



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

“ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA GENERADOR DE
CÓDIGO PARA LA CARGA ÓPTIMA, CORRECCIÓN DE ERRORES Y
ESTANDARIZACIÓN DE DATOS PARA EL POSTERIOR USO DE ESTA
INFORMACIÓN EN EL PROCESO DE MEDIACIÓN TELEFÓNICA DE EMPRESAS
DE TELECOMUNICACIONES”

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN

Presentada por:

SERGIO CASTRO MEJÍA

WALTHER MALDONADO MOREIRA

GUAYAQUIL – ECUADOR

2004

AGRADECIMIENTO

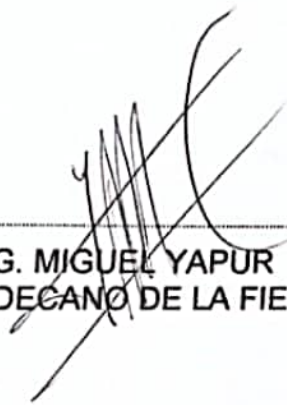
Agradecemos a cada una de las personas que hicieron posible la realización de esta tesis, de manera muy especial:

Al Msc. Xavier Ochoa, nuestro Director de Tesis, por su ayuda y valiosos comentarios para la realización de este trabajo.

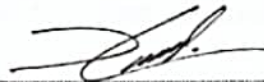
A los directivos de Yoveri S.A. y de Linkotel S.A., quienes confiaron en nosotros y nos facilitaron los recursos necesarios, permitiéndonos realizar este trabajo.

A nuestros padres, por su incondicional y constante apoyo.

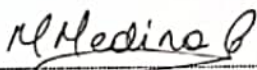
TRIBUNAL DE GRADUACION



ING. MIGUEL YAPUR
SUB DECANO DE LA FIEC



MSC. XAVIER OCHOA
DIRECTOR DE TESIS



ING. MERCEDES MEDINA
MIEMBRO DEL TRIBUNAL



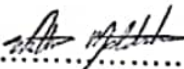
ING. KATHERINE CHILUIZA
MIEMBRO DEL TRIBUNAL

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, nos corresponden exclusivamente; y el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de Exámenes y Títulos profesionales de la ESPOL)


.....
Sergio Castro Mejía


.....
Walther Maldonado Moreira

RESUMEN

La problemática con la que se enfrentan las compañías de telecomunicaciones al lidiar con las distintas tecnologías que sus centrales usan para recolectar los consumos de voz, datos y servicios de valor agregado que estas proporcionan, transformarlos a un formato adecuado para su posterior consumo y realizar sobre estos datos procesos de auditoría, control anti-fraude y seguimiento de estadísticas, ha originado la necesidad de crear soluciones que permitan a los sistemas consumidores de dichos datos (por ej. facturación, control de costos, etc...) abstraerse de la complejidad de dicho proceso.

Una de las etapas más importantes de este problema es la transformación de los datos originales proporcionados por las distintas centrales de la compañía, a un formato común dependiente de las necesidades de la

empresa. De esta manera, los datos de múltiples orígenes pueden ser procesados unificadamente y de manera eficiente.

Este trabajo de Tesis presenta una solución a la interpretación y transformación de datos, aplicable a Linkotel S.A, por medio de la cual sus operadores podrán definir formatos de entrada de datos, así como los procesos que llevarán a cabo la transformación de dichos datos de manera eficiente y confiable.

En el capítulo 1, se da una introducción general y las justificaciones para la realización de este proyecto de tesis. Se analizan los conceptos más importantes de mediación, como son: sus objetivos principales, las etapas que lo conforman y las diversas industrias en las cuales puede ser aplicado este concepto. Se describe en particular la mediación en empresas de telefonía fija y se discute la etapa de transformación de datos de diversos orígenes hacia un formato convergente.

En el capítulo 2, se proporcionan los fundamentos y prerequisites teóricos sobre conceptos relacionados con estándares de definición, almacenamiento y manipulación de información, los cuales serán utilizados a lo largo de todo el proyecto.

En el capítulo 3, se describe el análisis de los requerimientos funcionales y operativos. Se incluyen descripciones detalladas de los patrones de formatos que el sistema debe permitir definir, así como de los tipos de procesos que deberán llevar a cabo la carga de información basada en los formatos definidos.

En el capítulo 4, se detalla el diseño del sistema, esto es su arquitectura tanto física como lógica, el diseño de los formatos y de los procesos que se van a llevar a cabo sobre estos.

En el capítulo 5, se revisa la implementación del sistema, las pruebas realizadas y los problemas que se presentaron durante la implementación del proyecto.

Finalmente se dan Conclusiones y Recomendaciones acerca de los siguientes pasos a seguir para la implementación completa de un sistema de mediación telefónica. Adicionalmente, se incluyen apéndices que comprenden: El Manual del Usuario, Casos de Uso y Escenarios del Sistema, los Diagramas de Clases, los Diagramas de Secuencia, Documentos XSD utilizados por el Sistema, Documentos XML generados por el Sistema para la descripción de formatos de centrales telefónicas y definición de un XML Schema Definition.

ÍNDICE GENERAL

RESUMEN	V
ÍNDICE GENERAL	VIII
ÍNDICE DE FIGURAS	XIV
ÍNDICE DE TABLAS	XXI
1. ANTECEDENTES Y JUSTIFICACIÓN	1
1.1 CONCEPTOS GENERALES DE MEDIACIÓN	1
1.1.1 DEFINICIÓN DE UN SISTEMA DE MEDIACIÓN	1
1.1.2 OBJETIVOS DE UN SISTEMA DE MEDIACIÓN.....	5
1.1.2.1 REQUERIMIENTOS FUNCIONALES DE UN SISTEMA DE MEDIACIÓN	6
1.1.2.2 REQUERIMIENTOS OPERATIVOS DE UN SISTEMA DE MEDIACIÓN.....	7
1.1.3 SUBSISTEMAS EN UN SISTEMA DE MEDIACIÓN	8
1.2 MEDIACIÓN EN EMPRESAS DE TELEFONÍA FIJA.....	14
1.2.1 SISTEMAS DE SOPORTE DE NEGOCIOS EN EMPRESAS DE TELEFONÍA FIJA.....	14

1.2.2	MEDIACIÓN DE TELEFONÍA FIJA EN EL ECUADOR – REGIÓN COSTA.....	18
1.2.3	SISTEMAS DE MEDIACIÓN TELEFÓNICA EN OTROS PAÍSES.....	20
1.2.3.1	INTER-MEDIATE.....	21
1.2.3.2	N*VISION	23
1.3	JUSTIFICACIÓN Y OBJETIVOS	28
1.3.1	JUSTIFICACIÓN DEL PROYECTO	28
1.3.2	OBJETIVOS Y ALCANCE DEL PROYECTO	29
2.	FUNDAMENTOS TEÓRICOS	31
2.1	ESTÁNDARES PARA REPRESENTACIÓN Y DEFINICIÓN DE INFORMACIÓN.....	31
2.1.1	XML COMO ESTÁNDAR DE REPRESENTACIÓN DE INFORMACIÓN	31
2.1.1.1	ANTECEDENTES.....	33
2.1.1.2	ESPECIFICACIÓN DEL XML ^[XXIX]	36
2.1.1.2.1	ESTRUCTURA LÓGICA DE LOS DOCUMENTOS XML	38
2.1.1.2.2	COMPONENTES Y DEFINICIONES	39
2.1.1.2.3	XML VERSIÓN 1.1	42
2.1.2	XML SCHEMA COMO ESTÁNDAR DE VALIDACIÓN DE DOCUMENTOS XML.....	44
2.2	DESCRIPCIÓN DE TECNOLOGÍAS AUXILIARES DE INFORMACIÓN BASADAS EN XML USADAS EN EL PROYECTO DE TESIS.	45
2.2.1	DOCUMENT OBJECT MODEL (DOM)	45
2.2.1.1	HISTORIA.....	46
2.2.1.2	INTRODUCCIÓN.....	47
2.2.1.3	ESPECIFICACIÓN DEL DOM	49

2.2.1.3.1	DOM NIVEL 0.....	49
2.2.1.3.2	DOM NIVEL 1.....	49
2.2.1.3.3	DOM NIVEL 2.....	51
2.2.1.3.4	DOM NIVEL 3.....	57
2.2.2	SAX.....	58
2.2.2.1	CARACTERÍSTICAS DE SAX.....	59
2.2.2.2	ESPECIFICACIÓN E INTERFACES USADAS.....	61
2.2.3	XERCES.....	62
2.2.3.1	INTERFAZ NATIVA DE XERCES.....	63
2.2.4	XPATH.....	65
2.2.4.1	ESTRUCTURA.....	66
2.2.4.1.1	CAMINOS DE LOCALIZACIÓN (LOCATION PATHS).....	66
2.2.4.1.2	PASOS DE LOCALIZACIÓN (LOCATION STEPS).....	67
2.2.4.1.3	EJES.....	68
2.2.4.1.4	LOS NODOS DE PRUEBA.....	69
2.2.4.1.5	PREDICADOS.....	70
2.2.5	XALAN.....	72
2.2.6	JAXP.....	74
2.2.6.1	LAS CAPAS DE CONEXIÓN.....	75
3.	ANÁLISIS.....	94
3.1	ANÁLISIS DE REQUERIMIENTOS.....	94

3.1.1	REQUERIMIENTOS FUNCIONALES	94
3.1.2	REQUERIMIENTOS OPERATIVOS.....	100
3.1.2.1	REQUERIMIENTOS DE EFICIENCIA	100
3.1.2.2	REQUERIMIENTOS DE PORTABILIDAD	102
3.2	ANÁLISIS TÉCNICO.....	103
3.2.1	TIPOS DE FORMATOS QUE EL SISTEMA DEBE PERMITIR DEFINIR	103
3.2.1.1	CAMPOS DE LONGITUD FIJA.....	108
3.2.1.2	CAMPOS DE LONGITUD VARIABLE	109
3.2.1.3	CAMPOS IDENTIFICADOS POR ID	110
3.2.1.4	CAMPOS MIXTOS.....	111
3.2.1.5	CAMPOS DIVIDIDOS POR SEPARADORES	113
3.2.2	TIPOS DE PROCESOS QUE EL SISTEMA DEBE PERMITIR DEFINIR Y EJECUTAR	114
3.2.3	ANÁLISIS DEL ENTORNO DE EJECUCIÓN DEL SISTEMA EN LINKOTEL S.A.	116
3.3	USUARIOS DEL SISTEMA	117
3.3.1	USUARIOS ADMINISTRADORES.....	118
3.3.2	USUARIOS OPERADORES.....	119
3.4	ESPECIFICACIÓN DE LOS CASOS DE USO DEL SISTEMA.....	120
4.	DISEÑO	129
4.1	DISEÑO ARQUITECTÓNICO DEL SISTEMA.....	129

4.1.1	ELECCIÓN DE LAS TECNOLOGÍAS Y LENGUAJES DE PROGRAMACIÓN ADECUADOS PARA SATISFACER LOS REQUERIMIENTOS	130
4.1.2	DISEÑO DE LA IMPLANTACIÓN DEL SISTEMA EN LINKOTEL S.A.	132
4.2	DISEÑO DE LOS MÓDULOS DEL SISTEMA.....	135
4.2.1	MÓDULO DE DEFINICIÓN DE FORMATOS	137
4.2.1.1	ARQUITECTURA GENERAL DEL DOCUMENTO DE DESCRIPCIÓN DE FORMATOS	139
4.2.1.2	PROPIEDADES DEL DOCUMENTO	141
4.2.1.3	DISTRIBUCIÓN DE LOS CAMPOS EN LOS REGISTROS.....	145
4.2.1.4	INFORMACIÓN SOBRE LA ESTRUCTURA INTERNA DE LOS CAMPOS	149
4.2.1.5	CREACIÓN Y EDICIÓN DEL DOCUMENTO DESCRIPTOR DE FORMATOS	154
4.2.2	MÓDULO DE DEFINICIÓN Y EJECUCIÓN DE PROCESOS	161
4.2.3	MÓDULO DE INTERPRETACIÓN Y TRANSFORMACIÓN DE DATOS.....	171
5.	IMPLEMENTACIÓN Y PRUEBAS	200
5.1	PROCESO DE IMPLEMENTACIÓN	200
5.1.1	CAPACITACIÓN EN LA TECNOLOGÍA INVOLUCRADA.....	200
5.1.2	SIMULACIÓN DEL AMBIENTE DE PRODUCCIÓN DEL PRODUCTO	202
5.1.3	DEFINICIÓN DE ESTÁNDARES DE DESARROLLO.....	203
5.2	PROBLEMAS PRESENTADOS EN LA IMPLEMENTACIÓN	204
5.3	PRUEBAS REALIZADAS	207
5.3.1	PRUEBAS DE FUNCIONALIDAD.....	207

5.3.2	PRUEBAS DE EFICIENCIA	209
5.3.3	PRUEBAS DE PORTABILIDAD	210
6.	CONCLUSIONES Y RECOMENDACIONES	212
6.1	CONCLUSIONES	212
6.2	RECOMENDACIONES.....	215
APÉNDICE A:	MANUAL DE USUARIO	218
APÉNDICE B:	ESPECIFICACIÓN DE CASOS DE USO Y ESCENARIOS DEL SISTEMA	253
APÉNDICE C:	DIAGRAMAS DE CLASES DEL SISTEMA.....	290
APÉNDICE D:	DIAGRAMAS DE SECUENCIA DEL SISTEMA	301
APÉNDICE E:	DOCUMENTOS XSD DEL SISTEMA	332
APÉNDICE F:	DESCRIPTORES DE FORMATOS PARA CENTRALES HUAWEI, ERICSSON Y ALCATEL	338
APÉNDICE G:	DEFINICIÓN DE UN XML SCHEMA DEFINITION.....	406
BIBLIOGRAFÍA	422

ÍNDICE DE FIGURAS

FIGURA 1.1: PROCESO DE MEDIACIÓN EN EMPRESAS OPERADORAS DE UNA RED DE SERVICIOS	4
FIGURA 1.2: SUBSISTEMAS DE UN SISTEMA DE MEDIACIÓN	9
FIGURA 1.3: PROCESO DE AGREGACIÓN	11
FIGURA 1.4: PROCESO DE CORRELACIÓN	12
FIGURA 1.5: FLUJO DE DATOS DE USO EN UNA EMPRESA DE TELEFONÍA FIJA.....	15
FIGURA 1.6: ACTUAL SISTEMA DE CARGA DE PACIFICTEL S.A.....	19
FIGURA 1.7: MÓDULOS DEL SISTEMA “INTER-MEDIATE”	22
FIGURA 1.8: MÓDULOS DEL SISTEMA “N*VISION”.....	25
FIGURA 2.1 EJEMPLO DE UN DOM	48
FIGURA 2.2 EJEMPLO DE UN RANGO	56
FIGURA 2.3 FLUJO DE INFORMACIÓN EN UNA CONFIGURACIÓN DE VALIDADOR	64
FIGURA 3.1: GENERACIÓN DE CDRS POR PARTE DE LAS CENTRALES TELEFÓNICAS DE TRÁFICO	97
FIGURA 3.2: GENERACIÓN DEL PROGRAMA CAPAZ DE REALIZAR LA INTERPRETACIÓN DE UN FORMATO DE CDRS	98

FIGURA 3.3: DEFINICIÓN DE UN PROCESO DE INTERPRETACIÓN	99
FIGURA 3.4: EJECUCIÓN DE UN PROCESO DE INTERPRETACIÓN	99
FIGURA 3.5 FRAGMENTO DEL DOCUMENTO DESCRIPTOR DE REGISTROS DE UNA CENTRAL HUAWEI	105
FIGURA 3.6: DESCRIPCIÓN DE LOS FORMATOS USADOS POR CENTRALES DE TRÁFICO.....	107
FIGURA 3.7: CAMPOS DE LONGITUD FIJA EN UN REGISTRO DE UNA CENTRAL ALCATEL.....	109
FIGURA 3.8: CAMPOS IDENTIFICADOS POR IDS EN UN REGISTRO DE UNA CENTRAL SIEMENS	111
FIGURA 3.9: CAMPOS MIXTOS EN UN REGISTRO DE UNA CENTRAL SIEMENS.....	112
FIGURA 3.10: AMBIENTE DE EJECUCIÓN DEL SISTEMA EN LINKOTEL S.A.....	116
FIGURA 3.11: IDENTIFICACIÓN DE LOS ACTORES PRIMARIOS DEL SISTEMA	122
FIGURA 3.12: IDENTIFICACIÓN DE LOS ACTORES SECUNDARIOS DEL SISTEMA ...	122
FIGURA 3.13: IDENTIFICACIÓN DE LAS METAS DE LOS ACTORES PRIMARIOS.....	123
FIGURA 4.1: IMPLANTACIÓN DEL SISTEMA EN LINKOTEL S.A.	133
FIGURA 4.2: ESQUEMA GENERAL DEL DOCUMENTO DE DEFINICIÓN DE FORMATOS	140
FIGURA 4.3: PROPIEDADES DEL DOCUMENTO.....	142
FIGURA 4.4: ESTRUCTURA DE LOS REGISTROS GENERADOS POR LA CENTRAL...	145
FIGURA 4.5: MANEJO DE ESTRUCTURAS CONDICIONALES	147
FIGURA 4.6: ESTRUCTURA INTERNA DE LOS CAMPOS.....	150
FIGURA 4.7: DEFINICIÓN DE EXPRESIONES.....	152
FIGURA 4.8: DIAGRAMA DE CLASES DEL MÓDULO DE DEFINICIÓN DE FORMATOS	155
FIGURA 4.9: USUARIO MUEVE GRUPO DESPUÉS DEL SIGUIENTE	159
FIGURA 4.10: USUARIO CREA UN CAMPO DE TIPO LEÍDO	160

