el sistema queda definido a través del contexto de interacción.
ACTORES	Usuario.
CASOS DE USO	<p><b>Gestión de la base de datos.</b> Proporciona soporte a las operaciones básicas que el usuario puede realizar sobre la base de datos para la manipulación de la información almacenada en la misma.</p> <p><b>Gestión de los formularios de entrada.</b> Permite al usuario añadir nueva información acerca del grupo de investigación.</p> <p><b>Gestión de los formularios de salida.</b> Permite al usuario obtener información acerca del grupo de investigación.</p> <p><b>Gestión de la ayuda.</b> Permite al usuario obtener información acerca del manejo de la herramienta.</p>
FLUJO DE EVENTOS PRINCIPAL	<p>Pasos de ejecución del camino básico del caso de uso:</p> <ol style="list-style-type: none"> <li>1. El usuario procede a la inserción de la información referente al grupo de investigación.</li> <li>2. Una vez insertada la información, podrá visualizarla</li> </ol>



	fácilmente a través de una serie de formularios.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , sino hay ninguna información almacenada del grupo de investigación no podrá visualizarla.

En la *Figura 3.5.1* se puede observar la proyección del contenido gráfico del diagrama de casos de uso *Contexto del Sistema*.



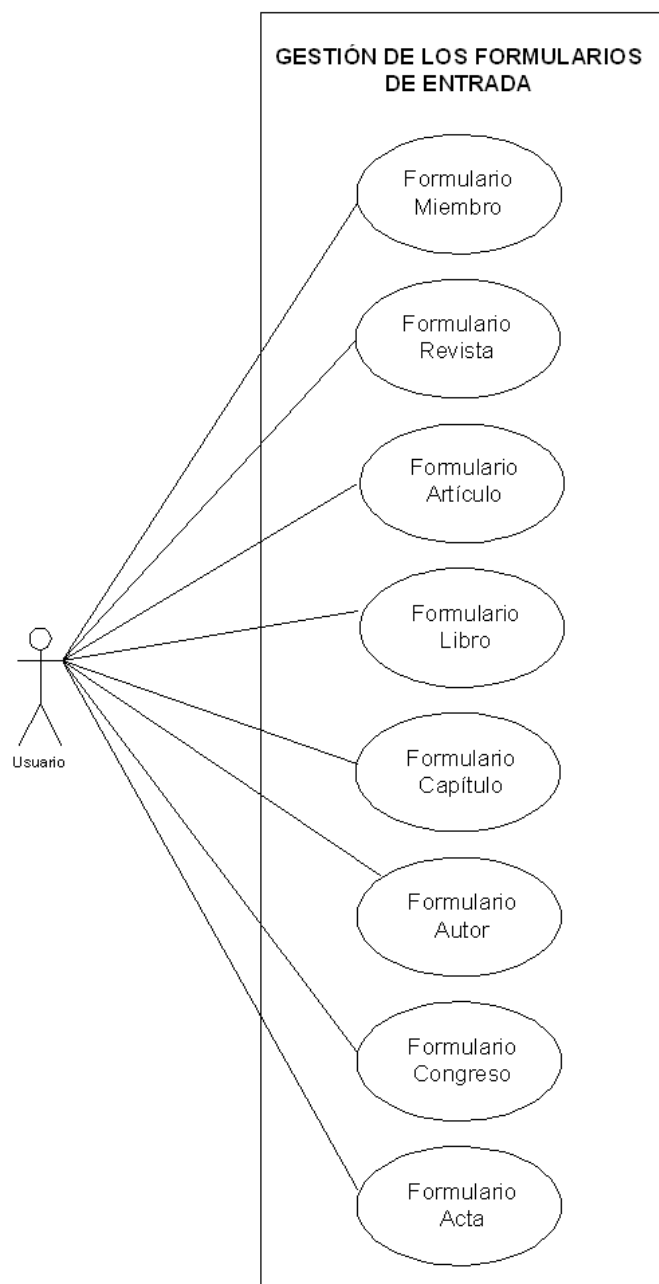
**Figura 3.5.1:** Diagrama de Casos de Uso: Contexto del Sistema

### 3.5.2.3 Diagrama de casos de uso: Gestión de los formularios de entrada

NOMBRE	<b>Gestión de los formularios de entrada</b>
DESCRIPCIÓN	Este caso de uso se encarga de gestionar los formularios de entrada, es decir, aquellos formularios que se utilizan para la manipulación de la información del grupo de investigación.
ACTORES	Usuario.
CASOS DE USO	<b>Formulario Miembro.</b> Proporciona soporte a las operaciones de manipulación de los miembros del grupo. <b>Formulario Revista.</b> Proporciona soporte a las

	<p>operaciones de manipulación de las revistas relacionadas con el grupo.</p> <p><b>Formulario Artículo.</b> Proporciona soporte a las operaciones de manipulación de los artículos de las revistas.</p> <p><b>Formulario Libro.</b> Proporciona soporte a las operaciones de manipulación de los libros relacionados con el grupo.</p> <p><b>Formulario Capítulo.</b> Proporciona soporte a las operaciones de manipulación de los capítulos de los libros.</p> <p><b>Formulario Congreso.</b> Proporciona soporte a las operaciones de manipulación de los congresos relacionados con el grupo.</p> <p><b>Formulario Actas.</b> Proporciona soporte a las operaciones de manipulación de las actas proporcionadas por los congresos.</p> <p><b>Formulario Autor.</b> Proporciona soporte a las operaciones de manipulación de los autores de cualquiera de las publicaciones (libros, revistas, etc.)</p>
FLUJO DE EVENTOS PRINCIPAL	<p>Pasos de ejecución del camino básico del caso de uso:</p> <ol style="list-style-type: none"> <li>1. El usuario selecciona cualquiera de los formularios para manipular la información del grupo de investigación.</li> </ol>

En la *Figura 3.5.2* se puede observar la proyección del contenido gráfico del diagrama de casos de uso *Gestión de los formularios de entrada*.



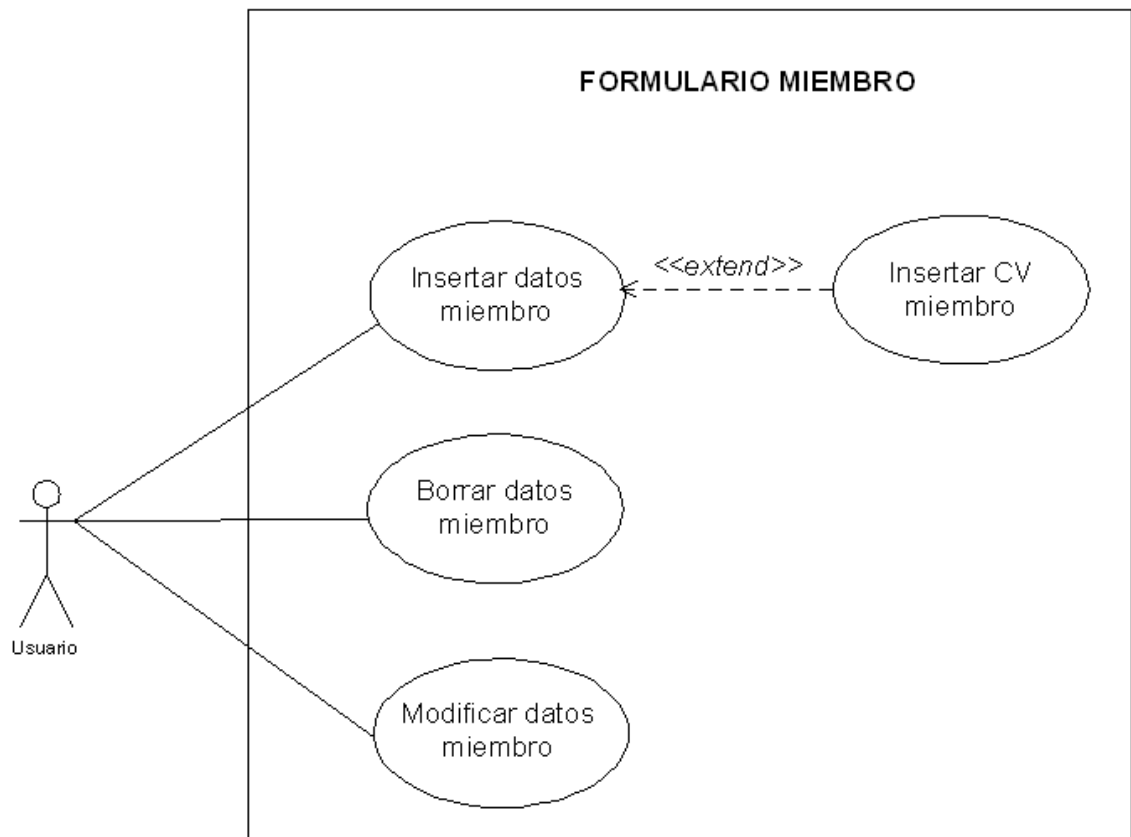
**Figura 3.5.2:** Diagrama de Casos de Uso: Gestión de los Formularios de Entrada

#### 3.5.2.4 Diagrama de casos de uso: *Formulario Miembro*

NOMBRE	<b><i>Formulario Miembro</i></b>
DESCRIPCIÓN	Mediante este caso de uso se manipula la información relativa a los miembros integrantes del grupo de investigación.

ACTORES	Usuario.
CASOS DE USO	<p><b>Insertar datos miembro.</b> Permite al usuario introducir los datos de un nuevo miembro e insertarlos en la base de datos.</p> <p><b>Insertar CV del miembro.</b> Permite al usuario añadir el currículum vital del miembro.</p> <p><b>Borrar datos miembro.</b> Permite al usuario eliminar los datos de un miembro, es decir, eliminar un miembro del grupo.</p> <p><b>Modificar datos miembro.</b> Permite al usuario modificar en cualquier momento, información acerca de un determinado miembro.</p>
FLUJO DE EVENTOS PRINCIPAL	<p>Pasos de ejecución del camino básico del caso de uso:</p> <ol style="list-style-type: none"> <li>1. El usuario selecciona el formulario de entrada de miembros.</li> <li>2. Una vez seleccionado el formulario, procederá a realizar alguna de las operaciones de manipulación establecidas: insertar, modificar y borrar.</li> </ol>
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , sino hay ningún miembro del grupo almacenado en la base de datos no se podrá ni modificar ni borrar.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario no podrá insertar un miembro del grupo que ya exista en la base de datos.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario deberá insertar toda aquella información que se considera necesaria para un miembro y que se le indicará a través del formulario.
FLUJO DE EVENTOS EXEPCIONAL	En cualquier paso, el usuario puede detenerse sin llegar a realizar ninguna operación de manipulación.

En la *Figura 3.5.3* se puede observar la proyección del contenido gráfico del diagrama de casos de uso *Formulario Miembro*.



**Figura 3.5.3:** Diagrama de Casos de Uso: Formulario Miembro

### 3.5.2.5 Diagrama de casos de uso: Formulario Revista

NOMBRE	<b>Formulario Revista</b>
DESCRIPCIÓN	Mediante este caso de uso se manipula la información relativa a las revistas relacionadas con el grupo de investigación.
ACTORES	Usuario.
CASOS DE USO	<p><b>Insertar nueva revista.</b> Permite al usuario introducir los datos de una nueva revista e insertarlos en la base de datos.</p> <p><b>Añadir artículos.</b> Permite al usuario añadir artículos a la revista ya que una revista no tiene existencia sin uno o más artículos publicados en la misma.</p> <p><b>Borrar revista.</b> Permite al usuario eliminar los datos de una revista.</p> <p><b>Modificar revista.</b> Permite al usuario modificar en cualquier momento, información acerca de una determinada revista.</p>

FLUJO DE EVENTOS PRINCIPAL	<p>Pasos de ejecución del camino básico del caso de uso:</p> <ol style="list-style-type: none"> <li>1. El usuario selecciona el formulario de entrada de revistas.</li> <li>2. Una vez seleccionado el formulario, procederá a realizar alguna de las operaciones de manipulación establecidas: insertar, modificar y borrar.</li> <li>3. Si se selecciona la opción de insertar nueva revista, tendrá que añadir artículos a la revista para que esta tenga existencia: relación de inclusión.</li> </ol>
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , sino hay ninguna revista almacenada en la base de datos no se podrá ni modificar ni borrar.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario no podrá insertar una revista que ya exista en la base de datos.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario deberá insertar toda aquella información que se considera necesaria para una revista y que se le indicará a través del formulario.
FLUJO DE EVENTOS EXEPCIONAL	En cualquier paso, el usuario puede detenerse sin llegar a realizar ninguna operación de manipulación.

En la *Figura 3.5.4* se puede observar la proyección del contenido gráfico del diagrama de casos de uso *Formulario Revista*.

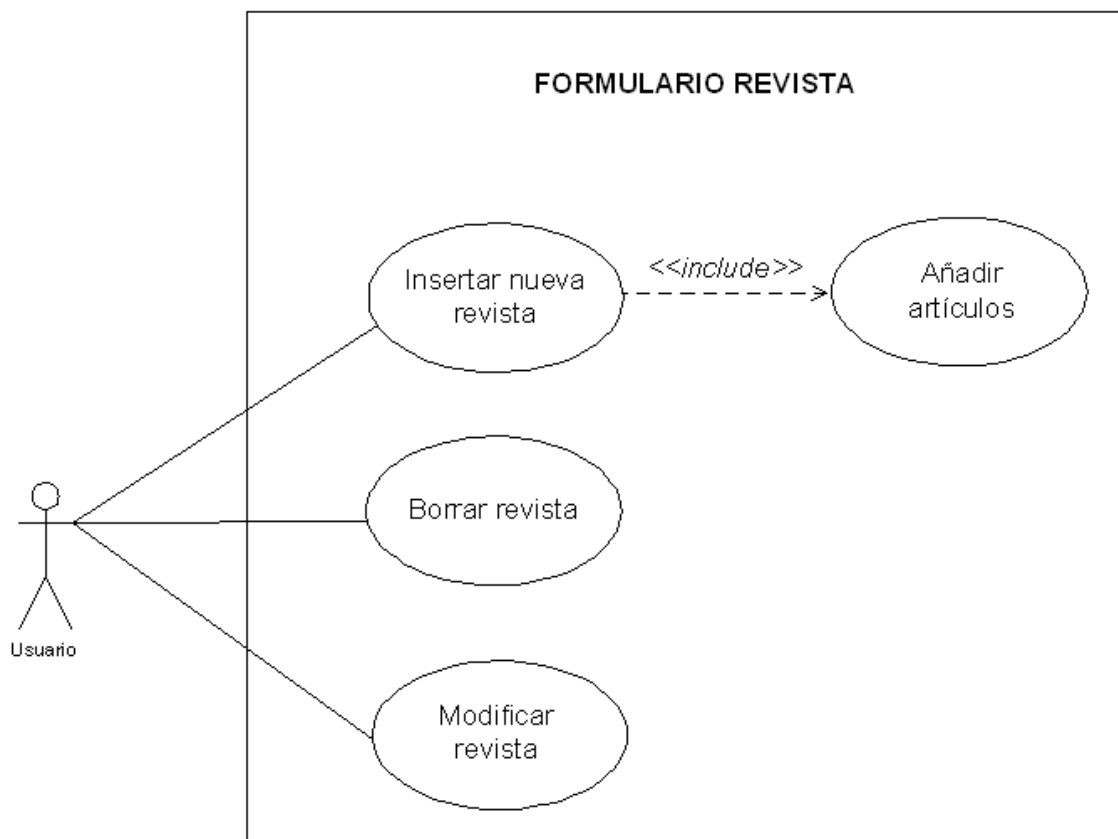


Figura 3.5.4: Diagrama de Casos de Uso: Formulario Revista

### 3.5.2.6 Diagrama de casos de uso: Formulario Artículo

NOMBRE	<b>Formulario Artículo</b>
DESCRIPCIÓN	Mediante este caso de uso se manipula la información relativa a los artículos publicados en cualquiera de las revistas.
ACTORES	Usuario.
CASOS DE USO	<p><b>Insertar nuevo artículo.</b> Permite al usuario introducir los datos de un nuevo artículo e insertarlos en la base de datos.</p> <p><b>Añadir revista.</b> Permite al usuario indicar la revista en la cual se ha publicado el artículo.</p> <p><b>Añadir autores.</b> Permite al usuario añadir los autores del artículo.</p> <p><b>Borrar artículo.</b> Permite al usuario eliminar los datos de</p>



	<p>un artículo.</p> <p><b>Modificar artículo.</b> Permite al usuario modificar en cualquier momento, información acerca de uno determinado artículo.</p>
FLUJO DE EVENTOS PRINCIPAL	<p>Pasos de ejecución del camino básico del caso de uso:</p> <ol style="list-style-type: none"> <li>1. El usuario selecciona el formulario de entrada de artículos.</li> <li>2. Una vez seleccionado el formulario, procederá a realizar alguna de las operaciones de manipulación establecidas: insertar, modificar y borrar.</li> <li>3. Si se selecciona la opción de insertar nuevo artículo, tendrá que añadir información de la revista en la que se ha publicado el artículo y los autores del artículo: relación de inclusión.</li> </ol>
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , sino hay ningún artículo almacenado en la base de datos no se podrá ni modificar ni borrar.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario no podrá insertar un artículo que ya exista en la base de datos.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario deberá insertar toda aquella información que se considera necesaria para un artículo y que se le indicará a través del formulario.
FLUJO DE EVENTOS EXEPCIONAL	En cualquier paso, el usuario puede detenerse sin llegar a realizar ninguna operación de manipulación.

En la *Figura 3.5.5* se puede observar la proyección del contenido gráfico del diagrama de casos de uso *Formulario Artículo*.

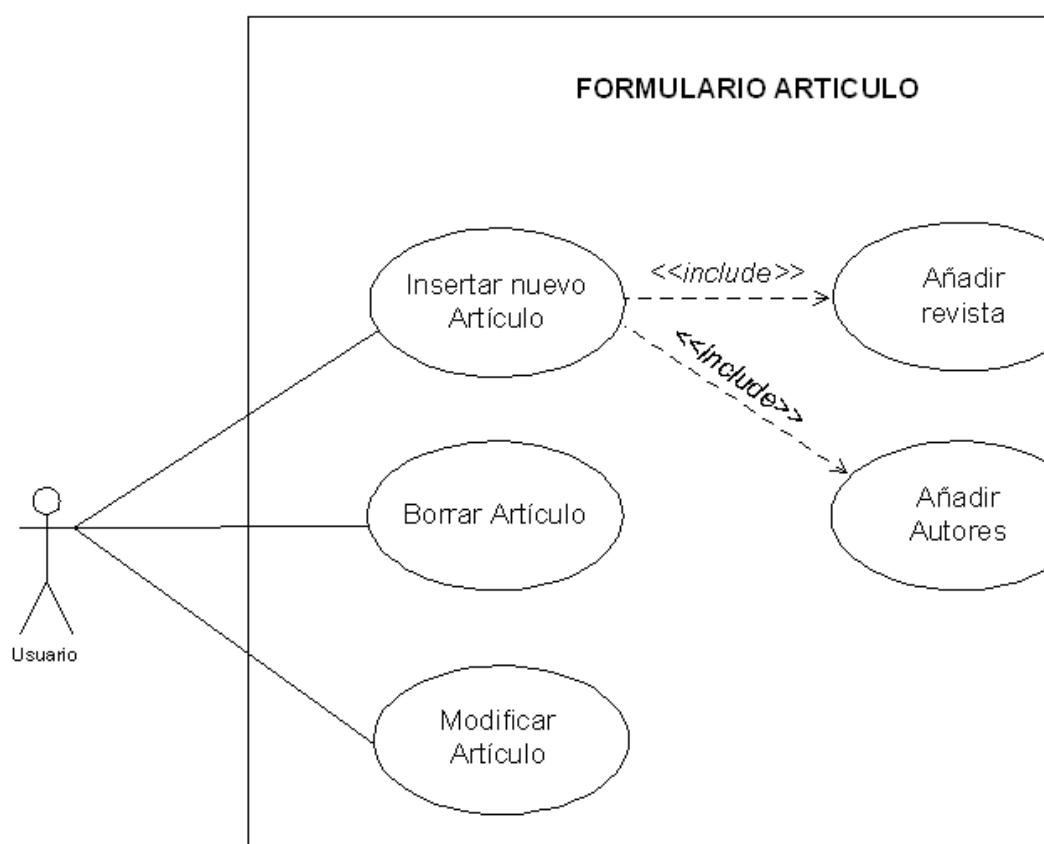


Figura 3.5.5: Diagrama de Casos de Uso: Formulario Artículo

### 3.5.2.7 Diagrama de casos de uso: Formulario Libro

NOMBRE	<b>Formulario Libro</b>
DESCRIPCIÓN	Mediante este caso de uso se manipula la información relativa a los libros relacionados con el grupo de investigación.
ACTORES	Usuario.
CASOS DE USO	<p><b>Insertar nuevo libro.</b> Permite al usuario introducir los datos de un nuevo libro e insertarlos en la base de datos.</p> <p><b>Añadir capítulos.</b> Permite al usuario indicar los capítulos que componen el libro y sin los que no tendría existencia el mismo.</p> <p><b>Añadir autores.</b> Permite al usuario añadir los autores del libro.</p>

	<p><b>Borrar libro.</b> Permite al usuario eliminar los datos de un libro.</p> <p><b>Modificar libro.</b> Permite al usuario modificar en cualquier momento, información acerca de uno determinado libro.</p>
FLUJO DE EVENTOS PRINCIPAL	<p>Pasos de ejecución del camino básico del caso de uso:</p> <ol style="list-style-type: none"> <li>1. El usuario selecciona el formulario de entrada de libros.</li> <li>2. Una vez seleccionado el formulario, procederá a realizar alguna de las operaciones de manipulación establecidas: insertar, modificar y borrar.</li> <li>3. Si se selecciona la opción de insertar nuevo libro, tendrá que añadir información de los capítulos que componen el libro y los autores del libro: relación de inclusión.</li> </ol>
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , sino hay ningún libro almacenado en la base de datos no se podrá ni modificar ni borrar.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario no podrá insertar un libro que ya exista en la base de datos.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario deberá insertar toda aquella información que se considera necesaria para un libro y que se le indicará a través del formulario.
FLUJO DE EVENTOS EXEPCIONAL	En cualquier paso, el usuario puede detenerse sin llegar a realizar ninguna operación de manipulación.

En la *Figura 3.5.6* se puede observar la proyección del contenido gráfico del diagrama de casos de uso *Formulario Libro*.

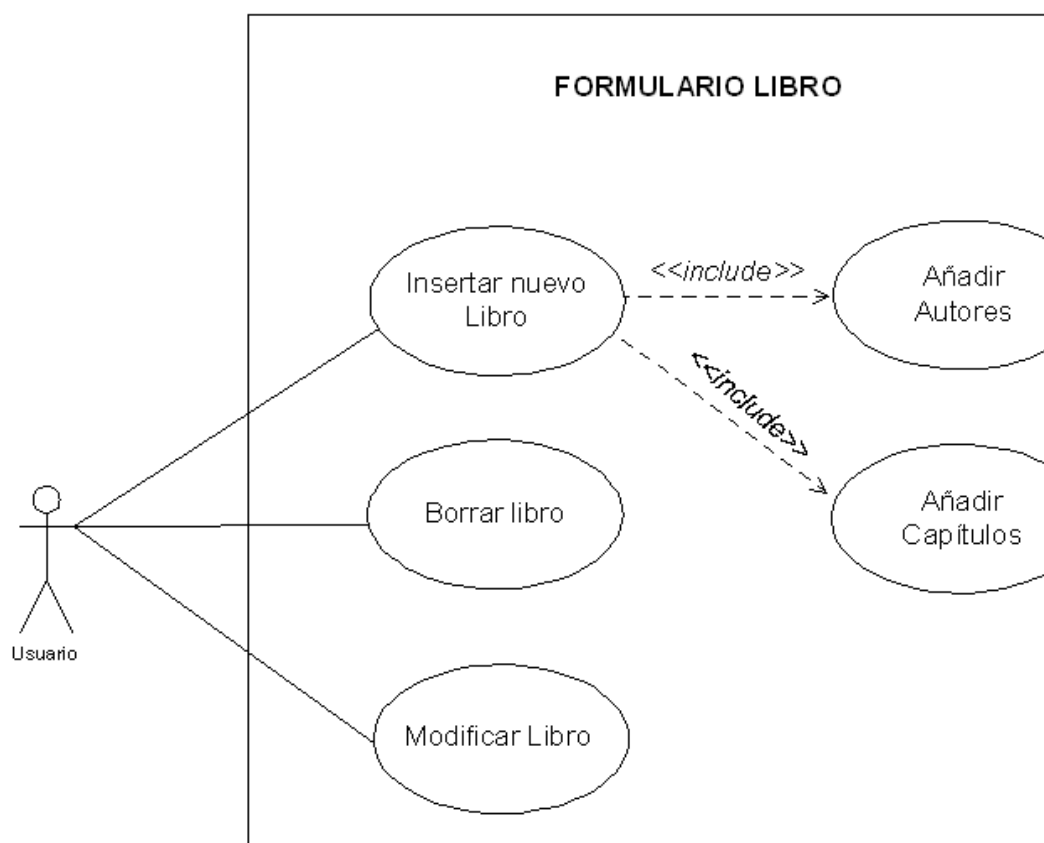


Figura 3.5.6: Diagrama de Casos de Uso: Formulario Libro

### 3.5.2.8 Diagrama de casos de uso: Formulario Capítulo

NOMBRE	<b>Formulario Capítulo</b>
DESCRIPCIÓN	Mediante este caso de uso se manipula la información relativa a los capítulos que componen un libro.
ACTORES	Usuario.
CASOS DE USO	<p><b>Insertar nuevo capítulo.</b> Permite al usuario introducir los datos de un nuevo capítulo e insertarlos en la base de datos.</p> <p><b>Añadir libro.</b> Permite al usuario indicar el libro al cuál pertenece el nuevo capítulo.</p> <p><b>Añadir autores.</b> Permite al usuario añadir los autores del capítulo.</p>

	<p><b>Borrar capítulo.</b> Permite al usuario eliminar los datos de un capítulo.</p> <p><b>Modificar capítulo.</b> Permite al usuario modificar en cualquier momento, información acerca de uno determinado capítulo.</p>
FLUJO DE EVENTOS PRINCIPAL	<p>Pasos de ejecución del camino básico del caso de uso:</p> <ol style="list-style-type: none"> <li>1. El usuario selecciona el formulario de entrada de capítulos.</li> <li>2. Una vez seleccionado el formulario, procederá a realizar alguna de las operaciones de manipulación establecidas: insertar, modificar y borrar.</li> <li>3. Si se selecciona la opción de insertar nuevo capítulo, tendrá que añadir información del libro al que pertenece el capítulo y los autores del capítulo: relación de inclusión.</li> </ol>
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , sino hay ningún capítulo almacenado en la base de datos no se podrá ni modificar ni borrar.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario no podrá insertar un capítulo que ya exista en la base de datos.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario deberá insertar toda aquella información que se considera necesaria para un capítulo y que se le indicará a través del formulario.
FLUJO DE EVENTOS EXEPCIONAL	En cualquier paso, el usuario puede detenerse sin llegar a realizar ninguna operación de manipulación.

En la *Figura 3.5.7* se puede observar la proyección del contenido gráfico del diagrama de casos de uso *Formulario Capítulo*.

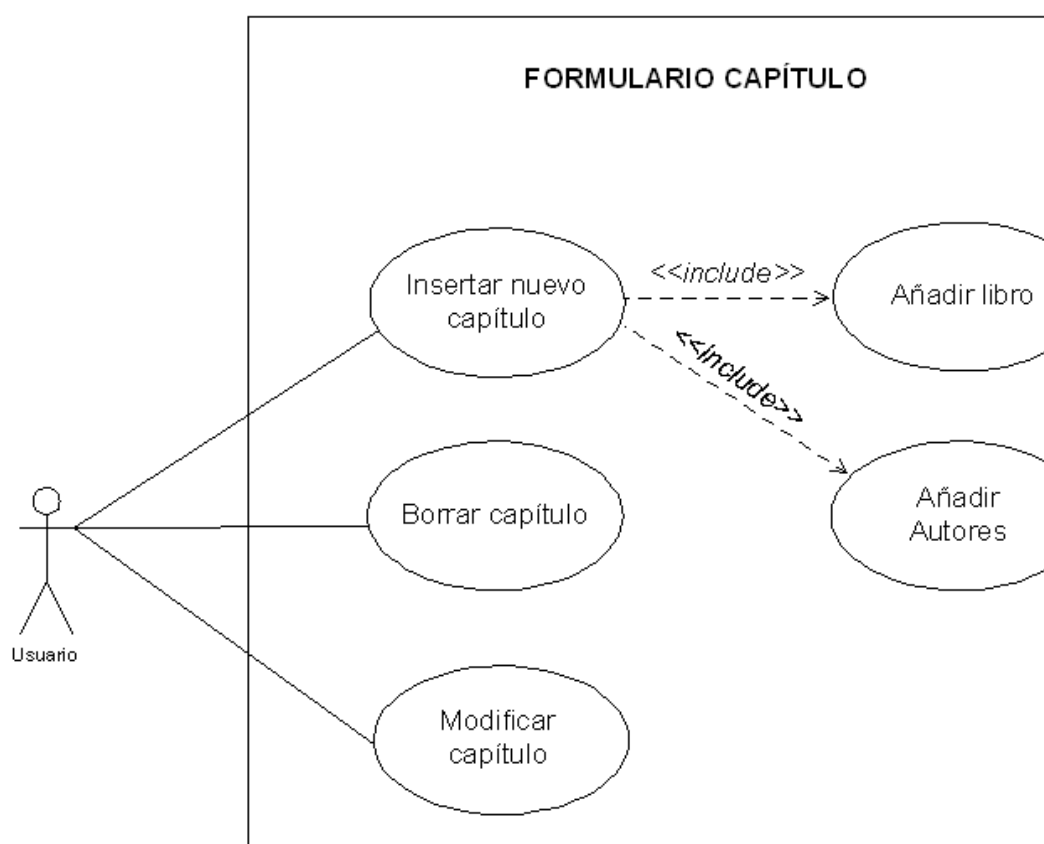


Figura 3.5.7: Diagrama de Casos de Uso: Formulario Capítulo

### 3.5.2.9 Diagrama de casos de uso: Formulario Congreso

NOMBRE	<b>Formulario Congreso</b>
DESCRIPCIÓN	Mediante este caso de uso se manipula la información relativa a los congresos relacionados con el grupo.
ACTORES	Usuario.
CASOS DE USO	<p><b>Insertar nuevo congreso.</b> Permite al usuario introducir los datos de un nuevo congreso e insertarlos en la base de datos.</p> <p><b>Añadir actas.</b> Permite al usuario añadir las actas del congreso.</p> <p><b>Borrar congreso.</b> Permite al usuario eliminar los datos de un congreso.</p> <p><b>Modificar congreso.</b> Permite al usuario modificar en</p>

	cualquier momento, información acerca de uno determinado congreso.
FLUJO DE EVENTOS PRINCIPAL	<p>Pasos de ejecución del camino básico del caso de uso:</p> <ol style="list-style-type: none"> <li>1. El usuario selecciona el formulario de entrada de congresos.</li> <li>2. Una vez seleccionado el formulario, procederá a realizar alguna de las operaciones de manipulación establecidas: insertar, modificar y borrar.</li> <li>3. Si se selecciona la opción de insertar nuevo congreso, tendrá que añadir información de las actas del congreso: relación de inclusión.</li> </ol>
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , sino hay ningún congreso almacenado en la base de datos no se podrá ni modificar ni borrar.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario no podrá insertar un congreso que ya exista en la base de datos.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario deberá insertar toda aquella información que se considera necesaria para un congreso y que se le indicará a través del formulario.
FLUJO DE EVENTOS EXEPCIONAL	En cualquier paso, el usuario puede detenerse sin llegar a realizar ninguna operación de manipulación.

En la *Figura 3.5.8* se puede observar la proyección del contenido gráfico del diagrama de casos de uso *Formulario Congreso*.

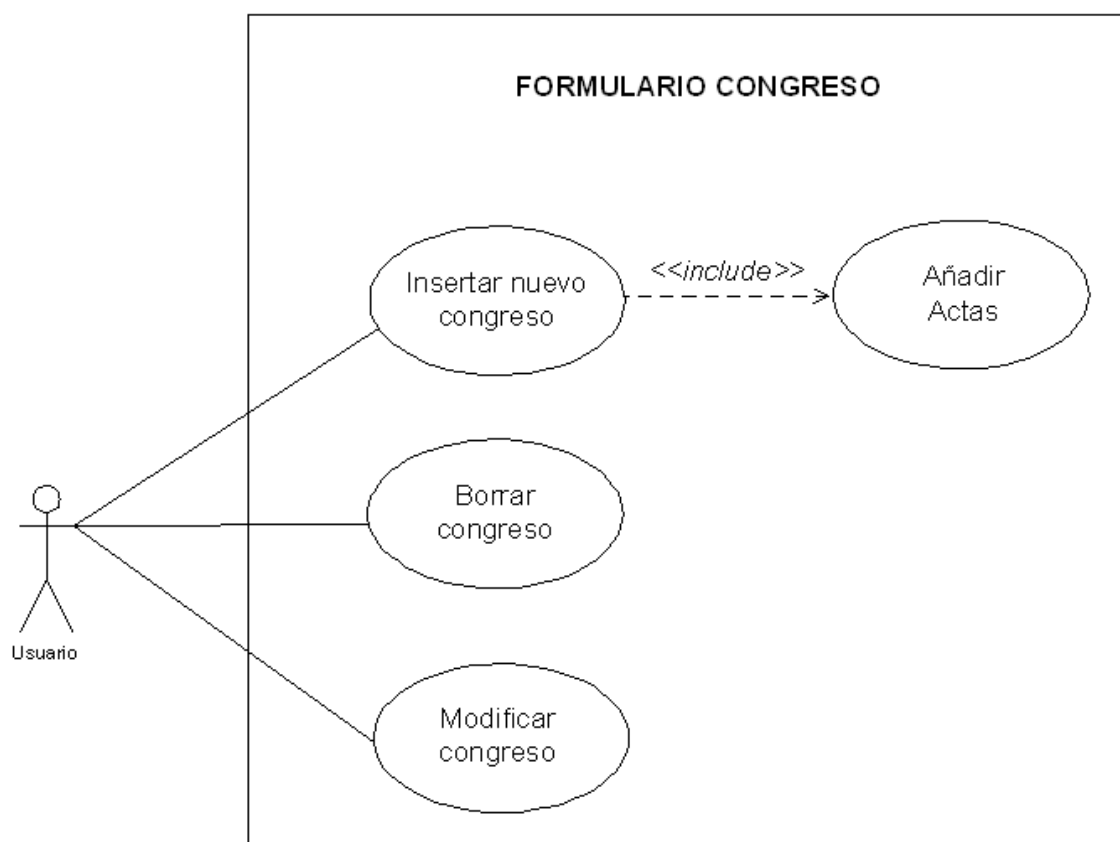


Figura 3.5.8: Diagrama de Casos de Uso: Formulario Miembro

### 3.5.2.10 Diagrama de casos de uso: Formulario Acta

NOMBRE	<b>Formulario Acta</b>
DESCRIPCIÓN	Mediante este caso de uso se manipula la información relativa a las actas de los congresos.
ACTORES	Usuario.
CASOS DE USO	<p><b>Insertar nueva acta.</b> Permite al usuario introducir los datos de una nueva acta e insertarlos en la base de datos.</p> <p><b>Añadir congreso.</b> Permite al usuario indicar el congreso al cual pertenece el acta.</p> <p><b>Añadir autores.</b> Permite al usuario añadir los autores del acta.</p> <p><b>Borrar acta.</b> Permite al usuario eliminar los datos de un acta.</p>



	<b>Modificar acta.</b> Permite al usuario modificar en cualquier momento, información acerca de uno determinado acta.
FLUJO DE EVENTOS PRINCIPAL	<p>Pasos de ejecución del camino básico del caso de uso:</p> <ol style="list-style-type: none"> <li>1. El usuario selecciona el formulario de entrada de actas.</li> <li>2. Una vez seleccionado el formulario, procederá a realizar alguna de las operaciones de manipulación establecidas: insertar, modificar y borrar.</li> <li>3. Si se selecciona la opción de insertar nueva acta, tendrá que añadir información del congreso al que pertenece el acta y los autores del acta: relación de inclusión.</li> </ol>
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , sino hay ningún acta almacenada en la base de datos no se podrá ni modificar ni borrar.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario no podrá insertar un acta que ya exista en la base de datos.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario deberá insertar toda aquella información que se considera necesaria para un acta y que se le indicará a través del formulario.
FLUJO DE EVENTOS EXEPCIONAL	En cualquier paso, el usuario puede detenerse sin llegar a realizar ninguna operación de manipulación.