FIGURA 4.11: EJECUCIÓN DE ACCIONES SOBRE PROCESOS.....	164
FIGURA 4.12: DIAGRAMA DE CLASES DEL MÓDULO DE DEFINICIÓN Y EJECUCIÓN DE PROCESOS – SECCIÓN DEFINICIÓN.....	165
FIGURA 4.13: DIAGRAMA DE CLASES DEL MÓDULO DE DEFINICIÓN Y EJECUCIÓN DE PROCESOS – SECCIÓN EJECUCIÓN.....	167
FIGURA 4.14: USUARIO DEFINE UN NUEVO PROCESO.....	169
FIGURA 4.15: USUARIO INICIA UN PROCESO	170
FIGURA 4.16: DIAGRAMA DE CLASES DEL MÓDULO DE INTERPRETACIÓN DE DATOS (PARTE 1)	173
FIGURA 4.17: DIAGRAMA DE CLASES DEL MÓDULO DE INTERPRETACIÓN DE DATOS (PARTE 2)	176
FIGURA 4.18: DIAGRAMA DE CLASES DEL MÓDULO DE INTERPRETACIÓN DE DATOS (PARTE 3)	179
FIGURA 4.19: USUARIO INICIA UN PROCESO (A)	180
FIGURA 4.20: USUARIO INICIA UN PROCESO (B)	181
FIGURA 4.21 USUARIO INICIA UN PROCESO (C)	183
FIGURA 4.22: REPRESENTACIÓN DE UN DESCRIPTOR DE FORMATOS A MANERA DE UN ÁRBOL	185
FIGURA 4.23: DIFERENCIAS ENTRE CAMPOS LEÍDOS Y CALCULADOS	186
FIGURA 4.24: DIAGRAMA DE CLASES DEL MÓDULO DE INTERPRETACIÓN DE DATOS (PARTE 4)	188
FIGURA 4.25: DIAGRAMA DE CLASES DEL MÓDULO DE INTERPRETACIÓN DE DATOS (PARTE 5)	192
FIGURA 4.26: ESTRUCTURA PARA LA EVALUACIÓN DE EXPRESIONES	193
FIGURA 4.27: DIAGRAMA DE CLASES DEL MÓDULO DE INTERPRETACIÓN DE DATOS (PARTE 6)	195
FIGURA 4.28: REPRESENTACIÓN DE UN REGISTRO YA LEÍDO	198

FIGURA C.1: DIAGRAMA DE CLASES DEL SISTEMA (PARTE 1)	291
FIGURA C.2: DIAGRAMA DE CLASES DEL SISTEMA (PARTE 2)	292
FIGURA C.3: DIAGRAMA DE CLASES DEL SISTEMA (PARTE 3)	293
FIGURA C.4: DIAGRAMA DE CLASES DEL SISTEMA (PARTE 4)	294
FIGURA C.5: DIAGRAMA DE CLASES DEL SISTEMA (PARTE 5)	295
FIGURA C.6: DIAGRAMA DE CLASES DEL SISTEMA (PARTE 6)	296
FIGURA C.7: DIAGRAMA DE CLASES DEL SISTEMA (PARTE 7)	297
FIGURA C.8: DIAGRAMA DE CLASES DEL SISTEMA (PARTE 8)	298
FIGURA C.9: DIAGRAMA DE CLASES DEL SISTEMA (PARTE 9)	299
FIGURA C.10: DIAGRAMA DE CLASES DEL SISTEMA (PARTE 10)	300
FIGURA D.1: USUARIO CREA FORMATO BINARIO DE ESTRUCTURA FIJA	302
FIGURA D.2: USUARIO CREA FORMATO BINARIO DE CAMPOS IDENTIFICADOS POR IDS.....	302
FIGURA D.3: USUARIO CREA FORMATO DE TEXTO BASADO EN SEPARADORES....	303
FIGURA D.4: USUARIO CREA UN CAMPO DE TIPO LEÍDO.....	303
FIGURA D.5: USUARIO GENERA UN CAMPO CON FORMATO FECHA Y MÁSCARA CORRECTA	304
FIGURA D.6: USUARIO GENERA UN CAMPO CON FORMATO FECHA Y MÁSCARA INCORRECTA.....	305
FIGURA D.7: USUARIO GENERA UN CAMPO DE DEFINICIÓN SIMPLE CORRECTAMENTE.....	306
FIGURA D.8: USUARIO GENERA UN CAMPO DE DEFINICIÓN SIMPLE INCORRECTAMENTE	307
FIGURA D.9: USUARIO GENERA UN CAMPO DE DEFINICIÓN COMPUESTA INCOMPLETO.....	308
FIGURA D.10: USUARIO GENERA UN CAMPO DE DEFINICIÓN COMPUESTA CORRECTAMENTE.....	309

FIGURA D.11: USUARIO GENERA UN CAMPO DE DEFINICIÓN COMPUESTA INCORRECTAMENTE	310
FIGURA D.12: USUARIO REFERENCIA CAMPOS INEXISTENTES EN EXPRESIÓN COMPUESTA.....	311
FIGURA D.13: USUARIO CREA UN GRUPO CON CONDICIÓN POR OMISIÓN.....	312
FIGURA D.14: USUARIO CREA UN GRUPO CON CONDICIÓN BASADA EN CAMPOS.	312
FIGURA D.15: USUARIO CREA UN GRUPO CON CONDICIÓN INVÁLIDA.....	313
FIGURA D.16: USUARIO MUEVE CAMPO A LA DERECHA.....	313
FIGURA D.17: USUARIO MUEVE CAMPO A LA IZQUIERDA.....	314
FIGURA D.18: USUARIO NO PUEDE MOVER CAMPO SELECCIONADO PORQUE ES EL ÚLTIMO	314
FIGURA D.19: USUARIO MUEVE CAMPO A LA IZQUIERDA, PERO INVALIDA EL CAMPO ACTUAL	315
FIGURA D.20: USUARIO MUEVE CAMPO A LA DERECHA, PERO INVALIDA EL CAMPO SIGUIENTE	315
FIGURA D.21: USUARIO MUEVE GRUPO DESPUÉS DEL SIGUIENTE	316
FIGURA D.22: USUARIO NO PUEDE MOVER GRUPO PORQUE NO EXISTEN OTROS	316
FIGURA D.23: USUARIO MODIFICA LAS OPCIONES GENERALES DEL DOCUMENTO	317
FIGURA D.24: USUARIO MODIFICA CON ERRORES LAS OPCIONES GENERALES DEL DOCUMENTO.....	318
FIGURA D.25: USUARIO GUARDA UN DOCUMENTO EXITOSAMENTE.....	319
FIGURA D.26: USUARIO INTENTA GUARDAR UN DOCUMENTO INVÁLIDO	319
FIGURA D.27: USUARIO ABRE UN DOCUMENTO.....	320
FIGURA D.28: USUARIO INTENTA ABRIR UN DOCUMENTO INVÁLIDO	320
FIGURA D.29: USUARIO DEFINE UN NUEVO PROCESO	321
FIGURA D.30: USUARIO DEFINE UN NUEVO PROCESO CON DATOS INCOMPLETOS	321

FIGURA D.31: USUARIO MODIFICA UN PROCESO	322
FIGURA D.32: USUARIO MODIFICA UN PROCESO CON ERRORES.....	322
FIGURA D.33 A: USUARIO INICIA UN PROCESO	323
FIGURA D.33 B: USUARIO INICIA UN PROCESO	323
FIGURA D.33 C: USUARIO INICIA UN PROCESO.....	324
FIGURA D.34: USUARIO INICIA UN PROCESO CON DATOS INCOMPLETOS.....	325
FIGURA D.35 A: USUARIO INICIA UN PROCESO QUE FALLA POR PARÁMETROS INCORRECTOS	325
FIGURA D.35 B: USUARIO INICIA UN PROCESO QUE FALLA POR PARÁMETROS INCORRECTOS	326
FIGURA D.36 A: USUARIO INICIA UN PROCESO QUE FALLA POR PROBLEMAS DE LECTURA.....	326
FIGURA D.36 B: USUARIO INICIA UN PROCESO QUE FALLA POR PROBLEMAS DE LECTURA.....	327
FIGURA D.37 A: USUARIO DETIENE UN PROCESO	327
FIGURA D.37 B: USUARIO DETIENE UN PROCESO	328
FIGURA D.38: USUARIO INTENTA DETENER PROCESOS QUE NO SE ESTÁN EJECUTANDO	329
FIGURA D.39 A: USUARIO REINICIA PROCESOS DETENIDOS.....	329
FIGURA D.39 B: USUARIO REINICIA PROCESOS DETENIDOS.....	330
FIGURA D.40 A: USUARIO REINICIA PROCESOS EN EJECUCIÓN.....	330
FIGURA D.40 B: USUARIO REINICIA PROCESOS EN EJECUCIÓN.....	331
FIGURA G.1: TIPO SIMPLE DE DATO	417
FIGURA G.2: ELEMENTOS DE ELECCIÓN	418
FIGURA G.3: ELEMENTOS REQUERIDOS	418
FIGURA G.4: ELEMENTOS SECUENCIALES.....	419
FIGURA G.5: CARDINALIDAD DE ELEMENTOS.....	419

FIGURA G.6: DEFINICIÓN DE TIPOS COMPUESTOS	420
FIGURA G.7: DEFINICIÓN DE UNIONES	421
IMAGEN G.8: DEFINICIÓN DE LISTAS.....	421

ÍNDICE DE TABLAS

TABLA 1.1: CENTRALES TELEFÓNICAS DE TRÁFICO CUYOS FORMATOS DE ARCHIVOS SERÁN DEFINIDOS POR EL SISTEMA	30
TABLA 2.1 ESTRUCTURAS Y COMPONENTES COMUNES DE UN DOCUMENTO XML	42
TABLA 2.7: INTERFACES FUNDAMENTALES DEL NÚCLEO DEL DOM, NIVEL 1	50
TABLA 2.8: INTERFACES AUXILIARES DEL NÚCLEO DEL DOM, NIVEL 1	51
TABLA 2.9: TERMINOLOGÍA USADA EN LOS EVENTOS	54
TABLA 2.10: INTERFACES DEL MÓDULO DE NAVEGACIÓN DEL DOM NIVEL 2	55
TABLA 2.11: INTERFACES DE SAX	61
TABLA 2.12: EJES DE XPATH	69
TABLA 2.13: NODOS DE PRUEBA PREDETERMINADOS DE XPATH	70
TABLA 2.14: FUNCIONES DISPONIBLES PARA LOS PREDICADOS	72
TABLA 3.1: CANTIDAD DE CDRS PROCESADOS POR PACIFICTEL S.A. EN LA SEMANA COMPRENDIDA DEL DOMINGO 12 DE SEPTIEMBRE AL SÁBADO 18 DE SEPTIEMBRE DEL 2004	101
TABLA 4.1: CUADRO COMPARATIVO ENTRE LOS LENGUAJES JAVA Y C++	131
TABLA 4.2: PARÁMETROS DE DEFINICIÓN DE PROCESOS	162

TABLA 4.3: ACCIONES A REALIZAR SOBRE LOS PROCESOS DEFINIDOS POR EL SISTEMA.....	163
TABLA G.1: PROPIEDADES COMUNES PARA ELEMENTOS Y ATRIBUTOS DE UN ESQUEMA XML	407
TABLA G.2: PROPIEDADES ESPECÍFICAS DE LOS ATRIBUTOS EN UN ESQUEMA XML	407
TABLA G.3: PROPIEDADES EXCLUSIVAS DE LOS ELEMENTOS EN UN ESQUEMA XML	408
TABLA G.4: PROPIEDADES DE LOS DATOS SIMPLES EN UN ESQUEMA XML.....	411
TABLA G.5: TIPOS DE NOTACIÓN EN LOS ESQUEMAS XML.....	417

CAPÍTULO 1

1. ANTECEDENTES Y JUSTIFICACIÓN

1.1 CONCEPTOS GENERALES DE MEDIACIÓN

1.1.1 DEFINICIÓN DE UN SISTEMA DE MEDIACIÓN

Un sistema de mediación permite a empresas que administran una red de servicios recolectar datos de consumos sobre los distintos elementos de dicha red, procesarlos de acuerdo a las reglas de negocio definidas y distribuirlos en tiempo real como información útil a las diferentes aplicaciones que los necesiten.

Los elementos integrantes de la red de servicios pueden poseer tecnologías completamente diferentes entre sí, por lo que un sistema de mediación debe proveer interfaces para cada uno de estos componentes, abstrayendo a los sistemas consumidores de los datos de uso, de la complejidad de obtener dichos datos y transformarlos a un formato común y estandarizado de acuerdo a las necesidades de la empresa. De esta manera un sistema de mediación actúa como un portal de trabajo entre los elementos de la red de servicios y los sistemas de soporte de negocios que generan ganancias a la empresa, facilitando la comunicación entre estos componentes y permitiendo conocer y entender a los clientes al poder llevarse a cabo análisis más granulares de los consumos que estos realizan. Como consecuencia, los sistemas de soporte de negocios de la empresa que se basan en el procesamiento de los datos de consumo, se ejecutan de manera más competitiva y eficiente ^[1].

El concepto de mediación es aplicable a todas aquellas industrias que basen sus operaciones en la administración de una red de centrales que provean algún tipo de servicio a sus usuarios finales, es decir: industrias que deben capturar, analizar, codificar y finalmente facturar eventos de uso en sus redes servicios, las cuales

son crecientes cada vez más tanto en tamaño como en complejidad^[I].

Entre las industrias que potencialmente pueden aprovechar el implantar un sistema de mediación están: las de telecomunicaciones, electricidad, distribución de agua potable, gas, servicios de valor agregado y similares^{[XVI][XVII][XVIII]}. La siguiente figura ilustra el contexto de un sistema de mediación en una empresa administradora de una red de servicios:

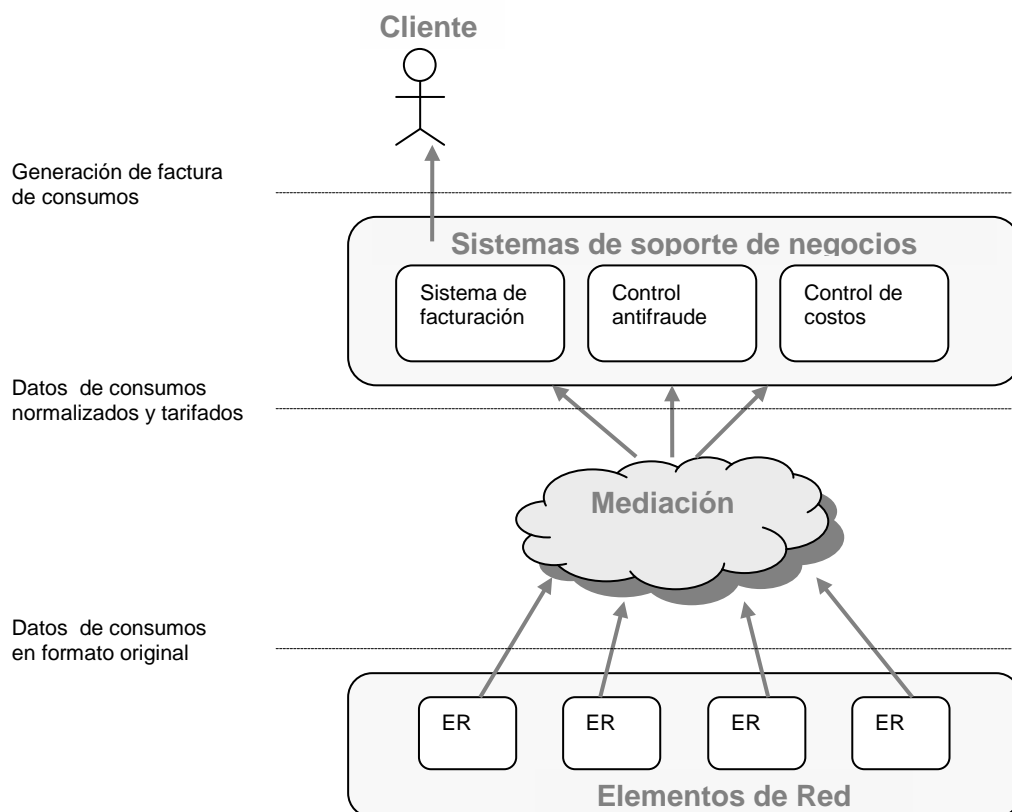


Figura 1.1: Proceso de mediación en empresas operadoras de una red de servicios^[1]

El gráfico muestra un cierto número de elementos de red, los cuales son responsables de recolectar datos sobre los consumos realizados por los clientes de la empresa. Cada uno de estos elementos de red, pone a disposición sus datos recolectados a través de algún protocolo específico y en un formato determinado. El sistema de mediación recolecta los datos de consumo en cada uno de estos puntos y los transforma a un formato estandarizado de acuerdo a las necesidades de la empresa. Una vez que los datos han sido normalizados son sometidos a un proceso de tarifación y finalmente

entregados a los distintos sistemas de soporte de negocios que posea la empresa, creando una capa de abstracción entre estos y la red de servicios ^[II]. Entre los sistemas dependientes de la mediación, el de facturación es uno de los más importantes, ya que es el que permite a la empresa obtener ganancias por las actividades realizadas. Este sistema crea en base a los datos previamente tarifados las facturas de consumos, las cuales son finalmente entregadas al cliente^[X].

1.1.2 OBJETIVOS DE UN SISTEMA DE MEDIACIÓN

Los objetivos de un sistema de mediación son los de abstraer a los sistemas que necesiten información sobre los consumos realizados en una red de servicios, de la complejidad de las diversas tecnologías usadas por los elementos presentes en dicha red. Para que un sistema de mediación cumpla con sus objetivos es necesario que este satisfaga ciertos requerimientos tanto funcionales como operativos. Dichos requerimientos serán listados a lo largo de esta sección.

1.1.2.1 REQUERIMIENTOS FUNCIONALES DE UN SISTEMA DE MEDIACIÓN

A continuación se listan los requerimientos funcionales con los que debe cumplir un sistema de mediación:

- Debe poder realizar la carga de consumos de múltiples servicios convergentes, así como soportar la definición de nuevos servicios creados por la empresa.
- Permitir modelar diversos esquemas de tarificación para todos los servicios prestados por la empresa, de acuerdo a las reglas de negocio definidas. La tarificación consiste en asociar un valor monetario a cierto tipo de consumo registrado por los elementos de la red de servicios.
- Incrementar la calidad de los datos que alimentan la facturación y otras áreas de la empresa, a través de mecanismos de filtración, validación de datos y corrección de errores.
- Permitir la definición de múltiples flujos de entrada y salida de datos, así como la definición flexible de los formatos a los que se ciñen estos datos.
- Convertir a un formato único todos los consumos ingresados a través de la recolección de datos de cada una de las centrales

(puntos de red) involucradas. De manera que se abstraiga a los sistemas que requieran los datos de la complejidad de trabajar con múltiples formatos.

- Permitir definir las reglas del negocio y la configuración de los parámetros del sistema en un lenguaje amigable para el usuario.

1.1.2.2 REQUERIMIENTOS OPERATIVOS DE UN SISTEMA DE MEDIACIÓN

A continuación se listan los requerimientos operativos con los que debe cumplir un sistema de mediación.

- Proporcionar un rendimiento adecuado en términos de velocidad de procesamiento, de acuerdo a las necesidades de la empresa.
- Debe ser escalable, de manera tal que pueda crecer de acuerdo a las necesidades de expansión del proveedor de servicios.
- Portable entre plataformas, de manera tal que pueda adaptarse flexiblemente a múltiples arquitecturas de hardware de acuerdo a la actividad y tipo de negocio de la empresa.
- Garantizar que la información sea cargada sin inconsistencias, de manera completa y una sola vez.

- Realizar en tiempo real la entrega de los datos procesados a los diferentes sistemas de soporte de negocios, de manera que puedan ser utilizados a medida que vayan siendo generados por los elementos de la red de servicios.

1.1.3 SUBSISTEMAS EN UN SISTEMA DE MEDIACIÓN

La siguiente figura ilustra los subsistemas que conforman un sistema de mediación, su interacción entre ellos y con los diversos sistemas de soporte de negocios de la empresa:

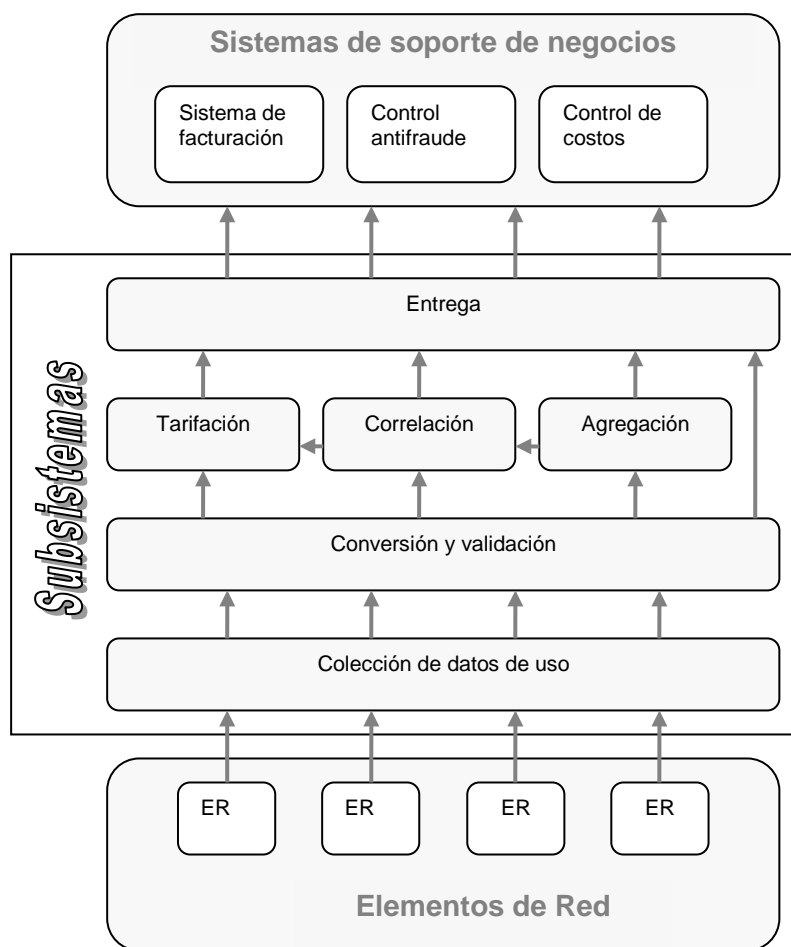


Figura 1.2: Subsistemas de un sistema de mediación

A continuación se da una descripción para cada uno de los subsistemas mostrados:

- **Colección de datos de uso.-** En esta etapa la información del uso de servicios es cargada desde las diferentes centrales que componen la red de servicios del operador, las cuales poseen distintos protocolos, formatos y métodos de acceso. Este

subsistema debe proveer interfaces para cada uno de los elementos de red, de manera que abstraiga del proceso de recolección de datos a los demás subsistemas involucrados en la mediación.

- **Conversión y validación.-** En este subsistema, la información del consumo de servicios en formato original es transformada a la representación completa de una transacción, de acuerdo al formato común definido como estándar para el proceso de mediación. Si se detectan errores o inconsistencias en los datos leídos se los corrige si es posible hacerlo, de lo contrario se registra el hecho para que sea considerado por el operador del sistema ^{[VIII][IX]}.
- **Agregación.-** Ocasionalmente, un elemento de la red de servicios registra en momentos distintos datos intermedios o parciales referentes a un mismo registro de consumo. Este subsistema es responsable de detectar la ocurrencia de dicho evento y de realizar un proceso de agregación sobre estos registros parciales y unificarlos en un simple registro normalizado. La siguiente figura muestra un ejemplo de un proceso de agregación realizado sobre tres registros intermedios generados por una misma central, en momentos diferentes y referentes al mismo registro de consumos:

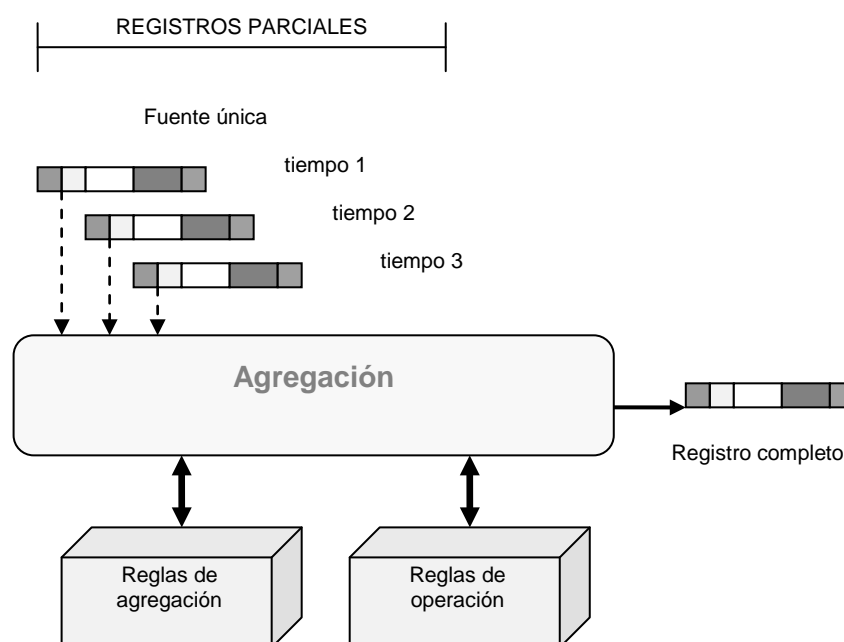


Figura 1.3: Proceso de agregación

El subsistema de agregación, al determinar que los tres registros generados por la central se refieren a la misma información de consumo, genera un solo registro normalizado en vez de tres con información incompleta ^[VIII].

Este módulo basa su comportamiento en dos tipos de reglas: las *reglas de agregación* definen cuándo los registros deben ser agregados con otros y transformados en uno solo, las *reglas de operación* determinan cómo se llevará a cabo el proceso de agregación sobre estos registros ^{[VIII][IX]}.

- **Correlación.-** Si distintos elementos de la red de servicios generan información redundante sobre una misma sesión, actividad, entidad o servicio, este subsistema es responsable de detectar dicho evento y de combinar los datos en un solo registro para evitar la duplicación de información. La siguiente figura muestra un ejemplo de un proceso de correlación sobre tres registros emitidos por distintas centrales, que se refieren al mismo consumo ^[VIII]:

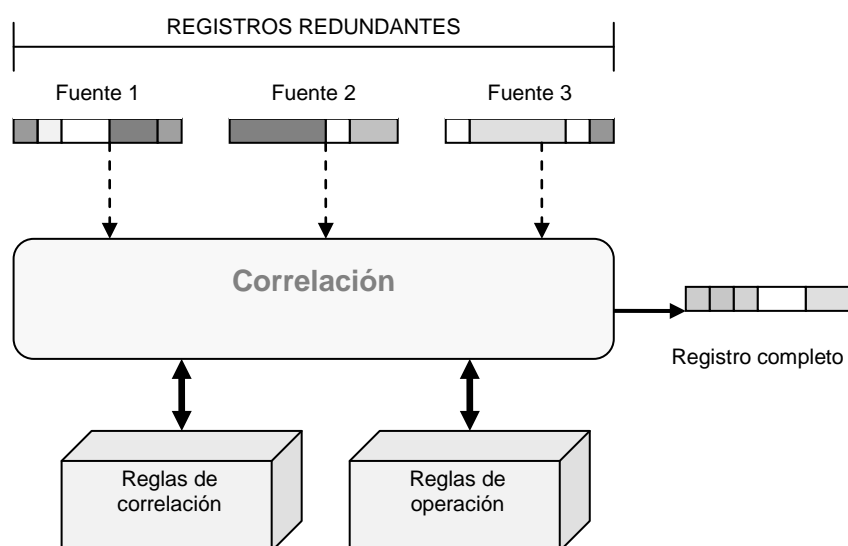


Figura 1.4: Proceso de correlación

En el gráfico se muestran tres registros generados por distintos elementos de la red de servicios, sin embargo, el subsistema de correlación determina que se refieren redundantemente a la misma información de consumo, por lo que los unifica en un solo registro con los datos apropiados.

Este módulo basa su comportamiento en dos tipos de reglas: las *reglas de correlación* definen cuándo los registros provenientes de distintas centrales deben ser correlacionados entre sí y transformados en uno solo, las *reglas de operación* determinan cómo se llevará a cabo el proceso de correlación sobre estos registros ^{[VIII][IX]}.

La importancia de los subsistemas de correlación y agregación radica en que evitan la facturación duplicada de un mismo consumo, a la vez que permiten reducir significativamente la cantidad de datos almacenados por largos periodos de tiempo, manteniendo solamente la información relevante a los sistemas de soporte de negocios de la empresa y eliminando los datos redundantes.

- **Tarifación.**- Los consumos registrados son tarifados de acuerdo a las reglas de negocio definidas, de esta manera, cada registro de consumo es asociado con un valor monetario.
- **Entrega.**- El registro de la transacción es entregado a las aplicaciones que lo necesitan en un formato comprensible para

ellas, abstrayéndolas del proceso de lectura e interpretación a bajo nivel de la misma.

1.2 MEDIACIÓN EN EMPRESAS DE TELEFONÍA FIJA

1.2.1 SISTEMAS DE SOPORTE DE NEGOCIOS EN EMPRESAS DE TELEFONÍA FIJA

En esta sección se describirán a los sistemas consumidores de los datos proporcionados por un sistema de mediación en empresas operadoras de telefonía fija. La siguiente figura ilustra el flujo de información en este tipo de empresas:

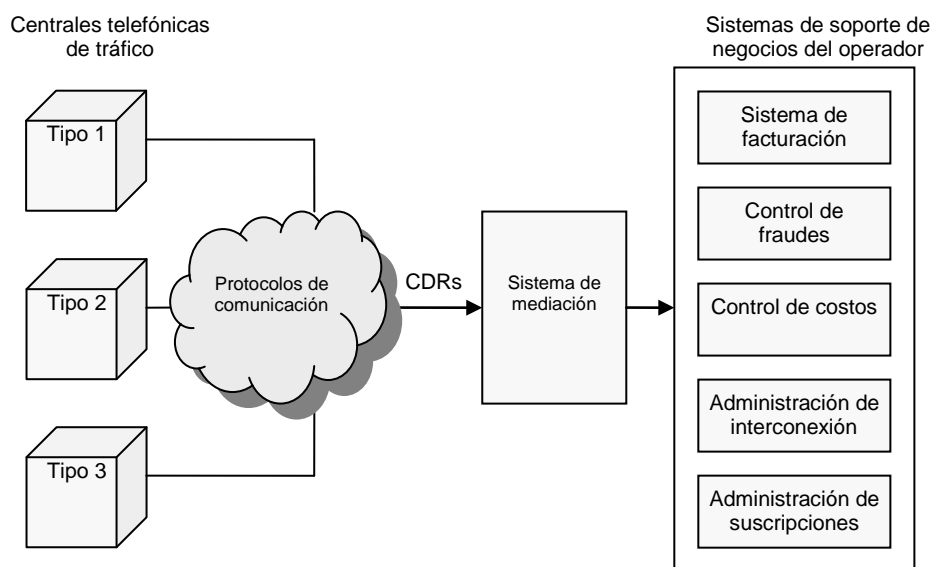


Figura 1.5: Flujo de datos de uso en una empresa de telefonía fija ^[1]

Como muestra el gráfico, los elementos de red están constituidos en este tipo de industria por centrales telefónicas, las cuales registran toda la información del tráfico de llamadas que pasa por ellas y generan con esta “detalles de registros de llamadas” o CDRs (por sus siglas en inglés). Son estos CDRs los que constituyen la información de uso de consumos que debe ser llevada a un formato común, analizada y distribuida a los distintos sistemas de soporte de negocios de la operadora telefónica ^[X]. Las principales aplicaciones que son alimentadas por un sistema de mediación telefónica son listadas a continuación:

- **Facturación.-** Este sistema genera las facturas de consumos a todos los clientes de la empresa, este proceso es llevado a cabo de manera ágil y eficiente debido a que los procesos de agregación, correlación y tarifación fueron ya resueltos por el sistema de mediación telefónica.
- **Control de fraudes.-** El sistema de control de fraudes de la empresa se basa en el sistema de mediación para realizar un análisis combinado tanto sobre los datos históricos de consumos como sobre los datos proporcionados en tiempo real, para poder determinar rápidamente si existen ciertos patrones que puedan indicar un fraude a la empresa (por ejemplo un consumo de servicios excepcionalmente alto). Los sistemas tradicionales de detección de fraudes, basan sus análisis solamente en información histórica, lo que origina que haya mucha latencia entre la ocurrencia del delito y su detección, lo que favorece a los perpetradores del fraude ^{[VI][VII]}.
- **Control de costos.-** El sistema de control de costos permite determinar como la red de servicios puede optimizarse, ayudando a decidir como esta debe desarrollarse, qué áreas de la misma necesitan fortalecerse, o qué redes de terceras partes tienen

debilidades. De esta manera puede asegurarse la existencia de un balanceo de carga entre los distintos elementos de la red ^{[VI][VI]}.

- **Interconexión con otras operadoras.-** El sistema de interconexión usa los datos generados por el sistema de mediación para encontrar información de interconexión con otros operadores de telefonía ^[III]. Este proceso de interconexión entre operadores es central para el crecimiento del mercado. En un ambiente con un simple operador, cuando un cliente realiza una llamada telefónica, el operador administra completamente el proceso de hacer llegar la llamada al destinatario y de cobrar por esta, sin embargo los beneficios para los usuarios fueran muy pocos si ellos estuvieran limitados a hablar con personas conectadas al mismo operador, es por esto que los operadores deben interconectar sus redes y pasarse tráfico de interconexión entre ellos ^[IV].
- **Administración de suscripciones.-** El sistema de administración de suscripciones permite la asignación o cancelación de servicios a los abonados de la telefónica.

1.2.2 MEDIACIÓN DE TELEFONÍA FIJA EN EL ECUADOR – REGIÓN COSTA

Esta sección describe los hechos en cuanto a la carga de datos generados por centrales de tráfico e interpretación de los mismos en las operadoras telefónicas de la región costa del Ecuador. En la actualidad, en esta región hay dos empresas de telefonía fija: Pacifictel y la nueva operadora Linkotel.

Pacifictel no posee un sistema de mediación telefónica que cumpla con los objetivos expuestos en este capítulo, realizando a bajo nivel todos los procesos que involucren interacción con sus centrales de tráfico, lo que representa realizar una programación dependiente del tipo, modelo y marca de la central, para la implementación de los procesos de carga, interpretación y distribución de los datos emitidos por estas. Por lo tanto, la actualización o corrección de los formatos que siguen los datos de tráfico generados por las centrales, son costosos en términos de tiempo y dinero, así como la posterior distribución de esta información a las diferentes aplicaciones que los requieran. La siguiente figura ilustra como Pacifictel lleva a cabo la interpretación y transformación de los formatos generados por sus centrales de tráfico:


```

$ su - oper
Password:
Sun Microsystems Inc. SunOS 5.9 Generic May 2002

YOVERI S.A.
Programa para cargar cintas de consumos telefonicos.

Nombre de Usuario: dnora
Contraseña: Verificando Usuario.....OK!!

Oficio: INF-7808

Descripcion: ALCATEL-TRAFICO

                1.- GUAYAQUIL
                2.- MACHALA
                3.- MANTA
                4.- GALAPAGOS
                5.- LOJA
                6.- CUENCA

Origen: MANTA

ALT -- Alcatel Trafico
ALC -- Alcatel Contadores
ERC -- Ericsson Contadores
EQC -- Equitel Contadores
ERT -- Ericsson Trafico
ENT -- Ericsson Trafico (Guayaquil)
SIT -- Siemens Trafico
RIN -- Red Inteligente Trafico
EDX -- Telex Trafico
HNT -- Honet Trafico

Tipo de Carga:

```

Figura 1.6: Actual sistema de carga de Pacifictel S.A. ⁽¹⁾

La figura muestra una aplicación de consola en la cual aparece un menú solicitando información sobre el tipo de central cuyos datos van a ser cargados. En base a la selección del usuario, la aplicación elige un programa realizado en lenguaje C para proceder a realizar la transformación de datos. Es necesario que un programador experimentado desarrolle y de el mantenimiento adecuado a los módulos en lenguaje C encargados de llevar a cabo esta tarea.