En la *Figura 3.5.9* se puede observar la proyección del contenido gráfico del diagrama de casos de uso *Formulario Acta*.

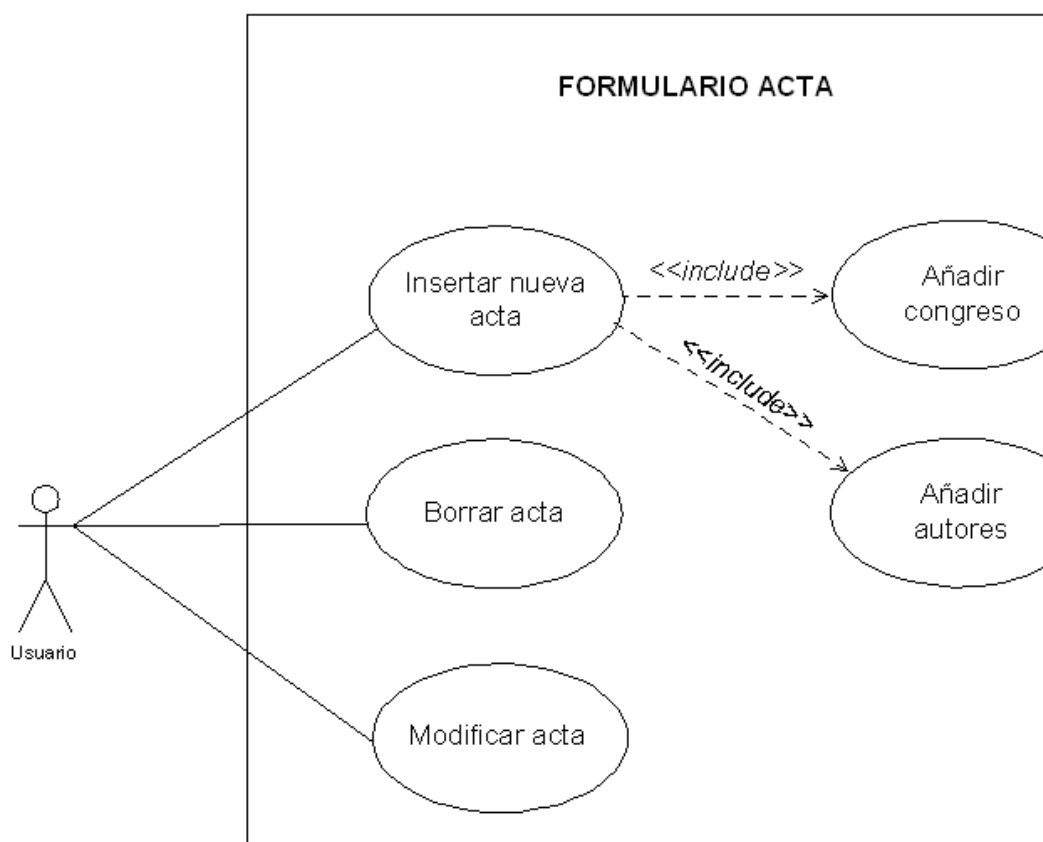


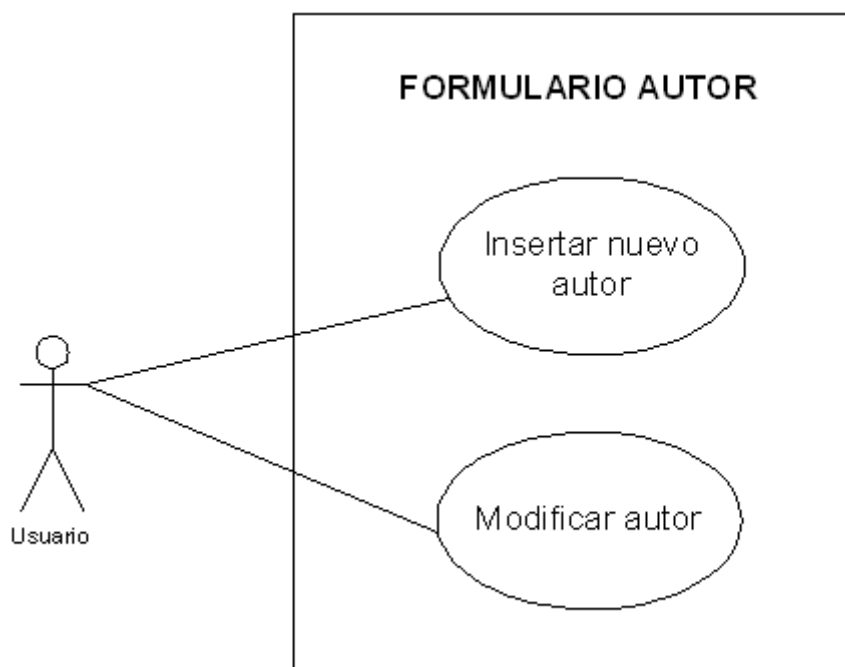
Figura 3.5.9: Diagrama de Casos de Uso: Formulario Acta

### 3.5.2.11 Diagrama de casos de uso: Formulario Autor

NOMBRE	<b>Formulario Artículo</b>
DESCRIPCIÓN	Mediante este caso de uso se manipula la información relativa a los autores de alguna de las publicaciones relacionadas con el grupo de investigación.
ACTORES	Usuario.
CASOS DE USO	<p><b>Insertar nuevo autor.</b> Permite al usuario introducir los datos de un nuevo autor e insertarlos en la base de datos.</p> <p><b>Modificar autor.</b> Permite al usuario modificar en cualquier momento, información acerca de uno determinado autor.</p>
FLUJO DE EVENTOS PRINCIPAL	<p>Pasos de ejecución del camino básico del caso de uso:</p> <ol style="list-style-type: none"> <li>1. El usuario selecciona el formulario de entrada de autores.</li> </ol>

	2. Una vez seleccionado el formulario, procederá a realizar alguna de las operaciones de manipulación establecidas: insertar y modificar.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , sino hay ningún autor almacenado en la base de datos no se podrá modificar.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario no podrá insertar un autor que ya exista en la base de datos.
FLUJO DE EVENTOS EXEPCIONAL	En el <i>paso 2</i> , el usuario deberá insertar toda aquella información que se considera necesaria para un autor y que se le indicará a través del formulario.
FLUJO DE EVENTOS EXEPCIONAL	En cualquier paso, el usuario puede detenerse sin llegar a realizar ninguna operación de manipulación.

En la *Figura 3.5.10* se puede observar la proyección del contenido gráfico del diagrama de casos de uso *Formulario Autor*.



**Figura 3.5.10:** Diagrama de Casos de Uso: Formulario Autor

# 3.6

## ANÁLISIS DEL SISTEMA

---

Los objetivos de este capítulo son: especificar las características funcionales del software, establecer la interfaz que dicho software va a tener con los demás elementos del sistema. Todo ello con el fin de crear un modelo de análisis que muestre, a un nivel de abstracción poco profundo, el funcionamiento del sistema, la información que maneja el mismo y como maneja dicha información. Mediante este modelo se obtendrá como resultado la implementación de las clases que utiliza el sistema dentro del contexto de este modelo.

Para la especificación de dicho análisis se empleará el Lenguaje Unificado de Modelado (UML), dado que proporciona las reglas sintácticas, semánticas y prácticas y el vocabulario suficiente para combinar palabras de éste con el objetivo de lograr la comunicación creando, a su vez, modelo bien formados. (de Booch, 1999)

### 3.6.1 MODELO DE ANÁLISIS

El *modelo de análisis* utiliza el modelo de casos de uso como entrada. Es una especificación detallada de los requisitos y funciona como primera aproximación del modelo de diseño. Se utiliza para comprender de manera más precisa el modelo de casos de uso *refinándolos* en forma de *clasificadores conceptuales* (*clases de análisis*, diferentes de los clasificadores de diseño que serán *objeto de implementación*).

El modelo de análisis es diferente del modelo de diseño en que es un modelo puramente conceptual en lugar de ser un esquema de implementación. Puede ser transitorio o mantenerse durante toda la vida del sistema en proyectos complejos. Cada elemento del modelo de análisis es *trazable* a partir de los elementos del modelo de diseño que lo realizan.

Cada caso de uso en el modelo de casos de uso se traducirá en una *realización* de caso de uso de colaboración en el modelo de análisis. Esta realización identifica cómo se lleva a cabo la funcionalidad representada por el caso de uso de colaboración bajo la dinámica de una sociedad de clases del modelo de análisis. En la mayoría de las ocasiones un caso de uso es realizado exactamente por una colaboración que se considera implícita en el modelo y no es necesario representarse en los diagramas de interacción. *La dualidad caso de uso/realización es la base de la trazabilidad directa entre los requisitos y el análisis.*

### **¿Cómo realizar el modelo de análisis?**

El modelo de análisis estructura los requisitos para facilitar su comprensión, preparación, modificación y mantenimiento. La estructuración del modelo (basada en clases de análisis y paquetes de análisis) es independiente de la estructura basada en casos de uso. Sin embargo, existe una trazabilidad directa entre ambas estructuras (casos de uso/realizaciones de casos de uso), de forma que es posible hacer una traza de diferentes descripciones (en diferentes niveles de detalle) del mismo requisito y mantener su consistencia mutua con facilidad. La estructura de los requisitos que proporciona el modelo de análisis sirve como entrada fundamental para dar forma al sistema en su totalidad incluyendo la arquitectura, debido a que el objetivo es construir el sistema como un todo mantenible y no sólo describir sus requisitos.

Es muy importante que el modelo de análisis se limite únicamente a mostrar aquellas clases que formen parte del vocabulario del dominio del problema (*requisitos de negocio*). Es preciso mantener el modelo de análisis como una declaración concisa y sencilla de la estructura y comportamiento del sistema.

Los artefactos de análisis son:

- Clases de análisis. Modelan conceptos clave en el dominio del negocio.
- Realizaciones de caso de uso. Representan cómo las instancias de las clases de análisis pueden interactuar para realizar el comportamiento especificado por un caso de uso.

### 3.6.1.1 Especificación del Modelo de Clases

Una vez analizados los casos de uso identificados anteriormente, se procede a realizar un refinamiento de los casos de uso más relevantes.

El modelo de clases describe la *estructura estática general del sistema*. Este modelo únicamente puede mostrar clases y cada clase aparece en él sólo una vez. (de Booch, 1999)

Una *clase* es una abstracción de las cosas que forman parte del vocabulario del sistema. Una clase representa un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Un *atributo* representa una propiedad del elemento que se está modelando, describe un valor que puede tomar un objeto concreto de la clase. Una *operación* o *método* es la implementación de un servicio que puede ser requerido a cualquier objeto de la clase para que muestre un comportamiento.

A continuación se realizará una descripción de cada una de las clases del modelo de clases especificando para cada una de ellas sus características significativas. La representación gráfica de cada clase se realizará utilizando la notación de UML.

#### 3.6.1.1.1 Clase: GestorInformacion

**Tabla3.6.1:** Especificación de la clase de análisis: GestorInformación

CLASE DE ANÁLISIS: <i>GestorInformacion</i>	
Descripción	La clase GestorInformación se encarga de realizar una gestión de la información del

sistema GIGI, para lo que necesita el empleo de formularios de entrada que se le ofrecen al usuario para manejo de la información del sistema.

#### Atributos

generadorFE: Mantiene una referencia a un objeto GeneradorFE.

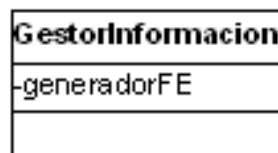
#### Operaciones

#### Estereotipos

<<control>>

#### Valores etiquetados

En la *Figura 3.6.1* se puede ver la representación en UML de la clase de análisis *GestorInformacion*.



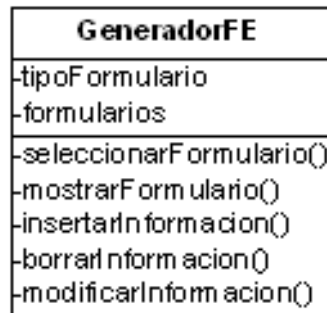
**Figura3.6.1:** Relación: GestorInformacion – GeneradorFE

### 3.6.1.1.2 Clase: GeneradorFE

**Tabla 3.6. 2:** Especificación de la clase de análisis: GeneradorFE

CLASE DE ANÁLISIS: <i>GeneradorFE</i>	
<b>Descripción</b>	Mediante esta clase se genera el formulario de entrada seleccionado por el usuario y que emplea para llevar a cabo operaciones de manipulación de información: insertar, borrar y modificar.
<b>Atributos</b>	<p><b>formularios:</b> Representa el conjunto de formularios disponibles para el usuario.</p> <p><b>tipoFormulario:</b> Se utiliza para conservar el formulario que ha seleccionado el usuario.</p>
<b>Operaciones</b>	<p><b>seleccionarFormulario:</b> Permite al usuario la elección, de entre los posibles, del formulario con el que desea trabajar.</p> <p><b>mostrarFormulario:</b> Muestra al usuario el formulario seleccionado.</p> <p><b>insertarInformacion:</b> almacena en la base de datos la información de entrada.</p> <p><b>borrarInformacion:</b> elimina de la base de datos la información seleccionada por el usuario.</p> <p><b>modificarInformacion:</b> modifica la información correspondiente de la base de datos.</p>
<b>Estereotipos</b>	<<interfaz>>
<b>Valores etiquetados</b>	

En la *Figura 3.6.2* se puede ver la representación en UML de la clase de análisis *GeneradorFE*.



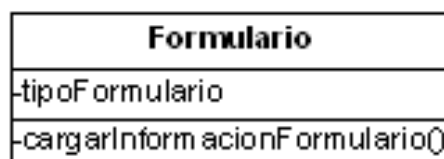
**Figura 3.6.2:** Representación en UML de la Clase: GeneradorFE

### 3.6.1.1.3 Clase: Formulario

**Tabla 3.6. 3:** Especificación de la clase de análisis: Formulario

CLASE DE ANÁLISIS: <i>Formulario</i>	
<b>Descripción</b>	Esta clase representa un formulario de entrada determinado que ha sido seleccionado por el usuario. En función del formulario se cargará en el GeneradorFE la información necesaria para cada formulario.
<b>Atributos</b>	tipoFormulario: almacena el tipo de formulario.
<b>Operaciones</b>	cargarInformacionFormulario: carga la información correspondiente del formulario.
<b>Estereotipos</b>	
<b>Valores etiquetados</b>	<<entity>>

En la *Figura 3.6.3* se puede ver la representación en UML de la clase de análisis *Formulario*.



**Figura 3.6.3:** Representación en UML de la Clase: Formulario



### 3.6.1.2 Especificación del modelo Objeto de Análisis – Relación

Mediante el modelo objeto de análisis – relación se mostrarán las relaciones entre las clases indicando la naturaleza, cardinalidades, dependencias, etc., de cada una de ellas. Cada clase tiene una serie de responsabilidades (servicios) que puede cumplir de dos formas distintas:

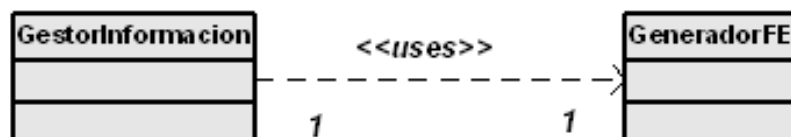
1. De forma particular usando operaciones para manipular sus propios atributos.
2. Colaborando con otras clases. La colaboración es una relación de interacción con otra clase para la realización de una responsabilidad.

Para una mayor claridad y comprensión del modelo objeto-relación se realizará, en primer lugar, una descripción de cada una de las relaciones entre los objetos del modelo de clases y, en segundo lugar, una representación del diagrama de clases resultante.

#### 3.6.1.2.1 Relación: GestorInformacion – GeneradorFE

Un objeto de tipo *GestorInformacion* provee de métodos para gestionar la información de entrada proporcionada por el *GeneradorFE*. Este objeto hará uso de un objeto de tipo *GeneradorFE*.

En la *Figura 3.6.4* se puede ver la representación gráfica en notación UML de la relación *GestorInformacion* – *GeneradorFE*. A efectos de simplificación no se muestran los atributos ni operaciones de las clases.

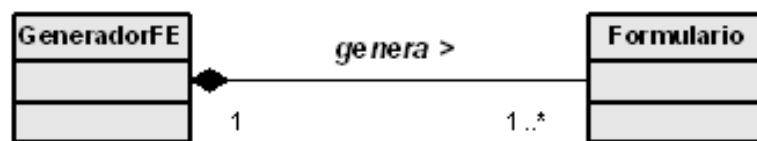


**Figura 3.6.4:** Relación: GestorInformacion – GeneradorFE

### 3.6.1.2.2 Relación: GeneradorFE – Formulario

Esta relación de agregación indica que un objeto *GeneradorFE* se compone de uno o varios objetos *Formulario*.

En la *Figura 3.6.5* se puede ver la representación gráfica en notación UML de la relación *GeneradorFE – Formulario*. A efectos de simplificación no se muestran los atributos ni operaciones de las clases.



**Figura 3.6.5:** Relación: GeneradorFE – Formulario

### 3.6.1.3 Diagrama de clases del Sistema

Una vez analizadas las relaciones una a una se puede generar el **diagrama de clases del modelo estructural estático del subsistema**. Mediante este diagrama se pretende mostrar los aspectos estructurales acerca de las clases consideradas dentro del dominio del problema y de las relaciones principales necesarias ente dichas clases para cumplir los requisitos funcionales del subistema.

En la *Figura 3.6.6* se puede observar el diagrama de clases del subistema propuesto.

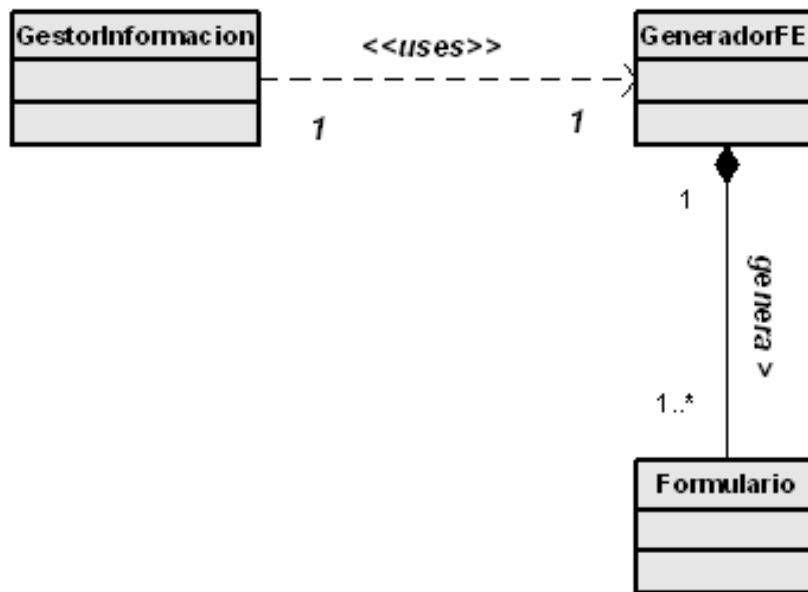


Figura 3.6.6: Diagrama de Clases del Subsistema

#### 3.6.1.4 Especificación del modelo Objeto de Análisis - Comportamiento

Una vez representados los elementos estáticos del sistema ahora es el momento de hacer una transición a su comportamiento dinámico. El aspecto dinámico del sistema combina la parte estructural con la dimensión del tiempo. El **modelo objeto-comportamiento** representa cómo interaccionan las clases, especificando un flujo de control (secuencia de mensajes entre objetos) a lo largo del tiempo, para la ejecución de una tarea.

UML proporciona dos tipos de diagramas de interacción (de Booch, 1999), los diagramas de *secuencia* y los diagramas de *colaboración*. Un *diagrama de secuencia destaca la ordenación temporal de los mensajes entre objetos*. Un *diagrama de colaboración destaca la organización estructural de los objetos que envían y reciben mensajes*. Ambos son consistentes entre sí; representan la misma información pero desde diferentes perspectivas. En el contexto de los casos de uso un diagrama de interacción permite representar un escenario, que a su vez, representa un flujo particular de la acción asociada al caso de uso.

En los apartados siguientes sólo se desarrollan los diagramas de interacción más significativos dentro de la funcionalidad requerida. Se va a presentar solamente el diagrama de colaboración perteneciente a la acción de insertar información nueva, ya que son similares los diagramas de colaboración para las acciones de modificación y borrado de información.

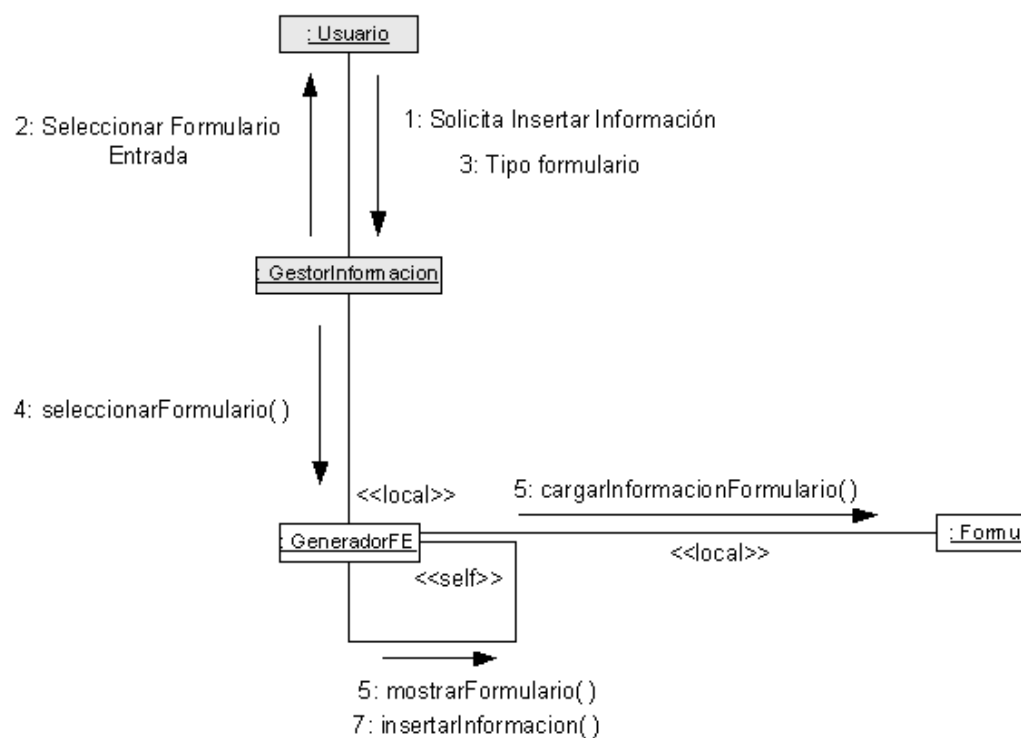
#### 3.6.1.4.1 Diagrama de Colaboración: Insertar Nueva Información

En este diagrama de colaboración se pretende representar el flujo de eventos principal correspondiente a la inserción de nueva información.

En la *Figura 3.6.7* se nos presenta el diagrama de colaboración **Insertar Nueva Información**.

##### **Flujo de sucesos:**

1. El usuario solicita insertar nueva información.
2. El sistema solicita al usuario que seleccione el tipo de formulario, en función de la información que pretenda insertar.
3. El usuario selecciona el formulario.
4. El sistema genera el formulario determinado.
5. Se inserta la nueva información.



**Figura3.6.7:** Diagrama de Colaboración: Insertar Nueva Información

# 3.7

## DISEÑO DEL SISTEMA

---

Una vez realizado el análisis del sistema vamos a describir la **vista de diseño**. En esta vista se describirá a un nivel mayor de refinamiento los objetos tal y como deberán ser implementados, lo que irá dando la arquitectura final del sistema. En este nivel de especificación, los objetos resultantes irán fuertemente vinculados con el código fuente que se obtendrá más adelante, ya que este es el objetivo principal de esta fase de diseño. Los diagramas de especificación de objetos que obtendremos describen la solución propuesta al problema analizado y no tendrán por qué ser idénticos a los diagramas de nivel conceptual utilizados en la fase de análisis ya que estos están vistos desde un mayor nivel de abstracción.

### 3.7.1 ESPECIFICACIÓN DEL MODELO DE OBJETOS

El *Modelo de Objetos* describe el conjunto de elementos que darán soporte a la implementación computacional de la solución propuesta. Cada objeto se presenta al nivel de refinamiento suficiente para cumplir sus responsabilidades de forma particular o colaborando con otros objetos. Algunos objetos pueden no tener correspondencia directa o indirecta con elementos del modelo de clases debido a que son objetos de soporte que colaboran para llevar a cabo una responsabilidad mayor.

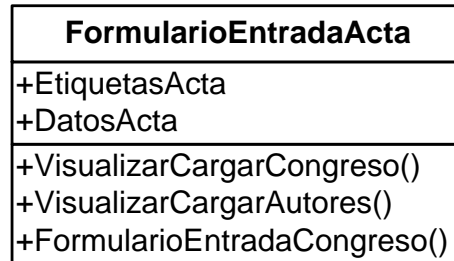
A continuación se realizará una descripción de cada uno de los objetos del modelo de objetos especificando para cada uno de ellos sus atributos y métodos más significativos. Se utilizará UML como notación gráfica de cada objeto.

### 3.7.1.1 Objeto: *FormularioEntradaActa*

**Tabla 3.7. 1:** Objeto: FormularioEntradaActas

FormularioEntradaActa	
<b>Descripción</b>	Clase que construye el formulario de entrada para las actas del grupo de investigación. El formulario permitirá insertar y mantener la información referente a las actas de la base de datos
<b>Estereotipos</b>	
<b>Atributos</b>	<p>EtiquetasActas: Etiquetas de los campos de los campos de información que serán mostrados.</p> <p>DatosActa: Cuadros de texto para la inserción de información por parte del usuario.</p>
<b>Operaciones</b>	<p>VisualizarCargarCongreso: Método que muestra los congresos cuya información es mantenida por el grupo de investigación. Se podrá seleccionar de la lista el congreso al cual pertenece el acta actual cuyos campos se están rellenando. La selección de un congreso dará lugar a la actualización del campo correspondiente.</p> <p>VisulizarCargarAutores: Método que muestra los autores cuya información es mantenida por el grupo de investigación. Se podrá seleccionar de la lista el autor o autores que han realizado el acta actual cuyos campos se están rellenando. La selección de un autor dará lugar a la actualización del campo correspondiente.</p> <p>FormularioEntradaCongreso: Método cuyo objetivo es invocar al formulario de entrada par los datos de los congresos. Este método también invocará la visualización de los datos de los congresos mantenidos en la base de datos actualmente.</p>
<b>Valores etiquetados</b>	
<b>Responsabilidades</b>	

En la *Figura 3.7.1* se puede ver la representación gráfica del objeto *FormularioEntradaActas..*



**Figura3.7.1:** Representación en UML del Objeto: FormularioEntradaActas

### 3.7.1.2 Objeto: *FormularioEntradaArticulo*

**Tabla 3.7.2:** Objeto: FormularioEntradaArticulo

FormularioEntradaArticulo	
<b>Descripción</b>	Clase que construye el formulario de entrada para los artículos del grupo de investigación. El formulario permitirá insertar y mantener la información referente a los artículos de la base de datos.
<b>Estereotipos</b>	
<b>Atributos</b>	<p>EtiquetasArticulo: Etiquetas de los campos de los campos de información que serán mostrados.</p> <p>DatosArticulo: Cuadros de texto para la inserción de información por parte del usuario.</p>
<b>Operaciones</b>	<p>VisualizarCargarRevista: Método que muestra las revistas cuya información es mantenida por el grupo de investigación. Se podrá seleccionar de la lista la revista a la cual pertenece el artículo actual cuyos campos se están rellenando. La selección de una revista dará lugar a la actualización del campo correspondiente.</p> <p>VisulizarCargarAutores: Método que muestra los autores cuya información es mantenida por el grupo de investigación. Se podrá seleccionar de la lista el autor o autores que han realizado el acta actual cuyos campos se están rellenando. La selección de un autor dará lugar a la actualización del campo correspondiente.</p> <p>FormularioEntradaRevista: Método cuyo objetivo es invocar al formulario de entrada para los datos de las revistas. Este método también invocará la visualización de los datos de las revistas mantenidos en la base de datos actualmente.</p>



Valores etiquetados
Responsabilidades

En la *Figura 3.7.2* se puede ver la representación gráfica del objeto *FormularioEntradaArticulo*.

<b>FormularioEntradaArticulo</b>
+DatosArticulo +EtiquetasArticulo
+VisualizarCargarRevista() +VisualizarCargarAutores() +FormularioEntradaRevista()

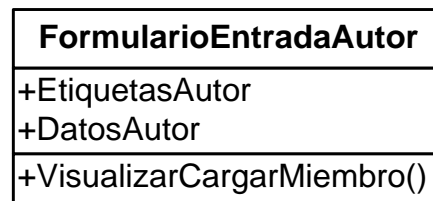
**Figura3.7.2:** Representación en UML de la Clase: FormularioEntradaArticulo

### 3.7.1.3 Objeto: *FormularioEntradaAutor*

**Tabla 3.7.3:** Objeto: FormularioEntradaAutor

<b>FormularioEntradaAutor</b>	
<b>Descripción</b>	Clase que construye el formulario de entrada para la información de los autores. El formulario permitirá insertar y mantener la información referente a los autores en la base de datos.
<b>Estereotipos</b>	
<b>Atributos</b>	EtiquetasAutor: Etiquetas de los campos de los campos de información que serán mostrados.  DatosAutor: Cuadros de texto para la inserción de información por parte del usuario.
<b>Operaciones</b>	VisualizarCargarMiembro: Método que muestra los miembros que actualmente forman parte del grupo de investigación. Se podrá seleccionar de la lista el miembro que se desea y sus datos serán cargados en el formulario de entrada de autor.
<b>Valores etiquetados</b>	
<b>Responsabilidades</b>	

En la *Figura 3.7.3* se puede ver la representación gráfica del objeto *FormularioEntradaAutor*.



**Figura3.7.3:** Representación en UML del Objeto: FormularioEntradaAutor

#### 3.7.1.4 Objeto: *FormularioEntradaCapitulo*

**Tabla 3.7.4:** Objeto: FormularioEntradaCapitulo

FormularioEntradaCapitulo	
<b>Descripción</b>	Clase que construye el formulario de entrada para los capítulos de libros mantenidos por grupo de investigación. El formulario permitirá insertar y mantener la información referente a los capítulos de la base de datos.
<b>Estereotipos</b>	
<b>Atributos</b>	<p>EtiquetasCapitulo: Etiquetas de los campos de los campos de información que serán mostrados.</p> <p>DatosCapitulo: Cuadros de texto para la inserción de información por parte del usuario.</p>
<b>Operaciones</b>	<p>VisualizarCargarLibro: Método que muestra los libros de capítulos cuya información es mantenida por el grupo de investigación. Se podrá seleccionar de la lista el libro de capítulos al cual pertenece el capítulo actual cuyos campos se están rellenando. La selección de un libro dará lugar a la actualización del campo correspondiente.</p> <p>VisualizarCargarAutores: Método que muestra los autores cuya información es mantenida por el grupo de investigación. Se podrá seleccionar de la lista el autor o autores que han realizado el acta actual cuyos campos se están rellenando. La selección de un autor dará lugar a la actualización del campo correspondiente.</p> <p>FormularioEntradaLibroCapitulo: Método público cuyo objetivo es invocar al formulario de entrada para los datos de los libros de capítulos. Este método</p>

	también invocará la visualización de los datos de los libros de capítulos mantenidos en la base de datos actualmente.
Valores etiquetados	
Responsabilidades	

En la *Figura 3.7.4* se puede ver la representación gráfica del objeto *FormularioEntradaCapitulo*.

<b>FormularioEntradaCapitulo</b>
+DatosCapitulo +EtiquetasCapitulo
+VisualizarCargarLibro() +VisualizarCargarAutores() +FormularioEntradaLibroCapitulo()

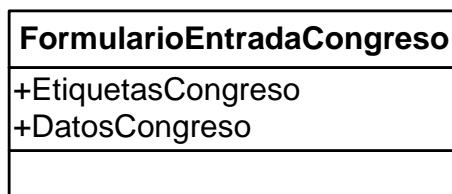
**Figura3.7.4:** Representación en UML del Objeto: FormularioEntradaCapitulo

### 3.7.1.5 Objeto: *FormularioEntradaCongreso*

**Tabla 3.7.5:** Objeto: FormularioEntradaCongreso

<b>FormularioEntradaCongreso</b>	
<b>Descripción</b>	Clase que construye el formulario de entrada para la información de los congresos. El formulario permitirá insertar y mantener la información referente a los congresos en la base de datos.
<b>Estereotipos</b>	
<b>Atributos</b>	EtiquetasCongreso: Etiquetas de los campos de los campos de información que serán mostrados.  DatosCongreso: Cuadros de texto para la inserción de información por parte del usuario.
<b>Operaciones</b>	
<b>Valores etiquetados</b>	
<b>Responsabilidades</b>	

En la *Figura 3.7.5* se puede ver la representación gráfica del objeto *FormularioEntradaCongreso*.



**Figura3.7.5:** Representación en UML del Objeto: FormularioEntradaCongeso

### 3.7.1.6 Objeto: *FormularioEntradaLibro*

**Tabla 3.7.6:** Objeto: FormularioEntradaLibro

FormularioEntradaLibro	
<b>Descripción</b>	Clase que construye el formulario de entrada para los libros completos del grupo de investigación. El formulario permitirá insertar y mantener la información referente a los libros de la base de datos.
<b>Estereotipos</b>	
<b>Atributos</b>	<p>EtiquetasLibro: Etiquetas de los campos de los campos de información que serán mostrados.</p> <p>DatosLibro: Cuadros de texto para la inserción de información por parte del usuario.</p>
<b>Operaciones</b>	<p>VisulizarCargarAutores: Método que muestra los autores cuya información es mantenida por el grupo de investigación. Se podrá seleccionar de la lista el autor o autores que han realizado el acta actual cuyos campos se están rellendo. La selección de un autor dará lugar a la actualización del campo correspondiente.</p>
<b>Valores etiquetados</b>	
<b>Responsabilidades</b>	

En la *Figura 3.7.6* se puede ver la representación gráfica del objeto *FormularioEntradaLibro*.

FormularioEntradaLibro
+EtiquetasLibro
+DatosLibro
+VisualizarCargarAutores()

**Figura3.7.6:** Representación en UML del Objeto: FormularioEntradaLibro

### 3.7.1.7 Objeto: FormularioEntradaLibroCapitulo

**Tabla 3.7.7:** Objeto: FormularioEntradaLibroCapitulo

FormularioEntradaLibroCapitulo	
<b>Descripción</b>	Clase que construye el formulario de entrada para la información de los libros de capítulos. El formulario permitirá insertar y mantener la información referente a los libros de capítulos en la base de datos.
<b>Estereotipos</b>	
<b>Atributos</b>	EtiquetasLibroCapitulo: Etiquetas de los campos de los campos de información que serán mostrados.  DatosLibroCapitulo: Cuadros de texto para la inserción de información por parte del usuario.
<b>Operaciones</b>	
<b>Valores etiquetados</b>	
<b>Responsabilidades</b>	

En la *Figura 3.7.7* se puede ver la representación gráfica del objeto *FormularioEntradaLibroCapitulo*.

FormularioEntradaLibroCapitulo
+EtiquetasLibroCapitulo
+DatosLibroCapitulo

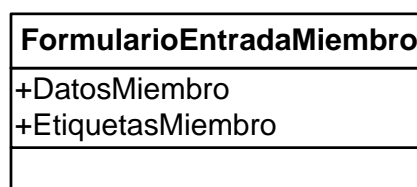
**Figura3.7.7:** Representación en UML del Objeto: FormularioEntradaLibroCapitulo

### 3.7.1.8 Objeto: *FormularioEntradaMiembro*

**Tabla 3.7.8:** Objeto: *FormularioEntradaMiembro*

<b>FormularioEntradaMiembro</b>	
<b>Descripción</b>	Clase que construye el formulario de entrada para los datos de los miembros del grupo de investigación. El formulario permitirá insertar y mantener la información referente a los miembros de la base de datos.
<b>Estereotipos</b>	
<b>Atributos</b>	<p>EtiquetasMiembro: Etiquetas de los campos de los campos de información que serán mostrados.</p> <p>DatosMiembro: Cuadros de texto para la inserción de información por parte del usuario.</p>
<b>Operaciones</b>	
<b>Valores etiquetados</b>	
<b>Responsabilidades</b>	

En la *Figura 3.7.8* se puede ver la representación gráfica del objeto *FormularioEntradaMiembro*.



**Figura3.7.8:** Representación en UML del Objeto: *FormularioEntradaMiembro*

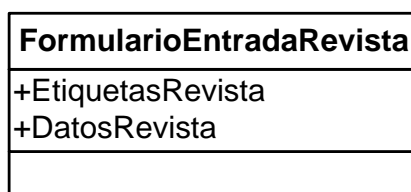
### 3.7.1.9 Objeto: *FormularioEntradaRevista*

**Tabla 3.7.9:** Objeto: *FormularioEntradaRevista*

<b>FormularioEntradaRevista</b>	
<b>Descripción</b>	Clase que construye el formulario de entrada para la información de las revistas. El formulario permitirá insertar y mantener la información referente a las revistas en la base de datos.
<b>Estereotipos</b>	

<b>Atributos</b>	EtiquetasRevista: Etiquetas de los campos de los campos de información que serán mostrados.  DatosRevista: Cuadros de texto para la inserción de información por parte del usuario.
<b>Operaciones</b>	
<b>Valores etiquetados</b>	
<b>Responsabilidades</b>	

En la *Figura 3.7.9* se puede ver la representación gráfica del objeto *FormularioEntradaRevista*.



**Figura3.7.9:** Representación en UML del Objeto: FormularioEntradaRevista

### 3.7.2 ESPECIFICACIÓN DEL MODELO OBJETO DE DISEÑO – RELACIÓN

El diagrama de objetos del sistema muestra la arquitectura final del sistema. En el se incluyen todos los objetos que a través de su funcionalidad contribuyen a la resolución del problema propuesto. Además se ha de tener en cuenta que existen otros objetos que dado su carácter auxiliar no han sido representados en este modelo porque no tienen relación con el núcleo del problema propuesto. Por ejemplo podríamos citar elementos simples de la interfaz como botones o barras de herramientas, los diálogos utilizados, etc.

En la *Figura 3.7.10* se muestra, utilizando la notación UML, el diagrama de objetos del sistema.

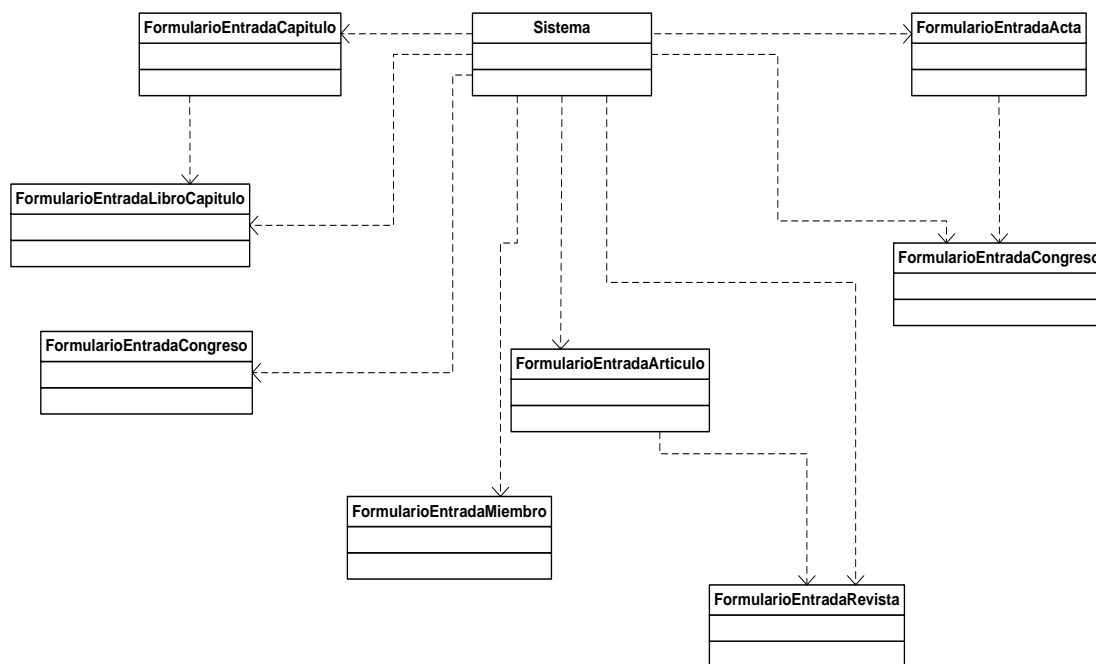


Figura 3.7.10: Diagrama de Objetos del sistema

### 3.7.3 ESPECIFICACIÓN PROCEDIMENTAL

Llegados a este punto, trataremos de modelar el sistema viendo sus aspectos dinámicos desde un alto nivel de abstracción, es decir, nos centraremos en las actividades de negocio más importantes en el dominio del problema que abarca este proyecto. Para ello no pretendemos describir en detalle los procedimientos u operaciones que tienen lugar entre dichos objetos ni los atributos que se intercambian entre sí sino simplemente mostrar las actividades que tienen lugar entre ellos. El flujo de trabajo del sistema quedará de esta manera definido como un conjunto de actividades coordinadas para realizar una tarea específica utilizando los objetos del sistema.

Para realizar esta representación utilizaremos uno de los diagramas que proporciona UML para modelar los aspectos dinámicos del un sistema: los **diagramas de actividades**. Estos no son más que un diagrama de flujo que



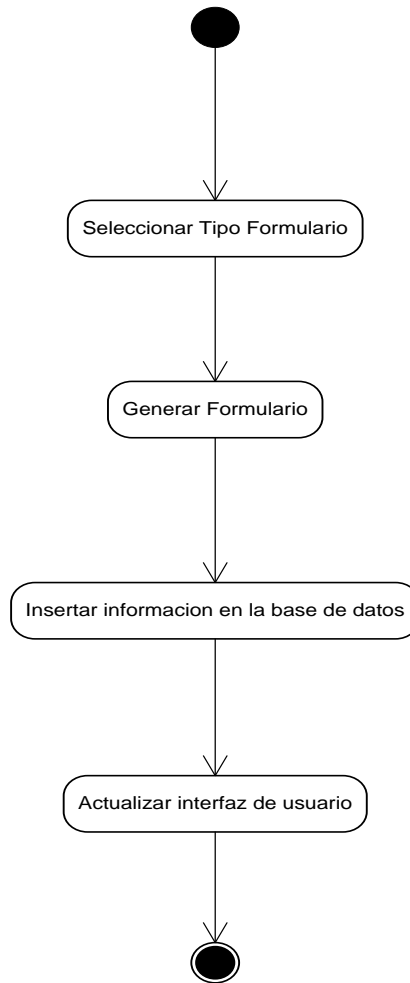
muestra el flujo de control entre actividades y permite dar relevancia a la actividad que ocurre a lo largo del tiempo.

Señalar que únicamente se va a presentar el diagrama de actividades correspondiente a insertar nueva información, sabiendo que tanto modificar como eliminar son similares.

### **3.7.3.1 Diagrama de Actividades: Insertar nueva información**

Para insertar nueva información referente al grupo de investigación, lo que se hace es mostrar al usuario un formulario con la información que quiere introducir, es decir, si se quiere insertar una nueva revista se presentará al usuario un formulario en el que se presentan aquellos campos que se han considerados necesarios para disponer de información acerca de la revista. Una vez que el usuario dispone del formulario de entrada, simplemente lo único que tiene que realizar es introducir la información y almacenarla en la base de datos. Todo esto es similar para el resto de información.

En la *Figura 3.7.11* se presenta, en notación UML, el desarrollo del diagrama de actividades *Insertar nueva información*.



**Figura3.7.11:** Representación en UML del Objeto: FormularioEntradaRevista

### 3.7.4 MODELO DE PAQUETES

Para una mejor comprensión de la arquitectura del sistema, los elementos del modelo de objetos se agrupan en *paquetes*. Un *paquete* organiza los objetos en grupos de modo que pueden ser manipulados de forma global. Cada paquete agrupa elementos cercanos semánticamente y funcionalmente. La visibilidad de los elementos del paquete va implícita en la propia especificación de cada elemento. La *interfaz del paquete* la conforman aquellos elementos que son visibles fuera del paquete. Cada paquete posee un nombre único dentro del sistema. Un paquete puede contener subpaquetes cuyo nombre va precedido del nombre del paquete contenedor. Cada objeto pertenece exclusivamente a un único paquete. Un paquete forma un *espacio de nombres*, lo que significa que la nomenclatura de los elementos del mismo grupo no debe

repetirse en el contexto de su paquete contenedor. Un paquete puede *importar* la interfaz de otro paquete para acceder a sus elementos. Esta importación es un permiso explícito de acceso en un solo sentido.

Un paquete puede ser representado gráficamente como una proyección de su modelo de contenido. Para la representación de cada paquete se hará uso de la notación gráfica de UML.

#### **3.7.4.1 Paquete: Sistema**

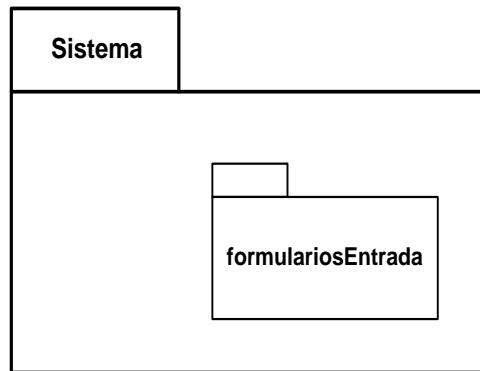
Este paquete es el de nivel superior en la jerarquía de paquetes y representa el sistema completo. Agrupa los objetos de control de la interfaz gráfica de usuario, objetos de soporte y el resto de paquetes.

En la *Figura 3.7.12* se puede ver la representación gráfica del paquete *Sistema*.

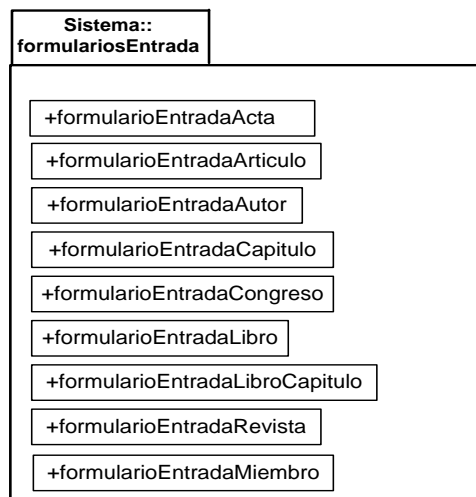
#### **3.7.4.1 Paquete: FormulariosEntrada**

Este paquete agrupa los objetos que implementan los procedimientos de administración sobre los formularios de entrada de información. Agrupa todas aquellas clases necesarias para llevar a cabo la manipulación de la información referente al grupo de investigación.

En la *Figura 3.7.13* se puede ver la representación gráfica del paquete *FormulariosEntrada*.



**Figura 3.7.12:** Representación en UML del paquete: Sistema



**Figura3.7.13:** Representación en UML del pauete: FormulariosEntrada

# 3.8

## DESCRIPCIÓN DE LA INTERFAZ

---

En este capítulo se presenta la descripción de la interfaz desarrollada para el subsistema. La interfaz servirá de unión entre el usuario y el propio subsistema, facilitando así la interacción suficiente para que éste pueda realizar todo el trabajo requerido por aquel. Es decir, se identificarán las pantallas cuyo desarrollo es necesario para permitir la explotación de toda la funcionalidad proporcionada por el sistema. Ello implica describir la navegación a realizar, esto es, desde qué otras pantallas se pueden acceder a cada una y hacia qué otras pantallas se puede acceder desde la actual, especificando su funcionalidad, es decir, qué acciones se permiten llevar a cabo en cada una de ellas.

Por norma general, la interfaz del subsistema debe cumplir una serie de características ergonómicas de forma que faciliten la interacción hombre-máquina haciendo el sistema más atractivo para los usuarios y logrando que la interfaz del subsistema sea lo más cómoda, sencilla e intuitiva posible dentro de los límites del problema planteado. Por tanto, la interfaz debe cumplir los siguientes objetivos básicos:

1. Ubicar toda la funcionalidad del sistema en el mínimo número de pantallas posible, evitando así los continuos cambios de pantalla como recurso a la realización de distintas operaciones.

2. Disminuir en la medida de lo posible la frecuencia de cambio entre los distintos dispositivos de entrada del sistema. Es decir, si el usuario maneja el sistema con el ratón, debe poder realizar la mayor cantidad de operaciones con el mismo, sin la necesidad de acudir al teclado u otro dispositivo para interactuar con el sistema.
3. Los elementos de la interfaz debe acomodarse de manera que su posición en la pantalla facilite la transición entre el pensamiento del usuario y la acción a realizar.

Estos objetivos básicos están englobados en lo que se denomina “reglas de oro” para el diseño de la interfaz:

- *Dar el control al usuario:* Se deben definir los modos de interacción de manera que no se obligue a que el usuario realice acciones innecesarias y no deseadas, considerando así una interacción flexible entre el mismo y el sistema. Al mismo tiempo, se han de ocultar al usuario ocasional los entresijos técnicos, evitando así que pueda interactuar a un nivel “interno” de la máquina.
- *Reducir la carga de memoria del usuario:* Para ello se debería reducir la demanda de memoria a corto plazo, estableciendo, por ejemplo, valores por defecto útiles. A su vez, el formato visual de la interfaz se deberá basar en una metáfora del mundo real, evitando así tener que memorizar una secuencia secreta de interacciones, y debiendo para ello organizar la interfaz de forma jerárquica y progresiva.
- *Construir una interfaz consecuente,* lo que implica que:
  - Toda la información visual se organice de acuerdo con el diseño estándar que se mantiene en todas las presentaciones de pantallas.

- Todos los mecanismos de entrada se limiten a un conjunto limitado y que se utilicen consecuentemente por toda la aplicación.
- Los mecanismos para ir de tarea a tarea se hayan definido e implementado consecuentemente.

En este capítulo se describe la interfaz obtenida finalmente no incorporando posibles mejoras o modificaciones que se realicen en versiones posteriores al proyecto que aquí se aborda. En base a la funcionalidad del subsistema, se ha definido el siguiente módulo que, junto con los módulos desarrollados por otras tareas, formarán la interfaz del sistema completo

- Módulo de entrada.

Cada uno de estos módulos utiliza un conjunto independiente de componentes gráficos de la IGU y otros compartidos. Todos los componentes de la IGU se sitúan bajo el marco principal del sistema que proporciona acceso a toda la funcionalidad implementada.

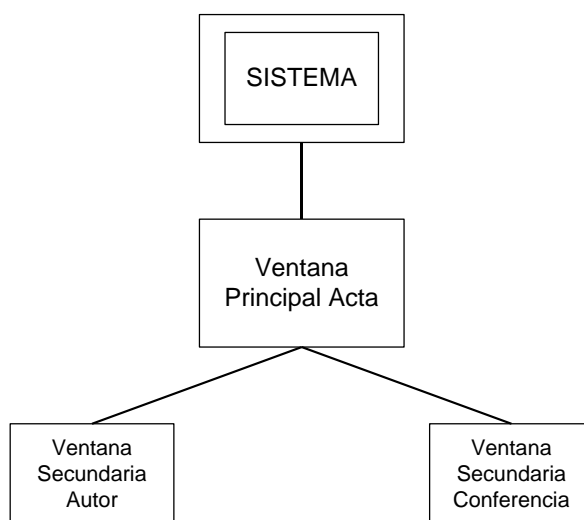
### **3.8.1 MÓDULO DE ENTRADA**

Este módulo contendrá todas las ventanas para la inserción y mantenimiento de información, las cuales se detallarán a continuación y se explicará brevemente su funcionalidad.

Hay que tener en cuenta que dentro de la interfaz, hay ventanas que podrán ser accedidas desde la interfaz general, llamadas ventanas principales, y otras que sólo podrán ser llamadas desde ventanas principales, por lo que podríamos considerarlas secundarias.

### 3.8.1.1 Ventana Principal Actas

Esta ventana es la encargada de la inserción y mantenimiento de la información de las actas. En la *Figura 3.8.1* se muestra las ventanas secundarias que pueden ser llamados desde ella.



**Figura 3.8.1:** Jerarquía Ventana Principal Actas

A continuación en la *Figura 3.8.2* se muestra una captura de la ventana en cuestión, para su posterior explicación.

Actas (Campos obligatorios marcados en negrita)

**Datos del Acta**

**Código** 3 **Contribución** Contribucion 1 al congreso 2

Volumen 5 **Página Inicial** 456 **Página Final** 500 Estado Publicado

**Referencias del Acta**

**Congresos**

**Código** 2 **Título** Congreso 2 Ver Editar

**Autores**

Apellidos Nombre

Ver Eliminar

**Documento**

Documento

Seleccionar Documento

**Figura 3.8.2:** Ventana Principal Acta



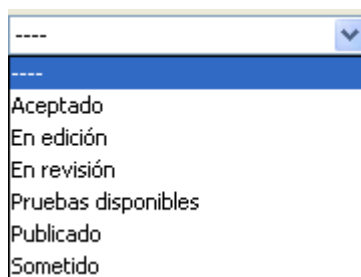
Como se ha dicho, la ventana Actas puede llamar tanto a la ventana de entrada Congreso como a la de Autor. A ambas se las puede invocar mediante el botón editar para insertar un nuevo elemento y posteriormente colocarlo de forma automática en el campo correspondiente.

El botón ver se utilizan para abrir un diálogo donde seleccionar un elemento y una que una vez elegido un elemento de esta segunda ventana se actualiza la información en el campo correspondiente de la primera.

Observamos que el campo Contribución se encuentra señalado como campo obligatorio. Observamos también que aparece el botón seleccionar, utilizado simplemente para poder seleccionar el fichero que vamos a insertar mediante un navegador de ficheros. También aparecen los botones editar, eliminar y cancelar comentados en la especificación de requisitos.

Cabe destacar también el uso de las listas enumeradas de valores, como podemos observar en el campo Estado de esta ventana. De esta forma el usuario simplemente tiene la opción de seleccionar un elemento existente en la lista.

Esto se ha realizado para cumplir un requisito expuesto por el cliente. En la *Figura 3.8.3* se muestra como se utiliza dicha lista de valores.



**Figura 3.8.3:** Combo de ejemplo

### **3.8.1.2 Ventana Principal Integrantes**

Esta ventana es la encargada de la inserción y mantenimiento de la información de los integrantes.

A continuación en *Figura 3.8.4* se muestra una captura de la ventana en cuestión, para su posterior explicación.

Observamos que los campos Apellidos, Nombre y Fecha de Ingreso se encuentran señalados como campos obligatorios. Observamos también que aparece el botón seleccionar, utilizado simplemente para poder seleccionar la imagen que vamos a insertar mediante un navegador de ficheros. También aparece los botones insertar, borrar y cancelar comentados en la especificación de requisitos.

The image shows a web form titled 'Ventana Principal Integrante' divided into two sections: 'Datos Personales' and 'Contacto'.

**Datos Personales:**

- Nombre: [Text Field]
- Apellidos: [Text Field]
- Tratamiento: [Dropdown Menu]
- Categoría: [Text Field]
- N.I.F.: [Text Field]
- Sexo: ☐ Hombre ☐ Mujer
- N.R.P.: [Text Field]
- Espec.: [Dropdown Menu]
- Fecha Nacimiento: [Text Field]
- Universidad: [Text Field]
- Dpto.: [Text Field]

**Contacto:**

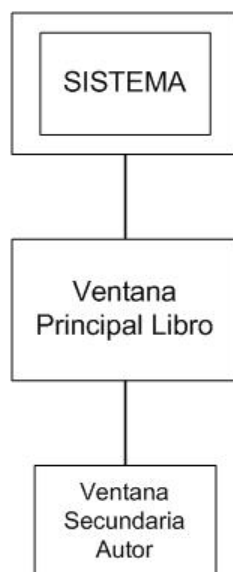
- Dirección: [Text Field]
- Provincia: [Text Field]
- Pais: [Dropdown Menu]
- Localidad: [Text Field]
- Email: [Text Field]
- Email2: [Text Field]
- C.P.: [Text Field]
- Teléfono: [Text Field]
- Móvil: [Text Field]
- Fax: [Text Field]
- Web: [Text Field]

**Figura 3.8.4:** Ventana Principal Integrante

### 3.8.1.3 Ventana Principal Libros

Esta ventana es la encargada de la inserción y mantenimiento de la información de los libros. A continuación se muestra mediante la *Figura 3.8.5* las ventanas secundarias que pueden ser llamados desde ella.

Como se ha dicho, la ventana Libro puede llamar a la ventana de entrada de Autor. Se la puede invocar mediante el botón editar para insertar un nuevo elemento y posteriormente colocarlo de forma automática en el campo correspondiente.



**Figura 3.8.5:** Jerarquía Ventana Principal Libro

A continuación en la *Figura 3.8.6* se muestra una captura de la ventana en cuestión, para su posterior explicación.

Libro (Campos obligatorios marcados en negrita)

**Datos del Libro**

**Código**  **ISBN**  **Título**

**Año**  **Páginas**  **Estado**  **Editorial**

**Ciudad**  **País**

**Información Adicional del Libro**

**Autores**

Apellidos	Nombre

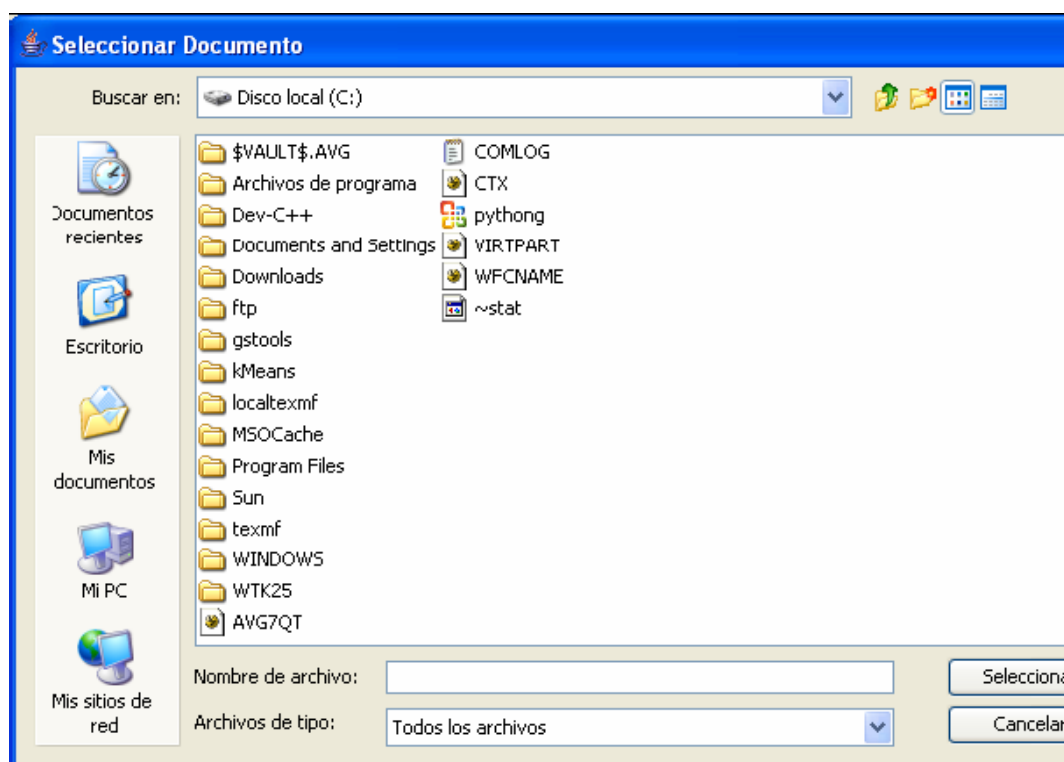
**Documento**

Documento Seleccionado

**Figura 3.8.6:** Ventana Principal Libro

El botón visualizar se utiliza para abrir un diálogo donde seleccionar un autor elemento y una que una vez elegido un elemento de esta segunda ventana se actualiza la información en el campo correspondiente de la primera.

Observamos que el campo Título se encuentra señalado como campo obligatorio. Observamos también que aparece el botón seleccionar, utilizado simplemente para poder seleccionar el fichero que vamos a insertar mediante un navegador de ficheros. Cuando se pulsa ese botón aparece el diálogo que podemos ver en la *Figura 3.8.7*.



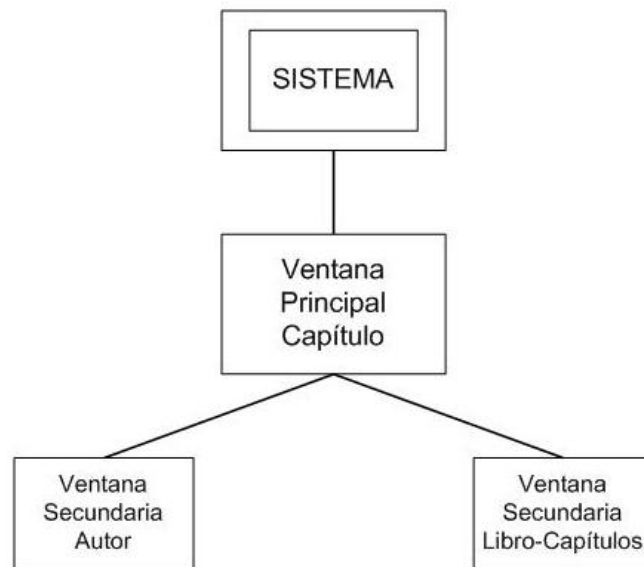
**Figura 3.8.7:** Diálogo selección de documento

También aparece los botones insertar, borrar y cancelar comentados en la especificación de requisitos.

Aparecen dos listas enumeradas de valores para el campo Estado y para el campo País.

#### **3.8.1.4 Ventana Principal Capítulo**

Esta ventana es la encargada de la inserción y mantenimiento de la información de los capítulos. A continuación se muestra mediante la *Figura 3.8.8* las ventanas secundarias que pueden ser llamados desde ella.



**Figura 3.8.8:** Jerarquía Ventana Principal Capítulo

A continuación en la *Figura 3.8.9* se muestra una captura de la ventana en cuestión, para su posterior explicación.

La imagen es una captura de pantalla de una interfaz de usuario web para la gestión de capítulos. El título principal es "Capítulo" con un subtítulo "(Campos obligatorios marcados en negrita)".

La interfaz está organizada en secciones:

- Datos del Capítulo:** Incluye campos para "Código" (valor 4), "Título" (valor "Capitulo 1 Libro Capitulos 2"), "Página Inicial" (valor 220), "Página Final" (valor 228) y "Estado" (valor "Aceptado" en un menú desplegable).
- Referencias del Capítulo:** Se divide en dos sub-secciones:
  - Libro de Capítulos:** Incluye campos para "ISBN" (valor 2) y "Título" (valor "Libro de capitulos 2"), con botones "Ver" y "Editar".
  - Autores:** Incluye una tabla con columnas "Apellidos" y "Nombre", y botones "Ver" y "Eliminar".
  - Documento:** Incluye un campo "Documento Seleccionado" y un botón "Seleccionar Documento".

**Figura 3.8.9:** Ventana Principal Capítulo

Como se ha dicho, la ventana Capítulo puede llamar tanto a la ventana Libro-Capítulos como a Autor. A ambas se las puede invocar mediante el botón editar y ver respectivamente, para insertar un nuevo elemento y posteriormente colocarlo de forma automática en el campo correspondiente.

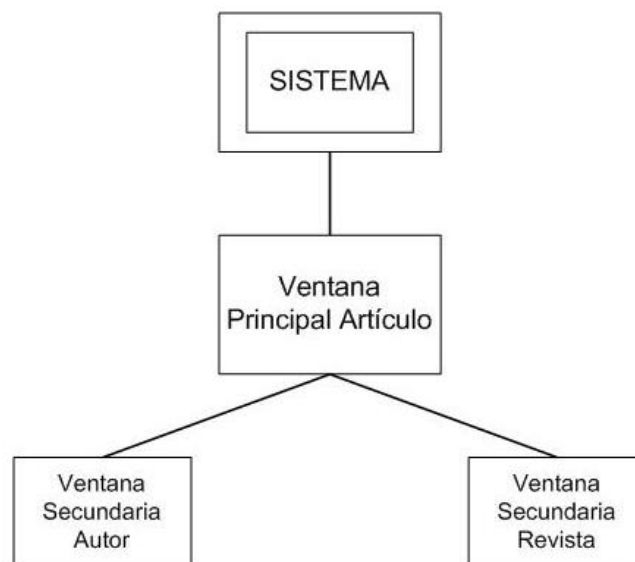
Los botones visualizar se utilizan para abrir un diálogo donde seleccionar un elemento y una que una vez elegido un elemento de esta segunda ventana se actualiza la información en el campo correspondiente de la primera.

Observamos que el campo Título se encuentra señalado como campo obligatorio. Observamos también que aparece el botón seleccionar, utilizado simplemente para poder seleccionar el fichero que vamos a insertar mediante un navegador de ficheros. También aparece los botones insertar, borrar y cancelar comentados en la especificación de requisitos.

Aparece una lista enumerada de valores para el campo Estado, la cual ya ha aparecido en ventanas anteriores.

### 3.8.1.5 Ventana Principal Artículo

Esta ventana es la encargada de la inserción y mantenimiento de la información de los artículos. A continuación se muestra mediante la *Figura 3.8.10* las ventanas secundarias que pueden ser llamados desde ella.



**Figura 3.8.10:** Jerarquía Ventana Principal Artículo

A continuación en *Figura 3.8.11* se muestra una captura de la ventana en cuestión, para su posterior explicación.

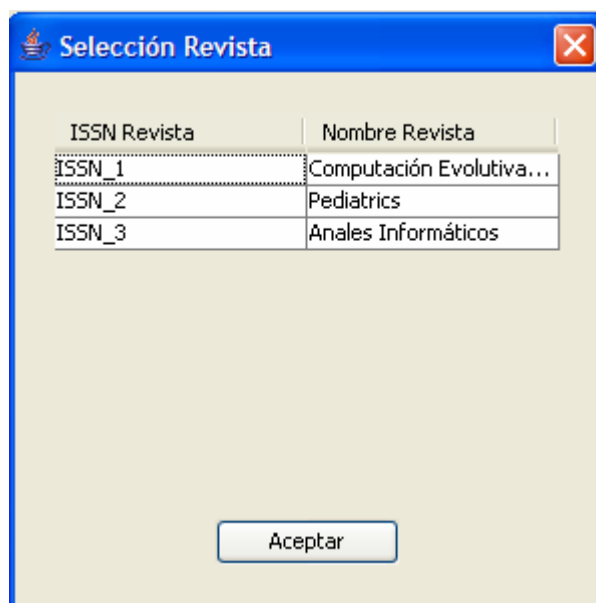
La imagen muestra una interfaz de usuario con dos secciones principales:

- Datos del Artículo:** Incluye campos para "Código" (un campo de texto gris), "Título" (un campo de texto largo), "Año" (un campo de texto), "Volumen" (un campo de texto), "Página Inicial" (un campo de texto), "Página Final" (un campo de texto) y "Estado" (un menú desplegable con "----" y una flecha hacia abajo).
- Referencias del Artículo:** Se divide en tres sub-secciones:
  - Revista:** Incluye "ISSN" (un campo de texto gris con el valor "ISSN\_3") y "Nombre" (un campo de texto con el valor "Anales Informáticos"). Hay botones "Ver" y "Editar" a la derecha.
  - Autores:** Incluye una tabla con columnas "Apellidos" y "Nombre". La tabla está vacía. Hay botones "Ver" y "Eliminar" debajo de la tabla.
  - Documento:** Incluye un campo "Documento Seleccionado" (un campo de texto gris) y un botón "Seleccionar Documento" debajo.

**Figura 3.8.11:** Ventana Principal Artículo

Como se ha dicho, la ventana Artículo puede llamar tanto a la ventana Revista como a Autor. A ambas se las puede invocar mediante el botón editar para insertar un nuevo elemento y posteriormente colocarlo de forma automática en el campo correspondiente.

Los botones visualizar se utilizan para abrir un diálogo donde seleccionar un elemento y una que una vez elegido un elemento de esta segunda ventana se actualiza la información en el campo correspondiente de la primera. Si pulsamos el botón visualizar del campo "ISSN revista" se abre el diálogo que aparece en la *Figura 3.8.12*.



**Figura 3.8.12:** Diálogo Selección de Revista

Observamos que el campo Título se encuentra señalado como campo obligatorio. Observamos también que aparece el botón seleccionar, utilizado simplemente para poder seleccionar el fichero que vamos a insertar mediante un navegador de ficheros. También aparece los botones insertar, borrar y cancelar comentados en la especificación de requisitos.

Aparece una lista enumerada de valores para el campo Estado, la cual ya ha aparecido en ventanas anteriores.

#### **3.8.1.6 Ventana Secundaria Autor**

Esta ventana es la encargada de la inserción y mantenimiento de la información de los autores. Es llamada desde el botón editar de las ventanas principales Libro, Artículo, Capítulo y Acta.

A continuación en la *Figura 3.8.13* se muestra una captura de la ventana en cuestión, para su posterior explicación.



**Datos Personales**

Apellidos  Nombre  Tratamiento

---

**Información Laboral**

Categoría  Universidad  Departamento

---

**Ubicación**

Dirección  Ciudad

Provincia  País   Código Postal

---

**Contacto**

E-mail  Telefono  Fax

**Figura 3.8.13:** Ventana Secundaria Autor

Observamos que los campos Apellidos y Nombre se encuentran señalados como campos obligatorios.

También aparece el botón Visualizar Miembros el cual muestra un diálogo mediante el cual podemos insertar un miembro como autor de forma mucho más cómoda ya que aparecerá una tabla con los miembros del grupo y sólo habrá que seleccionar la fila correspondiente y pulsar aceptar. El diálogo que aparece al pulsar el botón podemos verlo en la *Figura 3. 8. 14*.

**Selección Miembro**

Apelli...	Nombre	Trato	Cate...	Univ...	Depa...	Direc...	Provi...	Locali...	País	ZIP	E-mail	Telef...	FAI
Catalá ...	Maria	Lda.	Ayudante	Córdoba	Análisis ...	La suya	Córdoba	Córdoba	España	789	i12caca...	957223...	9572
García S...	Antonio	BSc.	Ayudante	Córdoba	Análisis ...	Calle An...	Córdoba	Córdoba	España	779	i12anga...	957887...	9957
Lozano ...	Esteban	Dr.	Director	Córdoba	Análisis ...	Calle Es...	Córdoba	Córdoba	España	305	i12eslor...	957232...	9957
Luque S...	Carlos	Prof.	Becario	Málaga	Análisis ...	Calle Ca...	Córdoba	Córdoba	España	200	i12calus...	957444...	9957
Martin	David	Profesor	Colabor...	Sevilla	Analisis ...	la calle ...	Córdoba	Córdoba	España	103	i02dam...	957122...	9957
Martíne...	Fernando	Ldo.	Jefe	Córdoba	Análisis ...	Calle Fe...	Córdoba	Córdoba	España	705	i12fema...	957777...	9957
Moreno	Manuel	Mr.	Ayudante	Córdoba	Analisis ...	la calle	Córdoba	Córdoba	España	123	i02mos...	957888...	9957
Muriel Z...	Javi	Ldo.	Ayudante	Córdoba	Análisis ...	Mi calle	Córdoba	Córdoba	España	456	i12muz...	957112...	9571
Olivera ...	Daniel	Prof.	Ayudante	Córdoba	Matemá...	Calle Da...	Córdoba	Córdoba	España	300	i12daga...	957222...	9957
Sánche...	Beatriz	Mss.	Becaria	Córdoba	Análisis ...	Calle Be...	Córdoba	Córdoba	España	100	i12besa...	957335...	9957

**Figura 3.8.14:** Diálogo de selección de miembro

### 3.8.1.7 Ventana Secundaria Congreso

Esta ventana es la encargada de la inserción y mantenimiento de la información de los congresos. Es llamada desde el botón editar de la ventana principal Acta.