¹ Cortesía de Pacifictel S.A.

Linkotel, empresa de telefonía fija nueva en el mercado ecuatoriano, ha comenzado a realizar sus procesos de interpretación de datos con el sistema desarrollado como parte de este proyecto de tesis, el mismo que deberá crecer hasta convertirse a mediano plazo en un sistema completo de mediación telefónica. Actualmente, debido a su temprana introducción en el mercado, esta empresa posee solamente una central de tráfico tipo Huawei con capacidad para seis mil usuarios, a medida que se incremente su número de abonados se irán adquiriendo paulatinamente nuevas centrales para satisfacer la demanda.

1.2.3 SISTEMAS DE MEDIACIÓN TELEFÓNICA EN OTROS PAÍSES

El objetivo de esta sección es el de proporcionar un acercamiento sobre los productos relacionados con mediación telefónica utilizados por empresas de telecomunicaciones representativas alrededor del mundo. No se pretende realizar una exposición en detalle sobre la arquitectura de dichos sistemas, sino más bien el ofrecer un resumen sobre la información disponible públicamente al respecto.

1.2.3.1 INTER-MEDIATE

Uno de los más grandes proveedores de sistemas de soporte de operaciones para empresas de telecomunicaciones en el mundo es INTEC Telecom Systems ^[XXI]. Esta empresa tiene más de 550 clientes en más de 70 países, entre las principales compañías que usan los sistemas de INTEC se cuentan: “A.T.& T.” (EEUU), “Bellsouth” (USA, Brazil, Chile, Colombia, Peru), “France Telecom” (Francia) y Swisscom (Suiza) entre otros ^[XL].

Entre los principales productos desarrollados por esta compañía, se encuentra el sistema de mediación convergente: “Inter-mediatE”. Este sistema, basado técnicamente en tecnologías Java y CORBA ^{[XLII][XLIII]}, tiene como objetivos funcionales el coleccionar, procesar y distribuir toda clase de datos en redes de cualquier tipo o tamaño.

La siguiente figura muestra de manera gráfica los módulos del sistema “Inter-mediatE”:

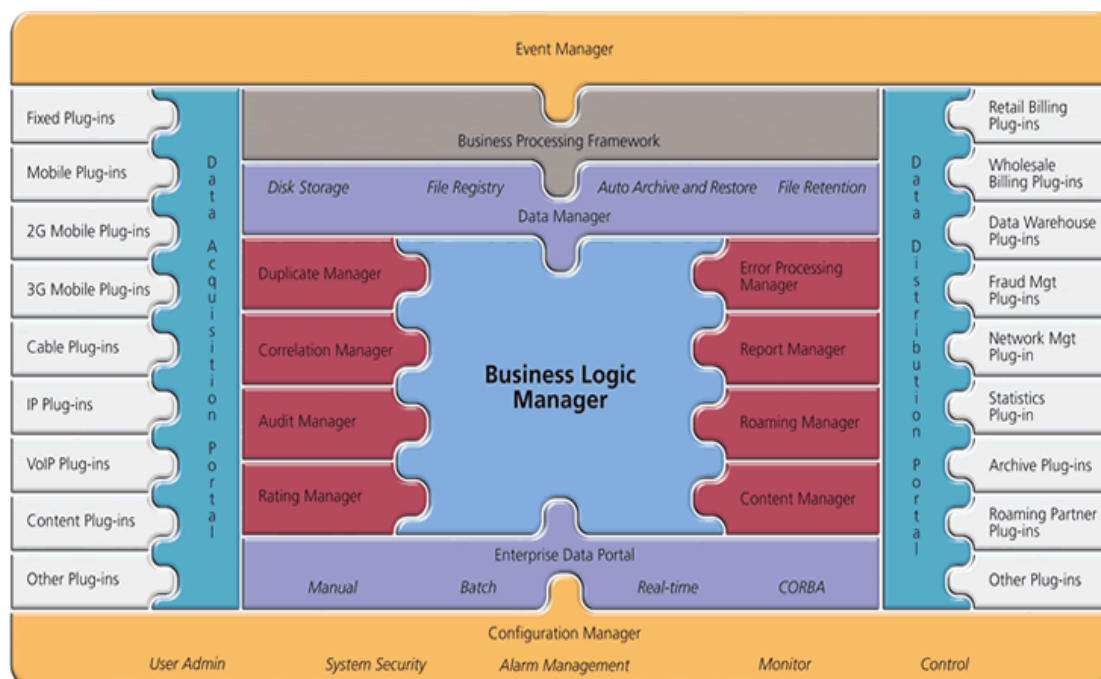


Figura 1.7: Módulos del sistema “Inter-mediatE” [XLII]

El gráfico muestra cómo la obtención de datos es realizada a través de un “Portal de adquisición de datos” y la distribución de los mismos es manejada por el “Portal de distribución de datos”. Ambos elementos son manejados como colecciones de “plug-ins”, cada uno de los cuales implementa una interfaz hacia diferentes tipos de redes o sistemas con los que es necesario intercambiar información.

Las funciones complejas de mediación son distribuidas a través de una serie de módulos. Las reglas básicas de procesamiento son manejadas por un “Administrador de lógica de negocios”, mientras que más complejos requerimientos son manejados por subsistemas

dedicados, tales como los módulos administradores de duplicados o de correlación.

Otro elemento importante en el Inter-mediatE es el “NGNode”. Este componente se ejecuta en un pequeño servidor dedicado ubicado cerca de los elementos de red que son monitoreados y sus principales funciones son las siguientes: coleccionar eventos de red, realizar tareas de premediación sobre estos eventos y enviar los datos ya procesados al servidor central en donde se ejecuta Inter-mediatE ^[XLII].

1.2.3.2 N*VISION

N*VISION es un sistema de mediación convergente desarrollado por ACE*COMM ^[XXII]. Esta compañía cuenta entre sus principales clientes al gobierno de los EEUU, British Telecom (Inglaterra), Telmex (México) y Korea Telecom (Corea del sur) entre otros y ha realizado más de 3500 instalaciones de sus productos en más de 65 países.

Tecnológicamente, N*VISION está implementado usando C y C++ para el núcleo del sistema y J2EE para sus interfaces de usuario ⁽²⁾.

Entre los principales objetivos de N*VISION se encuentran los de manejar en tiempo real los datos de uso de la red y permitir acceso instantáneo a información clave de negocios, tales como estadísticas del uso de la red, perfiles de los clientes, costos de acceso y métricas de calidad de servicio. De esta manera busca asistir en los eventos de fallas de red, minimizar el tráfico no facturado y maximizar la satisfacción de sus clientes.

El siguiente esquema representa los módulos del sistema N*Vision:

² Datos de tecnología cortesía de ACE*COMM

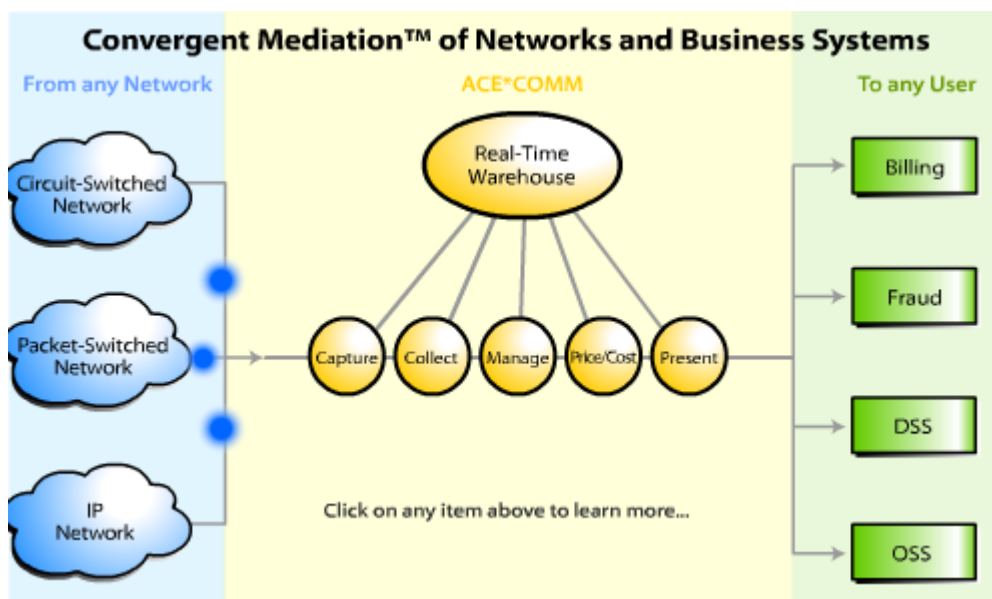


Figura 1.8: Módulos del sistema “N*VISION” ^[XLIV]

En la figura puede verse que el sistema es capaz de procesar eventos generados por varios tipos distintos de elementos de red, los cuales poseen diferentes protocolos y formatos.

Una vez procesada, la información puede ser accedida por los sistemas de facturación, detección de fraudes, sistemas de toma de decisiones y sistemas de soporte de negocios.

Los diferentes módulos de N*VISION que llevan a cabo el procesamiento son descritos a continuación:

- **Captura.-** Este módulo envuelve a los procesos de obtener eventos de diferentes tipos de redes, tales como: IP, VoIP, wireline, wireless, circuit-switched, packet-switched, etc ^[XLVI].
- **Colección.-** Este módulo realiza tareas de agregación y correlación sobre los datos obtenidos de los elementos de red. Puede ser configurado para ejecutarse prácticamente en tiempo real, o bien en procesos por lotes definidos por el usuario ^[XLVI].
- **Administración.-** Este módulo encapsula las diferentes habilidades de procesamiento y manipulación de datos del sistema de mediación. Entre sus tareas están las de normalizar los datos leídos a un simple formato meta, de manera que se simplifique las tareas de los subsistemas siguientes. También es responsable de corregir registros incompletos o que presenten errores ^[XLVII].
- **Tarifación.-** Permite monitorear los eventos de uso desde una perspectiva financiera. Soporta muchos tipos y planes de tarificación, permitiendo cambiar las tasas de cobro en tiempo real, o bien programarlas para una fecha determinada ^[XLVIII].

- **Presentación.-** Este módulo resume y reformatea datos de uso de la red, para su posterior consumo de los sistemas de soporte de negocios de la empresa. Este módulo puede realizar una transformación por registro, es decir: un registro de salida por un registro de entrada, o bien generar información resumida de los datos de consumos ^[XLIX].

Todos estos módulos interactúan con el componente de Data Warehousing del mediador, el cual añade valor a los datos obtenidos correlacionándolos y almacenándolos de acuerdo a las especificaciones dadas, al tiempo que proporciona herramientas para la toma de decisiones. El Data Warehousing del mediador almacena los registros de uso históricos y los de tiempo real tanto de manera detallada como resumida, con estadísticas y otros datos relevantes.

1.3 JUSTIFICACIÓN Y OBJETIVOS

1.3.1 JUSTIFICACIÓN DEL PROYECTO

En esta sección se presentan las justificaciones para la realización de este proyecto de tesis.

Al revisar el estado actual de los procedimientos empleados para la interpretación, transformación y carga de los datos generados por las centrales de tráfico de las empresas de telefonía fija de nuestro medio y compararlos con la forma en que son llevados a cabo por empresas de similares características en otros países, sale a la luz la necesidad de implementar una solución que permita la optimización de estos procesos. Si se agilizan los métodos de interpretación de datos y se mejora la calidad de estos últimos eliminando redundancias y corrigiendo errores, aumentará por ende la eficiencia de los sistemas de soporte de negocios consumidores de dichos datos, entre ellos el sistema de facturación, cuyos resultados son percibidos de manera directa por los clientes. De esta manera mejorará la imagen global de la empresa frente a sus consumidores.

1.3.2 OBJETIVOS Y ALCANCE DEL PROYECTO

El objetivo principal de este proyecto de tesis es el de realizar una implementación del módulo de lectura, interpretación y transformación de distintos formatos de datos de tráfico a un formato común y estandarizado, permitiendo la especificación de la estructura de los formatos de entrada por parte de un operador, sin que sea imprescindible que este posea conocimientos de programación.

Este módulo será el primero a implementarse como parte de un sistema completo de mediación telefónica que será desarrollado para la empresa Linkotel S.A, la implementación de todo el sistema de mediación está fuera del alcance de este proyecto de tesis.

El módulo a construir como parte de este proyecto de tesis, deberá permitir la definición de los formatos generados por las siguientes tipos de centrales de tráfico de Linkotel y Pacifictel:

Tipo de central	Operadora telefónica
Huawei	Linkotel
Alcatel	Pacifictel
Ericsson	Pacifictel

Tabla 1.1: Centrales Telefónicas de Tráfico cuyos formatos de archivos serán definidos por el sistema

CAPÍTULO 2

2. FUNDAMENTOS TEÓRICOS

2.1 ESTÁNDARES PARA REPRESENTACIÓN Y DEFINICIÓN DE INFORMACIÓN.

2.1.1 XML COMO ESTÁNDAR DE REPRESENTACIÓN DE INFORMACIÓN^[XXV]

El Extended Markup Language (XML) es un formato de texto derivado del SGML⁽¹⁾, originalmente diseñado para combatir las dificultades de publicaciones electrónicas a gran escala, pero rápidamente ha sido aceptado y adoptado para el intercambio de información en todo tipo de sistemas. XML se diferencia de SGML en que este es mucho más simple al ser sólo un subconjunto de SGML, y por tanto es más fácil para las aplicaciones soportar dicho formato (hasta el punto que los navegadores de Internet actuales puedan soportarlos).

Actualmente, la especificación XML es mantenida por el World Wide Web Consortium (W3C)⁽²⁾, el desarrollo y actividades del mismo se divide en grupos con diferentes objetivos y metas cada uno^[XXV].

¹ SGML: El "Standard Generalized Markup Language" es un meta-lenguaje para generar documentos mucho más complejo y avanzado que XML. SGML está descrito en el estándar ISO 8879:1986
(<http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16387>)

² World Wide Web Consortium (W3C): Es un foro de de información, comercio, comunicación y entendimiento colectivo. El W3C impulsa tecnologías inter-operativas con la misión de desarrollar el Web a todo su potencial. Para más información visitar su sitio web:
<<http://www.w3.org/>>

2.1.1.1 ANTECEDENTES

La historia del lenguaje XML es la historia de los lenguajes de marcado. A principios de la era de computación los primeros editores de texto necesitaban guardar documentos de texto con formato^[XXVII]. Como los procesadores de aquel entonces no eran muy potentes, la manera más fácil de definir documentos era usando elementos de marcado de formato, los cuales dictaban como debía presentarse el documento. Era la manera mas fácil de definir formatos en aquel entonces, pero tenía la desventaja de que los elementos de marcado estaban fijados a la aplicación en que se van a usar, el documento no es portable^[XXVIII].

En 1967 William Tunnicliffe presenta por primera vez la idea de separar el contenido del formato de un documento. William era el presidente del “Graphic Communication Association⁽³⁾” en aquel entonces. Eventualmente se generó el Comité GenCode para

³“Graphic Communication Association” (GCA): Asociación encargada de definir estándares para facilitar el intercambio de información entre empresas. Originalmente orientada a las empresas de publicación e impresión, actualmente abarca todo ámbito de negocio orientados a tecnologías de información. Su nombre actual es “International Digital Enterprise Alliance” (IDEAlliance: <http://www.idealliance.org/>)

estudiar y analizar la factibilidad de generar dicho lenguaje de marcado orientado a información.

El Comité llegó a las conclusiones de que es imposible describir todos los tipos de documentos con un sólo lenguaje de marcado, que el marcado debe ser de tipo descriptivo en vez de procedimental⁴, y que el marcado debe respetar la estructura jerárquica de los documentos.

El trabajo de la GCA fue luego continuado por Charles Goldfarb, quien trabajaba para IBM en un proyecto de aplicación de sistemas computacionales al ámbito legal. Dicho proyecto incluía entre sus metas, integrar un sistema editor de texto con un sistema de obtención de información y un sistema de composición de hojas. Viendo las inconveniencias de que cada sistema tenía su propio formato incompatible con los otros, y luego de ser presentado al trabajo del GCA, era obvia la necesidad de implementar un esquema de marcado genérico.

⁴ Marcado descriptivo es aquel que describe la funcionalidad, por ejemplo, marcado que indique cuando el texto es un título; mientras marcado procedimental indica directamente como el texto debe ser presentado, por ejemplo, marcado que indican que el texto debe ser centrado.

Goldfarb, junto con Ed Mosher y Ray Lorie desarrollaron el “General Markup Language”, el cual fue expuesto al mundo luego de su finalización junto con la primera aplicación que implementaba el estándar en 1973^[XXVI].

En 1978 el “American National Standards Institute” (ANSI)⁽⁵⁾ crea un comité de lenguajes de computación para procesamiento de texto. A este comité se integran Goldfarb y el CGA para generar un nuevo estándar basado en GML: “Standard General Markup Language” (SGML). El primer borrador del SGML fue finalizado en 1980, en 1984 el “International Standards Organization” (ISO)⁽⁶⁾ también se une al grupo de trabajo, y en 1986 SGML se vuelve un estándar internacional.

El SGML era un meta-lenguaje por medio del cual se podían especificar otros lenguajes. Usando definiciones de documento

⁵ El “American National Standards Institute” (ANSI) es una institución privada sin fines de lucro cuyo objetivo es administrar y coordinar el uso de estándares en las empresas afiliadas en los Estados Unidos <<http://www.ansi.org>>.

⁶ International Organization for Standardization (ISO): Red de estándares nacionales de 148 países (al momento de este escrito) que coordina el sistema de dichos estándares <<http://www.iso.org>>.