A continuación en la *Figura 3.8.15* se muestra una captura de la ventana en cuestión, para su posterior explicación.

Observamos que los campos Nombre, Abreviatura y Organización se encuentran señalados como campos obligatorios.

**Congreso** (Campos obligatorios marcados en negrita)

**Datos del Congreso**

**Código** 2 **Nombre** Congreso 2

Abreviatura C2 Organización Organización 2 Tipo I Estado ----

Ciudad Granada País ---- Fecha Inicio 2006-02-20

Fecha Final 2006-02-23 Deadline Marzo 2006 Intención intencion 2

**Información Adicional del Congreso**

ISBN ISBN-77777-77777 Título Acta Congreso Extraordinario 2

URL http://www.congreso2.es Editorial Editorial 2 Editores Editores 2

**Figura 3.8.15:** Ventana Secundaria Congreso

Esta ventana dispone de dos campos que utilizan las listas enumeradas, que son Estado y País.

### 3.8.1.8 Ventana Secundaria Revista

Esta ventana es la encargada de la inserción y mantenimiento de la información de las revistas. Es llamada desde el botón editar de la ventana principal Artículo.

A continuación en la *figura 3.8.16* se muestra una captura de la ventana en cuestión, para su posterior explicación.

Observamos que los campos ISSN, Nombre y Abreviatura se encuentran señalados como campos obligatorios.

**Revista** (Campos obligatorios marcados en negrita)

**Datos de la Revista**

**ISSN** ISSN\_1 Editorial Editorial Cúspide **Abreviatura** ECCEE

Nombre Computación Evolutiva en España

**ISI (Institute for Scientific Information)**

Año 2000 Factor de Impacto 1.77 Total citas 32000

Índice inmediato 0.453 Número de Artículos 2000 Vida media 2.3

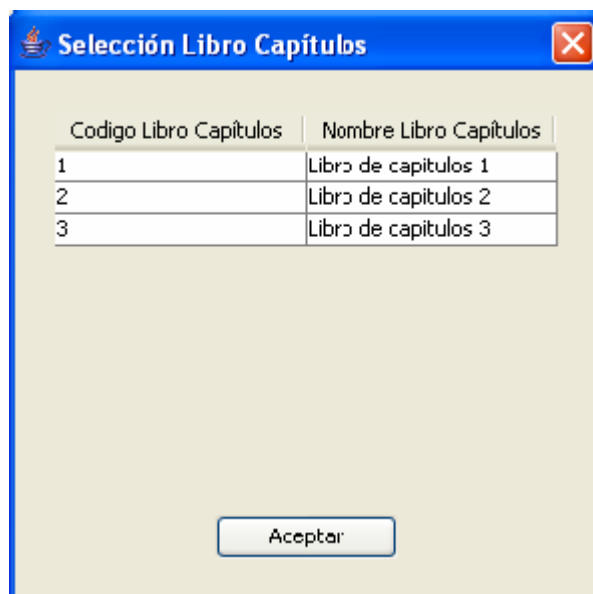
InsertarISI VerISI BorrarISI ModificarISI

**Figura 3.8.16:** Ventana Secundaria Revista

### 3.8.1.9 Ventana Secundaria Libro-Capítulo

Esta ventana es la encargada de la inserción y mantenimiento de la información de libros cuyos capítulos son mantenidos por el sistema. Es llamada desde el botón editar de la ventana principal Capítulo.

A continuación en la *Figura3. 8.17* se muestra una captura de la ventana en cuestión, para su posterior explicación.



**Figura 3.8.17:** Ventana Secundaria Libro - Capítulo

## 4. Formulario de Visualización

# 4.1

## DEFINICIÓN DEL PROBLEMA

### 4.1.1 DEFINICIÓN DEL PROBLEMA REAL

El problema que se plantea consiste en la revisión de los formularios de visualización de la aplicación encargada de llevar a cabo la Gestión de la Información de los Grupos de Investigación, revisando los formularios desarrollados por el grupo encargado de llevar a cabo su desarrollo en la fase anterior (fase 1), estamos hablando del grupo 14.

Para ello tendremos que adaptar los formularios existentes a las nuevas modificaciones producidas sobre la BD, así como terminar la funcionalidad de los mismos.

Se dotará al formulario de visualización nuevas funcionalidades, como la búsqueda por patrones introducidos por el usuario, aumentando así las posibilidades de nuestra aplicación.

### 4.1.2 DEFINICIÓN DEL PROBLEMA TÉCNICO.

Los formularios obtenidos en esta nueva fase tienen que estar basados en los que ya tenemos, adaptando los existentes a las nuevas necesidades, así como llevando a cabo una mejora de los mismos.

Será necesario conectar los formularios con la BD ya existente, así como adaptarlos a la nueva BD desarrollada en la fase 2 de nuestro proyecto.

El interfaz de los formularios tendrá que seguir la misma filosofía que el interfaz de la aplicación desarrollado hasta ahora, así como el mismo interfaz usado en los formularios originales.

Los formularios deben de permitir la visualización de la información relativa a la información perteneciente a los miembros, publicaciones, congresos, etc. En definitiva, toda la información relacionada con los grupos de investigación que almacene nuestro sistema.

El sistema nos debe de permitir el listado de todas las tuplas de una tabla determinada de la BD, así como llevar a cabo búsqueda mediante la elección de algún patrón de búsqueda.

La información obtenida en la visualización debe poder ser ordenada por algún patrón, así como mostrar un mayor detalle del registro mostrado. Además de ser una visualización sencilla y clara, usando para ello una interfaz intuitiva y amigable.

Estas es la definición de nuestra funcionalidad a implementar, para entender la envergadura de nuestro proyecto el lector tendrá que revisar la documentación de cada una de las tareas efectuadas a lo largo de las fases de desarrollo del proyecto.

# 4.2

## OBJETIVOS

En la presente sección vamos a describir los objetivos que se pretenden alcanzar con el desarrollo de la tarea asignada en la fase 2 del proyecto. Para ello vamos a distinguir entre los objetivos formales, los cuales son los más generales a conseguir con el desarrollo de nuestra tarea, y los objetivos operacionales, los cuales son los objetivos que deben de satisfacer la funcionalidad a desarrollar.

### 4.2.1 OBJETIVOS FORMALES

Como objetivos formales a destacar en la realización de nuestra tarea podríamos destacar los siguientes:

- Perfeccionamiento del lenguaje de programación Java.
- Aprendizaje del gestor de Base de Datos MySQL, así como la conexión de Java con el mismo.
- Conocer la forma de trabajar en un proyecto de amplia envergadura con varios equipos de trabajo cada uno con su tarea y objetivos establecidos.
- Poner en práctica los conocimientos adquiridos a lo largo de nuestros estudios universitarios.



- Perfeccionar el desarrollo de documentaciones técnicas, así como el uso de lenguajes de modelado como UML.2.2 Objetivos funcionales

A continuación vamos a hacer un listado de los objetivos que debe de cumplir la aplicación a desarrollar, en nuestro caso, los formularios de visualización de la aplicación de Gestión de la Información de los Grupos de Investigación. El listado es el siguiente:

- Revisar una funcionalidad ya existente de un proyecto, revisando el análisis, diseño y codificación del mismo.
- Dotar de una mayor funcionalidad a uno de los modulos que integran una aplicación modular.
- Llevar a cabo una conexión con una BD implementada usando MySQL a través de JDBC de Java.
- Desarrollar una interfaz común a lo ya existente.
- Desarrollar una funcionalidad que se pueda integrar perfectamente a lo ya existente.
- Permitir llevar a cabo la visualización total o cribada de la información almacenada por el grupo de investigación que disponga de la aplicación.
- Permitir llevar a cabo una ordenación de la información consultada.
- Construir un software que reúna todas las características deseables, características tales como fiabilidad, robustez, facilidad de uso, etc.

# 4.3

## RESTRICCIONES

En el presente epígrafe vamos a describir las restricciones con las que nos encontramos a la hora de llevar a cabo el análisis y desarrollo de nuestro sistema, para ello vamos a distinguir entre restricciones datos y entre restricciones estratégicos. Es conveniente señalar que los factores datos son aquellos impuestos por nuestro directores de proyecto; *Irene Luque Ruiz* y *Miguel Angel Gómez Nieto*, y los factores estratégicos son aquellos pertenecientes a la propia naturaleza del proyecto.

### 4.3.1 FACTORES DATOS

Se consideran los siguientes:

- El sistema debe presentar una interfaz de usuario homogénea e intuitiva. Desde la interfaz será accesible toda la funcionalidad del sistema.
- El sistema deberá ser capaz de manejar toda la información correspondiente al grupo de investigación.

- Debido a la naturaleza multiplataforma del sistema, y a la modularidad exigida, el desarrollo del sistema debe realizarse en el lenguaje de programación Java. Este lenguaje proporciona un paradigma de programación orientado a objetos. Tiene un API que proporciona una fuente de recursos lo suficientemente rica y en continua expansión para cubrir todas las necesidades de la fase de codificación del sistema.
- Restricciones de tiempo: para realizar la fase actual en la que nos encontramos, la segunda, es de 4 semanas.
- Restricciones de tipo presupuestario, debido a que se trata de una práctica de la carrera.
- Limitaciones de tipo humano, disponiendo de dos profesores que ejercen de directores y dos estudiantes de Ingeniería, para realizar ésta funcionalidad, ya que para desarrollar la aplicación completa se dispone de todos los alumnos de 5 curso de Ingeniería Informática.
- Limitaciones de hardware y software, disponiendo de los recursos de la universidad y de los personales.

### 4.3.2 FACTORES ESTRATÉGICOS

Se consideran los siguientes:

- El lenguaje de programación a utilizar es Java, usando la SDK 1.4 o superior, ya que fue el que se eligió por consenso al comienzo de la aplicación y debido a que se trata de una revisión de una funcionalidad ya creada tenemos que utilizar el mismo lenguaje de programación.

- Se utilizará la tecnología que MySql 5.0 proporciona para el desarrollo de la aplicación. A su vez, para lograr la interacción entre el sistema y la base de datos se utilizará JDBC, como puente de conexión entre ambos.
- Para optimizar la fase de codificación se utilizará el entorno gráfico de desarrollo Eclipse-SDK 3.2.1 para Java. Un entorno integral de este tipo mejora notablemente la codificación de los distintos componentes del sistema, ya que proporciona: herramientas de gestión de la configuración, diseñadores de entornos gráficos de usuario, plantillas de componentes reutilizables, configuración del acceso a librerías de clases de terceros y sistemas de depuración de código, entre otras muchas utilidades.

# 4.4

## RECURSOS

En esta sección vamos a identificar los diferentes recursos con los que disponemos para llevar a cabo la tarea asignada en la fase 2, en la cual nos encontramos. Para ello distinguiremos entre recursos humanos, hardware y software.

### 4.4.1 RECURSOS HUMANOS

Esta parte del sistema será realizada por el grupo 15 perteneciente al proyecto *“Gestión de información para grupos de investigación”*. Los componentes de dicho grupo son Rafael Montero Muñoz y Javier Mejías Real, bajo las pautas marcadas por los profesores de la asignatura AIS: Dña. Irene Luque Ruiz, Titular de Universidad y D.Miguel Ángel Gómez Nieto, Catedrático E.U.

### 4.4.2 RECURSOS HARDWARE

Los recursos hardware utilizados para el desarrollo del módulo del sistema descrito en el presente documento son los siguientes:

- Un PC AMD Atolón 64 3.2GHz, 1024 Mb RAM y 160Gb de Hd.
- Un Portátil AMD Turion 2000MHz. 1024 Mb de RAM y 200GB de Hd.

### 4.4.3 RECURSOS SOFTWARE

Para la realización de los diversos apartados del sistema se dispondrá de los siguientes recursos software:

- **MS Word 2003:** procesador de texto, muy útil a la hora de realizar la documentación del proyecto.
- **MS Visio 2003:** editor de diagramas, que nos será de utilidad a la hora de realizar los diagramas de flujo de datos en nuestra documentación.
- **Java 2 de Sun Microsystems**, que compone todas aquellas herramientas necesarias para la programación en Java. (librerías, compiladores, Máquina Virtual de Java, etc...)
- **Eclipse 3.2.1**, IDE para Java.
- **Microsoft Windows Xp**, Sistema Operativo bajo el que realizaremos el proyecto. La elección del SO no es relevante ya que nuestra aplicación será multiplataforma.
- **Mozilla Firefox**, explorador Web para todo lo relacionado con la navegación por Internet.
- **Adobe Reader 7.0**, visor de documentos .pdf.
- **MySQL 5:** Gestor y servidor de BD.

# 4.5

## ESPECIFICACIÓN DE REQUISITOS

### 4.5.1 IDENTIFICACIÓN DE REQUISITOS

Para llevar a cabo una correcta especificación de requisitos lo primero que vamos a hacer es llevar a cabo una descripción inicial de los mismos en un lenguaje formal, para llevar a cabo una especificación de los mismos.

Los requisitos que podemos identificar a priori son los siguientes:

- *Identificador:* fun001
- *Nombre:* listar la información relativa al grupo de investigación.
- *Descripción:* visualización de los artículos, libros, capítulos, actas, congresos, miembros del grupo de investigación, autores, tesis, financiación de proyectos y en definitiva toda la información perteneciente al grupo de investigación así como a los miembros del mismo. Permitiendo listar toda la información relativa a un determinado objeto sin realizar criba.
- *Tipo:* Requisito funcional

- *Identificador:* fun002
  - *Nombre:* buscar y visualizar información relativa al grupo de investigación.
  - *Descripción:* búsqueda de información, así como la visualización de los resultados de la búsqueda, llevada a cabo mediante patrones de búsqueda.
  - *Tipo:* Requisito funcional.
- 
- *Identificador:* fun003
  - *Nombre:* mostrar la información detallada.
  - *Descripción:* mostrar la información detallada de un determinado ítem, mostrando más allá de la información perteneciente al mismo, ya que mostrará información acerca de las relaciones de éste con otros elementos.
  - *Tipo:* Requisito funcional.
- 
- *Identificador:* fun004
  - *Nombre:* ordenar información por campos.
  - *Descripción:* ordenar la información mostrada por alguno de sus campos.
  - *Tipo:* Requisito funcional.
- 
- *Identificador:* in001.
  - *Nombre:* adaptación de la interfaz con la aplicación.



- *Descripción:* la aplicación debe de adaptarse perfectamente con la interfaz ya diseñada para la aplicación final, además de que quede en perfecta sintonía con lo ya desarrollado.
- *Tipo:* Requisito de interfaz

## 4.5.2 FORMALIZACIÓN DE REQUISITOS

A continuación vamos a llevar a cabo una formalización de los requisitos identificados y descritos con anterioridad. Para ello haremos uso del lenguaje de modelado UML, llevando a cabo primeramente una identificación de los actores del sistema, y posteriormente la identificación de los diferentes casos de uso que haya en el mismo, mostrando como interactúan ambos mediante los diagramas de caso de uso oportunos.

### 4.5.2.1 Identificación de los actores

En nuestra aplicación, formularios de visualización podemos encontrar un único actor, el actor principal, el cual estará desempeñado por el cliente de la aplicación, los cuales podrán ser cualquiera de los miembros del grupo de Investigación que haga uso de la aplicación.

**Tabla 4.5.1:** Descripción del actor cliente.

ACTOR: cliente	
Identificador	001
Descripción	El actor cliente es el encargado de interactuar con la aplicación para llevar a cabo la visualización de la información disponible en la BBDD.

#### 4.5.2.2 Casos de uso

En esta sección vamos a identificar las formas en las que el usuario va a interactuar con el sistema, para ello vamos a identificar los diferentes casos de uso que encontramos en nuestra aplicación, es decir, las diferentes formas que va a tener el usuario para llevar a cabo la visualización de información a través de los formularios.

##### 4.5.2.2.1 Casos de uso Inicial

Vamos pues a llevar a cabo una descripción de los diferentes casos de uso iniciales identificados para más tarde llevar a cabo una descripción más detallada de los mismos.

**Tabla 4.5.2** Descripción inicial caso de uso “Definir búsqueda”

Caso de uso: Definir búsqueda	
Identificador:	1
Id-requisito:	fun001, fun 002
Descripción:	seleccionar el elemento a buscar así las opciones de la misma, indicando si se desea listar todos los item o buscar mediante patrones de búsqueda, los cuales también tendrán que ser seleccionados.

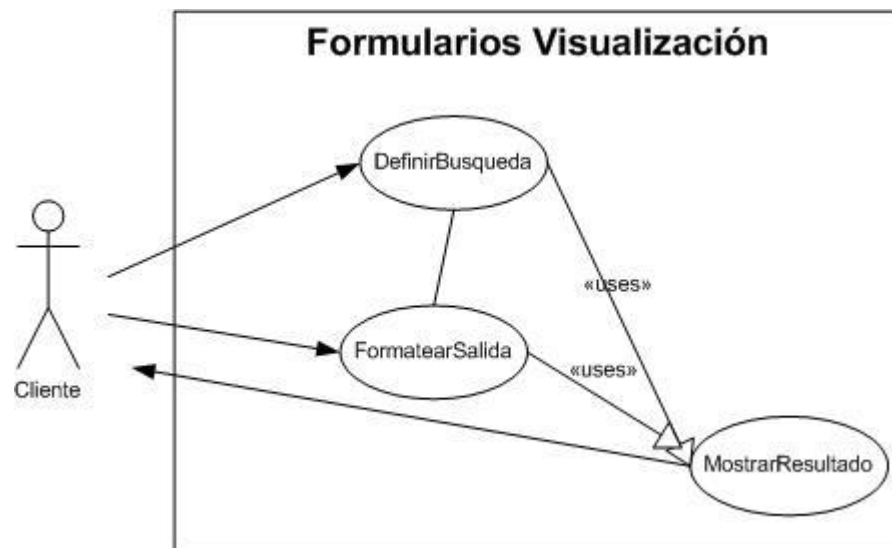
**Tabla 4.5.3** Descripción inicial caso de uso “Formatear salida”

Caso de uso: Formatear salida	
Identificador:	2
Id-requisito:	fun004
Descripción:	Darle formato a la salida de la búsqueda permitiendo ordenar la salida por algún campo.

**Tabla 4.5.4** Descripción inicial caso de uso “Mostrar Resultado”

Caso de uso: Mostrar Resultado	
Identificador:	3
Id-padre:	1, 2
Id-requisito:	fun003
Descripción:	Mostrar los resultados obtenidos por la búsqueda permitiendo mostrar con mayor nivel de detalle los ítem.

A continuación podemos ver en el siguiente diagrama de caso de uso como el actor principal interactúa con los casos de uso identificados anteriormente:

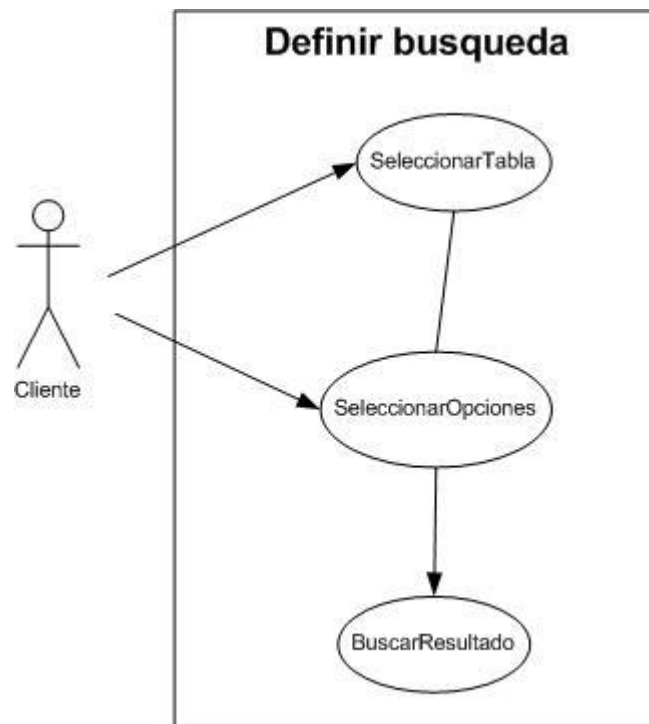
**Figura 4.5.1:** Diagrama Caso de Uso de Inicial

#### 4.5.2.2.2 Refinamiento Casos de uso

Una vez que hemos descrito los casos de uso iniciales y hemos visto como el actor de nuestro sistema interactúa con él, vamos a llevar a cabo un refinamiento de los casos de uso anteriores.

#### ***Subsistema “Definir Búsqueda”***

Un refinamiento del caso de uso “Definir Búsqueda” lo podemos observar en el siguiente diagrama:



**Figura 4.5.2:** Diagrama Caso de Uso “Definir Búsqueda”

Una vez mostrado el diagrama de caso de uso vamos a realizar una descripción tabular de cada uno de los diagramas que lo componen.

**Tabla 4.5.5.** Caso de uso “Seleccionar Tabla”.

Caso de uso: Seleccionar tabla	
Identificador	1-1
Id-Padre	1
Id-Requisito de soporte	func001, 002
Descripción	Selecciona la tabla de la BD sobre la cual se quiere realizar la búsqueda.
Actores principales	Cliente
Actores secundarios	
Precondiciones	Que el cliente haya seleccionado en la aplicación la opción

	de visualizar información.
Flujo principal	1- Cliente despliega el menú con las diferentes tablas. 2- Se muestran las diferentes opciones de dicha tabla.
Postcondicione	Se habrán mostrado las diferentes opciones para realizar una búsqueda.
Flujos alternativos	Ninguno

**Tabla 4.5.6.** Caso de uso “Seleccionar Tabla”.

Caso de uso: Seleccionar opciones	
Identificador	1-2
Id-Padre	1
Id-Requisito de soporte	func001, 002
Descripción	Selecciona las opciones de búsqueda, eligiendo entre un listado de todos los item de la tabla seleccionada o introducir los patrones de búsqueda.
Actores principales	Cliente
Actores secundarios	
Precondiciones	Que el cliente haya seleccionado un item para consultar (libro, articulo, etc...)
Flujo principal	1- Cliente elige listar todo o cribar 1.1- Si elige cribar introduce los datos que formarán el patrón de búsqueda.
Postcondicione	Se tendrán las opciones pertinentes para llevar a cabo la búsqueda.
Flujos alternativos	Seleccionar una nueva tabla a consultar.

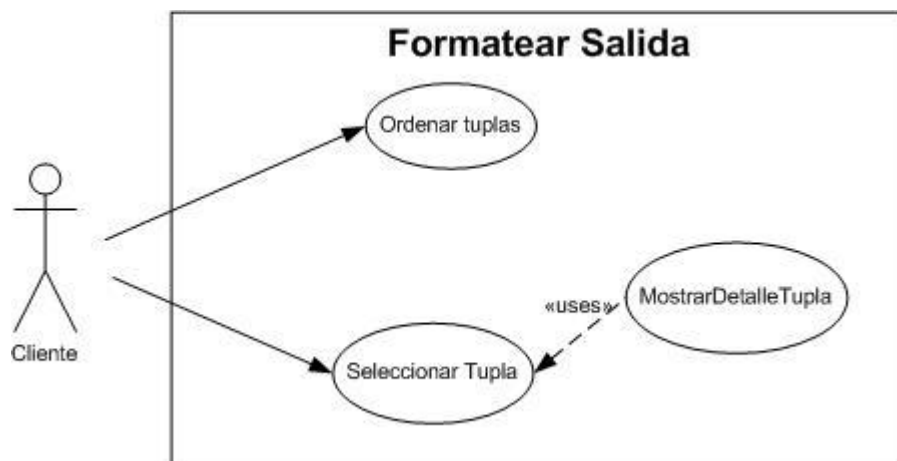
**Tabla 4.5.7.** Caso de uso “Seleccionar Tabla”.

Caso de uso: Buscar resultado	
Identificador	1-3
Id-Padre	1

<b>Id-Requisito de soporte</b>	func001, 002
<b>Descripción</b>	Lleva a cabo la búsqueda con las opciones indicadas
<b>Actores principales</b>	Ninguno.
<b>Actores secundarios</b>	
<b>Precondiciones</b>	Que el cliente haya seleccionado un ítem para consultar (libro, artículo, etc...) y las características de la búsqueda.
<b>Flujo principal</b>	1- Se lleva a cabo la conexión con la BD 2- Se buscan los elementos correspondientes a las opciones de búsqueda seleccionadas.
<b>Postcondicione</b>	Se tendrán los registro resultado de la búsqueda.
<b>Flujos alternativos</b>	Excepción al conectarse con la BD. Excepción al buscar información en la BD. No encontrar información.

### **Subsistema “Formatear Salida”**

El siguiente caso de uso que vamos a refinar es el caso de uso “Formatear salida”, el cual podemos verlo en su refinamiento inicial en la figura 4.5.1.



**Figura 4.5.3:** Diagrama Caso de Uso “Formatear Salida”

El diagrama de caso de uso es el que nos encontramos a continuación, en el cual podemos ver como el actor interactúa con los diferentes casos de uso en los que se refina el caso de uso general “Formatear Salida”

**Tabla 4.5.8.** Caso de uso “Ordenar tuplas”.

Caso de uso: Ordenar tuplas	
Identificador	2-1
Id-Padre	2
Id-Requisito de soporte	func004
Descripción	Lleva a cabo una ordenación de los resultados obtenidos en la búsqueda realizada.
Actores principales	Cliente
Actores secundarios	
Precondiciones	Que el cliente haya llevado a cabo una búsqueda y estén todavía los registros resultantes de la misma.
Flujo principal	1- Se elige el campo por el cual llevar a cabo la ordenación.
Postcondicione	Se tendrán los registro resultado de una búsqueda anterior ordenados por un campo determinado.
Flujos alternativos	Realizar alguna otra búsqueda. Seleccionar alguna otra opción de formateo de salida.

**Tabla 4.5.9.** Caso de uso “Seleccionar tuplas”.

Caso de uso: Seleccionar tupla	
Identificador	2-2
Id-Padre	2
Id-Requisito de soporte	In001
Descripción	Lleva a cabo una selección de una de las tuplas obtenidas como resultado de la búsqueda.
Actores principales	Cliente
Actores secundarios	
Precondiciones	Que se haya realizado alguna búsqueda que haya devuelto algún resultado.
Flujo principal	1- Seleccionar una fila con los datos del registro a mostrar detallados.
Postcondicione	Se tendrán los registro resultado de una búsqueda anterior

	ordenados por un campo determinado.
Flujos alternativos	Realizar alguna otra búsqueda. Seleccionar alguna otra opción de formateo de salida.

**Tabla 4.5.10.** Caso de uso “Mostrar detalle registro”.

Caso de uso: Mostrar detalle registro	
Identificador	2-3
Id-Padre	2
Id-Requisito de soporte	Fun003
Descripción	Lleva a cabo una visualización detallada del registro seleccionado.
Actores principales	Cliente
Actores secundarios	
Precondiciones	Que se haya seleccionado alguna tupla
Flujo principal	1- (Incluye) Seleccionar tupla. 2- Elegir la opción de visualizar los datos detallados.
Postcondicione	Se habrán mostrado los datos detallados.
Flujos alternativos	Excepción al acceder a la BD.

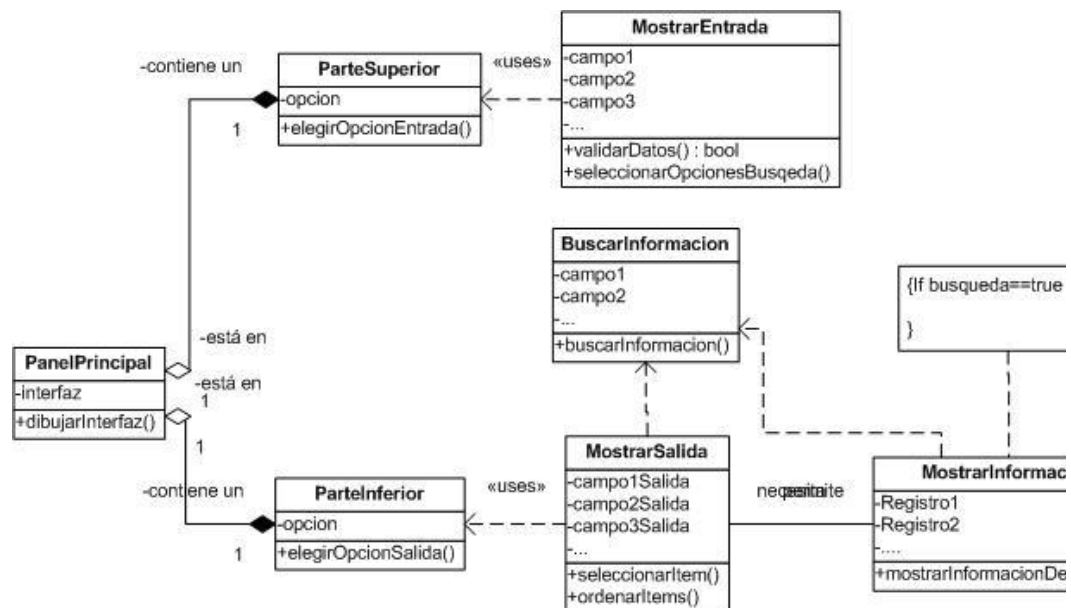


# 4.6

## ANÁLISIS DEL SISTEMA

Una vez hemos descrito los requisitos vamos a llevar a cabo un análisis del sistema, generando un modelo de análisis que va a servir de base para el diseño del mismo, centrándonos principalmente en qué debe de hacer el sistema y dejando el cómo para la fase de diseño.

A continuación podemos ver el diagrama de clases de análisis, el cual nos muestra una abstracción de una o varias clases que se obtendrán en el diseño del sistema, centrándose principalmente en los requisitos funcionales.



**Figura 4.6.1.** Diagrama de Clases de Análisis

Una vez que hemos mostrado el diagrama de clase de análisis vamos a realizar una descripción tabular de cada una de las clases que aparecen en el mismo.

**Tabla 4.6.1.** Especificación de la clase PanelPrincipal

CLASE DE ANÁLISIS: PanelPrincipal	
Descripción	Será la clase encargada de actuar de interfaz entre el usuario y la aplicación, para permitirle visualizar la información requerida.
Atributos	interfaz: este atributo es un atributo abstracto que representa al conjunto de atributos que conforman la interfaz gráfica.
Operaciones	-dibujarInterfaz(): es un método abstracto por el cual se dibuja la interfaz de la aplicación encargada de visualizar los formularios de visualización.
Estereotipos	Interfaz
Valores etiquetados	

**Tabla 4.6.2.** Especificación de la clase PanelSuperior

CLASE DE ANÁLISIS: ParteSuperior	
Descripción	Será la clase encargada de permitir al usuario seleccionar el elemento sobre el que realizar la búsqueda.
Atributos	-opcion: representa el tipo de item sobre el que realizar búsquedas, por ejemplo, revistas, libros, etc.
Operaciones	-elegirOpcionEntrada(): método que permite al usuario elegir la opción del item a consultar.
Estereotipos	Control
Valores etiquetados	

**Tabla 4.6.3.** Especificación de la clase MostrarEntrada

CLASE DE ANÁLISIS: MostrarEntrada	
Descripción	Clase encargada de mostrar la información necesaria para llevar a cabo patrones de búsqueda dependiendo del item que se haya seleccionado.
Atributos	-campo1: será uno de los atributos a través del cual se podrá realizar un búsqueda.  -campo2: será otro de los atributos.  Etc.
Operaciones	-validarDatos(); operación que valida los datos de búsqueda.  -seleccionarOpcionesBusqueda(): permite elegir entre una búsqueda u otra.
Estereotipos	Interfaz
Valores etiquetados	

**Tabla 4.6.4.** Especificación de la clase PanelInferior

CLASE DE ANÁLISIS: ParteInferior
----------------------------------

Descripción	Será la clase encargada de permitir al usuario elegir las opciones de visualización.
Atributos	-opcion: representa el tipo de visualización.
Operaciones	-elegirOpcionSalida(): método que permite al usuario elegir la opción de salida
Estereotipos	Control
Valores etiquetados	

**Tabla 4.6.5** Especificación de la clase MostrarSalida

CLASE DE ANÁLISIS: MostrarSalida	
Descripción	Será la clase encargada de mostrar la información esquematizada resultado de la consulta.
Atributos	-campoSalida1: uno de los datos obtenidos. -campoSalida2: otro de los datos obtenidos en la consulta. Etc.
Operaciones	-seleccionarItem(): método que permite al usuario elegir uno de los ítem obtenidos al realizar la búsqueda. -ordenarItems(): método que permite ordenar los resultados de la salida obtenida.
Estereotipos	Interfaz
Valores etiquetados	

**Tabla 4.6.6** Especificación de la clase BuscarInformacion

CLASE DE ANÁLISIS: BuscarInformacion	
Descripción	Será la clase encargada de buscar en la BD la información solicitada.
Atributos	-campo1: uno de los campos de búsqueda.

	-campo2: otro de los campos de búsqueda. Etc.
Operaciones	-buscarInformacion(): método que lleva a cabo las operaciones de búsqueda en la BD teniendo en cuenta los patrones de búsqueda elegidos por el usuario.
Estereotipos	Entidad
Valores etiquetados	

**Tabla 4.6.7** Especificación de la clase MostrarInformacion

CLASE DE ANÁLISIS: MostrarInformacion	
Descripción	Será la clase encargada de visualizar un ítem determinado de forma detallada.
Atributos	-Registro1: abstracción de los datos a visualizar de uno de los registros.  -Registro2: abstracción de los datos a visualizar de otro de los registros.  Etc.
Operaciones	-mostrarInformacionDetallada(): método que muestra la información perteneciente a uno o más registros buscados.
Estereotipos	Interfaz
Valores etiquetados	

# 4.7

## DISEÑO DEL SISTEMA

En este apartado vamos a llevar a cabo el diseño de nuestro sistema, para ello haremos un refinamiento de las clases obtenidas en el análisis del mismo, así como la utilización de otros diagramas que nos permitan un correcto diseño del mismo.

### 4.7.1 DIAGRAMA DE PAQUETES

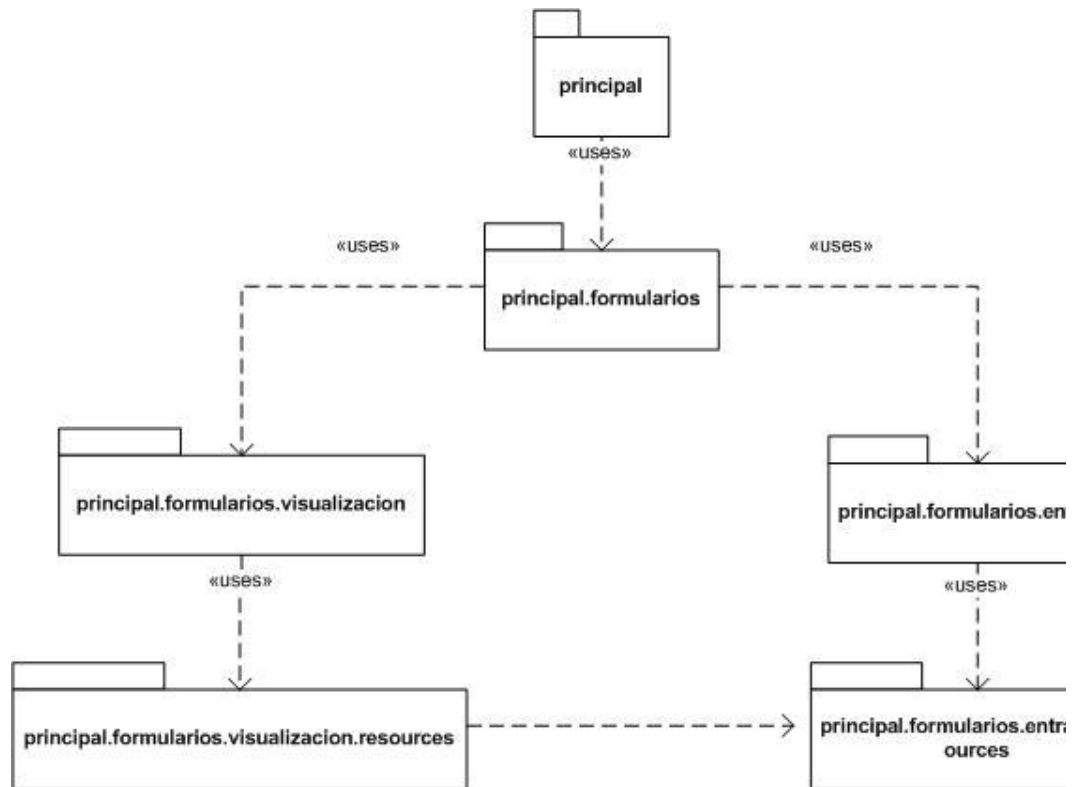
A continuación vamos a mostrar el diagrama en UML que muestra los diferentes paquetes de nuestra aplicación y como están relacionados entre ellos.

Hemos dividido nuestra aplicación en 6 paquetes, habiendo agrupado en cada uno de ellos las clases que hemos creído convenientes, además de haber identificado cada uno de los paquetes con un nombre que a nuestro parecer da una idea del objetivo del paquete.

Los paquetes que hemos obtenido son los siguientes:

- principal
- principal.formularios
- principal.formularios.entrada
- principal.formularios.entrada.resources

- principal.formularios.visualizacion
- principal.formularios.visualizacion.resources



**Figura 4.7.1** Diagrama de Paquetes

Una vez que hemos mostrado el diagrama de paquetes vamos a realizar una breve descripción de cada uno de ellos.

#### **4.7.1.1 PAQUETE PRINCIPAL**

El criterio que hemos elegido para agrupar las clases en este paquete es el de agrupar aquellas clases que conforman la ventana principal de la aplicación, estando formado dicho paquete por dos clases:

- Principal (publica)
- MarcoPrincipal(publica)

#### **4.7.1.2 PAQUETE PRINCIPAL.FORMULARIOS**

Al igual que ocurría con el paquete anterior, hemos decidido crear éste paquete para agrupar aquellas clases relacionadas con la creación de dos zonas en la ventana principal, una para colocar la zona de entrada de datos y otra para colocar la zona de visualización de la información.

Dicho paquete consta a priori de una sólo clase, la cual es la siguiente:

- PanelGeneral (publica)

#### **4.7.1.3 PAQUETE PRINCIPAL.FORMULARIOS.ENTRADA**

Para crear éste paquete hemos optado por agrupar la clase encargada de servir de interfaz entre el panel de la aplicación encargado de localizar la zona de visualización de la entrada de datos, y las clases encargadas de llevar a cabo dicha visualización así como la captura de los datos de consulta.

Dicho paquete está compuesto por la clase:

- FormularioEntrada (public)

#### **4.7.1.4 PAQUETE PRINCIPAL.FORMULARIOS.ENTRADA.RESOURCES**

El presente paquete es uno de los paquetes con mayor número de clases, y en él hemos decidido agrupar todas aquellas clases encargadas de visualizar los ítem pertinentes para llevar a cabo la inserción de información para efectuar consultas, así como las encargadas de recoger la información introducida por el usuario para llevar a cabo dichas búsquedas.

Las clases que conforman dicho paquete son las siguientes:

- ElementosEntrada
- MostrarEntradaActa
- MostrarEntradaActivity
- MostrarEntradaArticulos
- MostrarEntradaAuthor



- MostrarEntradaBookChapter
- MostrarEntradaCapitulo
- MostrarEntradaDegree
- MostrarEntradaFinanceProject
- MostrarEntradaFinstitution
- MostrarEntradaISI
- MostrarEntradaLenguaje
- MostrarEntradaLibro
- MostrarEntradaMeeting
- MostrarEntradaMember
- MostrarEntradaOMerits
- MostrarEntradaPatents
- MostrarEntradaPStatus
- MostrarEntradaResearchContract
- MostrarEntradaRevista
- MostrarEntradaThesis

#### **4.7.1.5 PAQUETE PRINCIPAL.FORMULARIOS.VISUALIZACION**

El criterio elegido para crear el presente paquete ha sido el de agrupar aquellas clases que se encargan de servir de interfaz entre el panel inferior, en el que se mostrará la visualización, y las clases encargadas de mostrar la información buscada.

Las clases que conforman dicho paquete son las siguientes:

- FormularioVisualizacion (public)
- MostrarInformacion (public)
- Soporte (public)
- Imagen (private)

#### **4.7.1.6 PAQUETE PRINCIPAL.FORMULARIOS.VISUALIZACION.RESOURCE**

Por último, el paquete principal.formularios.visualizacion.resources está compuesto por aquellas clases que llevan a cabo la búsqueda de información de un elemento en concreto y las muestra por pantalla.

Las clases que componen dicho paquete son las siguientes

- ModeloTabla (public)
- MostrarTablaActas (public)
- TablaActas (private)
- MostrarTablaActivity(public)
- TablaActivity (private)
- MostrarTablaArticulo (public)
- TablaArticulo (private)
- MostrarTablaAutor (public)
- TablaAutor (private)
- MostrarTablaCongreso (public)
- TablaCongreso (private)
- MostrarTablaDegree (public)
- TablaDegree (private)
- MostrarTablaFinanceProject(public)
- TablaFinanceProject (private)
- MostrarTablaForeignInstitution (public)
- TablaForeignInstitution (private)
- MostrarTablaISI (public)
- TablaISI (private)
- MostrarTablaLenguaje (public)
- TablaLenguaje (private)
- MostrarTablaLibroCapitulo (public)
- TablaLibroCapitulo (private)
- MostrarTablaLibros (public)
- TablaLibros (private)

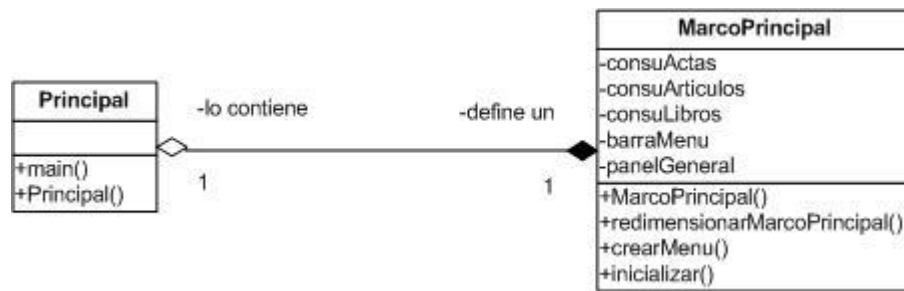
- MostrarTablaMiembros (public)
- Tablamiembros (private)
- MostrarTablaOtherMerits (public)
- TablaOtherMerits (private)
- MostrarTablaPatent (public)
- TablaPatent (private)
- MostrarTablaProfessionalStatus (public)
- TablaProfessionalStatus (private)
- MostrarTablaResearchContract (public)
- TablaResearchContract (private)
- MostrarTablaRevistas (public)
- TablaRevistas (private)
- MostrarTablaThesis (public)
- TablaThesis (private)

## 4.7.2 DIAGRAMA DE CLASES

Una vez que hemos desglosado los diferentes paquetes que componen nuestra aplicación vamos a llevar a cabo el refinamiento de cada uno de ellos a nivel de diagrama de clases, haciendo una breve descripción de las clases principales que lo conforman.

### 4.7.2.1 *Paquete principal*

En la siguiente figura podemos observar el diagrama de clase correspondiente al paquete “principal”. Como ya mencionamos anteriormente cuando describimos el diagrama de paquetes, decidimos agrupar las clases que lo conforman bajo éste paquete por agrupar aquellas clases que forman el armazón gráfico principal de la aplicación.



**Figura 4.7.2.** Diagrama de clase del paquete “principal”

Vamos a continuación a describir las clases que componen dicho paquete.

#### 4.7.2.1.1 Clase Principal

Esta es la clase encargada de crear un objeto de la clase **MarcoPrincipal**, la cual constituirá el interfaz principal de la aplicación.

##### **Visión preliminar de la clase**

La interfaz de nuestra aplicación va a depender de una serie de clases, y ésta va a ser la encargada de lanzarlo todo en cierta manera, luego podemos decir que es la clase principal de la aplicación.

##### **Dependencias de la clase**

La clase **Principal** depende de la clase **MarcoPrincipal**, ya que hace uso de ésta para dibujar el marco principal de la aplicación.

##### **Componentes de la clase**

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

##### **a) Atributos:**

**b) Métodos:**

- **main():**
  - Descripción: función principal, comienza a ejecutar la aplicación.
  - Parámetros: String args (línea de comandos)
  - Devuelve: nada
  
- **Principal().**
  - Descripción: constructor de la clase
  - Parámetros: ninguno
  - Devuelve: nada

**4.7.2.1.2 Clase MarcoPrincipal**

Esta es la clase encargada de crear el marco principal de la aplicación, sobre el cual se va a desarrollar el resto de operaciones.

***Visión preliminar de la clase***

La interfaz de nuestra está basada principalmente en esta clase, la cual establece el marco o ventana principal de la aplicación y sobre él se sustentará el resto de iconos gráficos.

***Dependencias de la clase***

Hereda de la clase JPanel de la librería javax.swing de Java. Además hace uso de la clase PanelGeneral del paquete principal.formularios.

***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

**a) Atributos:**

- *JMenuItem* **consuActas**: constituye un ítem del menú general.
- *JMenuItem* **consuLibros**: constituye el ítem correspondiente a la consulta de libros en el menú general
- Etc. Hay un objeto del tipo *JMenuItem* para cada uno de los componentes sobre los que la aplicación contiene información.
- *PanelGeneral* **panelGeneral**: instancia de la clase *PanelGeneral* del paquete *principal.formularios*.
- *JMenuBar* **barraMenu**: objeto del tipo *JMenuBar* donde se colocarán los *JMenuItem*.

**b) Métodos:**

- **MarcoPrincipal()**:

- Descripción: constructor de la clase.
- Parámetros: nada
- Devuelve: nada

- **crearMenu ()**.

- Descripción: crea el menú principal de la aplicación que posteriormente será colocado en la ventana principal.
- Parámetros: ninguno
- Devuelve: *JMenuBar*

- **Inicializar ()**.

- Descripción: crea el menú principal de la aplicación que posteriormente será colocado en la ventana principal.
- Parámetros: ninguno
- Devuelve: nada.

#### 4.7.2.2 Paquete “principal.formularios”

A continuación vamos a describir el paquete principal.formularios, el cual está compuesto por la clase PanelGeneral, como ya comentamos anteriormente en la sección relacionada con los diagramas de paquetes.

El diagrama de clases correspondiente a dicho paquete es el siguiente:



Figura 4.7.3. Diagrama de clase del paquete “principa.formularios!”

##### 4.7.2.2.1 Clase PanelGeneral

Esta es la clase encargada de dividir el marco principal en dos y añadir en cada uno de ellos la zona correspondiente.

##### *Visión preliminar de la clase*

La clase PanelGeneral se encarga de servir de interfaz entre el marco principal de la aplicación y las clases encargadas de dibujar el interfaz de búsqueda en la parte superior del marco principal y las clases encargadas de dibujar el panel que muestra el resultado de la búsqueda en la parte inferior del marco.

##### *Dependencias de la clase*

Hereda de la clase JSplitPane de la librería javax.swing de Java. Además hace uso de las clases FormularioEntrada y formularioVisualizacion de

los paquetes `principal.formularios.entrada` y `principal.formularios.visualizacion` respectivamente.

### **Componentes de la clase**

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

#### **a) Atributos:**

- *FormularioEntrada* **formularioEntrada**: constituye una instancia de la clase `FormularioEntrada`.
- *FormularioVisualizacion* **formularioVisualizacion**: constituye una instancia de la clase `FormularioVisualizacion`.
- *int* **TIPOFORMULARIOSALIDA**: indica el tipo de formulario de salida que tienen que mostrar, es decir, si es de una revista, miembro, autor, etc.
- *int* **TIPOFORMULARIOENTRADA**: indica el tipo de formulario de entrada que tiene que mostrar.

#### **b) Métodos:**

- **PanelGeneral()**:
  - Descripción: constructor de la clase.
  - Parámetros: nada
  - Devuelve: nada
- **dibujarFormulario()**.
  - Descripción: dibuja el formulario de visualizacion con los datos resultado de la busqueda.
  - Parámetros: `ResultSet`, objeto con el resultado de la consulta en SQL.
  - Devuelve: nada
- **setFormulario()**.

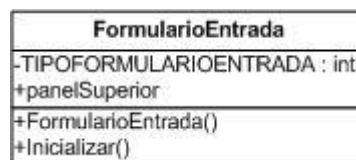


- Descripción: dibuja el formulario de visualización sin haber realizado una consulta
- Parámetros: entero, tipo de formulario a visualizar.
- Devuelve: nada

#### 4.7.2.3 Paquete “*principal.formularios.entrada*”

El siguiente paquete es el que contiene la clase encargada de servir de interfaz entre el panel de la aplicación encargado de localizar la zona de visualización de la entrada de datos, y las clases encargadas de llevar a cabo dicha visualización así como la captura de los datos de consulta.

El diagrama de clase de dicho paquete es el siguiente:



**Figura4.7.4.**Diagrama de clase del paquete “*principa.formularios.entrada*”

##### 4.7.2.3.1 Clase *FormularioEntrada*

###### ***Visión preliminar de la clase***

La clase *FormularioEntrada* se encarga de llamar a la clase oportuna según la opción que haya elegido el usuario en el marco principal para visualizar los formularios de entrada de datos para llevar a cabo una consulta sobre alguna de las tablas que conforman la BD.

###### ***Dependencias de la clase***

Hereda de la clase *JPanel* de la librería *javax.swing* de Java. Además hace uso de las clases del paquete *principal.formularios.entrada.resources* así como de la clase *PanelGeneral*.

###### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

**c) Atributos:**

- *int* **TIPOFORMULARIOENTRADA**: indica el tipo de formulario de entrada que tiene que mostrar.
- *PanelGeneral* **pg**: es un objeto que apuntará al panelGeneral desde el cual se llamó a la clase, de ésta manera permitirá estar conectado por si tiene que pasarle información.

**d) Métodos:**

- **FormularioEntrada()**:
  - Descripción: constructor de la clase.
  - Parámetros: PanelGeneral: referencia desde el panel que lo llamo.
  - Parámetros: entero: tipo de formulario de entrada.
  - Devuelve: nada
- **Inicializar()**.
  - Descripción: llama a la clase pertinente para dibujar en el panel las opciones de búsqueda oportunas.
  - Parámetros: ninguno
  - Devuelve: nada

#### **4.7.2.4 Paquete “principal.formularios.entrada.resources”**

Es uno de los paquetes más numerosos y es el encargado de mostrar la información oportuna para llevar a cabo las búsquedas de información. Dependiendo del tiempo a buscar mostrará unas opciones de búsqueda u otras.

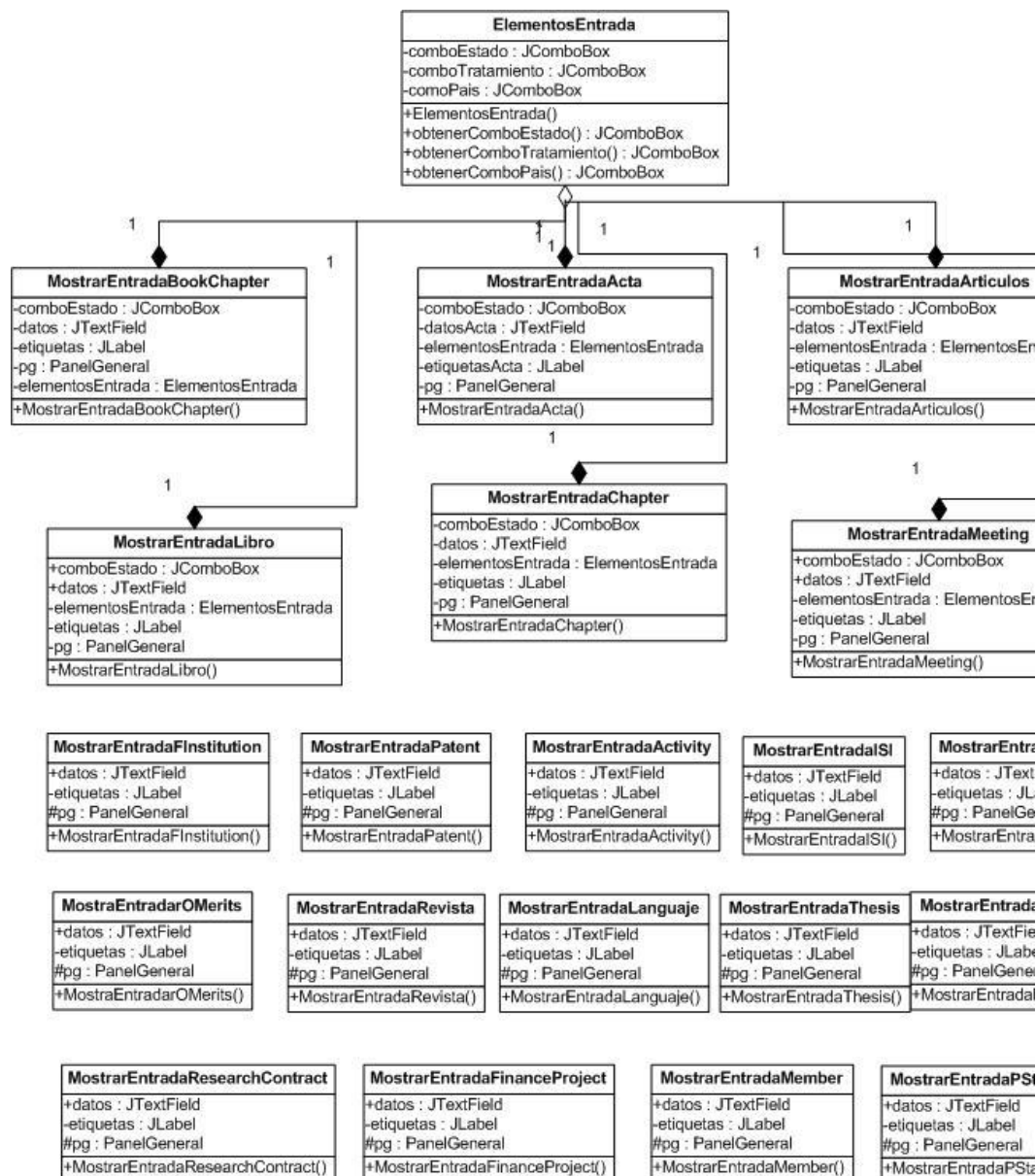


Figura4.7.5:Diagrama de clase del paquete “principa.formularios.entrada.resources”

#### 4.7.2.4.1 Clase ElementosEntrada

##### Visión preliminar de la clase

La clase ElementosEntrada, contiene utilidades que pueden ser necesarias utilizar al mostrar las opciones de búsqueda de algunas entidades, como por ejemplo, combos desplegables con los posibles países que tiene el usuario para llevar a cabo la búsqueda o el status social de los miembros del grupo de investigación.

### **Dependencias de la clase**

Dicha clase no depende de ninguna otra para funcionar como tal, aunque no tiene sentido, ya que implementa menús desplegables que tendrán que ser colocado en algún interfaz.

### **Componentes de la clase**

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

#### **a) Atributos:**

- *JComboBox* **comoPais::** indica los posibles paises sobres los que puede realizar el usuario búsquedas.
- *JComboBox* **comoStatus:** indica los posibles status de las personas cuya información esté reflejada en la aplicación.
- *JComboBox* **comoEstado:** indica los posibles estados de las publicaciones.

#### **b) Métodos:**

- **ElementosEntrada():**
  - Descripción: constructor de la clase.
  - Parámetros: PanelGeneral: referencia desde el panel que lo llamo.
  - Parámetros: ninguno
  - Devuelve: nada
- **obtenerComboPais().**
  - Descripción: añade los posibles estados al comboPais.
  - Parámetros: nada
  - Devuelve: JComboBox, combo con los posibles paises.

- **obtenerComboTratamiento().**
  - Descripción: añade el posible trato, Sr, Srta,etc.
  - Parámetros: nada
  - Devuelve: JComboBox, combo con los posibles tratos.
  
- **obtenerComboEstado().**
  - Descripción: añade el posible estado de las publicaciones: Aceptado, En edicion, En revision, etc.
  - Parámetros: nada
  - Devuelve: JComboBox, combo con los posibles estados.

#### **4.7.2.4.2 Clase *MostrarEntradaBookChapter***

##### ***Visión preliminar de la clase***

Es la clase encargada de dibujar los elementos necesarios para efectuar una búsqueda sobre la tabla que contenga información acerca de los libros de capítulos.

##### ***Dependencias de la clase***

Dicha clase hace uso de la clase anteriormente descrita ElementoEntrada. Además por otra parte hereda de la clase JPanel.

##### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

##### **a) Atributos:**

- *JComboBox* **comoEstado**: indica los posibles status de las personas cuya información esté reflejada en la aplicación.
- *TextField* **datos**: vector de *TextField* donde el usuario podrá introducir los patrones de búsqueda.
- *JLabel* **etiquetas**: vector de *JLabel* para añadir las etiquetas de lo que es cada *TextField*.
- *ElementosEntrada* **elementosEntrada**: objeto del tipo *ElementosEntrada* para poder obtener los combos oportunos.
- *PanelGeneral* **pg**: objeto del tipo *PanelGeneral*.

**b) Métodos:**

- **MostrarEntradaBookChapter():**
  - Descripción: constructor de la clase.
  - Parámetros: *FormularioEntrada* referencia desde el panel que lo llamo.
  - Devuelve: nada

Las clases “*MostrarEntradaActa*”, “*MostrarEntradaArticulos*”, “*MostrarEntradaLibro*”, “*mostrarEntradaChapter*” y “*MostrarEntradaMeeting*” tienen un comportamiento igual a la descrita anteriormente, *MostrarEntradaBookChapter*, con lo cual vamos a prescindir de su documentación, ya que con la documentación de la clase *MostrarEntradaBookChapter* y el diagrama de clases de la figura 7.4 creemos que es suficiente para entender el funcionamiento de las mismas y vamos a centrarnos en aquellas que contengan alguna peculiaridad.

Decir también que si la clase *MostrarEntradaBookChapter* mostraba los ítem necesarios para llevar a cabo una búsqueda de un libro de Capítulos (Book of Chapter) la clase *MostrarEntradaActa*, por ejemplo, añade los ítem para llevar a cabo una búsqueda de Actas, o por ejemplo la clase *MostrarEntradaArticulos*, para una búsqueda de Artículos. Podemos ver que el

nombre es lo suficientemente intuitivo como para saber el comportamiento de la misma, ayudado como ya hemos comentado del diagrama de clases y la clase ya documentada `MostrarEntradaBookChapter`.

#### 4.7.2.4.3 Clase *MostrarEntradaPatent*

##### ***Visión preliminar de la clase***

Es la clase encargada de dibujar los elementos necesarios para efectuar una búsqueda sobre la tabla que contenga información acerca de los patentes efectuadas por algún miembro del grupo de investigación o que estén relacionadas con el mismo.

##### ***Dependencias de la clase***

La clase `MostrarEntradaPatent` hereda de la clase `JPanel`, luego mantiene una relación de herencia con la misma.

##### ***Componentes de la clase***

Los elementos que encontramos en dicha clase son los siguientes:

###### **a) Atributos:**

- *JTextField* **datos**: vector de `JTextField` donde el usuario podrá introducir los patrones de búsqueda.
- *JLabel* **etiquetas**: vector de `JLabel` para añadir las etiquetas de lo que es cada `JTextField`.
- *PanelGeneral* **pg**: objeto del tipo `PanelGeneral`.

###### **b) Métodos:**

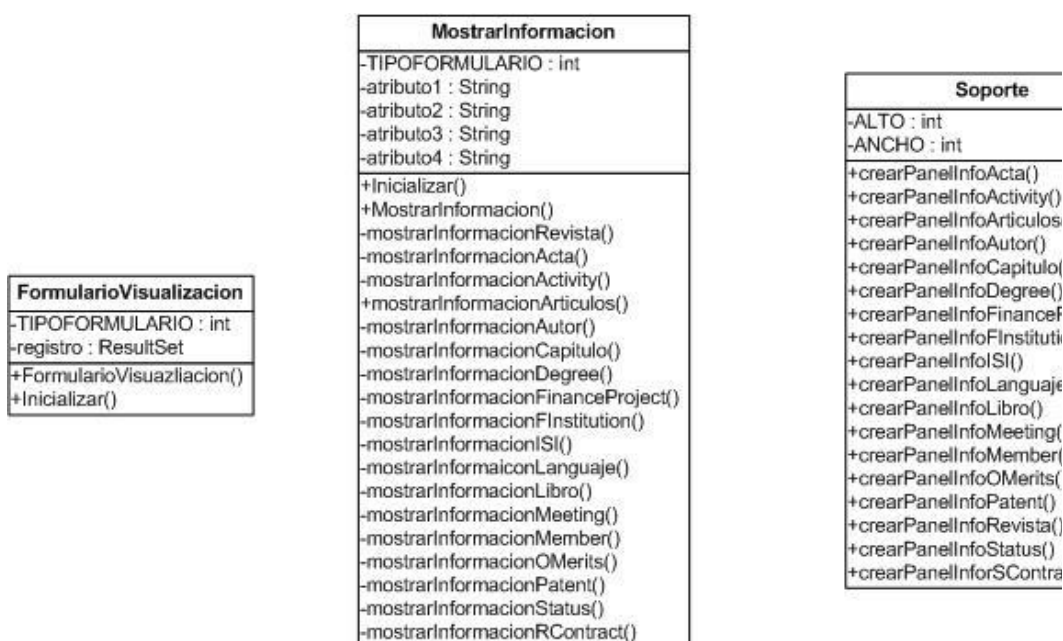
- **MostrarEntradaPatent()**:
  - Descripción: constructor de la clase.
  - Parámetros: `FormularioEntrada` referencia desde el panel que lo llamo.
  - Devuelve: nada

Al igual que ocurría antes, las clases “MostrarEntradaFinstitution”, “MostrarEntradaActivity”, “MostrarEntradaISI”, “MostrarEntradaAutor”, “MostrarEntradaOMerits”, “MostrarEntradaRevista”, “MostrarEntradaLenguaje”, “MostrarEntradaThesis”, “MostrarEntradaDegree”, “MostrarEntradaMember”, “MostrarEntradaResearchContract”, “MostrarEntradaFinanceProject” y “MostrarEntradaPStatus” tienen un comportamiento similar a la clase MontrarEntradaPatent, descrita en éste epígrafe, por lo que para no cargar la documentación de información redundante vamos a prescindir de la documentación explícita de las mismas, ya que creemos que se puede entender su comportamiento perfectamente con la ayuda del diagrama de clases y la descripción de la clase MostrarEntradaPatent.

#### 4.7.2.5 Paquete “principal.formularios.visualizacion”

El presente paquete contiene aquellas clases que sirven de interfaz entre el panel inferior y las clases que se encargan de mostrar los resultados obtenidos en la búsqueda realizada.

El diagrama de clases de dicho paquete puede verse a continuación:



**Figura4.7.6:**Diagrama de clase del paquete principal.formularios.visualizacion



#### 4.7.2.5.1 Clase *FormularioVisualizacion*

##### **Visión preliminar de la clase**

La clase *FormularioVisualizacion* se encarga de llamar a la clase oportuna según la opción que haya elegido el usuario en el marco principal para visualizar los formularios de salida de datos en los que se mostrará la salida de la búsqueda realizada.

##### **Dependencias de la clase**

La clase *FormularioVisualizacion* Hereda de la clase *JPanel* de la librería *javax.swing* de Java. Además hace uso de las clases del paquete *principal.formularios.visualizacion.resources*.

##### **Componentes de la clase**

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

##### **a) Atributos:**

- *int* **TIPOFORMULARIO**: indica el tipo de formulario de visualizacion que tiene que mostrar.
- *ResultSet* **registro**: objeto del tipo *ResultSet* que contendrá el resultado de la consulta realizada.

##### **b) Métodos:**

- **FormularioVisualizacion()**:
  - Descripción: constructor de la clase. (sobrecargado)
  - Parámetros: entero: tipo de formulario
  - Parámetros: *ResultSet*: resultado de la consulta
  - Devuelve: nada
- **FormularioVisualizacion()**:
  - Descripción: constructor de la clase. (sobrecargado)
  - Parámetros: entero: tipo de formulario

- Devuelve: nada
- **Inicializar().**
  - Descripción: inicializa los componentes gráficos que van a ir en el panel.
  - Parámetros: ninguno
  - Devuelve: nada

#### 4.7.2.5.2 Clase *MostrarInformacion*

##### ***Visión preliminar de la clase***

La clase *MostrarInformacion* es la encargada de mostrar la información detallada de alguno de los ítem que se hayan obtenido como resultado de la búsqueda y mostrarlos por pantalla.

##### ***Dependencias de la clase***

Hereda de la clase *JDialog*

##### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

##### **a) Atributos:**

- *int* **TIPOFORMULARIO**: indica el tipo de formulario de visualización que tiene que mostrar.
- *String* **atributo1**: es uno de los atributos identificadores de búsqueda.
- *String* **atributo2**: es uno de los atributos identificadores de búsqueda ( en caso de que la clave sea el combinado de dos atributos)
- *String* **atributo3**: es uno de los atributos identificadores de búsqueda ( en caso de que la clave sea el combinado de tres atributos)

- *String atributo4*: es uno de los atributos identificadores de búsqueda, en caso de que la clave sea el combinado de cuatro atributos.

**b) Métodos:**

- **MostrarInformacion()**.
  - Descripción: constructor de la clase
  - Parámetros: entero, tipo de formulario a mostrar
  - Parámetros: string: clave para buscar en la BD.
  - Parámetros: string: clave para buscar en la BD, en caso de que sea compuesta dos atributos.
  - Parámetros: string: clave para buscar en la BD, en caso de que sea compuesta por tres atributos
  - Parámetros: string: clave para buscar en la BD, en caso de que sea compuesta por cuatro atributos.
  - Devuelve: nada
- **Inicializar()**.
  - Descripción: inicializa los componentes gráficos que van a ir en el panel.
  - Parámetros: ninguno
  - Devuelve: nada
- **Inicializar()**.
  - Descripción: inicializa los componentes gráficos que van a ir en el panel.
  - Parámetros: ninguno
  - Devuelve: nada
- **mostrarInformacionActa()**.

- Descripción: muestra información de un acta de un congreso.
  - Parámetros: ninguno
  - Devuelve: nada
- **mostrarInformacionArticulo().**
  - Descripción: muestra información de un artículo.
  - Parámetros: ninguno
  - Devuelve: nada
- **mostrarInformacionCapitulo().**
  - Descripción: muestra información acerca de un capítulo.
  - Parámetros: ninguno
  - Devuelve: nada
- **mostrarInformacionCongreso().**
  - Descripción: muestra información acerca de un congreso.
  - Parámetros: ninguno
  - Devuelve: nada

El resto de procedimiento de ésta clase tienen una misión similar, con la diferencia de que en cada caso muestra información acerca de un ítem determinado, libro, miembro, patentes, etc.

#### **4.7.2.5.3 Clase Soporte**

##### ***Visión preliminar de la clase***

La clase Soporte como su propio nombre indica, se encarga de dar el soporte necesario para mostrar los formularios de visualización detallada.

**Dependencias de la clase**

Ninguna.

**Componentes de la clase****a) Atributos:**

- *int* **ALTO**: indica el ALTO del formulario.
- *Int* **ANCHO**: indica el ancho del formulario.

**b) Métodos:**

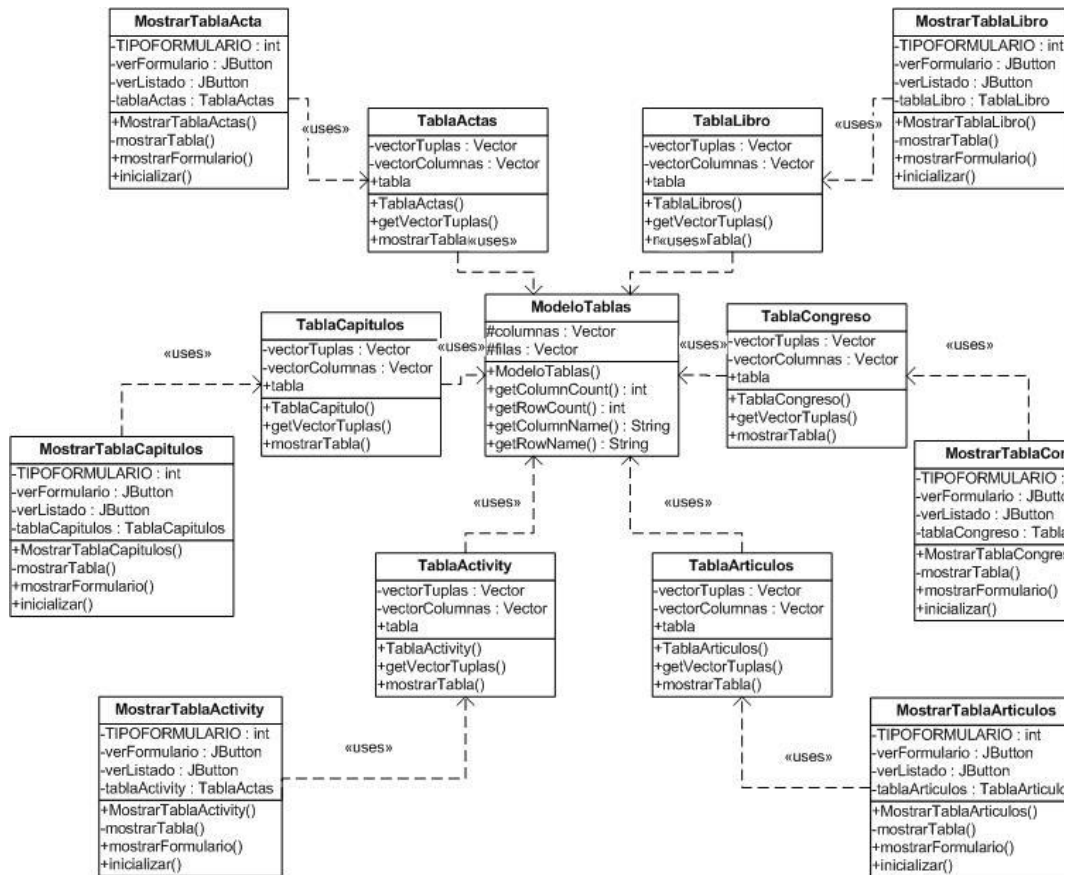
- **crearPanellInfoActa()**.
  - Descripción: método que muestra la información relativa a un acta de un congreso.
  - Parámetros: ninguno
  - Devuelve: nada
- **crearPanellInfoActivity()**.
  - Descripción: método que muestra la información relativa a una actividad
  - Parámetros: ninguno
  - Devuelve: nada

El resto de métodos tienen una utilidad similar a los aquí descritos, cada uno mostrando la información que le corresponde.

**4.7.2.6 Paquete “*principal.formularios.visualizacion.resources*”**

Este último paquete es el que abarca más clases, y éstas son las encargadas de mostrar la información acerca de las búsquedas realizadas con los patrones de búsqueda especificados y mostrarlas en el panel inferior de la aplicación, preparado para tal propósito.

El diagrama de clases es el que se muestra a continuación.



**Figura4.7.7:**Diagrama de clases del paquete “principal.formularios.visualizacion.resources”

El diagrama anterior no muestra todas las clases que forman el paquete, véase epígrafe 7.1, ya que por espacio hemos creído conveniente utilizar éste diagrama reducido, ya que cumple con la finalidad del mismo. El resto de clases que faltan se comportan de la misma manera que las reflejadas en el diagrama.

Una vez hecho este inciso vamos a describir las clases que conforman dicho paquete.

#### 4.7.2.6.1 Clase *ModeloTablas*

##### *Visión preliminar de la clase*

Esta clase permite obtener el valor de la celda seleccionada de una tabla, indicándonos el número de la fila, la columna, así como el valor que se encuentre en la misma.