(“Document Type Definition” - DTDs) se podían definir otros lenguajes, entre ellos^[XXVIII]:

- EAD (Encoded Archival Description)
- Docbook
- TEI (Text Encoded Initiative)
- HTML (Hyper Text Meta-Language)

Eventualmente se quería aplicar SGML en el Web, pero su implementación y nivel de complejidad era muy alto como para ser usado a través de una red. Es por este motivo que se emprende el diseño del XML, el cual es mucho más simple que el SGML para poder usarlo a través de la red.

2.1.1.2 ESPECIFICACIÓN DEL XML^[XXIX]

Actualmente existen dos especificaciones del XML, la versión 1.0 (cuya especificación actual es la tercera edición) y la versión 1.1. La versión 1.0 contiene todas las bases y detalles acerca del formato requerido, mientras la especificación 1.1 agrega otras características que no fueron consideradas anteriormente.

La especificación XML describe a construcciones de datos conocidas como “documentos XML”, las cuales están compuestas de “entidades”. La especificación incluye una descripción del funcionamiento de un “procesador XML”, el cual es un módulo de software encargado de leer la información de un archivo XML y pasarla a otras aplicaciones. Al diseñar el “Extended Markup Language” se tenía entre sus metas definidas:

- Poder ser usado directamente a través del Internet
- Soportar un gran rango de aplicaciones
- Ser compatible con SGML
- Que sea fácil escribir programas que procesen XML
- El número de características opcionales debe ser mantenidas al mínimo.
- Documentos XML deben ser leíbles por humanos, y relativamente claros.
- Los documentos XML deben ser fáciles de crear.

2.1.1.2.1 ESTRUCTURA LÓGICA DE LOS DOCUMENTOS XML

El documento esta formado por uno o más elementos, cada elemento está delimitado por su etiqueta de inicio y fin, y si el elemento está vacío, puede estar delimitado por una etiqueta vacía. Cada elemento tiene un tipo, el cual es identificado por el nombre del elemento. Adicionalmente, los elementos pueden contener atributos, cada atributo tiene un nombre y un valor.

Las etiquetas de inicio están delimitadas por los signos “<” y “>”, entre los cuales debe ir el tipo del elemento, seguido de los atributos que tenga. Los atributos se definen colocando el nombre del atributo, seguido del signo “=” y a continuación colocando el valor del atributo, entre comillas (simples o dobles, ambas son aceptadas para permitir el uso de la otra como parte del valor). La etiqueta de finalización los delimitadores “</” y “>”, y debe tener el mismo nombre de elemento usado en la etiqueta de apertura. La etiqueta de cierre no puede tener atributos. Por ejemplo, así se define un elemento llamado “moneda” con el atributo unidad:

```
<moneda unidad="dolar">  
  <!-- Aquí van los contenidos del elemento moneda  
  -->
```

```
</moneda>
```

Si el elemento está vacío, existe una etiqueta especial que se puede usar para señalar esto (en vez de colocar las etiquetas de inicio y final juntas), es la misma etiqueta de inicio, excepto que se cierra con la secuencia “/” en vez de “>”. Usando el mismo ejemplo anterior:

```
<moneda unidad="dolar" />
```

2.1.1.2.2 COMPONENTES Y DEFINICIONES

Documento XML

El documento XML es una entidad que cumple con las siguientes características^[XXIX]:

- Está bien formado. Un documento está bien formado cuando cumple con las siguientes tres características: El documento tomado como un todo, cuadra con el patrón de producción llamado “document”, cumple con todas restricciones de formación descrito, y que cada una de la entidades validadas a las que se hacen referencia a la vez están bien formadas.

- Está compuesto de uno o más elementos, cada uno de estos elementos también están bien formados.
- Para todo elemento que sea definido dentro del ámbito de otro elemento, el elemento previo debe contener en su totalidad al elemento posterior (lo cual quiere decir que el anidamiento de elementos debe ser correcto). La relación entre elemento y elemento anidado se conoce como la relación padre/hijo de elementos.

Procesadores XML

Los procesadores XML se dividen en dos tipos: en procesadores validadores y procesadores no validadores. Se dice que un procesador de documentos XML es validador cuando puede verificar que el documento XML esté correctamente construido según el DTD del mismo. Ambos tipos de procesadores deben siempre verificar que el documento XML esté bien formado según las especificaciones del formato XML^[XXIX].

El usuario del procesador debe estar en la capacidad de deshabilitar (o habilitar) la validación del DTD según sea conveniente.

Otros Conceptos y Definiciones

A continuación se enlistan varias de las definiciones y estructuras usadas en los documentos y una breve explicación de que son:

Nombre	Descripción
Datos de caracteres y marcado	Se refieren a todo texto dentro del documento que no son datos textuales, sino que representan información XML.
Comentarios	Permiten introducción de información textual de información para quienes lean el documento. Usa los delimitadores “<!--” y “-->”
Instrucciones de Procesamiento	Almacenan información que se le pasa a la aplicación o procesador XML (como la codificación del archivo), sus delimitadores son “<?” y “?>”
Sección CDATA	Permite almacenar cadenas de texto en el cual el formato es ignorado. Útil para almacenar información que pueda contener texto interpretable como XML. Sus delimitadores son “<![CDATA[“ y “]]>”
Prólogo	Es la declaración inicial de que el documento es XML, en el se declara la versión y la codificación del documento entre otros.
Document Type Definition ⁽⁷⁾	Permite la declaración de ellos dentro del mismo documento XML. Usa los delimitadores “<!DOCTYPE” y “>”.
Lenguaje	permite especificar el lenguaje del texto contenido dentro del nodo actual y sus descendientes. Usa el atributo “xml:lang” para ello. El valor del atributo es el descriptor de lenguaje según el IETF RFC 3066 ⁽⁸⁾

⁷ Document Type Definition (DTD) es un esquema para poner reglas sobre la estructura de un documento XML.

⁸ Internet Engineering Task Force: RFC 3066: Etiquetas para la identificación de lenguajes. Define una manera estándar para que las aplicaciones puedan definir el lenguaje de su contenido por medio de cadenas únicas para identificar el lenguaje y región. Su contenido se puede encontrar en el sitio del IETF <<http://www.ietf.org/rfc/rfc3066.txt>>.

Nombre	Descripción
Entidades de parámetros	Permite la definición de entidades que se comportan como constantes que se pueden reutilizar en varias partes del documento. A una entidad se la referencia con un "&" seguida por el nombre de la entidad. Las entidades se declaran usando los delimitadores "<!ENTITY" y ">".
Entidades de caracteres	Permite hacer referencia a caracteres que de otra forma serían difíciles de incluir en el documento XML. Se les hace referencia por su valor decimal (de manera &#NNNN; donde NNNN es el valor decimal), o por su valor hexadecimal (&#xNNNN; donde NNNN es el valor hexadecimal).
Notaciones	Son entidades que no deben ser validadas por el procesador XML. Las notaciones contienen un identificador externo el cual se puede usar para validar dicha entidad externamente. Sus delimitadores son "<!NOTATION" y ">".

Tabla 2.1 Estructuras y componentes comunes de un documento XML^[xxix]

2.1.1.2.3 XML VERSIÓN 1.1

La versión 1.1 del estándar XML fue hecho para incrementar la compatibilidad futura con Unicode, también define reglas más robustas sobre la normalización del documento. Los procesadores XML deben soportar ambas versiones, cuando las nuevas características de la versión posterior no son necesarias, no hay problemas en usar la versión 1.0. A continuación se explican los cambios más importantes ocurridos en esta versión:

- Independencia de la versión de Unicode. Mientras la especificación 1.0 indicaba que caracteres eran válidos dentro del formato XML y descartaba todos los demás como inválidos, la versión 1.1 hace lo

contrario, al especificar explícitamente que caracteres son inválidos y permitiendo todos los demás como válidos, facilita la incorporación de nuevos caracteres Unicode a medida que este evoluciona, volviendo a la especificación XML independiente de la especificación Unicode, y reduciendo así la necesidad de actualizar la especificación.

- La cantidad de caracteres representables por medio de caracteres de referencia fue incrementado drásticamente para facilitar la utilización de dichos caracteres en el documento.
- Finalmente, se definen restricciones adicionales sobre el documento para lograr una “normalización completa”, la cual garantiza que la comparación de cadenas de entidades y valores dentro del documento puede ser realizada a nivel binario entre las cadenas Unicode sin causar problemas.

2.1.2 XML SCHEMA COMO ESTÁNDAR DE VALIDACIÓN DE DOCUMENTOS XML

XML Schema es un estándar recomendado por el W3C el cual define un vocabulario por medio del cual es posible definir la estructura, contenido y semántica de un documento XML. La versión actual es 1.0 (finalizada el 2 de Mayo del 2001), y la versión 1.1 está, al momento de este escrito, bajo desarrollo.

Existen algunas diferencias entre XML Schemas y DTDs:

- Los DTDs tienen su propia estructura, mientras un esquema se define en XML propio, los requerimientos de una aplicación para tratar con esquemas no requieren validadores adicionales al del XML.
- Los esquemas tienen mayor flexibilidad al definir datos, por ejemplo, se puede definir el “tipo” de dato de un elemento (booleano, numérico, fechas, tiempos) mientras un DTD sólo puede validar numeraciones de elementos (donde los valores posibles están restringidos a los especificados explícitamente en la definición del DTD).

- Como XML Schema fue definido después de los DTD, es fácil notar las ventajas de este nuevo formato. Adicionalmente, las definiciones generadas por los esquemas se suelen almacenar en sus propios documentos externos. Una definición de esquema XML (XML Schema Definition “XSD”) se suele almacenar en un documento “.xsd” lo cual facilita su reutilización.

En el Apéndice G se explica como se definen archivos XSD para definir la estructura de varios elementos de un documento XML.

2.2 DESCRIPCIÓN DE TECNOLOGÍAS AUXILIARES DE INFORMACIÓN BASADAS EN XML USADAS EN EL PROYECTO DE TESIS.

2.2.1 DOCUMENT OBJECT MODEL (DOM)

El Document Object Model (DOM) o Modelo de Objetos del Documento es una especificación del World Wide Web Consortium (W3C) para acceder y manipular los contenidos de un documento. La definición de DOM expresada por el W3C es: “El Modelo de Objetos

del Documento (DOM) es una interfaz de programación de aplicaciones (API) para documentos HTML y XML”^[XXXIX].

2.2.1.1 HISTORIA^[XXXIV]

Los “Documentos” al que se refieren la especificación DOM originalmente era a páginas Web, el contenido de dichas páginas era estático en su totalidad (o generadas en el servidor) al principio, pero con el arribo del HTML versión 4.0, se comenzó a hablar del concepto de “Dynamic HTML” (DHTML). La idea era que las páginas ya no sean de contenido estático, sino que desde el lado del usuario se pueda acceder y modificar los contenidos. Para lograr esto se desarrollaron lenguajes como JavaScript y JScript hechos por Netscape y Microsoft respectivamente⁽⁹⁾. Pero estos lenguajes no eran compatibles entre ellos, por lo que las páginas escritas con uno en mente no funcionarían en el otro. Es aquí cuando interviene el W3C y se decide formar un estándar para acceso y manipulación de los contenidos del documento independiente de la plataforma, DOM. Se esperaba que el primer nivel del DOM (DOM Nivel 1) fijara la base

⁹ En aquel entonces, Microsoft y Netscape eran los únicos competidores en el área de navegadores de internet.

sobre la cual los lenguajes de “script” podrían acceder y modificar los contenidos de documentos HTML, pero Microsoft decidió implementar su propia versión del DOM con extensiones únicas para su navegador “Internet Explorer” lo cual fragmentó los sitios que eran compatibles con DOM y los que eran compatibles con la versión de DOM de Microsoft.

La evolución del DOM se divide en niveles, de tal manera que se puede identificar con facilidad que número de funciones del DOM están soportados por un programa en particular. Reciente se aprobó el tercer nivel del DOM como recomendación oficial del W3C⁽¹⁰⁾ mientras el cuarto nivel todavía está en etapa de desarrollo.

2.2.1.2 INTRODUCCIÓN

La idea atrás de un DOM es representar un Documento como un conjunto de nodos organizados en una jerarquía y proveer métodos de navegación y acceso a los datos de dicha jerarquía. Se podría decir que los nodos del documento están organizados a manera de

¹⁰ 07 de Abril, 2004

árbol, excepto que la especificación no indica que el documento deba tener sólo un árbol; el documento puede tener cualquier número de árboles. A continuación se muestra un ejemplo de DOM basado en el contenido de una página HTML cualquiera:

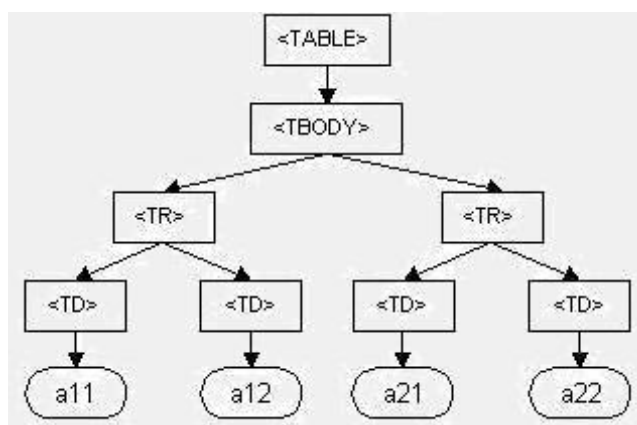


Figura 2.1 Ejemplo de un DOM⁽¹¹⁾

La idea atrás de la especificación del DOM es permitir métodos de navegación a través de estos nodos, así como métodos de extracción de información de los mismos.

¹¹ Fuente: Artículo 'A Gift of "Life": The Document Object Model
<<http://tech.irt.org/articles/js143/>>

2.2.1.3 ESPECIFICACIÓN DEL DOM^[xxxv]

A continuación se listan las diferentes secciones de los niveles del DOM y una breve descripción de cada una.

2.2.1.3.1 DOM NIVEL 0

Antes de que el W3C se propusiera a definir la especificación formal del DOM, ya existía en el mercado el concepto del mismo con la introducción de las tecnologías JScript y Javascript. El W3C se basó en estas tecnologías existentes para generar a DOM y por tanto a estos conceptos originales se les denomina DOM de nivel 0.

2.2.1.3.2 DOM NIVEL 1

El primer nivel del DOM se concentra en asegurar que ambos navegadores compartan un DOM común. Se concentra en el núcleo del DOM, y sobre este núcleo construye especificaciones especiales orientadas al HTML.

DOM Nivel 1 – Núcleo:

El núcleo del DOM está orientado a procesamiento de información en formato XML. Define los fundamentos sobre la estructura de elementos, nodos, atributos y otros componentes del DOM. El núcleo del DOM se divide en dos tipos de interfaz: Las interfaces fundamentales, y las interfaces extendidas. A continuación se listan dichas interfaces con una muy breve explicación:

Nombre	Descripción
DOMException	Clase raíz para el manejo de excepciones en el manejo de un DOM.
DOMImplementation	Proporciona funciones independientes de cualquier instancia del DOM incluyendo el acceso a las propiedades de la implementación de DOM usada.
DocumentFragment	Interfaz con funcionalidad muy limitada, para ser usada como almacenamiento temporal de secciones de un DOM.
Document	Raíz del documento que encapsula a todos los demás nodos. Todo nodo debe siempre pertenecer a algún documento.
Node	Interfaz genérica que encapsula el comportamiento global de los nodos dentro del DOM.
NodeList	Permite una manera de acceder a una lista de nodos de manera ordenada.
NamedNodeMap	Permite una manera de acceder a una lista de nodos por medio del nombres de los mismos.
CharacterData	Interfaz con los atributos y métodos necesarios para la manipulación de texto.
Attr	Interfaz que define a un atributo.
Element	Interfaz que define a un elemento.
Text	Interfaz que define a un conjunto de texto del documento.
Comment	Interfaz para almacenar los comentarios del documento.

Tabla 2.7: Interfaces fundamentales del Núcleo del DOM, Nivel 1

Nombre	Descripción
CDATASection	Encapsulan las secciones de texto especial llamadas CDATA.
DocumentType	Describe el tipo de documento y permite el acceso a las entidades del mismo.
Notation	Encapsulan a las notaciones de un DTD.
Entity	Modelan a las entidades del DOM.
EntityReference	Modelan a las referencias de entidades dentro del DOM.
ProcessingInstruction	Almacenan las instrucciones de procesamiento, las cuales son relevantes al procesador de texto del documento.

Tabla 2.8: Interfaces Auxiliares del Núcleo del DOM, Nivel 1

DOM Nivel 1 – HTML:

Este nivel de la especificación adapta el núcleo del DOM para ser utilizado en páginas HTML. Define interfaces orientadas a los diferentes elementos definidos en los documentos HTML y sus atributos pertinentes.

2.2.1.3.3 DOM NIVEL 2

El siguiente nivel de DOM extiende mucho más la especificación y está dividido en recomendaciones para varias secciones:

- núcleo
- Vistas

- Eventos
- Estilos
- Navegación y Rangos
- HTML

A continuación se describen brevemente estas secciones:

DOM Nivel 2 – Núcleo:

Entre los nuevos conceptos agregados al DOM se tienen soporte para los “Espacio de Nombre XML”⁽¹²⁾, en cuanto a las interfaces definidas, no se agregaron nuevas estructuras, y algunas de las interfaces existentes fueron ligeramente alteradas⁽¹³⁾.

DOM Nivel 2 – Vistas:

La especificación de vistas es una recomendación diseñada para proveer acceso y actualizar el contenido de la representación de un

¹² Los “XML NameSpace” definen un ámbito de contexto en los XML, de tal manera que los elementos de cierto NameSpace tienen una cantidad de elementos y atributos ya definidos.

¹³ Ver el Apéndice A de la especificación del DOM Nivel 2 – Núcleo para los detalles <<http://www.w3.org/TR/DOM-Level-2-Core/>>

documento, es decir, permitir a programas modificar como se describe visualmente un documento. Este módulo define una única interfaz “AbstractView”, la cual es la interfaz base de la cual todas las vistas del documento deben heredar.