##### *Dependencias de la clase*

La clase hereda de la clase `AbstractTableModel`

##### *Componentes de la clase*

###### a) Atributos:

- **Vector *columnas*:** vector con las columnas de la tabla.
- **Vector *filas*:** vector con las filas de la tabla.

###### b) Métodos:

- **ModeloTablas():**
  - Descripción: constructor de la clase.
  - Parámetros: vector: vector de columnas
  - Parámetros: vector: vector de filas
  - Devuelve: nada
- **getColumnCount():**
  - Descripción: obtiene el valor de la columna de la casilla seleccionada.
  - Parámetros: ninguno
  - Devuelve: entero: valor de la posición.
- **getRowCount():**
  - Descripción: obtiene el valor de la fila de la casilla seleccionada.

- Parámetros: ninguno
  - Devuelve: entero: valor de la posición.
- **getColumnName():**
  - Descripción: obtiene el nombre de la columna
  - Parámetros: ninguno
  - Devuelve: cadena de caracteres.
- **getRowName():**
  - Descripción: obtiene el nombre de la fila
  - Parámetros: ninguno
  - Devuelve: cadena de caracteres.
- **getValueAt():**
  - Descripción: obtiene el valor de la celda
  - Parámetros: entero: fila
  - Parámetros: entero: columna
  - Devuelve: Object, objeto genérico de Java.

#### **4.7.2.6.2 Clase *MostrarTablaActas***

##### ***Visión preliminar de la clase***

Muestra una tabla con todas las actas resultantes de la búsqueda realizada.

##### ***Dependencias de la clase***

Depende de la clase *TablaActas*, además existe cierta dependencia con el paquete *principal.formularios.entrada.resources*, del cual se obtiene el resultado de la búsqueda a mostrar. Si se desea mostrar otro resultado y no el obtenido por el paquete mencionado, no existiría dicha dependencia.



### **Componentes de la clase**

#### **a) Atributos:**

- *JButton* **verFormulario**: botón para ver el formulario.
- *JButton* **verListad**: botón para efectuar un listado.
- *TablaActas* **tablaActas**: objeto del tipo *TablaActas*

#### **b) Métodos:**

- **MostrarTablaActas()**:
  - Descripción: constructor de la clase.
  - Parámetros: ninguno
  - Devuelve: nada
- **MostrarFormulario ()**:
  - Descripción: muestra el formulario de la tupla seleccionada.
  - Parámetros: ninguno
  - Devuelve: nada
- **MostrarTabla ()**:
  - Descripción: crea la tabla donde se va a mostrar la información
  - Parámetros: ninguno
  - Devuelve: *JTabbedPane*
- **Inicializar ()**:
  - Descripción: inicializa los elementos gráficos a utilizar.
  - Parámetros: ninguno
  - Devuelve: nada

El resto de clases que tienen la misma nomenclatura, MostrarTablaXXXX. Presetan un comportamiento igual que la descrita aquí, por lo que vamos a prescindir de la descripción de cada una de ellas.

#### 4.7.2.6.3 Clase TablaActas

##### ***Visión preliminar de la clase***

Es una clase Interna utilizada por la clase MostrarTablaActas. Dicha clase crea una tabla del tipo actas para representar la información pertinente.

##### ***Dependencias de la clase***

Es conveniente señalar que es una clase de tipo privado y que tiene una dependencia con la clase ModeloTabla.

##### ***Componentes de la clase***

###### **a) Atributos:**

- Vector **vectorTuplas**: botón para ver el formulario.
- Vector **vectorColumnas**: botón para efectuar un listado.
- *tablaActas* **tabla**: objeto del tipo TablaActas

###### **b) Métodos:**

- **TablaActas():**
  - Descripción: constructor de la clase.
  - Parámetros: ninguno
  - Devuelve: nada
- **getVectorTuplas ():**
  - Descripción: retorna el vector de tuplas seleccionado.
  - Parámetros: ninguno
  - Devuelve: Vector

- **mostrarTabla ():**
  - Descripción: llama a la clase ModeloTabla y crea una tabla optima para las actas.
  - Parámetros: ninguno
  - Devuelve: nada

El resto de clases que tienen la misma nomenclatura, TablaXXXX, al igual que ocurría antes, presentan un comportamiento igual que la descrita aquí, por lo que vamos a prescindir de la descripción de cada una de ellas.

### 4.7.3 DIAGRAMA DE INTERACCIÓN

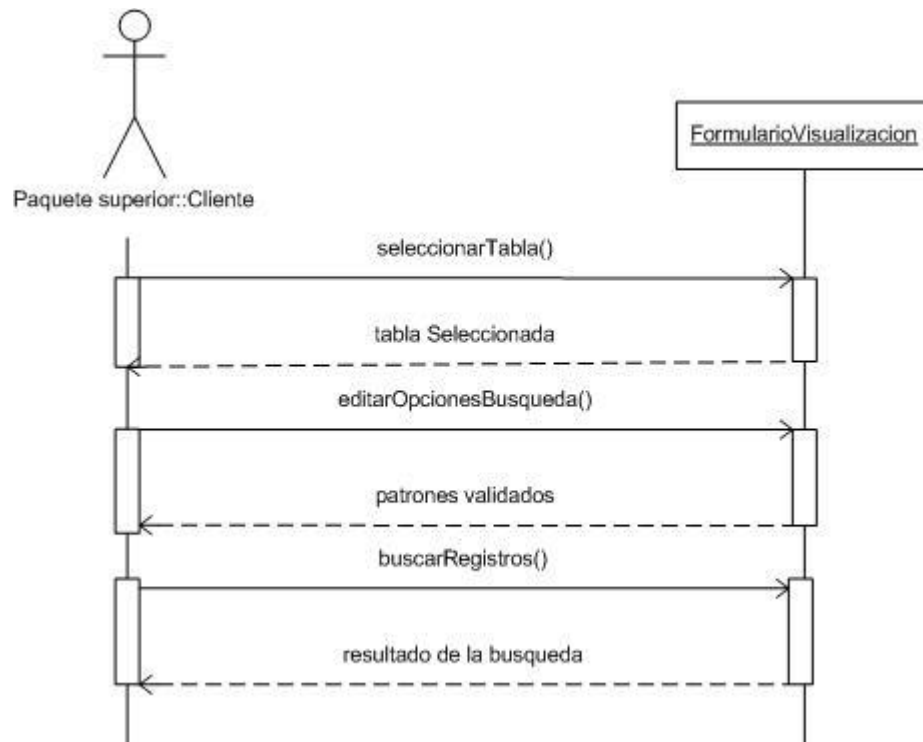
Los diagramas de interacción nos muestran el comportamiento dinámico del sistema, es decir, con dichos diagramas podemos ver como los objetos interaccionan.

Existen dos tipos de diagramas de interacción, los diagramas de secuencia los cuales nos dan una idea de la dimensión temporal, y los diagramas de colaboración, los cuales nos dan una idea de la dimensión estructural.

En esta práctica vamos a hacer solo mención a los diagramas de secuencia, ya que estamos más interesados en obtener la dimensión temporal de nuestro sistema, que la dimensión estructural.

#### 4.7.3.1 Diagrama de secuencia

El siguiente diagrama de secuencia corresponde a la actividad de visualizar información, el cual comprende, desde llevar a cabo la elección de las opciones de búsqueda, hasta que se produzca la búsqueda y el usuario reciba el resultado.



**Figura 4.7.8:** Diagrama de secuencia "Visualizar información"

## 4.7.4 DIAGRAMA DE ACTIVIDADES

Los diagramas de actividad pueden ser considerados como un diagrama de flujo que destaca la actividad que tiene lugar a lo largo del tiempo.

Vamos a realizar los diagramas de actividad de los procedimientos que consideramos son los más importantes del sistema y cuyo diagrama de actividad creemos es interesante para el lector.

### 4.7.4.1 BUSCAR INFORMACIÓN

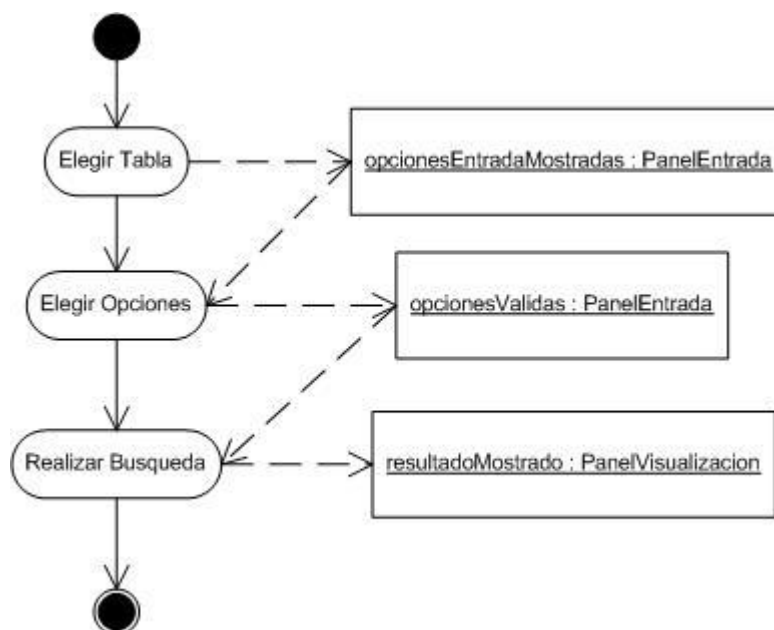


Figura 4.7.9: Diagrama de actividad "Buscar información"

#### 4.7.4.2 MOSTRAR DETALLE

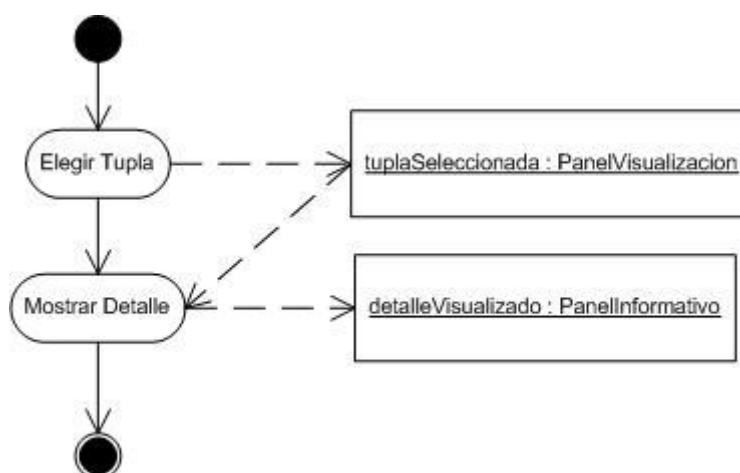
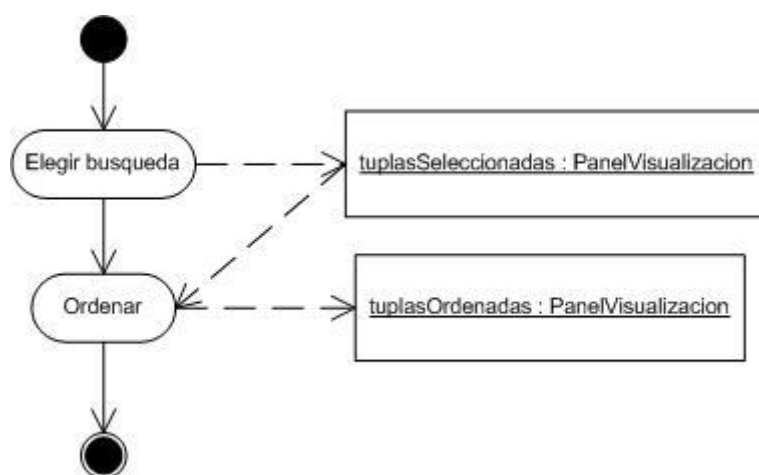


Figura 4.7.10: Diagrama de actividad "Mostrar detalle"

#### 4.7.4.3 ORDENAR TUPLAS



**Figura 4.7.11:** Diagrama de actividad "Ordenar tuplas"

# 4.8

## IMPLEMENTACIÓN DEL SISTEMA

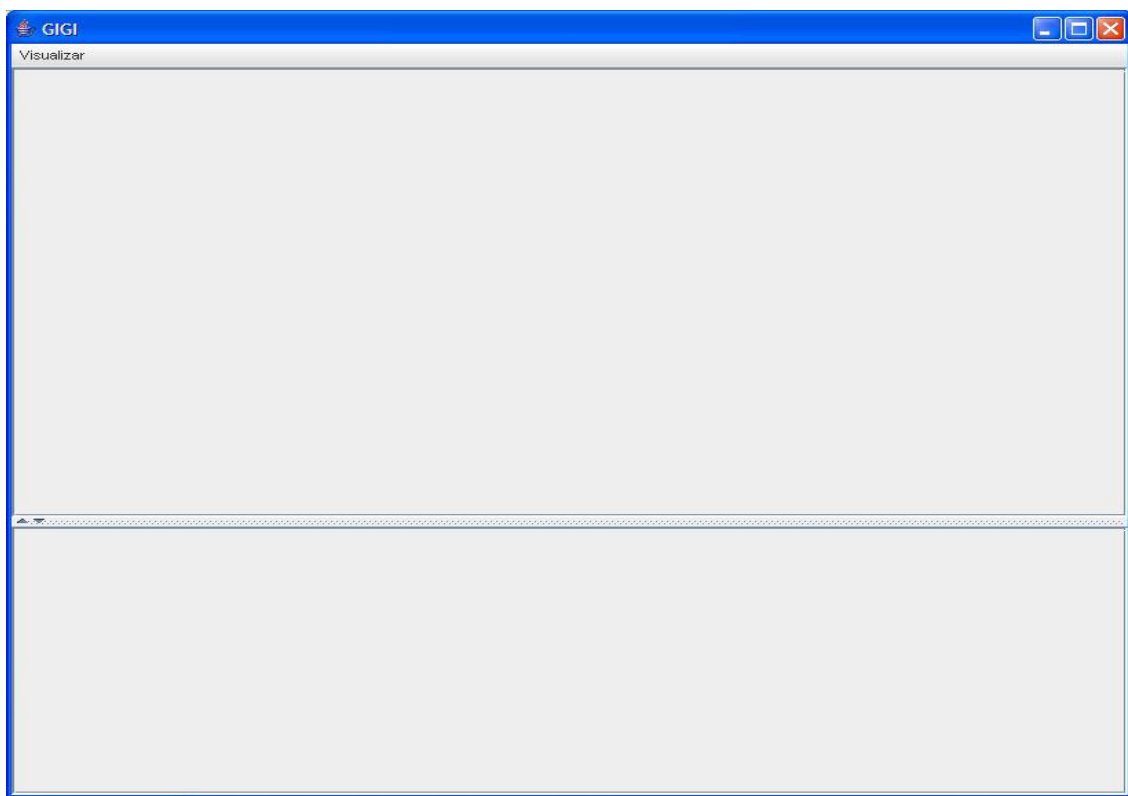
En esta sección vamos a describir la interfaz obtenida en los formularios de visualización.

El código fuente hemos creído conveniente no incluirlo, ya que al ser demasiado extenso dificultaría la lectura del presente documento, además al ser éste libre, el lector interesado podrá acceder a el mismo sin ningún tipo de problema.

### 4.8.1 INTERFAZ DEL SISTEMA

A continuación vamos a mostrar una serie de capturas ya del sistema implementado, las cuales podrán mostrar el resultado de toda la fase de desarrollo que hemos llevado a cabo.

La primera captura que nos encontramos pertenece al marco principal de la aplicación, sin haber escogido ninguna opción de consulta.



**Figura 4.8.1** Captura Marco Principal

Podemos observar como la pantalla se encuentra dividida en dos zonas, una servirá para llevar a cabo las consultas, la superior, y la otra para mostrar el resultado de las mismas. En la parte superior encontramos un menú, el cual al desplegarse muestra cada una de las tablas a las que puede acceder el usuario para efectuar consultas sobre las mismas, es decir, muestra la opción de Article, Books, Chapters, etc. Indicando las entidades que puede consultar el usuario, las cuales estarán relacionadas con el Grupo de Investigación.





**Figura 4.8.2** Captura Menú

Si el usuario elige una de esas opciones se le mostrará en la parte superior del marco principal las opciones para llevar a cabo una búsqueda, entre las que podemos señalar, las casillas de texto en las que introducir los patrones de búsqueda, así como los botones para elegir entre una búsqueda cribada y una búsqueda total.

Por otra parte en la zona inferior del panel aparecerá un listado de las características a mostrar en forma de matriz, de tal manera que los elementos localizados se añadirán a dicha matriz.

**Revista** (Campos obligatorios marcados en negrita)

Datos de la Revista

ISSN  Editorial  Abreviatura

Nombre

ISI (Institute for Scientific Information)

Año  Factor de Impacto  Total citas

Índice inmediato  Número de Artículos  Vida media

**Ver Detalle** **Imprimir Listado**

Revistas

**Información de Revistas**

Issn	Editorial	Abreviatura	Nombre
54654	hola	hola	hgolaaa
ISSN_1	Editorial Cúspide	ECCEE	Computación Evolutiva en España
ISSN_2	Publicaciones META	PMP	Pediatrics
ISSN_3	Grupo POSIT	GPAI	Anales Informáticos

29/05/07 18:59

**Figura 4.8.3.** Ejemplo de ejecución

Para cada una de las opciones se mostrará una parte superior e inferior diferente, acorde a los atributos de la entidad seleccionada.

En la parte inferior cuando realicemos una búsqueda se mostrarán los datos en forma de filas de la siguiente forma:

**Ver Detalle** **Imprimir Listado**

Revistas

**Información de Revistas**

Issn	Editorial	Abreviatura	Nombre
54654	hola	hola	hgolaaa
ISSN_1	Editorial Cúspide	ECCEE	Computación Evolutiva en España
ISSN_2	Publicaciones META	PMP	Pediatrics
ISSN_3	Grupo POSIT	GPAI	Anales Informáticos

**Figura 4.8.4.** Matriz de resultados

A continuación podemos ver un formulario de visualización detallado, este ejemplo corresponde a una revista, pero para el resto de elementos es igual con la salvedad de que cada uno mostrará los ítem correspondientes.

The screenshot shows a software window titled 'REVISTAS' with a blue header bar. Inside, there are two main sections: 'Datos Revista' and 'Datos Articulos'. The 'Datos Revista' section contains a table with journal information. The 'Datos Articulos' section contains a table with article information and a 'Descargar Artículo' button. At the bottom right, there are 'Imprimir' and 'Cerrar' buttons.

Datos Revista	
ISSN:	ISSN_3
NOMBRE:	Grupo POSIT
ABREVIATURA:	GPAI
EDITORIAL:	Anales Informáticos

Datos Articulos	
Artículos 1	
CODIGO:	5
TITULO:	Chip multiproceso
AÑO:	2006
VOLUMEN:	1
PAGINA INICIAL:	103
PAGINA FINAL:	107
DOCUMENTO:	<a href="#">Descargar Artículo</a>
ESTADO:	Aceptado
ISSN:	ISSN_3

**Figura 4.8.5:** Formulario de visualización de una revista

Por ejemplo el formulario correspondiente a la visualización de los integrantes del equipo es como sigue a continuación:



The screenshot shows a web application window titled "INFORMACION MIEMBROS". It contains three main sections: "Datos Miembros", "Datos Thesis", and "Other Merits".

**Datos Miembros**

**Miembro**



FAMILY NAME:	Luque Serrano
NAME:	Carlos
E-MAIL:	otroCLS@uco.es
PHONE:	686444444
WEB:	www.uco.es
FROM DATE:	2005-03-19
TO DATE:	2007-02-10
DNI:	44444444B
BIRTH DATE:	1980-07-10
SEX:	H
OFFICIAL NUMBER	4
PC:	14016

**Datos Thesis**

**Thesi**

TITLE:	Inteligencia Artificial
DOCTORATE:	doctorado
UNIVERSITY:	cordoba
FACULTY:	cordoba
YEAR:	2001
QUALIFICATION:	7.00

**Other Merits**

**Merit**

NAME PT:	Otros
----------	-------

Buttons: Imprimir, Cerrar

Figura 4.8.6: Formulario de visualización de un miembro

## 5. Formulario de Impresión

# 5.1

## DEFINICIÓN DEL PROBLEMA

---

En este apartado se tratarán de identificar las necesidades que pretendemos cubrir así como de definir el principal objetivo a alcanzar con el desarrollo de la práctica, mostrando las alternativas y posibilidades mediante las cuales será posible conseguir el resultado que deseamos.

La enumeración de las necesidades y la definición del problema podrán llevarse a cabo desde dos puntos de vista diferentes. Por un lado, se intentará identificar el problema desde la perspectiva del cliente (problema real) y, por otro lado, se intentará presentar la forma de dar solución a dichas necesidades (problema técnico).

### 5.1.1 DEFINICIÓN DEL PROBLEMA REAL

Desde el punto de vista real, el problema que se plantea es simple, ya que es la realización de la parte de la aplicación encargada de generar informes sobre los datos almacenados en la base de datos. Se podrán generar ya sean informes específicos de detalle de un elemento de la base de datos, o listados de los mismos.

Los formularios se generarán en el formato PDF, ya que es un estándar en cuanto a documentos, y ya que existen herramientas para su manipulación sobre cualquier sistema operativo.

Por tanto, se deberá desarrollar un módulo que sea capaz de realizar estas funcionalidades, es decir, permitir al usuario la creación de informes para cualquier elemento de la base de datos.

## **5.1.2 DEFINICIÓN DEL PROBLEMA TÉCNICO**

El desarrollo de la solución consistirá en darle funcionalidad a los enlaces creados a través de la interfaz general para la generación de los informes pertinentes. La generación de los informes se realizará de forma totalmente transparente al usuario, utilizando para ello una librería de generación de informes desde el lenguaje seleccionado para la aplicación.

Una vez identificado el problema real a resolver, será necesario realizar la definición del mismo desde un punto de vista técnico. Para ello, se utilizará la técnica de Ingeniería conocida con el nombre de PDS (Product Design Specification) [Pressman, 2002]. Esta metodología propone realizar un análisis de los principales condicionantes técnicos del problema a resolver, mediante la respuesta a una serie de cuestiones básicas, las cuales se exponen a lo largo de los siguientes subapartados.

### **5.1.2.1 Funcionamiento**

El módulo de generación de informes será el encargado de generar los informes pertinentes de acuerdo a las peticiones realizadas por el usuario a través de la interfaz del sistema.

La funcionalidad que ofrece este módulo es totalmente transparente al usuario, ya que éste simplemente interactúa con la interfaz general del sistema, y este módulo se encarga de generar el informe solicitado sin necesitar para nada interacción con usuario.

### **5.1.2.2 Entorno**

El módulo de impresión no dispone de ninguna ventana ya que es simplemente código que se llama en los momentos requeridos por el usuario de forma asíncrona para generar los informes en formato PDF.

### **5.1.2.3 Vida esperada**

La estimación de la vida de un software es un parámetro muy importante pues puede que no merezca la pena realizar un gran esfuerzo en el diseño de un software si después va a tener una vida muy corta. Sin embargo, este parámetro es difícil de calcular pues depende de varios factores, como pueden ser su mantenimiento o nuevas necesidades por parte de los usuarios.

Lo primero que debemos tener en cuenta es que este sistema se sustenta sobre una base de datos dirigida a un grupo de investigación, por lo cual, si el diseño de dicha base de datos es correcto, su estructura no debe ser modificada en un largo periodo de tiempo. Como es lógico esto no supone que no se modifique la información contenida en dicha base de datos, ya que la idea de la aplicación es la manipulación de dicha información.

Otra idea que debemos tener en cuenta es que este módulo, el de impresión, no tiene sentido como aplicación por sí misma, sino que debe ser integrada junto con los distintos módulos que forman la aplicación, por lo que si se invierte tanto esfuerzo en el desarrollo es de suponer que su tiempo de vida será elevado.

### **5.1.2.4 Ciclo de mantenimiento**

El software está preparado para cualquier modificación que se deba realizar pues está diseñado de tal forma que es fácil añadir tanto una nueva opción como una nueva funcionalidad.



### **5.1.2.5 Competencia**

En el mundo del desarrollo de sistemas de información existe mucha competencia, aunque es probable que en cuanto al desarrollo de una aplicación tan específica para manipular toda la información requerida por un grupo de investigación la competencia sea bastante reducida, y sobre todo para el módulo de generación de informes que es tan específico.

### **5.1.2.6 Aspecto externo**

El aspecto externo de una aplicación hace referencia no solamente al impacto visual que recibe el usuario al realizar la ejecución de la misma, sino también a la propia presentación física del sistema.

El módulo de impresión no dispone de ningún elemento que se muestre al usuario, simplemente se mostrarán los informes en formato PDF de acuerdo a estándares como por ejemplo el proporcionado por el MEC para el currículum.

### **5.1.2.7 Estandarización**

El diseño de la aplicación se ha realizado de forma tal que la aplicación usuario pueda ejecutarse en cualquier Sistema Operativo, ya sea de tipo Windows, Linux o de cualquier otro. Además, se ha procurado utilizar los elementos de la forma más estándar posible, ya sea el paradigma de programación, los botones, los menús, etc.

### **5.1.2.8 Calidad y fiabilidad**

La calidad y la fiabilidad son campos que cada vez van tomando mayor importancia en la sociedad. Por ello, ambos han sido muy tenidos en cuenta a la hora de diseñar el módulo para la aplicación global.

De esta manera, este módulo ha sido desarrollado identificando los puntos o elementos de riesgo o de mayor probabilidad de fallo e intentando minimizar esta probabilidad.

Ha sido necesario llevar a cabo gran cantidad de pruebas ya que aunque este módulo no tenga interacción directa con el usuario, tiene que acceder a la base de datos para obtener los datos, y estos datos deben estar almacenados de forma correcta y cumpliendo las especificaciones de diseño para que todo funcione de forma correcta.

### 5.1.2.9 Programa de tareas

A continuación se va a desarrollar el programa detallado de la realización del proyecto a lo largo del tiempo: Se conoce como estilo *Normal* aquel estilo cuyo uso es el predominante, esto es, se trata del estilo principal que se sigue para documentar los distintos aspectos de un proyecto o trabajo. En nuestro caso se ha definido con las siguientes características:

- **Fase de estudio.** En esta primera fase se estudia como se ha realizado el módulo de impresión. Se estudiarán también las herramientas necesarias para el desarrollo de la nueva funcionalidad que se incorporará en la misma.
- **Análisis y diseño** de la impresión. En esta fase se realizará el diseño y la implementación del módulo en cuestión. Para la implementación se utilizará el lenguaje Java, utilizando fundamentalmente la librería swing para la implementación gráfica, e iText para la manipulación de ficheros en PDF.
- **Realización de pruebas.** Para asegurar el correcto comportamiento del módulo así como su integración dentro de la aplicación global se deberán realizar una serie de pruebas y un análisis exhaustivo de los resultados.
- **Documentación de la memoria.** En esta fase se ha elaborado la documentación técnica, es decir, la memoria. La elaboración de la

memoria se ha realizado conjuntamente con el diseño del módulo de la aplicación para dotar al documento de una mayor coherencia.

#### **5.1.2.10 Pruebas**

Las pruebas que se llevan a cabo en esta etapa serán tanto de caja negra o pruebas funcionales como de caja blanca. Las pruebas de caja negra son de especial interés en este módulo del proyecto ya que tratan fundamentalmente las entradas (acceso a BD) y salidas (generación del informe) del sistema, y al tratarse de la impresión es normal que este tipo de pruebas tengan el mayor peso posible.

Se realizarán también pruebas de caja blanca en aquellas partes donde haya un funcionamiento complicado, respecto a implementación propiamente dicha, es decir, que tengan una algoritmia compleja.

Por último, se realizará la prueba de aceptación, en donde intervendrán una o varias personas distintas al autor, y la prueba del sistema, para comprobar que se cumplen todos los requisitos y que el propio sistema responde a situaciones límite de sobrecarga.

El módulo desarrollado para la aplicación se dará por válido cuando se hayan corregido todos los errores encontrados en las pruebas anteriores y se cumplan con los objetivos para los que se diseñó.

#### **5.1.2.11 Seguridad**

El módulo se ha diseñado de forma que el usuario no pueda modificar el diseño de la misma. Como ya se ha comentado, el usuario podrá interactuar con la información recogida en la base de datos que sustenta el sistema, pero nunca podrá acceder a la base de datos como tal.

Además antes de poder realizar cualquier operación que afecte a la base de datos es necesario que el usuario se identifique introduciendo su nombre de

usuario y su contraseña a través de la interfaz general. Estos datos se utilizarán para conectarse a la base de datos, sirviendo esta conexión a su vez de autenticación del usuario.

# 5.2

## OBJETIVOS

---

En la presente sección vamos a describir los objetivos que se pretenden alcanzar con el desarrollo de la tarea asignada en la fase 2 del proyecto. Para ello vamos a distinguir entre los objetivos formales, los cuales son los más generales a conseguir con el desarrollo de nuestra tarea, y los objetivos operacionales, los cuales son los objetivos que deben de satisfacer la funcionalidad a desarrollar.

### 5.2.1 OBJETIVOS FORMALES

Como objetivos formales a destacar en la realización de nuestra tarea podríamos destacar los siguientes:

- Perfeccionamiento del lenguaje de programación Java.
- Aprendizaje de manejo del paquete iText para impresión de información.
- Conocer la forma de trabajar en un proyecto de amplia envergadura con varios equipos de trabajo cada uno con su tarea y objetivos establecidos.

- Poner en práctica los conocimientos adquiridos a lo largo de nuestros estudios universitarios.
- Perfeccionar el desarrollo de documentaciones técnicas, así como el uso de lenguajes de modelado como UML.2.2 Objetivos funcionales.

### 5.2.1 OBJETIVOS FORMALES

El objeto principal de este módulo es el diseño e implementación del módulo de impresión de la aplicación, permitiendo al usuario la generación de informes.

Podemos puntualizar los siguientes objetivos formales:

- Se debe permitir la generación de tantos informes como entidades principales aparezcan en la base de datos del sistema.
- La generación de los informes cumplirá con estándares cuando existan, como es el caso del currículum.
- Se debe controlar que los informes generados tengan buena apariencia, es decir, que no se corte texto de forma no controlada al saltar de páginas, y que todo el texto esté encuadrado de forma correcta.
- Se debe llevar a cabo la implementación de forma que pueda ser entendida fácilmente por los demás miembros del proyecto, para lo cual se utilizarán comentarios cada vez que sea necesario.
- Se debe realizar una documentación de forma correcta para que en las siguientes fases del proyecto los grupos que lo necesiten puedan consultarla de forma cómoda, y ésta les sea útil.
- Se debe realizar el módulo de forma independiente a los demás módulos de la aplicación.

# 5.3

## RESTRICCIONES TÉCNICAS Y DE GESTIÓN

---

Las limitaciones a las que se ha visto sometido el desarrollo de esta aplicación son:

- El sistema debe generar los informes de acuerdo a unas especificaciones proporcionadas por la empresa a la que va destinada o por estándares.
- El sistema deberá ser capaz de generar informes para toda la información correspondiente al grupo de investigación.
- Debido a la naturaleza multiplataforma del sistema, y a la modularidad exigida, el desarrollo del sistema debe realizarse en el lenguaje de programación Java. Este lenguaje proporciona un paradigma de programación orientado a objetos. Tiene un API que proporciona una fuente de recursos lo suficientemente rica y en continua expansión para cubrir todas las necesidades de la fase de codificación del sistema.
- Para el desarrollo del entorno gráfico se utilizará la librería SWING de JAVA. Esto se ha decidido entre los grupos de desarrollo de las distintas partes de la aplicación, por tratarse de una librería estándar de JAVA y por lo tanto la cantidad de

documentación existente de la misma es muy útil a la hora de desarrollar.

- La utilización del lenguaje de programación Java para la codificación hará necesaria la aplicación de una perspectiva orientada a objetos tanto en el proceso de especificación de requisitos como en el proceso de diseño.
- Se utilizará la tecnología que MySql 5.0 proporciona para el desarrollo de la aplicación. A su vez, para lograr la interacción entre el sistema y la base de datos se utilizará JDBC, como puente de conexión entre ambos.
- Para la generación de los ficheros PDF se utilizará la librería iText. iText es una librería de libre disposición que permite generar todo tipo de ficheros PDF a través de JAVA, lo cual como es evidente cumple perfectamente los requisitos que se buscan para la generación de informes.
- Como se crearán y se mostrarán archivos PDF, es necesario que ese instalado la aplicación Adobe Acrobat Reader en alguna de sus versiones en la máquina del cliente.



## 5.4 RECURSOS

---

Los recursos utilizados para el desarrollo de este proyecto podemos dividirlos en recursos humanos y recursos materiales, ya sean hardware o software.

### 5.4.1 RECURSOS HUMANOS

Los recursos humanos para la realización del módulo de impresión consisten en los autores del proyecto y los gestores del mismo.

Los autores del proyecto son los integrantes del grupo 13 de desarrollo, ingenieros técnicos y estudiantes de 2º curso de Ingeniería Informática:

- Antonio Cid García.
- Jorge Fernández Marín.

Por otro lado, los gestores del proyecto, cuyas pautas deben seguir los autores mencionados anteriormente son:

- Dña. Irene Luque Ruiz, Titular de Universidad.
- Miguel Ángel Gómez Nieto, Catedrático E.U.

## 5.4.2 RECURSOS HARDWARE

Se han utilizado tres máquinas principalmente para el desarrollo del proyecto, las cuáles se describen a continuación:

- Intel Pentium IV 2,4 GHz, 1 GB RAM.
- AMD Athlon 64 Bits 3500+, 1 GB RAM.
- Intel Centrino 1.6 GHz, 512 MB RAM.

## 5.4.3 RECURSOS SOFTWARE

Entre los recursos software se encuentran cada uno de los programas, sistemas operativos, herramientas de programación y de documentación que han sido utilizadas a lo largo del desarrollo y documentación del prototipo.

- **Sistema Operativo**
  - Windows XP Professional.
  - Distribución de Linux basada en Debian: Ubuntu.
- **Herramientas de programación y librerías**
  - Máquina virtual de Java JDK 6.
  - Entorno de programación de dominio público Eclipse.
  - Librería de Java JRE 5.0.
  - Servidor de Base de Datos *MySQL 5.0*, ejecutado de forma local y *MySQLClient*.
  - Librería para manipulación de PDF a través de JAVA, iText.
- **Herramientas de documentación**
  - Procesador de textos Microsoft Office Word 2003.
  - Microsoft Office Visio 2003.
  - Adobe Acrobat Professional 6.0.

# 5.5

## ESPECIFICACIÓN DE REQUISITOS

En los apartados anteriores se ha llevado a cabo una descripción detallada del problema, y cómo se plantea solucionarlo mediante este proyecto. En la sección de especificación de requisitos nos centraremos en analizar los requisitos del sistema para obtener una idea clara de qué debe hacer el sistema, sin tener en cuenta por el momento el cómo debe hacerlo.

Para la especificación de requisitos se utilizará UML ya que éste se basa en una metodología orientada a objetos, ideal para el proyecto [Booch et al., 1999].

Esta sección dentro de la documentación se encuentre dividida en dos apartados:

- Identificación de requisitos.
- Formalización de requisitos.

### 5.5.1 IDENTIFICACIÓN DE REQUISITOS

Para llevar a cabo una correcta especificación de requisitos lo primero que vamos a hacer es llevar a cabo una descripción inicial de los mismos en un lenguaje formal, para llevar a cabo una especificación de los mismos.

#### **5.5.1.1 Requisito Imprimir Información Autores**

- *Identificador:* req1.
- *Nombre:* imprimir información autores.
- *Descripción:* impresión de la información almacenada por el grupo de investigación sobre los autores que colaboran con el mismo. Debe permitir imprimir tanto un listado de todos los autores que maneja como la información detallada de un autor previamente seleccionado.
- *Tipo:* Requisito funcional.
- *Prioridad:* Debe tener.

#### **5.5.1.2 Requisito Imprimir Información Integrante**

- *Identificador:* req2.
- *Nombre:* imprimir información integrante.
- *Descripción:* impresión de la información almacenada por el grupo de investigación sobre los integrantes del mismo. Debe permitir imprimir tanto un listado de todos los integrantes del grupo como el currículum vitae de un integrante previamente seleccionado.
- *Tipo:* Requisito funcional.
- *Prioridad:* Debe tener.