DOM Nivel 2 – Eventos:

Esta especificación define un sistema de eventos genéricos. El flujo general de eventos es que para cada evento existe un objeto objetivo al cual llegan los mensajes, una vez que el mensaje llega a este objeto, se lo pasa a todos los objetos que se han registrado para escuchar eventos con este objetivo en particular. No se especifica el orden en que los objetos registrados deben ser atendidos, y excepciones sucedidas durante el manejo de los eventos en uno de los objetos registrados no detiene a los demás de procesar el mismo evento posteriormente. El uso de eventos no fue necesario para el desarrollo de este trabajo de tesis.

A continuación se explica la terminología usada para eventos:

Nombre	Descripción
---------------	--------------------

Nombre	Descripción
UI Events	Eventos generados por la intervención directa del usuario por medio de dispositivos de entrada como el teclado o ratón.
UI Logical Events	Eventos de interfaz de usuario independientes del dispositivos (incluye cambios de foco, como lanzadores de eventos).
Mutation Events	Eventos generados cambios en la estructura del documento.
Capturing	El proceso de captura implica que los ancestros del objeto que está esperando algún evento puedan manipular y manejar el evento antes de que llegue al objeto que se registró para aquel evento.
Bubbling	Este proceso implica que luego de que el objeto manejador del evento maneja el evento, este evento se puede propagar por sus ancestros, para que todos ellos lo manejen posteriormente.
Cancelable	Permitir a objetos cancelar la propagación de un evento, de tal manera que el manejo por omisión por parte de la implementación DOM sea cancelada.

Tabla 2.9: Terminología usada en los Eventos

DOM Nivel 2 – Estilos

Esta especificación provee métodos de acceso por DOM a las hojas de estilos y hojas de estilo en cascadas (“Style Sheets” y “Cascading Style Sheets” – CSS). Las hojas de estilo son un lenguaje para definir la presentación de documentos, permiten separar las capas de información y datos de la capa de presentación e interfaz al usuario.

DOM Nivel 2 – Navegación y Rangos

Esta especificación define interfaces y métodos para poder navegar a través de nodos de un DOM y para poder definir y manipular rangos del documento.

Navegación (Traversal)

Esta especificación del DOM provee de tres interfaces para poder navegar a través de un subconjunto del DOM:

Nombre	Descripción
Nodelterator	Permite iterar sobre los nodos de manera lineal, no preserva la información de jerarquía en el DOM.
TreeWalker	Permite almacenar y navegar subárboles del DOM, preservando la relación entre los nodos contenidos.
NodeFilters	Permite definir reglas de filtrado, para especificar que nodos serán visibles dentro de las colecciones de un TreeWalker o un Nodelterator.

Tabla 2.10: Interfaces del módulo de Navegación del DOM Nivel 2

Rangos (Ranges)

Las interfaces de rangos permiten la selección de subconjuntos de nodos dentro del DOM y proveen métodos de manipulación sobre estos para realizar operaciones de edición comunes.

Los rangos se definen por medio de dos posiciones (inicial y final), y cada posición está compuesta de dos partes: un nodo y un desface. El nodo es un elemento, atributo, o un bloque de texto que contiene el límite del rango, y el desface define el índice entre los hijos de este que “es” el límite, aunque en el caso de bloques de texto, el desface especifica que carácter dentro del bloque marca el límite. Descrito de manera gráfica:

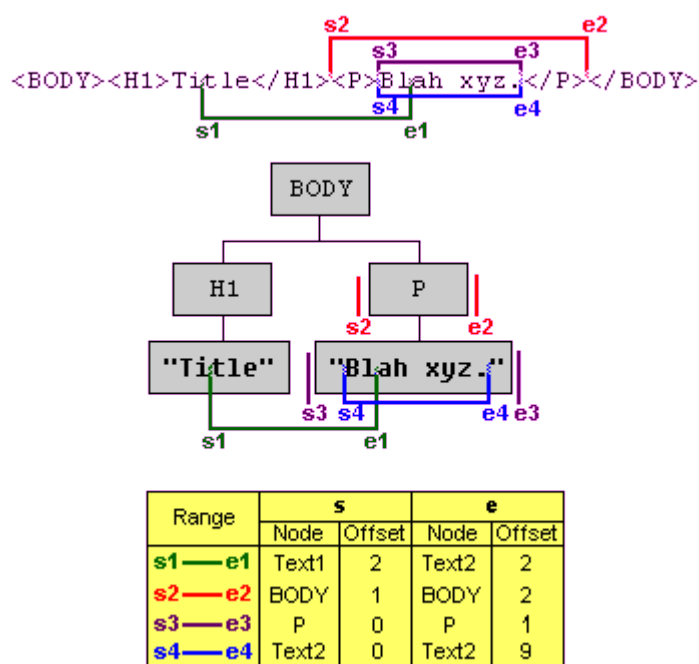


Figura 2.2 Ejemplo de un Rango⁽¹⁴⁾

¹⁴ Fuente: Document Object Model Range <<http://www.w3.org/TR/2000/REC-DOM-Level-2-Traversal-Range-20001113/ranges.html>>

Nótese que para que el rango definido sea válido, es necesario que ambos puntos de inicio y fin tengan un ancestro común.

Se puede profundizar mucho más en el concepto de rangos y sus métodos, pero esta es una funcionalidad del DOM que no fue necesitada para este proyecto de tesis.

DOM Nivel 2 – HTML

En el nivel 2 del componente de HTML del DOM se agregó soporte para documentos XHTML 1.0, el cual es una versión más robusta del HTML, pues se apega más fielmente a las reglas de sintaxis de los documentos XML. Aparte de esto, hubieron pequeños cambios en la especificación de las interfaces HTML las cuales están fuera del alcance de este proyecto de tesis.

2.2.1.3.4 DOM NIVEL 3

Recientemente (Abril 7, 2004) el nivel 3 de la especificación DOM se convirtió en una recomendación oficial del W3C. Esta especificación agrega nuevas características que permiten facilitar la validación y

guardado de un DOM. Hay que notar que esta recomendación es muy reciente, y por ende, no hay todavía implementaciones de la misma disponibles para usar en este proyecto de tesis.

Los módulos de cargado y guardado especifican una manera estándar e independiente de plataforma y lenguaje para permitir a lenguajes y scripts convertir un DOM a un archivo XML y cargar un archivo XML en un DOM de manera programática⁽¹⁵⁾.

La especificación de validación provee un API por medio del cual aplicaciones pueden ser notificadas de la validez del documento a través del tiempo (es decir, pueden revisar la validez actual del documento, y la validez futura luego de aplicar algún cambio).

2.2.2 SAX^[XXXVI]

SAX significa “Simple API for XML”, lo cual indica la meta de SAX: Ser una interfaz ligera y fácil de acceso y manipulación de documentos XML.

¹⁵ Debido a que esta especificación no estaba disponible a la hora de implementar la aplicación, en el diseño se decidió optar por Xerces como método de guardado y cargado.

2.2.2.1 CARACTERÍSTICAS DE SAX

La arquitectura de SAX está basada en eventos, de tal manera que las aplicaciones se pueden registrar para ser notificadas de diferentes eventos que suceden a medida que el validador va recorriendo el archivo. Los APIs de programación de XML usualmente son de dos tipos, basados en árboles y basados en eventos. La ventaja de usar un entorno orientado a eventos es que los requerimientos de memoria son mucho menores al no tener que armar y mantener un árbol en memoria todo el tiempo, además de permitir que las aplicaciones puedan usar los datos tomados para construir su propio árbol adaptado a las necesidades de la aplicación, en vez de intentar reusar el árbol genérico generado por los otros APIs.

Características y Propiedades del Validador:

SAX tiene una manera estándar de alterar y fijar las características del validador a usar. Con los mismos dos métodos se puede obtener y fijar los valores de cualquier parámetro soportado por los validadores, errores o funciones no soportadas son manejadas por medio de excepciones.

Sólo existen dos propiedades que obligatoriamente todos los validadores XML deben soportar: “espacios de nombre” y “prefijos de espacios de nombre”⁽¹⁶⁾, ya que este par de características son necesarias para asegurar que todo validador tenga al menos soporte mínimo de nombres de espacio, los cuales son necesarios para niveles más altos de lenguajes XML (como RDF, XML Schema, XLink, etc).

Filtros:

SAX también contiene el concepto de filtros, por el cual los mensajes de notificación de eventos pasan por varios escuchas, de tal manera que cada parte del flujo puede modificar el mensaje según sea necesario. Los filtros pueden ser de dos tipos, aquellos usados para post-proceso de eventos SAX, y aquellos filtros que se encapsulan para formar una única fuente de lectura para la aplicación (ver XMLReader más adelante).

¹⁶ Las cadenas que representan estas características son "http://xml.org/sax/features/namespace" y "http://xml.org/sax/features/namespace-prefixes" respectivamente.

2.2.2.2 ESPECIFICACIÓN E INTERFACES USADAS

A continuación se enlistan las interfaces y clases que componen al paquete principal de SAX. Cada una de las interfaces y clases es descrita brevemente.

Nombre	Descripción
Attributes	Provee un método de manipulación de atributos.
ContentHandler	Interfaz principal, la cual recibe notificaciones sobre los datos a medida que se van leyendo.
DTDHandler	Interfaz por la cual la aplicación se registra para escuchar eventos relacionados con los DTD.
EntityResolver	Usada cuando se desea que la aplicación intercepte y resuelva entidades en vez de dejar dicho trabajo a SAX.
ErrorHandler	Interfaz por medio de la cual la aplicación se puede enterar de errores no fatales que suceden durante el procesamiento del documento.
Locator	Interfaz que permite la asociación de eventos con la posición exacta en el documento donde sucede.
XMLFilter	Interfaz para la implementación de filtros. Son como lectores XML, excepto que leen la información pasada de este (o de otros filtros).
XMLReader	Es la interfaz implementada por cada validador XML. Permite registrarse como escucha de eventos y provee los métodos para configurar el validador. Todos sus métodos deben ser sincrónicos.

Tabla 2.11: Interfaces de SAX

SAX implementa una única clase: `InputSource`. Esta clase encapsula a la fuente de información de la cual leer el documento XML. Esta clase puede contener información de un flujo de datos (de caracteres

o binario), un atributo de codificación, y un identificador de sistema^[XL].

Las excepciones usadas por SAX son:

- SAXException
- SAXNotRecognizedException
- SAXNotSupportedException
- SAXParseException

2.2.3 XERCES^[XXXVII]

Xerces es un validador XML que forma parte de la fundación Apache⁽¹⁷⁾. La versión mas reciente es Xerces 2.6.2 y tiene las siguientes características:

- Soporta la recomendación XML 1.0 Tercera Edición

¹⁷ La fundación de Software Apache provee soporte para la comunidad de proyectos de fuente libre afiliados. Dichos proyectos son caracterizados por en método de desarrollo colaborativo, licencias abiertas y pragmáticas, y un desea de generar Software de alta calidad que lidere su campo. Ver <<http://www.apache.org/>> para más información sobre la Fundación Apache.

- Soporta la recomendación XML 1.1
- Soporte de Nombres de Espacio (versiones 1.0 y 1.1)
- Soporta las siguientes recomendaciones DOM nivel 2: Núcleo, Eventos, Recorrido y Rangos.
- Implementa SAX 2.0.1 (núcleo y extensiones)
- Implementa JAXP 1.2
- Implementa Esquemas XML versión 1.0

Aparte de los APIs estándares, Xerces también da acceso a su interfaz nativa (Xerces Native Interface – XNI) para aquellos casos en que los APIs estándares no ofrezcan la funcionalidad deseada.

2.2.3.1 INTERFAZ NATIVA DE XERCES

El XNI es un marco de trabajo para comunicar un flujo de información de documento y construcción de validadores genéricos para los mismos. XNI es parte del desarrollo de Xerces, pero el XNI es un modelo de interfaz sólido sobre el cual se pueden construir otros validadores. Xerces simplemente es una implementación construida sobre XNI que cumple con las especificaciones de DOM y SAX.

El XNI mantiene un flujo de información a través de distintos componentes, los cuales pueden ser modificados o intercambiados por otros, lo cual permite la generación de validadores genéricos altamente configurables. El flujo de información XNI es mucho más flexible que el flujo especificado por SAX, ya que se preserva la mayor cantidad de información posible, y los filtros por los cuales pasa la información pueden modificar al flujo (mientras en SAX los escuchas básicamente ven un flujo de sólo lectura. Un esquema del flujo de información es como el que sigue:

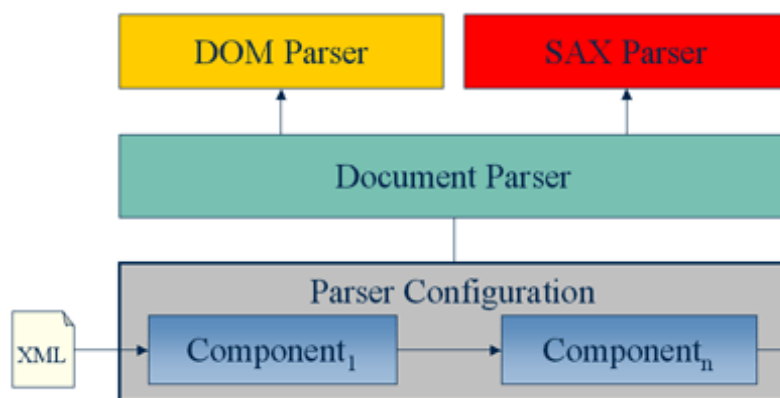


Figura 2.3 Flujo de Información en una configuración de Validador⁽¹⁸⁾

¹⁸ Fuente: XNI Parser Configuration <<http://xml.apache.org/xerces2-j/xni-config.html>>

Las primeras etapas son la configuración del validador, mientras la última etapa define el API que se presenta a las aplicaciones exteriores.

2.2.4 XPATH

XML Path Language es un lenguaje especificado por el W3C para definir “caminos” dentro de un documento XML relativos a la posición donde se encuentra definido dicho camino. El objetivo era proveer un sintaxis, funcionalidad y semántica común entre las implementaciones de XLST⁽¹⁹⁾ y XPointer⁽²⁰⁾ ya existentes.

Las expresiones XPath trabajan sobre el documento XML a su nivel de estructura lógica y pueden tener cuatro diferentes tipos de resultados básicos:

¹⁹ XLS Transformations: Es un lenguaje diseñado para transformar documentos XML a otros documentos también XML pero de diferente estructura. Ver <<http://www.w3.org/TR/xslt>> para acceder a la especificación.

²⁰ XML Pointer Language: Es un lenguaje especificado para poder identificar fragmentos de cualquier referencia URI de un documento XML. Como su nombre lo indica, es para poder apuntar y referirse a porciones específicas dentro del documento XML. Ver <<http://www.w3.org/XML/Linking>> para los detalles sobre XPointer.

- Conjunto de Nodos
- Booleano
- Número
- Cadena

2.2.4.1 ESTRUCTURA^[XXXVIII]

Cada expresión XPath trabaja sobre el contexto en el que se encuentra. La definición del contexto para cada expresión es definida por el entorno en que XPath fue usado (XSLT o XPointer). Las expresiones XPath pueden estar compuestas de algunos elementos, pero el más importante y usado es el camino de localización.

2.2.4.1.1 CAMINOS DE LOCALIZACIÓN (LOCATION PATHS)

Son una secuencia de pasos de localización (location steps) separados por el carácter “/”, cada paso sirve para generar un conjunto de nodos que servirán como nodos de contexto para el siguiente paso. La expresión XPath puede ser absoluta en vez de relativa, en tal caso la expresión empieza con “/”, y el nodo de contexto para el primer paso es el nodo raíz del documento.

2.2.4.1.2 PASOS DE LOCALIZACIÓN (LOCATION STEPS)

Cada paso de localización tiene como objetivo seleccionar un conjunto de nodos a partir del nodo de contexto actual, para usar este nuevo conjunto de nodos como nodos de contexto para el siguiente paso. El último paso del camino se lo puede evaluar a los otros tipos básicos de resultado para la expresión XPath, pero los pasos intermedios son exclusivamente para seleccionar nodos de contexto para los pasos siguientes.

Cada paso está compuesto de un eje, un nodo de prueba y opcionalmente uno o más predicados. A continuación se muestra un ejemplo con cero, uno y dos predicados:

```
child::ejemplo  
descendant::valor[position()>1]  
descendant::valor[position()>1][child::dolar]
```

Como se puede ver, los ejes van separado de los nodos de prueba por un par de dos puntos, los predicados van encerrados entre corchetes cuadrados (“[” “]”).

2.2.4.1.3 EJES

Los ejes indican hacia que sentido buscar e identificar nodos en relación con el nodo de contexto. Por ejemplo, “self” indica que la evaluación debe ser sobre el mismo nodo de contexto, mientras “child” se refiere a los nodos hijos directos del nodo contexto. La lista de ejes disponibles con una breve explicación es:

Eje	Descripción
child	Contiene a los hijos directos del nodo.
descendant	Contiene a todos los descendientes del nodo.
parent	El padre del nodo actual, vacío para la raíz del documento.
ancestor	Contiene al padre del nodo actual, y al padre de este, y así sucesivamente hasta la raíz del documento.
following-sibling	Contiene a los nodos que se encuentran lógicamente después del nodo actual que tienen al mismo padre que el nodo de contexto.
preceding-sibling	Contiene a los nodos ubicados antes del nodo de contexto que tienen al mismo padre que el nodo actual.
following	Contiene a todos los nodos ubicados lógicamente después del nodo actual, pero que no son descendientes del mismo.
preceding	contiene a todos los nodos ubicados lógicamente antes de nodo actual exceptuando a los ancestros.
attribute	Eje que selecciona los atributos pertenecientes al nodo actual.
namespace	Contiene a los nodos de espacio de nombre al que pertenece el nodo actual.
self	El nodo de contexto.
descendant-or-self	Unión de los ejes “descendant” y “self”.
ancestor-or-self	Unión de los ejes “ancestor” y “self”.