#### **5.5.1.3 Requisito Imprimir Información Acta**

- *Identificador:* req3
- *Nombre:* imprimir información acta.
- *Descripción:* impresión de la información almacenada por el grupo de investigación sobre las actas manejadas por el mismo. Debe

permitir imprimir tanto un listado de todas las actas del grupo como los detalles de un acta previamente seleccionado.

- *Tipo:* Requisito funcional.
- *Prioridad:* Debe tener.

#### **5.5.1.4 Requisito Imprimir Información Artículo**

- *Identificador:* req4.
- *Nombre:* imprimir información artículo.
- *Descripción:* impresión de la información almacenada por el grupo de investigación sobre los artículos manejados por el mismo. Debe permitir imprimir tanto un listado de todos los artículos del grupo como los detalles de un artículo previamente seleccionado.
- *Tipo:* Requisito funcional.
- *Prioridad:* Debe tener.

#### **5.5.1.5 Requisito Imprimir Información Capítulo**

- *Identificador:* req5.
- *Nombre:* imprimir información capítulo.
- *Descripción:* impresión de la información almacenada por el grupo de investigación sobre los capítulos de libros manejados por el mismo. Debe permitir imprimir tanto un listado de todos los capítulos del grupo como los detalles de un capítulo de libro previamente seleccionado.
- *Tipo:* Requisito funcional.
- *Prioridad:* Debe tener.

#### **5.5.1.6 Requisito Imprimir Información Congreso**

- *Identificador:* req6.
- *Nombre:* imprimir información congreso.
- *Descripción:* impresión de la información almacenada por el grupo de investigación sobre los congresos manejados por el mismo. Debe permitir imprimir tanto un listado de todos los congresos en los que participa el grupo como los detalles de un congreso previamente seleccionado.
- *Tipo:* Requisito funcional.
- *Prioridad:* Debe tener.

#### **5.5.1.7 Requisito Imprimir Información Libro**

- *Identificador:* req7.
- *Nombre:* imprimir información libro.
- *Descripción:* impresión de la información almacenada por el grupo de investigación sobre los libros manejados por el mismo. Debe permitir imprimir tanto un listado de todos los libros del grupo como los detalles de un libro previamente seleccionado.
- *Tipo:* Requisito funcional.
- *Prioridad:* Debe tener.

#### **5.5.1.8 Requisito Imprimir Información Revista**

- *Identificador:* req8.
- *Nombre:* imprimir información revista.
- *Descripción:* impresión de la información almacenada por el grupo de investigación sobre las revistas manejadas por el mismo. Debe

permitir imprimir tanto un listado de todas las revistas del grupo como los detalles de una revista previamente seleccionada.

- *Tipo:* Requisito funcional.
- *Prioridad:* Debe tener.

## 5.5.2 FORMALIZACIÓN DE REQUISITOS

A continuación vamos a llevar a cabo una formalización de los requisitos identificados y descritos con anterioridad. Para ello haremos uso del lenguaje de modelado UML como ya se comentó anteriormente [Booch et al., 1999].

Dentro de la metodología UML utilizaremos los diagramas que mejor describe funcionalmente el sistema, dichos diagramas son los Diagramas de casos de uso, el cual representa la interacción de entidades externas con el sistema en estudio.

Para la formalización de los requisitos seguiremos unos pasos determinados, que son:

- Identificación de los actores del sistema.
- Especificación de los casos de uso.
- Matriz requisitos / casos de uso.

### 5.5.2.1 Identificación de los actores

En nuestra aplicación, formularios de impresión podemos encontrar un único actor, el actor principal, el cual estará desempeñado por el usuario de la aplicación, los cuales podrán ser cualquiera de los miembros del grupo de investigación que haga uso de la aplicación.

**Tabla 5.5.3:** Descripción Actor *Usuario*

ACTOR: Usuario	
Identificador	1
Id – Padre	-

**Descripción**

Representa a una persona que interactúa con el sistema a través de su interfaz gráfica de usuario.

### **5.5.2.2 Especificación de los casos de uso**

En esta sección vamos a identificar las formas en las que el usuario va a interactuar con el sistema, para ello vamos a identificar los diferentes casos de uso que encontramos en nuestra aplicación, es decir, las diferentes formas que va a tener el usuario para llevar a cabo la visualización de información a través de los formularios.

#### **5.5.2.2.1 Casos de uso de trazo grueso**

Los casos de uso de trazo grueso identificados en nuestro sistema son:

7. Imprimir Información Autor.
8. Imprimir Listado Autores.
9. Imprimir Currículum Vitae.
10. Imprimir Listado Integrantes.
11. Imprimir Información Acta.
12. Imprimir Listado Actas.
13. Imprimir Información Artículo.
14. Imprimir Listado Artículos.
15. Imprimir Información Capítulo.
16. Imprimir Listado Capítulos.
17. Imprimir Información Congreso.
18. Imprimir Listado Congresos.
19. Imprimir Información Libro.
20. Imprimir Listado Libros.



21. Imprimir Información Revista.

22. Imprimir Listado Revistas.

23. Crear, Formatear y Mostrar Pdf.

Ahora vamos a llevar a cabo una descripción de los diferentes casos de uso de trazo grueso identificados.

**Tabla 5.5.4:** Descripción del caso de uso *Imprimir Información Autor*

CASO DE USO: Imprimir Información Autor	
<b>Identificador</b>	1
<b>Id – requisito</b>	req1
<b>Descripción</b>	Imprimir información detallada de alguno de los autores que son mantenidos en la base de datos del grupo de investigación.

**Tabla 5.5.5:** Descripción del caso de uso *Imprimir Listado Autores*

CASO DE USO: Imprimir Listado Autores	
<b>Identificador</b>	2
<b>Id – requisito</b>	req1
<b>Descripción</b>	Imprimir un listado de todos los autores que son mantenidos en la base de datos del grupo de investigación.

**Tabla 5.5.6:** Descripción del caso de uso *Imprimir Currículo Vitae*

CASO DE USO: Imprimir Currículum Vitae	
<b>Identificador</b>	3
<b>Id – requisito</b>	req2
<b>Descripción</b>	Imprimir el currículum vitae de alguno de los integrantes del grupo de investigación.

**Tabla 5.5.7:** Descripción del caso de uso *Imprimir Listado Integrantes*

CASO DE USO: Imprimir Listado Integrantes	
<b>Identificador</b>	4
<b>Id – requisito</b>	req2
<b>Descripción</b>	Imprimir un listado de todos los integrantes del grupo de investigación.

**Tabla 5.5.8:** Descripción del caso de uso *Imprimir Información Acta*

CASO DE USO: Imprimir Información Acta	
<b>Identificador</b>	5
<b>Id – requisito</b>	req3
<b>Descripción</b>	Imprimir información detallada de alguna de las actas que son mantenidas en la base de datos del grupo de investigación.

**Tabla 5.5.9:** Descripción del caso de uso *Imprimir Listado Actas*

CASO DE USO: Imprimir Listado Actas	
<b>Identificador</b>	6
<b>Id – requisito</b>	req3
<b>Descripción</b>	Imprimir un listado de todas las actas que son mantenidas en la base de datos del grupo de investigación.

**Tabla 5.5.10:** Descripción del caso de uso *Imprimir Información Artículo*

CASO DE USO: Imprimir Información Artículo	
<b>Identificador</b>	7
<b>Id – requisito</b>	req4
<b>Descripción</b>	Imprimir información detallada de alguno de los artículos que son mantenidos en la base de datos del grupo de investigación.

**Tabla 5.5.11:** Descripción del caso de uso *Imprimir Listado Actas*

CASO DE USO: Imprimir Listado Actas	
<b>Identificador</b>	8
<b>Id – requisito</b>	req4
<b>Descripción</b>	Imprimir un listado de todos los artículos que son mantenidos en la base de datos del grupo de investigación.

**Tabla 5.5.12:** Descripción del caso de uso *Imprimir Información Capítulo*

CASO DE USO: Imprimir Información Capítulo	
<b>Identificador</b>	9
<b>Id – requisito</b>	req5

<b>Descripción</b>	Imprimir información detallada de alguno de los capítulos que son mantenidos en la base de datos del grupo de investigación.
--------------------	--

**Tabla 5.5.13:** Descripción del caso de uso *Imprimir Listado Capítulos*

CASO DE USO: Imprimir Listado Capítulos	
<b>Identificador</b>	10
<b>Id – requisito</b>	req5
<b>Descripción</b>	Imprimir un listado de todos los capítulos que son mantenidos en la base de datos del grupo de investigación.

**Tabla 5.5.14:** Descripción del caso de uso *Imprimir Información Congreso*

CASO DE USO: Imprimir Información Congreso	
<b>Identificador</b>	11
<b>Id – requisito</b>	req6
<b>Descripción</b>	Imprimir información detallada de alguno de los congresos que son mantenidos en la base de datos del grupo de investigación.

**Tabla 5.5.15:** Descripción del caso de uso *Imprimir Listado Congresos*

CASO DE USO: Imprimir Listado Congresos	
<b>Identificador</b>	12
<b>Id – requisito</b>	req6
<b>Descripción</b>	Imprimir un listado de todos los congresos que son mantenidos en la base de datos del grupo de investigación.

**Tabla 5.5.16:** Descripción del caso de uso *Imprimir Información Libro*

CASO DE USO: Imprimir Información Libro	
<b>Identificador</b>	13
<b>Id – requisito</b>	req7
<b>Descripción</b>	Imprimir información detallada de alguno de los libros que son mantenidos en la base de datos del grupo de investigación.

**Tabla 5.5.17:** Descripción del caso de uso *Imprimir Listado Libros*

CASO DE USO: Imprimir Listado Libros	
<b>Identificador</b>	14

<b>Id – requisito</b>	req7
<b>Descripción</b>	Imprimir un listado de todos los libros que son mantenidos en la base de datos del grupo de investigación.

**Tabla 5.5.18** Descripción del caso de uso *Imprimir Información Revista*

CASO DE USO: Imprimir Información Revista	
<b>Identificador</b>	15
<b>Id – requisito</b>	req8
<b>Descripción</b>	Imprimir información detallada de alguno de las revistas que son mantenidas en la base de datos del grupo de investigación.

**Tabla 5.5.19:** Descripción del caso de uso *Imprimir Listado Revistas*

CASO DE USO: Imprimir Listado Revistas	
<b>Identificador</b>	16
<b>Id – requisito</b>	req8
<b>Descripción</b>	Imprimir un listado de todas las revistas que son mantenidas en la base de datos del grupo de investigación.

**Tabla 5.5.20:** Descripción del caso de uso *Crear, Formatear y Mostrar Pdf*

CASO DE USO: Crear, Formatear y Mostrar Pdf	
<b>Identificador</b>	17
<b>Id – padre</b>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 y 16
<b>Id – requisito</b>	req1, req2, req3, req4, req5, req6, req7 y req8
<b>Descripción</b>	Creación del pdf donde se realizará la impresión. Formateado del mismo, definiendo estilos e introduciendo el número de página. Y lanzamiento del pdf creado.

En la Figura 5.5.1 y 5.5.2 se muestra el diagrama de caso de uso de trazo grueso mostrando la interacción del usuario del sistema con éstos.

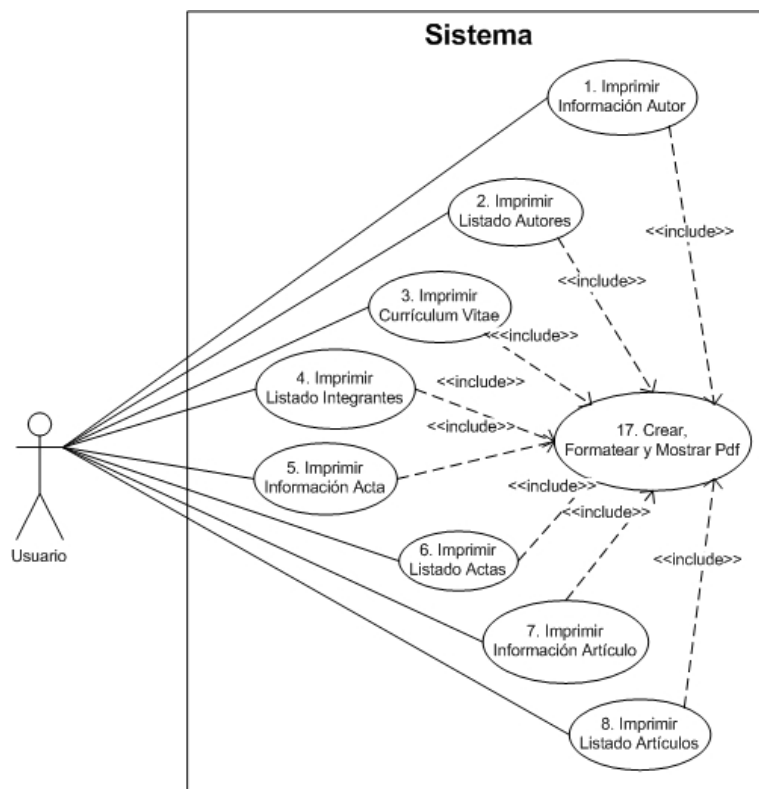


Figura 5.5.11: Diagrama de casos de uso de trazo grueso. Parte 1

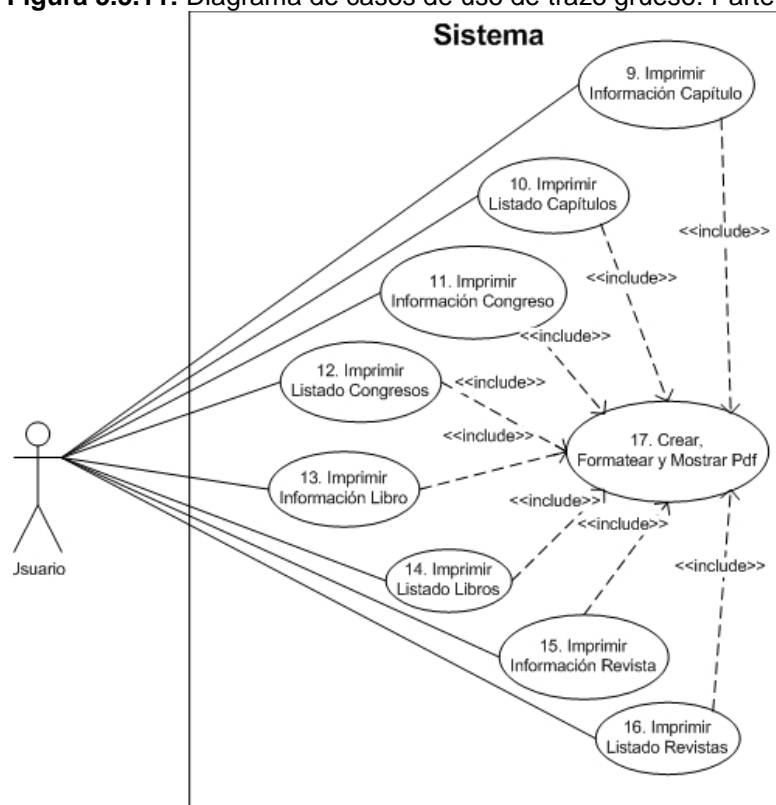


Figura 5.5.12: Diagrama de casos de uso de trazo grueso. Parte 2

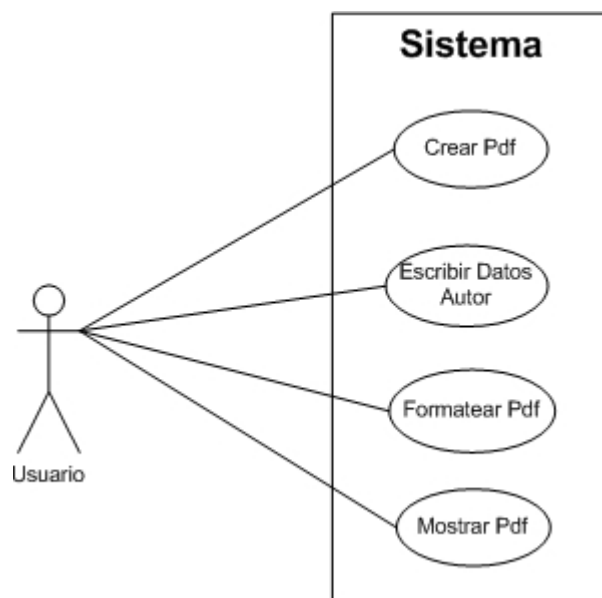
### 5.5.2.2.2 Refinamiento Casos de uso

Una vez que hemos descrito los casos de uso de trazo grueso y hemos visto como el actor de nuestro sistema interactúa con ellos, vamos a llevar a cabo un refinamiento de los casos de uso anteriores.

Todos los casos de uso que se refieren a la impresión de la información de un elemento determinado (acta, autor,...) trabajan de la misma manera, por lo tanto sólo refinaremos uno de ellos siendo el refinamiento de resto similares (sólo se diferencia en el contenido de la información que imprimen). Igualmente, todos los casos de uso de listado también trabajan de manera similar, por lo tanto también se refinará un único caso de uso de éstos.

#### 5.5.2.2.2.1 Caso de uso Imprimir Información Autor

El diagrama de caso de uso de imprimir la información de un autor se muestra en la Figura 5.5.3.



**Figura 5.5.13:** Diagrama caso de uso *Imprimir Información autor*

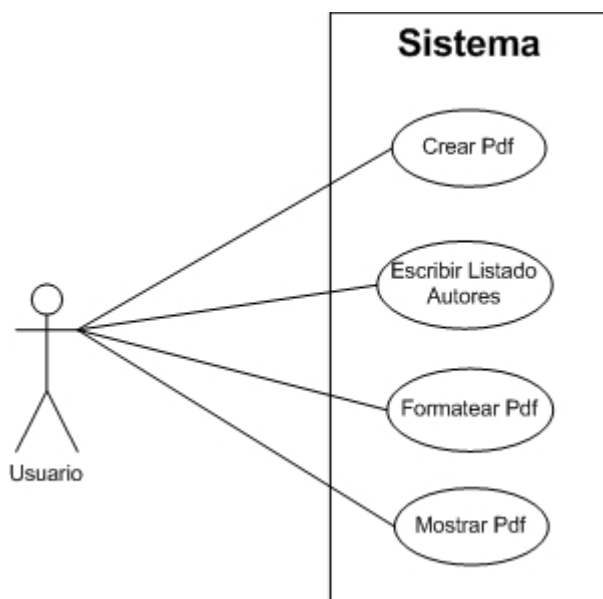
La especificación detallada del caso de uso *Imprimir Información Autor* se observa en la Tabla 5.5.19.

**Tabla 5.5.21.** Refinamiento caso de uso *Imprimir Información Autor*

<b>Caso de uso: Imprimir Información Autor</b>	
<b>Identificador</b>	1
<b>Ámbito</b>	Módulo Impresión
<b>Nivel</b>	Usuario
<b>Contexto Utilización</b>	El usuario quiere imprimir los datos de un autor concreto
<b>Actor Principal</b>	Usuario
<b>Precondiciones</b>	Que el usuario haya seleccionado en la aplicación el autor cuyos datos quiere imprimir y haya pulsado el botón de impresión correspondiente.
<b>Flujo principal</b>	<ol style="list-style-type: none"> <li>1. Se crea el pdf donde se imprimirán los datos del autor.</li> <li>2. Se escriben en el pdf los datos del autor especificado.</li> <li>3. El pdf se formatea y se introduce el número de página en cada hoja.</li> <li>4. Se muestra por pantalla el pdf creado.</li> </ol>
<b>Postcondiciones</b>	Se habrán impreso los datos en un pdf y se estará mostrando por pantalla.
<b>Flujos alternativos</b>	Ninguno

#### 5.5.2.2.2 Caso de uso Imprimir Listado Autores

El diagrama de caso de uso de imprimir el listado de autores se muestra en la Figura 5.5.4.



**Figura 5.5.14:** Diagrama caso de uso *Imprimir Listado Autores*

La especificación detallada del caso de uso *Imprimir Listado Autores* se observa en la Tabla 5.5.20.

**Tabla 5.5.22.** Refinamiento caso de uso *Imprimir Listado Autores*

Caso de uso: Imprimir Listado Autores	
Identificador	2
Ámbito	Módulo Impresión
Nivel	Usuario
Contexto Utilización	El usuario quiere imprimir un listado de todos los usuarios mantenidos por el sistema
Actor Principal	Usuario
Precondiciones	Que el usuario haya pulsado el botón de impresión correspondiente.
Flujo principal	<ol style="list-style-type: none"> <li>1. Se crea el pdf donde se imprimirá el listado</li> <li>2. Se escribe en el pdf el listado de los autores mantenidos por el sistema. De cada autor se escribirán los datos más importantes.</li> <li>3. El pdf se formatea y se introduce el número de página en cada hoja.</li> <li>4. Se muestra por pantalla el pdf creado.</li> </ol>
Postcondiciones	Se habrán impreso el listado en un pdf y se estará



	mostrando por pantalla.
Flujos alternativos	Ninguno

### 5.5.2.3 Matriz requisitos / casos de uso

En la Tabla 5.5.21 se muestra la matriz que resalta que requisitos son cubiertos por qué casos de uso, dejando claro que todos los requisitos impuestos han sido cumplido mediante el planteamiento que se ha seguido con los casos de uso detallados.

**Tabla 5.5.23:** Matriz Requisitos / Casos de Uso

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Req1	X	X															X
Req2			X	X													X
Req3					X	X											X
Req4							X	X									X
Req5									X	X							X
Req6											X	X					X
Req7													X	X			X
Req8															X	X	X

# 5.6

## ANÁLISIS DEL SISTEMA

Los objetivos de este capítulo son: especificar las características funcionales del software, establecer la interfaz que dicho software va a tener con los demás elementos del sistema. Todo ello con el fin de de crear un modelo de análisis que muestre, a un nivel de abstracción poco profundo, el funcionamiento del sistema, la información que maneja el mismo y como maneja dicha información. Mediante este modelo se obtendrá como resultado la implementación de las clases que utiliza el sistema dentro del contexto de este modelo.

Para la especificación de dicho análisis se empleará el Lenguaje Unificado de Modelado (UML), dado que proporciona las reglas sintácticas, semánticas y prácticas y el vocabulario suficiente para combinar palabras de éste con el objetivo de lograr la comunicación creando, a su vez, modelo bien formados [Booch, 1999]

### 5.6.1 MODELO DE ANÁLISIS

El *modelo de análisis* utiliza el modelo de casos de uso como entrada. Es una especificación detallada de los requisitos y funciona como primera aproximación del modelo de diseño. Se utiliza para comprender de manera más precisa el modelo de casos de uso *refinándolos* en forma de

*clasificadores conceptuales (clases de análisis, diferentes de los clasificadores de diseño que serán objeto de implementación).*

El modelo de análisis es diferente del modelo de diseño en que es un modelo puramente conceptual en lugar de ser un esquema de implementación. Puede ser transitorio o mantenerse durante toda la vida del sistema en proyectos complejos. Cada elemento del modelo de análisis es *trazable* a partir de los elementos del modelo de diseño que lo realizan.

Cada caso de uso en el modelo de casos de uso se traducirá en una *realización* de caso de uso de colaboración en el modelo de análisis. Esta realización identifica cómo se lleva a cabo la funcionalidad representada por el caso de uso de colaboración bajo la dinámica de una sociedad de clases del modelo de análisis. En la mayoría de las ocasiones un caso de uso es realizado exactamente por una colaboración que se considera implícita en el modelo y no es necesario representarse en los diagramas de interacción. *La dualidad caso de uso/realización es la base de la trazabilidad directa entre los requisitos y el análisis.*

### **¿Cómo realizar el modelo de análisis?**

El modelo de análisis estructura los requisitos para facilitar su comprensión, preparación, modificación y mantenimiento. La estructuración del modelo (basada en clases de análisis y paquetes de análisis) es independiente de la estructura basada en casos de uso. Sin embargo, existe una trazabilidad directa entre ambas estructuras (casos de uso/realizaciones de casos de uso), de forma que es posible hacer una traza de diferentes descripciones (en diferentes niveles de detalle) del mismo requisito y mantener su consistencia mutua con facilidad. La estructura de los requisitos que proporciona el modelo de análisis sirve como entrada fundamental para dar forma al sistema en su totalidad incluyendo la arquitectura, debido a que el objetivo es construir el sistema como un todo mantenible y no sólo describir sus requisitos.

Es muy importante que el modelo de análisis se limite únicamente a mostrar aquellas clases que formen parte del vocabulario del dominio del

problema (*requisitos de negocio*). Es preciso mantener el modelo de análisis como una declaración concisa y sencilla de la estructura y comportamiento del sistema.

Los artefactos de análisis son:

- Clases de análisis. Modelan conceptos clave en el dominio del negocio.
- Realizaciones de caso de uso. Representan cómo las instancias de las clases de análisis pueden interactuar para realizar el comportamiento especificado por un caso de uso.

#### **5.6.1.1 Especificación del Modelo de Clases**

Una vez analizados los casos de uso identificados anteriormente, se procede a realizar un refinamiento de los casos de uso más relevantes.

El modelo de clases describe la *estructura estática general del sistema*. Este modelo únicamente puede mostrar clases y cada clase aparece en él sólo una vez [Booch, 1999].

Una *clase* es una abstracción de las cosas que forman parte del vocabulario del sistema. Una clase representa un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Un *atributo* representa una propiedad del elemento que se está modelando. Una *operación* o *método* es la implementación de un servicio que puede ser requerido a cualquier objeto de la clase para que muestre un comportamiento.

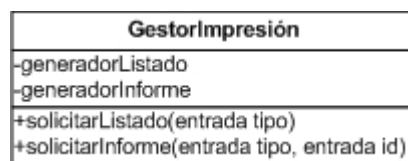
A continuación se realizará una descripción de cada una de las clases del modelo de clases especificando para cada una de ellas sus características significativas. La representación gráfica de cada clase se realizará utilizando la notación de UML.

### 5.6.1.1.1 Clase: GestorImpresionImpresión

**Tabla 5.6.4:** Especificación de la clase de análisis: GestorInformación

CLASE DE ANÁLISIS: <i>GestorImpresión</i>	
<b>Descripción</b>	La clase GestorImpresión se encarga de realizar la gestión de los procesos de impresión del sistema GIGI. Se encarga de recibir las solicitudes de impresión de los usuarios y tramitarlas.
<b>Atributos</b>	<p><b>generadorListado:</b> Mantiene una referencia a un objeto GeneradorListado.</p> <p><b>generadorInforme:</b> Mantiene una referencia a un objeto GeneradorInforme</p>
<b>Operaciones</b>	<p><b>solicitarListado (tipo):</b> Permite al usuario solicitar la impresión de algunos de los listados. El parámetro tipo indica sobre que entidad se quiere el listado.</p> <p><b>solicitarInforme (tipo,id):</b> Permite al usuario solicitar la impresión de la información detalla de algún elemento de una entidad. El parámetro <i>tipo</i> indica de que entidad es el elemento cuya información se quiere imprimir y el parámetro <i>id</i> sirve para identificar al elemento como tal.</p>
<b>Estereotipos</b>	<<control>>
<b>Valores etiquetados</b>	

En la Figura 5.6.1 se puede ver la representación en UML de la clase de análisis *GestorImpresión*.



**Figura 5.6.8:** Clase Análisis *GestorImpresión*

### 5.6.1.1.2 Clase: GeneradorListado

**Tabla 5.6.5:** Especificación de la clase de análisis: GeneradorListado

CLASE DE ANÁLISIS: <i>GeneradorListado</i>	
<b>Descripción</b>	Mediante esta clase se genera el listado impreso seleccionado por el usuario y para ello llevara a cabo operaciones de obtención de información de la base de datos.
<b>Atributos</b>	

**listados:** Representa el conjunto de listados que genera esta clase

**tipoListado:** Se utiliza para conservar el listado que desea el usuario.

#### Operaciones

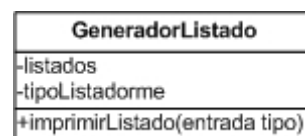
**imprimirListado (tipo):** Imprime el listado indicado en el parámetro *tipo*. Posteriormente lo muestra.

#### Estereotipos

<<interfaz>>

#### Valores etiquetados

En la Figura 5.6.2 se puede ver la representación en UML de la clase de análisis *GeneradorListado*.



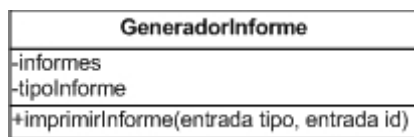
**Figura 5.6.9:** Clase Análisis GeneradorListado

### 5.6.1.1.3 Clase: GeneradorInforme

**Tabla 5.6.6:** Especificación de la clase de análisis: GeneradorInforme

CLASE DE ANÁLISIS: <i>GeneradorInforme</i>	
<b>Descripción</b>	Mediante esta clase se genera un informe impreso detallado del elemento seleccionado por el usuario y para ello llevara a cabo operaciones de obtención de información de la base de datos.
<b>Atributos</b>	<p><b>informes:</b> Representa el conjunto de informes que genera crear esta clase</p> <p><b>tipoInforme:</b> Se utiliza para conservar el tipo de entidad sobre la que desea el informe.</p> <p><b>idInforme:</b> Se utiliza para conservar el elemento sobre el que desea el informe.</p>
<b>Operaciones</b>	<b>imprimirInforme (tipo,id):</b> Imprime el informe del elemento <i>id</i> perteneciente al tipo de entidad <i>tipo</i> .
<b>Estereotipos</b>	<<interfaz>>
<b>Valores etiquetados</b>	

En la Figura 5.6.3 se puede ver la representación en UML de la clase de análisis *GeneradorInforme*.



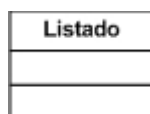
**Figura 5.6.10:** Clase Análisis GeneradorInforme

#### 5.6.1.1.4 Clase: Listado

**Tabla 5.6.7:** Especificación de la clase de análisis: Listado

CLASE DE ANÁLISIS: <i>Listado</i>	
<b>Descripción</b>	Esta clase representa un listado determinado que ha sido solicitado por el usuario y generado por el GeneradorListado.
<b>Atributos</b>	
<b>Operaciones</b>	
<b>Estereotipos</b>	
<b>Valores etiquetados</b>	<<entidad>>

En la Figura 5.6.4 se puede ver la representación en UML de la clase de análisis *Listado*.



**Figura 5.6.11:** Clase Análisis Listado

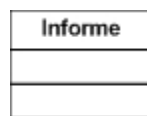
#### 5.6.1.1.5 Clase: Informe

**Tabla 5.6.8:** Especificación de la clase de análisis: Informe

CLASE DE ANÁLISIS: <i>Informe</i>	
<b>Descripción</b>	Esta clase representa un informe determinado que ha sido solicitado por el usuario y generado por el GeneradorInforme.
<b>Atributos</b>	



En la Figura 5.6.5 se puede ver la representación en UML de la clase de análisis *Informe*.



**Figura 5.6.12:** Clase Análisis Informe

### **5.6.1.2 Especificación del modelo Objeto de Análisis – Relación**

Mediante el modelo objeto de análisis – relación se mostrarán las relaciones entre las clases indicando la naturaleza, cardinalidades, dependencias, etc., de cada una de ellas. Cada clase tiene una serie de responsabilidades (servicios) que puede cumplir de dos formas distintas:

3. De forma particular usando operaciones para manipular sus propios atributos.
4. Colaborando con otras clases. La colaboración es una relación de interacción con otra clase para la realización de una responsabilidad.

Para una mayor claridad y comprensión del modelo objeto-relación se realizará, en primer lugar, una descripción de cada una de las relaciones entre los objetos del modelo de clases y, en segundo lugar, una representación del diagrama de clases resultante.



#### 5.6.1.2.1 Relación: GestorImpresión – GeneradorListado

Un objeto de tipo *GestorImpresión* provee de métodos para acceder a los procedimientos de impresión de listados proporcionados por el *GeneradorListado*. Este objeto hará uso de un objeto de tipo *GeneradorListado*.

En la Figura 5.6.6 se puede ver la representación gráfica en notación UML de la relación *GestorImpresión* – *GeneradorListado*. A efectos de simplificación no se muestran los atributos ni operaciones de las clases.



Figura 5.6.13: Relación: GestorImpresión – GeneradorListado

#### 5.6.1.2.2 Relación: GestorImpresión – GeneradorInforme

Un objeto de tipo *GestorImpresión* provee de métodos para acceder a los procedimientos de impresión de informes proporcionados por el *GeneradorInforme*. Este objeto hará uso de un objeto de tipo *GeneradorInforme*.

En la Figura 5.6.7 se puede ver la representación gráfica en notación UML de la relación *GestorImpresión* – *GeneradorInforme*. A efectos de simplificación no se muestran los atributos ni operaciones de las clases.

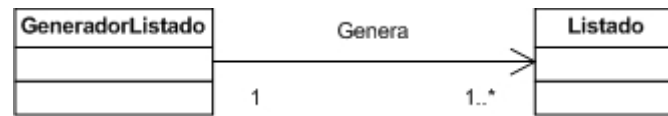


Figura 5.6.14: Relación: GestorImpresión – GeneradorInforme

#### 5.6.1.2.3 Relación: GeneradorListado – Listado

Esta relación de asociación indica que un objeto *GeneradorListado* genera uno o varios objetos *Listado*.

En la Figura 5.6.8 se puede ver la representación gráfica en notación UML de la relación *GeneradorListado* – *Listado*. A efectos de simplificación no se muestran los atributos ni operaciones de las clases.



**Figura 5.6.15:** Relación: GeneradorListado – Listado

#### 5.6.1.2.4 Relación: GeneradorInforme – Informe

Esta relación de asociación indica que un objeto *GeneradorInforme* genera uno o varios objetos Informe.

En la Figura 5.6.9 se puede ver la representación gráfica en notación UML de la relación *GeneradorInforme* – *Informe*. A efectos de simplificación no se muestran los atributos ni operaciones de las clases.

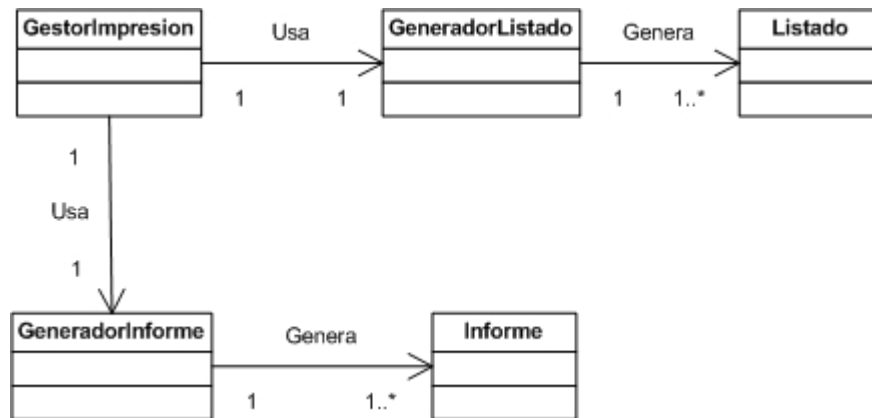


**Figura 5.6.16:** Relación: GeneradorInforme – Informe

#### 5.6.1.3 Diagrama de clases de análisis del sistema

Una vez analizadas las relaciones una a una se puede generar el diagrama de clases del modelo estructural estático del subsistema. Mediante este diagrama se pretende mostrar los aspectos estructurales acerca de las clases consideradas dentro del dominio del problema y de las relaciones principales necesarias entre dichas clases para cumplir los requisitos funcionales del subsistema.

En la Figura 5.6.10 se puede observar el diagrama de clases del subsistema propuesto.



**Figura 5.6.17:** Diagrama de Clases de Análisis del Subsistema

#### 5.6.1.4 Especificación del modelo Objeto de Análisis - Comportamiento

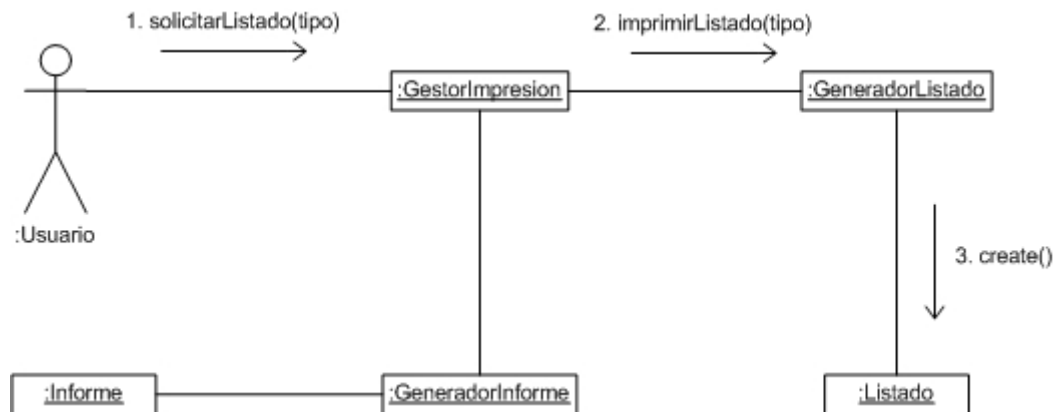
Una vez representados los elementos estáticos del sistema ahora es el momento de hacer una transición a su comportamiento dinámico. El aspecto dinámico del sistema combina la parte estructural con la dimensión del tiempo. El modelo *objeto-comportamiento* representa cómo interaccionan las clases, especificando un flujo de control (secuencia de mensajes entre objetos) a lo largo del tiempo, para la ejecución de una tarea.

UML proporciona dos tipos de diagramas de interacción [Booch, 1999], los diagramas de *secuencia* y los diagramas de *colaboración*. *Un diagrama de secuencia destaca la ordenación temporal de los mensajes entre objetos. Un diagrama de colaboración destaca la organización estructural de los objetos que envían y reciben mensajes.* Ambos son consistentes entre sí; representan la misma información pero desde diferentes perspectivas. En el contexto de los casos de uso un diagrama de interacción permite representar un escenario, que a su vez, representa un flujo particular de la acción asociada al caso de uso.

En los apartados siguientes sólo se desarrollan los diagramas de interacción más significativos dentro de la funcionalidad requerida.

#### 5.6.1.4.1 Diagrama de Colaboración: Imprimir Listado

En la Figura 5.6.11 se muestra el diagrama de colaboración que pretende representar el flujo de eventos principal correspondiente a la impresión de un listado cualquiera.



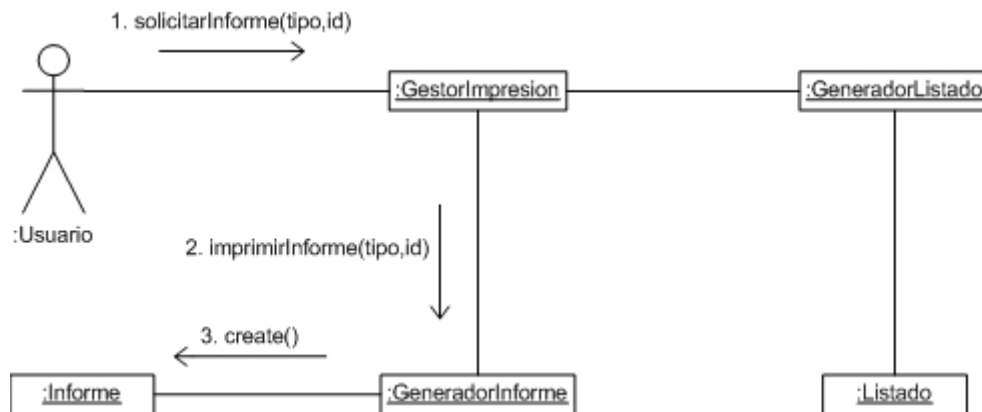
**Figura 5.6.18:** Diagrama de Colaboración Imprimir Listado

#### Flujo de sucesos:

6. El usuario solicita imprimir un listado.
7. El sistema recibe la solicitud de impresión, la procesa y llama al generador de listados.
8. El sistema crea y muestra el listado pedido.

#### 5.6.1.4.1 Diagrama de Colaboración: Imprimir Informe

En la Figura 5.6.12 se muestra el diagrama de colaboración que pretende representar el flujo de eventos principal correspondiente a la impresión de un informe cualquiera.



**Figura 5.6.19:** Diagrama de Colaboración Imprimir Informe

### Flujo de sucesos:

1. El usuario solicita imprimir un informe.
2. El sistema recibe la solicitud de impresión, la procesa y llama al generador de informes.
3. El sistema crea y muestra el informe pedido.

# 5.7

## DISEÑO DEL SISTEMA

En el capítulo anterior se ha realizado un análisis, desde el punto de vista técnico, de las principales características de la aplicación a desarrollar, así como del conjunto de requisitos que la misma deberá reunir para garantizar la correcta resolución del problema propuesto.

Dicho análisis se ha realizado considerando especialmente aquellos aspectos relativos a la funcionalidad que debe presentar el sistema, sin plantear la manera en la que ésta debe implementarse.

En este capítulo se proporcionará una descripción de la manera en que debe realizarse el Diseño del Sistema para garantizar que el mismo satisfaga la especificación de requisitos anteriormente desarrollada. El Diseño del Sistema se realizará utilizando la metodología UML, al igual que sucedió en la etapa dedicada a la Especificación de Requisitos [Booch et al., 1999].

Esta parte de la documentación la dividiremos en 4 apartados:

- Diagrama de paquetes
- Diagrama de clases
- Diagrama de interacción
- Diagrama de actividades

### 5.7.1 DIAGRAMA DE PAQUETES

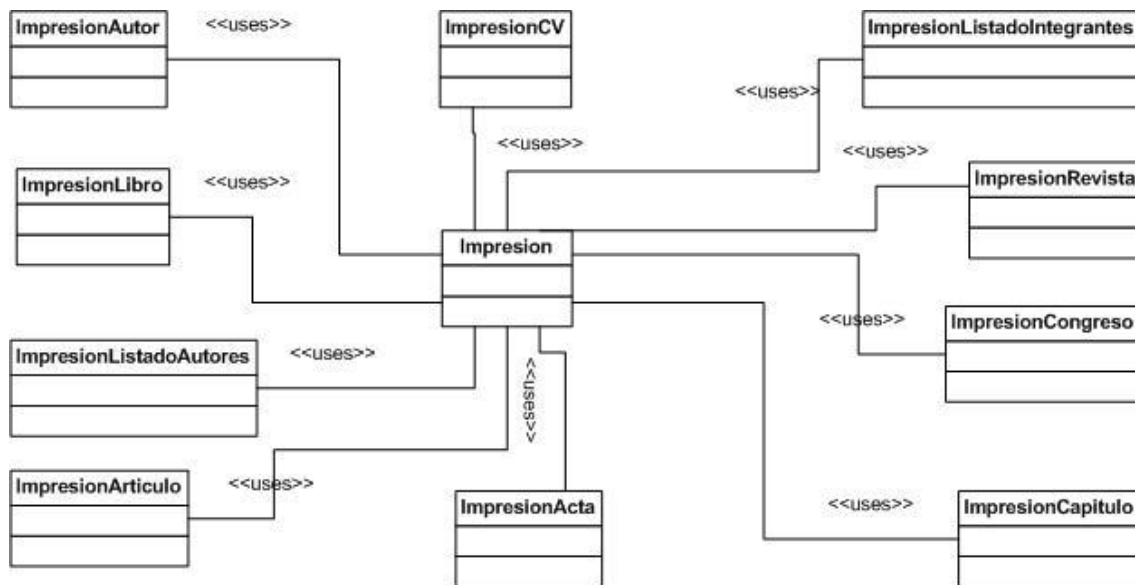
Este módulo ha sido desarrollado bajo un único paquete, llamado Imprimir que se comentará a continuación.

Este paquete está formado por las siguientes clases:

- Impresión
- ImpresiónAutor
- ImpresionCV
- ImpresionListadoAutores
- ImpresionListadoIntegrantes
- ImpresionActa
- ImpresionArticulo
- ImpresionCapitulo
- ImpresionCongreso
- ImpresionLibro
- ImpreisonRevista

### 5.7.2 DIAGRAMA DE CLASES

A continuación se explicará la relación que aparece entre las clases comentadas en el apartado anterior que conforman el paquete que contiene este módulo.



**Figura 5.7.14.** Diagrama de Clases de Diseño del Subsistema

A continuación vamos a detallar cada una de las clases que aparecen en este módulo.

### 5.7.2.1 Clase Impresión

Esta clase es la encargada de poner a disposición de las demás clases del módulo funciones comunes.

#### ***Visión preliminar de la clase***

Esta clase simplemente dará funcionalidad a las demás clases del módulo.

#### ***Dependencias de la clase***

No aparecen dependencias en esta clase.

#### ***Componentes de la clase***



Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

**c) Atributos:**

**d) Métodos:**

- **crearCabecera():**
  - Descripción: encargada de generar una cabecera dentro del documento PDF. Función estática.
  - Parámetros: Document doc (Documento PDF) y String texto (Texto para la cabecera)
  - Devuelve: nada
- **insertarNumerosPagina().**
  - Descripción: encargado de añadir el número de página a los documentos
  - Parámetros: String nombreFichero y bool primeraPagina (indica si la primera página debe llevar o no número)
  - Devuelve: nada

### **5.7.2.2 Clase ImpresiónAutor**

Esta clase es la encargada de generar el informe para la entidad Autor.

#### ***Visión preliminar de la clase***

Genera un fichero PDF y lo muestra con los datos obtenidos de la base de datos relacionados con autor.

#### ***Dependencias de la clase***

Utiliza la clase Impresion.

### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

#### **e) Atributos:**

- **nombre:** identifica a un autor
- **apellidos:** junto con el nombre identifica al autor
- **numeroLineas:** para controlar saltos de página
- **doc:** para escribir el documento
- **writer:** para escribir pdf.

#### **f) Métodos:**

- **ImpresionAutor():**
  - Descripción: constructor
  - Parámetros: nombre y apellidos.
  - Devuelve: nada
- **generarInforme():**
  - Descripción: crea el informe como tal llamando a las demás funciones.
  - Parámetros: nada
  - Devuelve: nada
- **imprimirDatosPersonales():**
  - Descripción: rellena sobre el pdf los datos personales del autor, accediendo a la BD.
  - Parámetros: nada
  - Devuelve: nada
- **imprimirArticulos():**

- Descripción: rellena sobre el pdf los datos de artículos del autor, accediendo a la BD.
  - Parámetros: nada
  - Devuelve: nada
- **imprimirLibros():**
  - Descripción: rellena sobre el pdf los datos de libros del autor, accediendo a la BD.
  - Parámetros: nada
  - Devuelve: nada
- **imprimirCapitulos():**
  - Descripción: rellena sobre el pdf los datos de capítulos del autor, accediendo a la BD.
  - Parámetros: nada
  - Devuelve: nada
- **imprimirActas():**
  - Descripción: rellena sobre el pdf los datos de las actas del autor, accediendo a la BD.
  - Parámetros: nada
  - Devuelve: nada
- **crearTitulo():**
  - Descripción: rellena sobre el pdf el título que se le pasa como argumento.
  - Parámetros: String titulo
  - Devuelve: nada
- **ImprimirLineaSeparadora():**

- Descripción: Introduce una línea separadora en el pdf.
- Parámetros: nada
- Devuelve: nada

- **numeroElementos():**

- Descripción: calcula el número de elementos que contiene un resultset pasado como argumento
- Parámetros: Resultset
- Devuelve: nada

### **5.7.2.3 Clase ImpresiónCV**

Esta clase es la encargada de generar el informe con el currículum de los integrantes.

#### ***Visión preliminar de la clase***

Genera un fichero PDF y lo muestra con los datos obtenidos de la base de datos generando el CV.

#### ***Dependencias de la clase***

Utiliza la clase Impresion.

#### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

##### **g) Atributos:**

- **nombre:** identifica a un autor

- **apellidos:** junto con el nombre identifica al autor
- **numeroLineas:** para controlar saltos de página
- **doc:** para escribir el documento
- **writer:** para escribir pdf.

#### h) Métodos:

- **ImpresionCV():**
  - Descripción: constructor
  - Parámetros: nombre y apellidos.
  - Devuelve: nada
- **generarInformeCV():**
  - Descripción: crea el informe como tal llamando a las demás funciones.
  - Parámetros: nada
  - Devuelve: nada
- **imprimirDatosPersonales():**
  - Descripción: rellena sobre el pdf los datos personales del integrante, accediendo a la BD.
  - Parámetros: nada
  - Devuelve: nada
- **imprimirArticulos():**
  - Descripción: rellena sobre el pdf los datos de artículos del integrante, accediendo a la BD.
  - Parámetros: nada
  - Devuelve: nada
- **imprimirLibros():**
  - Descripción: rellena sobre el pdf los datos de libros del integrante, accediendo a la BD.

- Parámetros: nada
  - Devuelve: nada
- **imprimirCapitulos():**
  - Descripción: rellena sobre el pdf los datos de capítulos del integrante, accediendo a la BD.
  - Parámetros: nada
  - Devuelve: nada
- **imprimirActas():**
  - Descripción: rellena sobre el pdf los datos de las actas del integrante, accediendo a la BD.
  - Parámetros: nada
  - Devuelve: nada
- **crearTitulo():**
  - Descripción: rellena sobre el pdf el título que se le pasa como argumento.
  - Parámetros: String titulo
  - Devuelve: nada
- **imprimirLineaSeparadora():**
  - Descripción: Introduce una línea separadora en el pdf.
  - Parámetros: nada
  - Devuelve: nada
- **numeroElementos():**
  - Descripción: calcula el número de elementos que contiene un resultset pasado como argumento
  - Parámetros: Resultset

- Devuelve: nada

- **ImprimirPortada():**

- Descripción: genera la portada del CV
- Parámetros: nada
- Devuelve: nada

- **ImprimirFormacionAcademica():**

- Descripción: rellena sobre el pdf los datos de la formación académica del integrante.
- Parámetros: nada
- Devuelve: nada

- **ImprimirSituacionProfesional():**

- Descripción: rellena sobre el pdf los datos de la situación profesional del integrante.
- Parámetros: nada
- Devuelve: nada

- **ImprimirActividades ():**

- Descripción: rellena sobre el pdf los datos sobre las actividades realizadas por el integrante.
- Parámetros: nada
- Devuelve: nada

- **ImprimirIdiomas():**

- Descripción: rellena sobre el pdf los datos de los idiomas estudiados por el integrante.
- Parámetros: nada
- Devuelve: nada

- **ImprimirProyectos():**
  - Descripción: rellena sobre el pdf los datos de los proyectos del integrante.
  - Parámetros: nada
  - Devuelve: nada
- **ImprimirContratos():**
  - Descripción: rellena sobre el pdf los datos de los contratos del integrante.
  - Parámetros: nada
  - Devuelve: nada
- **ImprimirPatentes():**
  - Descripción: rellena sobre el pdf los datos de las patentes del integrante.
  - Parámetros: nada
  - Devuelve: nada
- **ImprimirCentrosExtranjeros():**
  - Descripción: rellena sobre el pdf los datos de los centros en el extranjero en los que ha estado el integrante.
  - Parámetros: nada
  - Devuelve: nada
- **ImprimirTesis():**
  - Descripción: rellena sobre el pdf los datos de la tesis del integrante.
  - Parámetros: nada
  - Devuelve: nada



#### **5.7.2.4 Clase ImpresiónListadoAutores**

Esta clase es la encargada de generar el informe con el listado de autores del sistema.

##### ***Visión preliminar de la clase***

Genera un fichero PDF y lo muestra con los datos obtenidos de la base de datos en forma de listado de autores.

##### ***Dependencias de la clase***

Utiliza la clase Impresion.

##### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

##### **i) Atributos:**

- **numeroLineas:** para controlar saltos de página
- **doc:** para escribir el documento

##### **j) Métodos:**

- **ImpresionListadoAutores():**
  - Descripción: constructor
  - Parámetros: nada
  - Devuelve: nada
- **generarInformeListado():**
  - Descripción: crea el informe como tal llamando a la función encargada de añadir el listado.
  - Parámetros: nada
  - Devuelve: nada

- **imprimirAutores():**
  - Descripción: rellena sobre el pdf los datos personales más importantes de todos los autores en forma de listado.
  - Parámetros: nada
  - Devuelve: nada

#### **5.7.2.5 Clase ImpresiónListadoIntegrantes**

Esta clase es la encargada de generar el informe con el listado de integrantes del grupo de investigación.

##### ***Visión preliminar de la clase***

Genera un fichero PDF y lo muestra con los datos obtenidos de la base de datos en forma de listado de integrantes.

##### ***Dependencias de la clase***

Utiliza la clase Impresion.

##### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

##### **k) Atributos:**

- **numeroLineas:** para controlar saltos de página
- **doc:** para escribir el documento

##### **l) Métodos:**

- **ImpresionListadoIntegrantes():**
  - Descripción: constructor
  - Parámetros: nada

- Devuelve: nada
- **generarInformeListado():**
  - Descripción: crea el informe como tal llamando a la función encargada de añadir el listado.
  - Parámetros: nada
  - Devuelve: nada
- **imprimirMiembros():**
  - Descripción: rellena sobre el pdf los datos personales más importantes de todos los integrantes del grupo en forma de listado.
  - Parámetros: nada
  - Devuelve: nada

#### **5.7.2.6 Clase ImpresiónActa**

Esta clase es la encargada de generar el informe de las actas, bien sea de un acta concreta para lo cual generará un informe con los detalles, o un listado de las actas que se encuentran en la base de datos.

##### ***Visión preliminar de la clase***

Genera un fichero PDF y lo muestra con los datos obtenidos de la base de datos en forma de listado de actas, o conteniendo los detalles de una en particular. Para realizar uno u otro se sobrecarga el constructor de la clase.

##### ***Dependencias de la clase***

Utiliza la clase Impresion.

##### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

**m) Atributos:**

- **writer:** utilizado para escribir sobre el pdf.
- **rs:** resultset utilizado para la manipulación de la base de datos
- **rs2:** resultset utilizado para la manipulación de la base de datos
- **rs3:** resultset utilizado para la manipulación de la base de datos
- **doc :** documento que manipula el PDF
- **nombreFichero:** cadena que contiene el nombre del fichero a generar.

**n) Métodos:**

- **ImpresionActa ():**
  - Descripción: constructor que genera el listado de las actas
  - Parámetros: nada
  - Devuelve: nada
- **ImpresionActa():**
  - Descripción: genera el informe del acta que se le pasa como argumento
  - Parámetros: String id (identificador del acta)
  - Devuelve: nada

### **5.7.2.7 Clase ImpresiónArticulo**

Esta clase es la encargada de generar el informe de los artículos, bien sea de uno en concreto para lo cual generará un informe con los detalles, o un listado de los que se encuentran en la base de datos.

#### ***Visión preliminar de la clase***

Genera un fichero PDF y lo muestra con los datos obtenidos de la base de datos en forma de listado de artículos, o conteniendo los detalles de uno en particular. Para realizar uno u otro se sobrecarga el constructor de la clase.

#### ***Dependencias de la clase***

Utiliza la clase Impresion.

#### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

##### **o) Atributos:**

- **writer:** utilizado para escribir sobre el pdf.
- **rs:** resultset utilizado para la manipulación de la base de datos
- **rs2:** resultset utilizado para la manipulación de la base de datos
- **rs3:** resultset utilizado para la manipulación de la base de datos
- **rs4:** resultset utilizado para la manipulación de la base de datos
- **doc :** documento que manipula el PDF

- **nombreFichero:** cadena que contiene el nombre del fichero a generar.

**p) Métodos:**

- **ImpresionArticulo ():**
  - Descripción: constructor que genera el listado de los artículos
  - Parámetros: nada
  - Devuelve: nada
- **ImpresionArticulo():**
  - Descripción: genera el informe del artículo que se le pasa como argumento
  - Parámetros: String id (identificador del artículo)
  - Devuelve: nada

#### **5.7.2.8 Clase ImpresiónCapítulo**

Esta clase es la encargada de generar el informe de los capítulos, bien sea de uno en concreto para lo cual generará un informe con los detalles, o un listado de los que se encuentran en la base de datos.

##### ***Visión preliminar de la clase***

Genera un fichero PDF y lo muestra con los datos obtenidos de la base de datos en forma de listado de capítulos, o conteniendo los detalles de uno en particular. Para realizar uno u otro se sobrecarga el constructor de la clase.

##### ***Dependencias de la clase***

Utiliza la clase Impresion.

### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

#### **q) Atributos:**

- **writer:** utilizado para escribir sobre el pdf.
- **rs:** resultset utilizado para la manipulación de la base de datos
- **rs2:** resultset utilizado para la manipulación de la base de datos
- **rs3:** resultset utilizado para la manipulación de la base de datos
- **rs4:** resultset utilizado para la manipulación de la base de datos
- **doc :** documento que manipula el PDF
- **nombreFichero:** cadena que contiene el nombre del fichero a generar.

#### **r) Métodos:**

- **ImpresionCapitulo():**
  - Descripción: constructor que genera el listado de los capítulos
  - Parámetros: nada
  - Devuelve: nada
- **ImpresionCapítulo():**
  - Descripción: genera el informe del capítulo que se le pasa como argumento
  - Parámetros: String id (identificador del capítulo)
  - Devuelve: nada

### 5.7.2.9 Clase *ImpresiónCongreso*

Esta clase es la encargada de generar el informe de los congresos, bien sea de uno en concreto para lo cual generará un informe con los detalles, o un listado de los que se encuentran en la base de datos.

#### ***Visión preliminar de la clase***

Genera un fichero PDF y lo muestra con los datos obtenidos de la base de datos en forma de listado de congresos, o conteniendo los detalles de uno en particular. Para realizar uno u otro se sobrecarga el constructor de la clase.

#### ***Dependencias de la clase***

Utiliza la clase Impresion.

#### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

##### **s) Atributos:**

- **writer:** utilizado para escribir sobre el pdf.
- **rs:** resultset utilizado para la manipulación de la base de datos
- **rs2:** resultset utilizado para la manipulación de la base de datos
- **rs3:** resultset utilizado para la manipulación de la base de datos
- **doc :** documento que manipula el PDF
- **nombreFichero:** cadena que contiene el nombre del fichero a generar.

##### **t) Métodos:**



- **ImpresionCongreso():**
  - Descripción: constructor que genera el listado de los artículos
  - Parámetros: nada
  - Devuelve: nada
  
- **ImpresionCongreso():**
  - Descripción: genera el informe del artículo que se le pasa como argumento
  - Parámetros: String id (identificador del congreso)
  - Devuelve: nada

#### **5.7.2.10 Clase ImpresiónLibro**

Esta clase es la encargada de generar el informe de los libros, bien sea de uno en concreto para lo cual generará un informe con los detalles, o un listado de los que se encuentran en la base de datos.

##### ***Visión preliminar de la clase***

Genera un fichero PDF y lo muestra con los datos obtenidos de la base de datos en forma de listado de libros, o conteniendo los detalles de uno en particular. Para realizar uno u otro se sobrecarga el constructor de la clase.

##### ***Dependencias de la clase***

Utiliza la clase Impresion.

##### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

**u) Atributos:**

- **writer:** utilizado para escribir sobre el pdf.
- **rs:** resultset utilizado para la manipulación de la base de datos
- **rs2:** resultset utilizado para la manipulación de la base de datos
- **rs3:** resultset utilizado para la manipulación de la base de datos
- **rs4:** resultset utilizado para la manipulación de la base de datos
- **doc :** documento que manipula el PDF
- **nombreFichero:** cadena que contiene el nombre del fichero a generar.

**v) Métodos:**

- **ImpresionLibro():**
  - Descripción: constructor que genera el listado de los libros
  - Parámetros: nada
  - Devuelve: nada
- **ImpresionLibro():**
  - Descripción: genera el informe del libro que se le pasa como argumento
  - Parámetros: String id (identificador del libro)
  - Devuelve: nada

### **5.7.2.11 Clase ImpresiónRevista**

Esta clase es la encargada de generar el informe de las revistas, bien sea de uno en concreto para lo cual generará un informe con los detalles, o un listado de los que se encuentran en la base de datos.

#### ***Visión preliminar de la clase***

Genera un fichero PDF y lo muestra con los datos obtenidos de la base de datos en forma de listado de revistas, o conteniendo los detalles de uno en particular. Para realizar uno u otro se sobrecarga el constructor de la clase.

#### ***Dependencias de la clase***

Utiliza la clase Impresion.

#### ***Componentes de la clase***

Como componentes de la clase vamos a distinguir entre atributos y entre métodos.

##### **w) Atributos:**

- **writer:** utilizado para escribir sobre el pdf.
- **rs:** resultset utilizado para la manipulación de la base de datos
- **rs2:** resultset utilizado para la manipulación de la base de datos
- **rs3:** resultset utilizado para la manipulación de la base de datos
- **rs4:** resultset utilizado para la manipulación de la base de datos
- **doc :** documento que manipula el PDF

- **nombreFichero:** cadena que contiene el nombre del fichero a generar.

**x) Métodos:**

- **ImpresionRevista():**
  - Descripción: constructor que genera el listado de las revistas
  - Parámetros: nada
  - Devuelve: nada
- **ImpresionRevista():**
  - Descripción: genera el informe de la revista que se le pasa como argumento
  - Parámetros: String id (identificador de la revista)
  - Devuelve: nada

### 5.7.3 DIAGRAMA DE INTERACCIÓN

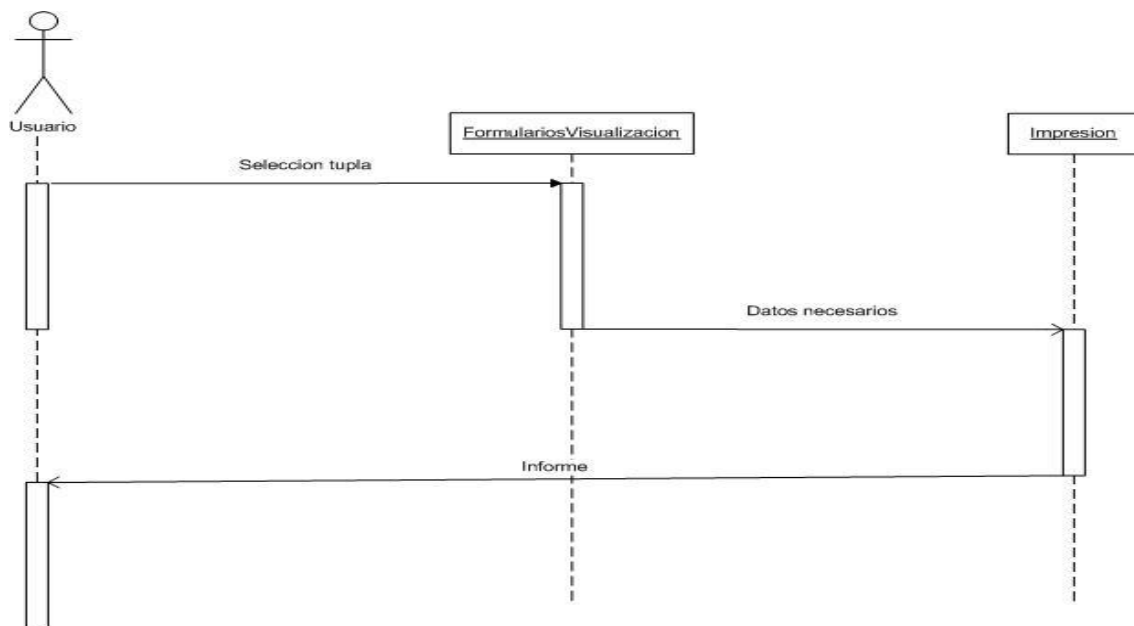
Los diagramas de interacción nos muestran el comportamiento dinámico del sistema, es decir, con dichos diagramas podemos ver como los objetos interaccionan.

Existen dos tipos de diagramas de interacción, los diagramas de secuencia los cuales nos dan una idea de la dimensión temporal, y los diagramas de colaboración, los cuales nos dan una idea de la dimensión estructural.

En esta práctica vamos a hacer solo mención a los diagramas de secuencia, ya que estamos más interesados en obtener la dimensión temporal de nuestro sistema, que la dimensión estructural.

### 5.7.3.1 Diagrama de secuencia

Este diagrama muestra la forma en que el usuario decide imprimir un informe de un elemento determinado mostrando sus detalles. Debe seleccionar el elemento y presionar el botón introducido para tal efecto.



**Figura 4.7.15:** Diagrama de Secuencia Imprimir Informe

Para generar un informe con un listado de los elementos de la base de datos simplemente se tendrá que presionar el botón de la interfaz imprimirListado, por lo cual no se ha realizado diagrama explicando el proceso.

### 5.7.4 DIAGRAMA DE ACTIVIDADES

Los diagramas de actividad pueden ser considerados como un diagrama de flujo que destaca la actividad que tiene lugar a lo largo del tiempo.

No se han incluido en esta documentación diagramas de actividad ya que las operaciones que se realizan son demasiado simples como para

detallarlas a tanto nivel, ya que se ha pensado que sería contraproducente, para el entendimiento del documento.

# 5.8

## IMPLEMENTACIÓN DEL SISTEMA

En esta sección se debe dar una visión de la interfaz utilizada por el módulo objeto del documento, y el código que la implementa.

El código fuente hemos creído conveniente no incluirlo, ya que al ser demasiado extenso dificultaría la lectura del presente documento, además al ser éste libre, el lector interesado podrá acceder a el mismo sin ningún tipo de problema.

Este módulo carece de interfaz ya que su funcionalidad es llamada a través del módulo formularios de visualización creado en una fase anterior del desarrollo.

# 5.9

## PRUEBAS DEL SISTEMA

A continuación se detalla la fase de pruebas a realizar sobre el sistema. Esta fase intentará provocar el mayor número de errores en nuestra aplicación, ya que su objetivo primordial es el de detectar posibles fallos en los requisitos, diseño e incluso implementación del sistema.

Para esto lo que intentaremos es realizar un número determinado de pruebas sobre el sistema, que abarque el mayor número posible de caminos dentro del código, o de requisitos. Es decir, si en el código aparecen dos caminos claramente diferenciados por una estructura condicional, debemos intentar que las pruebas analicen ambos caminos, para comprobar el correcto funcionamiento de ambos. Otro tipo de pruebas que se utilizarán son aquellas en las que se busca comprobar el correcto funcionamiento de las interfaces entre los distintos módulos de la aplicación, asegurando una correcta interacción entre los mismos.

El tipo de pruebas que llevaremos a cabo son:

- Pruebas de caja blanca.
- Pruebas de caja negra.



### **5.9.1 Pruebas de caja blanca**

En este tipo de pruebas se busca recorrer el mayor número de caminos dentro de la implementación. Como se comentó anteriormente, ante determinados tipos de estructuras, es conveniente probar todas las combinaciones posibles. Por ejemplo, las estructuras condicionales simples, o anidadas, en las que el número de posibilidades a probar sube exponencialmente.

Otro aspecto a tener en cuenta normalmente es el de los bucles, que suelen acaparar gran parte del tiempo de cómputo de las aplicaciones, y sobre los que tendremos que probar principalmente los puntos límite.

Este módulo en concreto hace uso exclusivo de la información introducida en la base de datos, sin embargo se pueden eliminar las pruebas que comprueban la corrección de dichos datos, ya que podemos suponer que los datos que extraemos de la base de datos se encuentran de forma correcta, ya que han tenido que pasar por el módulo de entrada, el cual controla dichos errores. También para acceder a las opciones de impresión han tenido que mostrarse los datos a través de los formularios de visualización, que también controlarían los errores en la extracción de información.

### **5.9.2 Pruebas de caja negra**

Estas pruebas son las encargadas de comprobar el correcto funcionamiento de las distintas interfaces de los distintos módulos que forma la aplicación, probando como cada módulo responde a través de sus interfaces, sin fijarnos en como internamente el código genera los valores.

Todas las pruebas que se comentan a continuación han sido desarrolladas en paralelo a la implementación y son:

- Comprobación de la comunicación entre los módulos de visualización e impresión, ya que las llamadas a este ultimo se harán siempre a través de visualización.

- Comprobación de que los informes generados son los adecuados, cumpliendo los estándares o los requisitos impuestos por el cliente.
- Comprobación de que los datos obtenidos de la base de datos son los adecuados accediendo en cada caso a los campos necesarios de cada una de las tablas para formar el informe, de forma que se cumplan las restricciones impuestas durante el diseño de la base de datos.

Podemos concluir que las pruebas llevadas a cabo han dado como resultado un funcionamiento adecuado del sistema, consiguiendo cumplir todos los requisitos impuestos para el sistema.

# BIBLIOGRAFÍA Y REFERENCIAS WEB

## BIBLIOGRAFIA

[Beckel, 2000]

Beckel, B. *Piensa en Java* Madrid. Prentice Hall. 2000. 960. ISBN:84-205-3192-8

[Booch, 1999]

Booch, G., Rumbaugh, J., Jacobson, I. *El Lenguaje Unificado de Modelado*. Madrid. Addison Wesley. 1999. 464. 84-7829-028-1.

[Gómez Nieto, 2001]

Gómez Nieto, M.A; Luque Ruiz, I. *Diseño y uso de Bases de datos relacionales*. 1ª Edicion. Ed. RAMA. ISBN: 84-7897-279-X.

[Irene, 2006]

Luque Ruiz, I, Transparencias de la asignatura Análisis y Diseño de Sistemas Informáticos de 4º I.Informática. Universidad de Córdoba. 2006.

[Pressman, 2002]

Pressman, R. S. Ingeniería del Software. Un enfoque práctico. 5ª ed. Madrid. Mc Graw Hill. 2002. 958 p. ISBN: 84-481-3214-9.

## REFERENCIAS WEB

- Enciclopedia libre en multiples idiomas

. <http://es.wikipedia.org>.

Ultimo Acceso: Abril de 2007.

- Tutorial de UML perteneciente a la universidad de Chile.

<http://www.dcc.uchile.cl/~psalinas/uml>.

Ultimo Acceso: Abril de 2007.

- Web dedicada a la programación.

<http://www.lawebdelprogramador.com>.

Ultimo Acceso: Abril de 2006.

- Web del grupo Alarcos de la universidad de Castilla la Mancha

<http://alarcos.inf-cr.uclm.es/>

Consulta: 15 Marzo 2007

- Articulo sobre la inserción de objetos binarios de gran tamaño en la base de datos.

[http://www.programacion.com/php/articluo/datos\\_blob](http://www.programacion.com/php/articluo/datos_blob)

Consulta: 15 Marzo 2007

- Foro sobre tecnología Java y MySql  
<http://forum.java.sun.com/thread.jspa?threadID576315>  
Consulta: 16 Marzo 2007
- Manual de referencia de MySQL 5.0  
<http://dev.mysql.com/doc/refman/5.0/es>  
Consulta: 16 Marzo 2007
- Herramientas GUI para descarga para MySQL 5.0  
<http://dev.mysql.com/downloads/gui-tools/5.0.html>  
Consulta: 16 Marzo 2007
- Ejemplos de procedimientos en MySQL  
<http://messagequeue.lenoxway.net/examples.php>  
Consulta: 17 Marzo
- Ejecutar sentencias SQL desde un fichero de texto  
<http://dev.mysql.com/doc/refman/5.0/es/batch-commands.html>  
Consulta: 20 Marzo 2007
- Copiar bases de datos MySQL a otra máquina  
<http://dev.mysql.com/doc/refman/5.0/es/upgrading-to-arch.html>  
Consulta: 25 Marzo 2007
- Ejemplos para cargar datos en una tabla

<http://www.webestilo.com/mysql/cargar-datos-tabla.phtml>

Consulta: 5 Abril 2007

- curso MySQL

<http://mysql.conclase.net/curso/index.php?cap=TimeZone>

Consulta: 8 Abril 2007

- Ejemplo para insertar ficheros en un campo de una tabla MySQL

<http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=262>

Consulta: 12 Abril 2007

- Biblioteca Electrónica acceso a base de datos de ISI Journal Citation Reports para analizar los índices de impacto y otros factores y criterios de calificación de revistas.

<http://www.uco.es>

Consulta: 14 Abril 2007

- Página perteneciente al grupo Programación en castellano

<http://www.programacion.net/java/tutorial/swing/>.

Consulta: Abril de 2007.

- Página perteneciente a: Chaiding

<http://www.chuidiang.com/java/> .

Consulta: Abril de 2007.