Tabla 2.12: Ejes de XPATH

2.2.4.1.4 LOS NODOS DE PRUEBA

Sirven para filtrar los nodos a seleccionar entre los nodos contenidos en el eje. Según el tipo de eje, existe un nodo de tipo principal; por ejemplo, si el eje es attribute, el tipo principal de nodos son los nodos de atributo, para el eje de espacios de nombre, el tipo principal es de namespace, y para los demás nodos, el tipo principal son los elementos. El nodo de prueba también se puede validar contra un espacio de nombre: child::proyecto:precio, el cual selecciona los hijos del nodo contexto que contienen al elemento llamado precio del espacio de nombre “proyecto”.

Existen nodos de prueba predeterminados en XPATH, y su sintaxis es como la de funciones de los lenguajes funcionales, a continuación se las enlistan junto con una breve descripción:

Nombre	Descripción
text()	Selecciona los nodos de texto.
comment()	Selecciona los comentarios.
processing-instruction()	Selecciona aquellos nodos que son instrucciones de procesamiento.

Nombre	Descripción
node()	Selecciona todos los nodos.

Tabla 2.13: Nodos de prueba predeterminados de XPATH

2.2.4.1.5 PREDICADOS

Cada paso de localización puede tener opcionalmente uno o más predicados. Los predicados son expresiones que se evalúan sobre cada uno de los nodos resultantes de los dos pasos anteriores (ejes y pruebas de nodo) para generar un nuevo subconjunto de nodos, si el predicado da verdadero, el nodo se preserva dentro de la lista. Para cada nodo en que se evalúa el predicado, ese nodo se convierte en el nodo contexto para la evaluación.

Las expresiones usadas en los predicados pueden ser de los siguientes tipos:

- referencia a una variable
- otra expresión encerrada en paréntesis
- un valor literal
- un número
- una llamada a una función

A continuación se listan y describen brevemente las funciones disponibles para los predicados, agrupadas según el tipo de función.

Nombre	Descripción
Funciones basadas en conjuntos de Nodos	
last()	Retorna el índice del último nodo en la lista.
position()	Retorna el índice del nodo de contexto.
count(node-set)	Retorna el número de nodos en el parámetro enviado.
local-name()	Retorna el nombre local del nodo.
namespace-uri()	Retorna el espacio de nombre del nodo correspondiente.
id(object)	Retorna el ID único ⁽²¹⁾ del elemento pasado.
Funciones basadas en Cadenas de Texto	
string(object)	Convierte a una cadena de texto el parámetro pasado.
concat(str1, str2, ...)	Concatena las cadenas pasadas como texto.
starts-with(str1, str2)	Retorna verdadero si la primera cadena empieza con la segunda.
contains(str1, str2)	Retorna verdadero si la primera cadena contiene a la segunda.
substring-before(str1, str2)	Retorna la parte de la primera cadena desde el principio hasta antes de la primera aparición de la segunda cadena.
substring-after()	Retorna la parte de la segunda cadena desde después de la primera aparición de la segunda cadena, hasta el final de la misma.
substring(str, int1, int2)	Retorna la subcadena del primer elemento, usando el segundo parámetro como índice de inicio y el tercero como longitud de la cadena resultante.
string-length(str)	Retorna la longitud de la cadena.

²¹ ID único: Atributo de nombre "id" que tienen los elementos cuando este fue definido en el DTD del documento. El valor de este atributo no puede ser el mismo en más de un elemento a la vez, debe ser único en todo el documento. Si el documento no tiene DTD, no pueden existir IDs únicos.

Nombre	Descripción
normalize-space()	Normaliza la cadena según las reglas para colapsar espacios de XML.
translate(str1, str2, str3)	Devuelve la primera cadena, reemplazando los caracteres que se encuentren en el segundo parámetro por su equivalente (según el índice) del tercero. Por ejemplo, translate(str, "abc", "ABC") hace que todas las ocurrencias de a, b y c sean reemplazadas por "A", "B" y "C".
Funciones de Operadores Booleanos	
boolean(object)	Convierte el parámetro a un valor booleano.
not(boolean)	Retorna la negación del parámetro enviado.
true()	Retorna siempre verdadero.
false()	Retorna siempre falso.
lang(str)	Retorna verdadero cuando el nodo de contexto está en el lenguaje especificado por el argumento enviado.
Funciones Numéricas	
number(object)	Convierte el parámetro a un número.
sum(nodes)	Retorna la suma de los nodos enviados como parámetros (luego de convertir cada uno a un número)
floor(number)	Retorna el entero inmediatamente inferior al valor dado.
ceiling(number)	Retorna el entero inmediatamente superior al valor dado.
round(number)	Redondea el valor entero más cercano. En caso de haber dos, se elige el mayor.

Tabla 2.14: Funciones disponibles para los Predicados

2.2.5 XALAN

Xalan fue creado como parte del "Proyecto Apache XML". Xalan es un procesador XSLT de alto rendimiento que implementa completamente las recomendaciones de XSLT y XPath del W3C,

actualmente disponible en los lenguajes de Java y C++. La versión usada es Xalan-Java 2.6.0, y entre sus características tenemos:

- Se puede usar el procesador XPath independientemente (desde la línea de comando).
- Usa el Bean Scripting Framework (BSF)⁽²²⁾ para implementar la extensión de Java y scripts.
- usa Xerces-Java⁽²³⁾ como validador por omisión, pero se puede configurar por medio de parámetros globales para usar cualquier otro validador XML que implemente el API para procesamiento de XML versión 1.2.
- implementa XSLT versión 1.0
- implementa XPath versión 1.0
- implementa TrAX (API de transformación para XML), el cual ahora es parte del Java API para el procesamiento de XML versión 1.2 (JAXP)
- Xalan está construido sobre las especificaciones de SAX 2 y DOM nivel 2.

²² Bean Scripting Framework: Conjunto de clases de Java que proveen soporte de lenguajes tipo script a aplicaciones, y acceso a clases y métodos Java a lenguajes de tipo script. Ver <<http://jakarta.apache.org/bsf/>> para más información.

²³ Xerces-Java es la implementación de un validador XML hecho por la fundación Apache. Ver <<http://xml.apache.org/xerces-j/index.html>>

- Puede procesar como cadenas de entrada: Un flujo (Stream bajo Java), SAX o DOM, y como salida se puede tener también un flujo, SAX o DOM.
- Las transformaciones se pueden encadenar.
- Se puede usar desde la línea de comando para uso inmediato y sencillo.
- Se puede usar desde un servlet⁽²⁴⁾ para generar documentos XML a HTML y mostrar los resultados a los clientes.
- Soporta la creación de extensiones en Java y lenguajes de script.
- Contiene varias extensiones que implementan funcionalidades como EXSLT⁽²⁵⁾, conjunto de nodos (nodesets), múltiples formatos de documentos de salida, soporte SQL, entre otros.

2.2.6 JAXP

El Java API para Procesamiento XML (Java API for XML Processing – JAXP)⁽²⁶⁾ permite a aplicaciones acceder y manipular documentos

²⁴ Servlet: La descripción más simple de un servlet es un applet que se ejecuta en el lado del servidor. Para mayor información ver el sitio acerca de Servlets de Sun <<http://java.sun.com/products/servlet/>>

²⁵ EXSLT es una iniciativa para intentar proveer de extensiones al XLST. Ver <<http://www.exslt.org/>> para mayor información.

XML independientemente de la implementación del procesador XML usado.

La versión actual al momento de escritura es JAXP 1.2.6, la versión anterior (1.0) era previamente definida como el Java API de validación XML (Java API for XML Parsing). En esta versión se incluye un esquema de trabajo para XSLT basado en TrAX, y actualizaciones al API para dar soporte a la especificación DOM nivel 2, y SAX versión 2.0. Además también se mejoró el esquema para usar implementaciones enchufables. JAXP también define soporte para Esquemas XML y el compilador de XSLT (XSLTC).

2.2.6.1 LAS CAPAS DE CONEXIÓN

Para que las implementaciones que soporten JAXP puedan ser accedidas por un API común de Java, es necesario establecer estas capas de interconexión entre la aplicación y la implementación en particular usada. A estas capas de conexión se les llaman “Plugability Layers”.

²⁶ La página oficial de SUN para JAXP es <<http://java.sun.com/xml/jaxp/index.jsp>>

Actualmente existen tres capas de conexión, las cuales proveen acceso a SAX, DOM, y XSLT.

Capa de conexión SAX:

La aplicación se registra para recibir las notificaciones SAX por medio de la interfaz `org.xml.sax.DefaultHandler`, el validador SAX usado es accedido por medio de la interfaz `SAXParser`, y el método de obtener el validador es por medio del `SAXParserFactory`, el cual se encarga de escoger la implementación de SAX a usar según la disponibilidad y configuración de la plataforma.

Capa de Conexión DOM

Esta capa permite a aplicaciones obtener una instancia de un documento DOM que implemente la interfaz `org.w3c.dom.Document` a partir de una implementación `DocumentBuilder`, la cual encapsula la implementación del DOM.

Para obtener el `DocumentBuilder` (ya que es dependiente de la implementación del DOM) se usa una instancia de un `DocumentBuilderFactory`., el cual define que implementación de

DOM a usar según la disponibilidad de dichas implementaciones y la configuración del sistema.

Capa de conexión XSLT

Esta capa permite a aplicaciones obtener una instancia de la clase Transformer basada en una hoja de estilo XSLT específica por medio por medio de una instancia de la clase TransformerFactory.

CAPÍTULO 3

3. ANÁLISIS

3.1 ANÁLISIS DE REQUERIMIENTOS

En esta sección se realizará el análisis tanto de los requerimientos funcionales como los de rendimiento y portabilidad.

3.1.1 REQUERIMIENTOS FUNCIONALES

Se requiere un módulo de software que ofrezca una solución al problema de la lectura, interpretación y transformación de los

archivos generados por centrales de tráfico de empresas operadoras de telefonía fija.

Los requerimientos funcionales que deberá satisfacer la solución son:

- Permitir a un usuario describir flexiblemente, mediante una interfaz gráfica, el formato de los archivos generados por una central telefónica de tráfico. Esta “descripción de formato” deberá ser algún tipo de código que permita, al ejecutarse sobre uno de los archivos generados por la central, obtener un nuevo archivo con la misma información pero en un formato legible y estandarizado de acuerdo a las necesidades de la empresa.
- Una vez definido el formato este deberá poder almacenarse y editarse en cualquier momento para realizar modificaciones sobre él.
- La solución deberá soportar la definición y administración de múltiples formatos de archivos provenientes de distintas centrales de tráfico.
- Se deberá poder definir procesos de transformación, mediante los cuales se especifique entre otros parámetros el archivo que se desea interpretar, el formato al cual se adhiere la estructura de dicho archivo y el archivo a generarse como resultado.

- Una vez definido el proceso de transformación, este deberá poder ejecutarse cuando sea necesario y soportará una futura edición de sus parámetros.
- El sistema deberá permitir la ejecución concurrente de varios procesos de transformación de archivos.
- El sistema deberá ofrecer una solución al problema de automatizar el acceso a la información de los archivos generados por la central de tráfico.
- El sistema dispondrá de algún tipo de configuración que permita establecer los recursos de hardware (memoria) que se utilizarán para la ejecución de cierto proceso de transformación.

Los siguientes diagramas muestran de manera gráfica el tipo de funcionalidad que el sistema debe poseer.

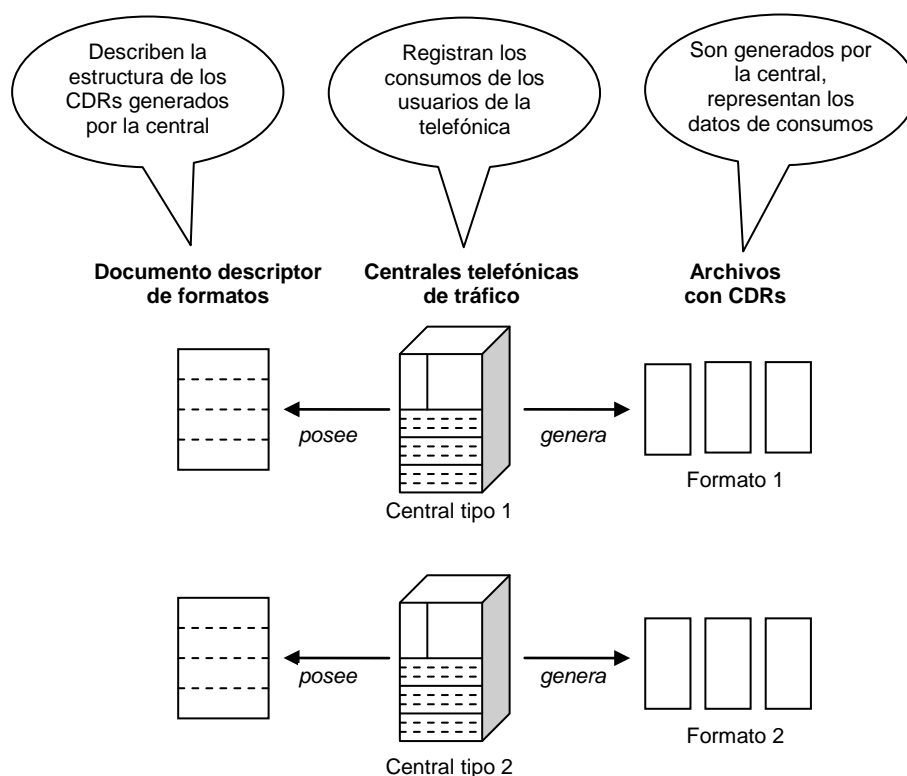


Figura 3.1: Generación de CDRs por parte de las centrales telefónicas de tráfico

La figura anterior muestra la forma en que son generados los archivos con CDRs por parte de las centrales telefónicas de tráfico. Cada central genera estos archivos con un formato diferente, para facilitar la interpretación de estos archivos, la documentación de la central incluye un “documento descriptor de formato”, el cual explica con detalle la estructura seguida por los archivos de CDRs generados por la central.

La siguiente figura ilustra como, en base a un “documento descriptor de formatos”, un administrador del sistema deberá ser capaz de generar un programa capaz de realizar la interpretación de los archivos de CDRs que se apeguen al formato especificado.

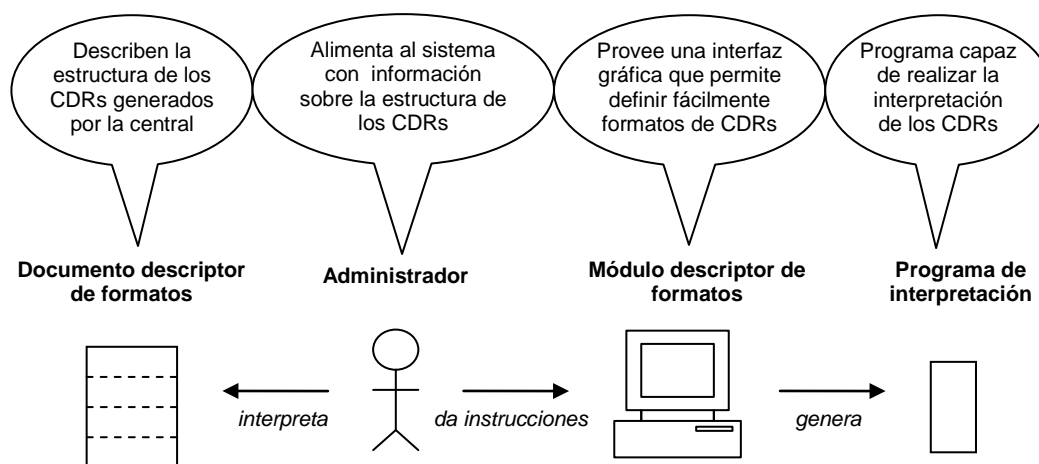


Figura 3.2: Generación del programa capaz de realizar la interpretación de un formato de CDRs

La siguiente figura muestra como un usuario del sistema deberá ser capaz de definir un “proceso de interpretación”, el cual debe contener parámetros de ejecución tales como el programa a utilizar para la interpretación, el archivo con CDRs a interpretarse y el archivo a generarse.

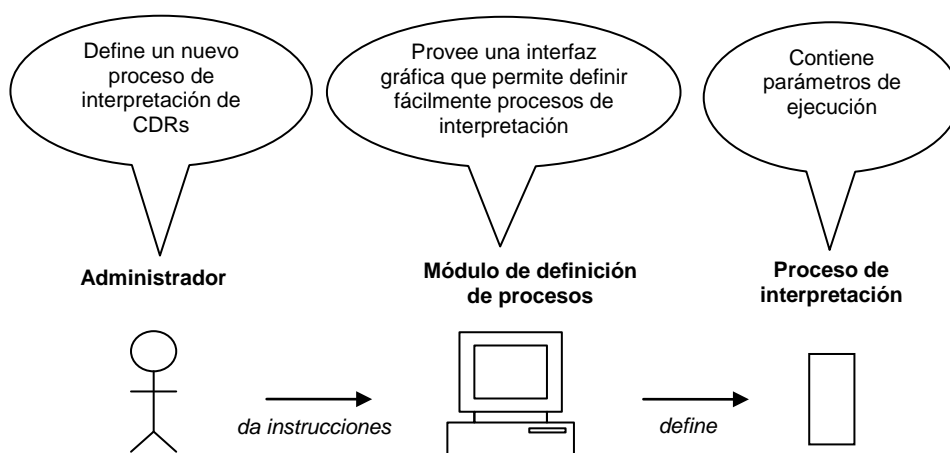


Figura 3.3: Definición de un proceso de interpretación

La siguiente figura muestra como un operador del sistema deberá poder ejecutar los procesos de interpretación previamente definidos. El resultado de esta ejecución es la generación de un archivo con CDRs en un formato adecuado a las necesidades de la telefónica.

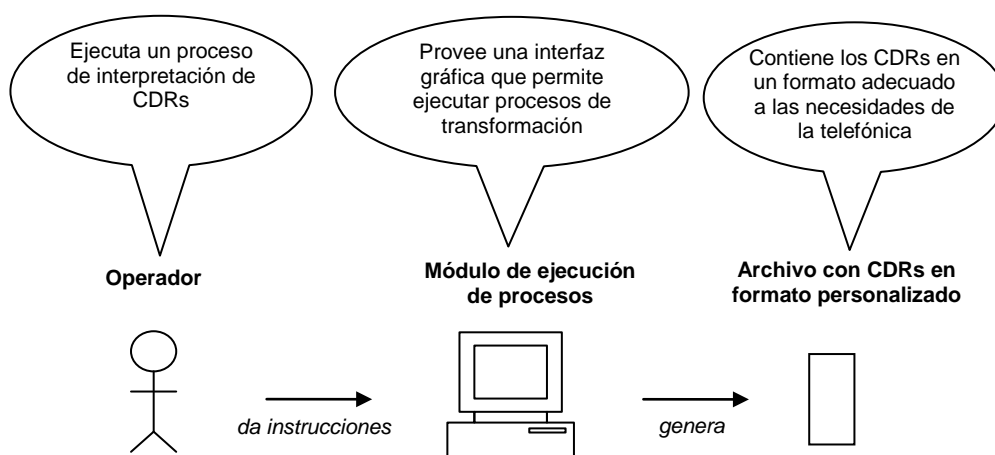


Figura 3.4: Ejecución de un proceso de interpretación

3.1.2 REQUERIMIENTOS OPERATIVOS

Los requerimientos operativos que deberá satisfacer la solución son listados en el resto de esta sección.

3.1.2.1 REQUERIMIENTOS DE EFICIENCIA

La velocidad del proceso de interpretación de datos debe estar acorde con el volumen del tráfico de llamadas registrado por la telefónica, ya que si se registran llamadas a una tasa superior a la velocidad de procesamiento del sistema, el mismo eventualmente colapsará.

Para estimar la velocidad adecuada a la que el sistema deberá ser capaz de procesar registros de llamadas, se tomó como muestra la cantidad de datos de tráfico recibidos por Pacifictel S.A. (una de las operadoras de telefonía fija con mayor número de abonados en el país). Dado que en esta telefónica el número de registros diarios que deben ser procesados es de aproximadamente 2.000.000 de CDRs

(ver tabla 3.1) y el tiempo diario invertido en el procesamiento de esta información es mayor a tres horas empleando los métodos actuales ⁽¹⁾, se tomó como base estos hechos para definir como eficiente, al inicio del proyecto, la capacidad de interpretar CDRs a una tasa de 2.000.000 de registros de llamadas en un lapso de tiempo menor a tres horas, tomando como muestra a los archivos generados por la central Huawei de Linkotel.

La siguiente tabla proporciona una muestra de la cantidad de registros diarios de llamadas que procesa una telefónica del tamaño de Pacifictel:

12 sept	13 sept	14 sept	15 sept	16 sept	17 sept	18 sept
1300869	2087459	2204608	2252299	2202068	2184827	1706732

Tabla 3.1: Cantidad de CDRs procesados por Pacifictel S.A. en la semana comprendida del domingo 12 de septiembre al sábado 18 de septiembre del 2004⁽²⁾

¹ Datos proporcionados por Yoveri S.A., empresa con más de seis años de experiencia en servicios de facturación y mediación para empresas de telefonía fija y móvil.

² Cortesía de Pacifictel S.A.

3.1.2.2 REQUERIMIENTOS DE PORTABILIDAD

Debe poder ser posible la implantación del sistema en diversas arquitecturas de hardware, así como en una gran variedad de sistemas operativos. Es importante para las telefónicas el contar con flexibilidad a la hora de tomar la decisión sobre qué equipo llevará la responsabilidad de realizar los procesos de mediación. Muchas veces existen ya equipos que realizan otro tipo de tareas, en los cuales puede desearse implantar el sistema de mediación.

También es posible que cierto tipo de centrales telefónicas tengan embebido un computador con un sistema operativo cualquiera, el cual podría servir como posible candidato para la implantación del sistema.

3.2 ANÁLISIS TÉCNICO

3.2.1 TIPOS DE FORMATOS QUE EL SISTEMA DEBE PERMITIR DEFINIR

Pese a que hay una gran cantidad de marcas y tipos de centrales telefónicas, y aunque cada una de estas genera datos de tráfico de manera diferente, en términos generales para que estas centrales sean útiles a la empresa de telefonía, deben proporcionar la misma clase de información en los archivos que generan.

Aunque es cierto que cada central telefónica genera información en un formato propietario, los cuales pueden llegar a ser sumamente diferentes al compararlos con formatos producidos por otras centrales, se observó que existe un número finito de “patrones” en los cuales las centrales se basan para estructurar la información que generan (los mismos que serán descritos más adelante en esta sección), por lo tanto puede describirse cualquier archivo complejo de tráfico, mediante el descubrimiento de dichos patrones de estructuración de datos.

Las centrales telefónicas tienen siempre un documento que describe la estructura en la cual los datos generados por estas son organizados. Este documento es proporcionado por el fabricante de la central y es útil para poder interpretar los datos que la misma genera.

La siguiente figura reproduce a manera de ejemplo un fragmento del documento que describe un registro de datos de tráfico generados por una central telefónica Huawei:

Ticket Format of OVS003B

1 Format of Detailed Ticket

No.	Field	Length (byte)	Meanings
1	Serial Number	4	Number for the tickets being generated since the switch has been started.
2	Ticket Type	1	Call type of this ticket record 0x01: Detailed ticket 0x02: DBO call record 0x03: IN record 0x05: TAX record 10xF0: Meter table ticket 0xF1: Meter table statistics 0xF2: Trunk duration statistics 0xF3: Free call statistics 0xFF: Warn ticket 0x55: Failed call ticket (Incomplete call watch ticket, Partial record ticket) <i>Note: 0x01 means that the ticket be explained by table</i>
3	Checksum	1	Byte checksum of all fields of the ticket except the first three fields.

Figura 3.5 Fragmento del documento descriptor de registros de una central Huawei ^[XIV]

Luego de examinar varios documentos de descripción de formatos de distintos tipos de centrales telefónicas, se los clasificó inicialmente de manera muy general en dos grandes grupos, los cuales son:

- Formatos de archivos binarios
- Formatos de archivos de texto

Estos grupos a su vez obedecen a los siguientes patrones de estructuración de datos:

- Campos de longitud fija.
- Campos de longitud variable.
- Campos identificados por IDs.
- Campos mixtos.
- Campos divididos por separadores.

La siguiente figura describe brevemente los tipos de formatos identificados:

I.- Campos de longitud fija.- El archivo contiene solamente información sobre los valores de los campos. ^{(xiv)(L)(U)} Ej:

dato1	Servi	dato3
-------	-------	-------

II.- Campos de longitud variable.- El archivo contiene información sobre la magnitud del campo y sus valores. ^(Lii) Ej:

tamaño	dato1	tamaño	dato2	tamaño	dato3
--------	-------	--------	-------	--------	-------

III.- Campos identificados por IDs.- El archivo contiene el ID del campo a leerse y el valor del mismo. ^(Lii) Ej:

ID1	dato1	ID2	dato2	ID3	dato3
-----	-------	-----	-------	-----	-------

IV.- Campos mixtos.- El archivo contiene el ID del campo a leerse, la magnitud del campo y el valor del mismo. ^(Lii) Ej:

ID1	tamaño	dato1	ID2	tamaño	dato2	ID3	tamaño	dato3
-----	--------	-------	-----	--------	-------	-----	--------	-------

V.- Campos divididos por separadores.- El archivo contiene solamente información sobre los valores de los campos. Estos se encuentran divididos por separadores. ^(Liii) Ej:

```
campo_1_1SEP campo_1_2SEP campo_1_3SEP
campo_2_1SEP campo_2_2SEP campo_2_3SEP
campo_3_1SEP campo_3_2SEP campo_3_3SEP
```

Figura 3.6: Descripción de los formatos usados por centrales de tráfico

En el resto de esta sección se dará una descripción para cada uno de estos patrones de datos de tráfico.

3.2.1.1 CAMPOS DE LONGITUD FIJA

En este patrón de datos, las longitudes de cada uno de los campos se conoce de antemano. Esto es gracias a la existencia de un documento que describe con detalle toda la información necesaria para la interpretación del archivo generado por la central telefónica, lo que incluye la magnitud de cada uno de sus campos. En este patrón, si un campo necesita menos caracteres que los que tiene asignados, usa típicamente caracteres de relleno o “padding” ^[XIV][L][L].

El siguiente ejemplo muestra un cierto grupo de bytes, usando notación hexadecimal, que forman parte de un archivo generado por una central Alcatel. El número total de bytes mostrados es de siete, y corresponden a los bytes situados en la posición cinco hasta la once en un registro de llamadas. La documentación de la central Alcatel indica que un registro de llamadas tiene una longitud fija de 32 bytes, los bytes en las posiciones cinco, seis y siete representan la hora en la que se realizó la llamada y los bytes situados en las posiciones de la ocho hasta la once, representan el número que realizó la llamada telefónica con su código de área incluido:

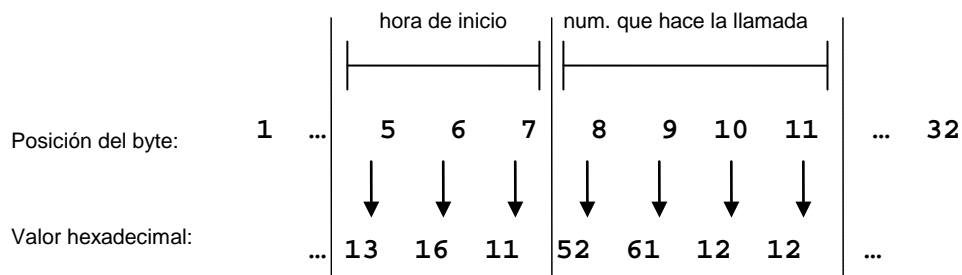


Figura 3.7: Campos de longitud fija en un registro de una central Alcatel

En este ejemplo puede observarse como la posición en la que es leído un byte es importante para poder determinar qué campo representa. De esta manera observamos que este registro fue leído a las 13:16:11 (HH:mm:SS), y el número que realizó la llamada tiene el valor 5 como código de área y 2611212 como número telefónico.

3.2.1.2 CAMPOS DE LONGITUD VARIABLE

Cada campo perteneciente a un registro de llamadas posee valores al inicio del mismo que definen su longitud. De esta manera puede trabajarse con campos de longitud variable.

Decimos entonces que en este patrón la longitud de los campos es dinámica, ya que no es conocida de antemano y se encuentra definida como los n primeros bits de cada campo ^[L1].

3.2.1.3 CAMPOS IDENTIFICADOS POR ID

En este patrón el orden de los campos no sigue ninguna estructura previamente definida, para poder determinar qué campo se está leyendo es necesario leer una determinada cantidad de bits, los cuales servirán como identificador del campo a procesarse ^[LIII].

El siguiente ejemplo muestra un grupo de bytes usando notación hexadecimal, que forman parte de un archivo generado por una central Siemens. La documentación de la central Siemens indica que cada campo es identificado por un código hexadecimal de 8 bits. El identificador 64 indica que el campo representa la fecha y la hora en la cual se realizó la llamada, este campo tiene una longitud fija de seis bytes:

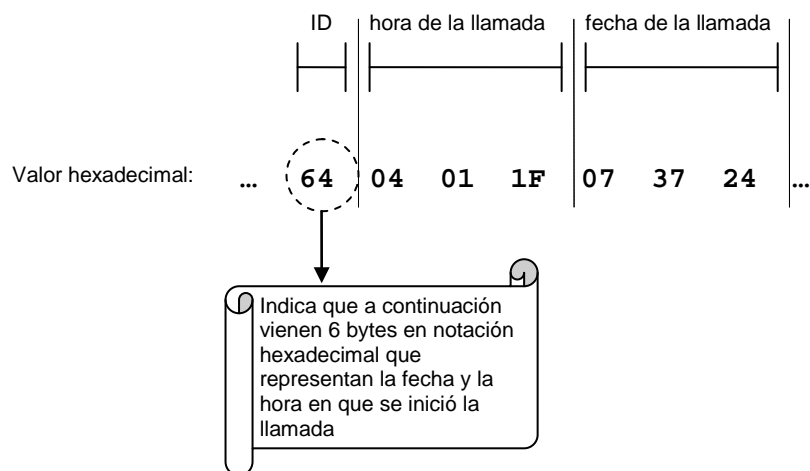


Figura 3.8: Campos identificados por IDs en un registro de una central Siemens

Como muestra la figura, en este tipo de campos es necesario leer primero una cantidad determinada de bits para saber qué campo es el que se está interpretando. La documentación de la central determina de antemano la longitud de cada campo, por lo que una vez reconocido su identificador, se procede a leer tantos bytes como sean necesarios.

3.2.1.4 CAMPOS MIXTOS

Estos formatos son una combinación de varios otros, típicamente están constituidos por campos identificados por IDs y cuya magnitud es variable. Para este tipo de campos deben aplicarse al mismo tiempo varias de las consideraciones explicadas para los formatos

anteriores. Es posible también que existan registros que posean más de un tipo de campo ^[LII].

El siguiente ejemplo muestra un grupo de bytes usando notación hexadecimal, que forman parte de un archivo generado por una central Siemens. La documentación de la central Siemens indica que cada campo es identificado por un código hexadecimal de 8 bits. El identificador 11 representa el número telefónico que realizó la llamada, la longitud de este campo no es conocida de antemano y para obtenerla deberá realizarse nuevamente una lectura de 8 bits, la unidad de la magnitud está dada en nibbles ⁽³⁾:

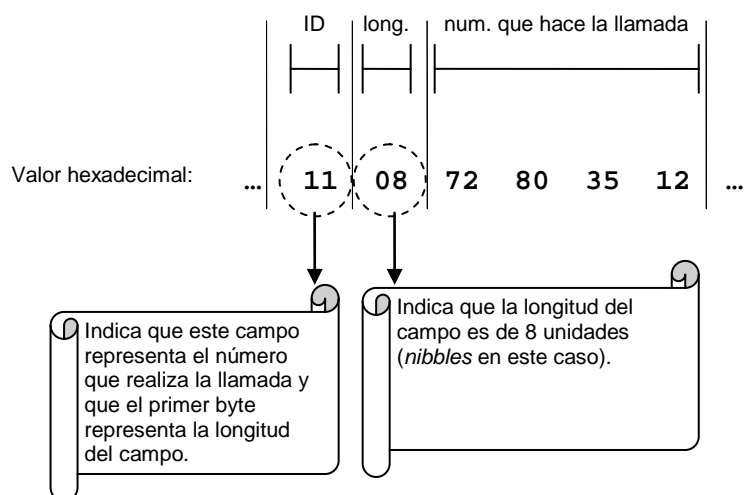


Figura 3.9: Campos mixtos en un registro de una central Siemens

³ Un nibble equivale a cuatro bits.

Como muestra la figura, una vez identificado que el ID del campo es el valor hexadecimal de 11, se procederá a leer otro byte que indicará la magnitud del mismo, como en esta lectura se obtuvo el valor de 8 y la unidad de la magnitud está dada en nibbles de acuerdo al documento descriptor de formatos de la central, se procederán a leer 4 bytes con la información del campo.

3.2.1.5 CAMPOS DIVIDIDOS POR SEPARADORES

Un símbolo predefinido actúa como separador entre los distintos campos. De manera similar a los campos de longitud fija, existe un documento que determina el orden en el cual están siendo leídos los campos. Es común en este patrón la existencia también de un “separador de registros”, el cual determina que se ha terminado de leer el registro actual y está por comenzar uno nuevo. Típicamente este es el carácter de nueva línea para archivos basados en texto [LIII].

3.2.2 TIPOS DE PROCESOS QUE EL SISTEMA DEBE PERMITIR DEFINIR Y EJECUTAR

Además de permitir especificar formatos para cada uno de los tipos de archivos que se desean interpretar con el sistema, el usuario debe también poder definir procesos de transformación, los cuales deben ser lo suficientemente parametrizables para automatizar gran parte del trabajo involucrado en la manipulación del archivo con los datos de tráfico.

A continuación se listan las consideraciones a tomar en cuenta para realizar la definición de procesos de transformación:

- **Definir archivos de entrada y de salida.-** Los archivos de entrada son aquellos generados por la central telefónica y los de salida son los producidos por el sistema al realizar la interpretación.
- **Seleccionar el formato de los archivos a ser interpretados.-** Un proceso de carga deberá estar asociado con un documento de descripción de formatos, el cual conoce como llevar a cabo la lectura e interpretación de los datos del archivo origen.

- **Elegir el número de ejecuciones, así como el intervalo de tiempo entre las mismas.-** Si la central telefónica está configurada para añadir sucesivamente nueva información a archivos previamente generados, debe ser posible la configuración de procesos de lectura en tiempo real, es decir, a medida que la central incrementa el tamaño del archivo generado. Para ello, se configura un proceso que deberá ejecutarse entre intervalos de tiempo iguales a los que la central se toma entre los incrementos de su archivo; el número de veces que se llevará a cabo la ejecución del proceso, depende del número de incrementos al archivo que la central está configurada para realizar.
- **Ingresar el momento en el que se iniciará el proceso de transformación.-** Debe ser posible la configuración de un espacio de tiempo que el proceso de transformación esperará antes de iniciar su ejecución.
- **Ingresar el número de registros que desean ser interpretados del archivo origen.-** Debe ser posible determinar cuántos registros deben ser leídos del archivo origen, o bien indicar que se desea realizar una lectura de todos los registros hasta llegar al final del archivo.

3.2.3 ANÁLISIS DEL ENTORNO DE EJECUCIÓN DEL SISTEMA EN LINKOTEL S.A.

El objetivo de esta sección es comprender bajo qué entorno deberá llevarse a cabo la implantación del sistema, es decir, la identificación de cuáles son los diferentes tipos de hardware y sistemas operativos con los que se interactuará en un ambiente de producción. La siguiente figura muestra el entorno de ejecución para Linkotel S.A:

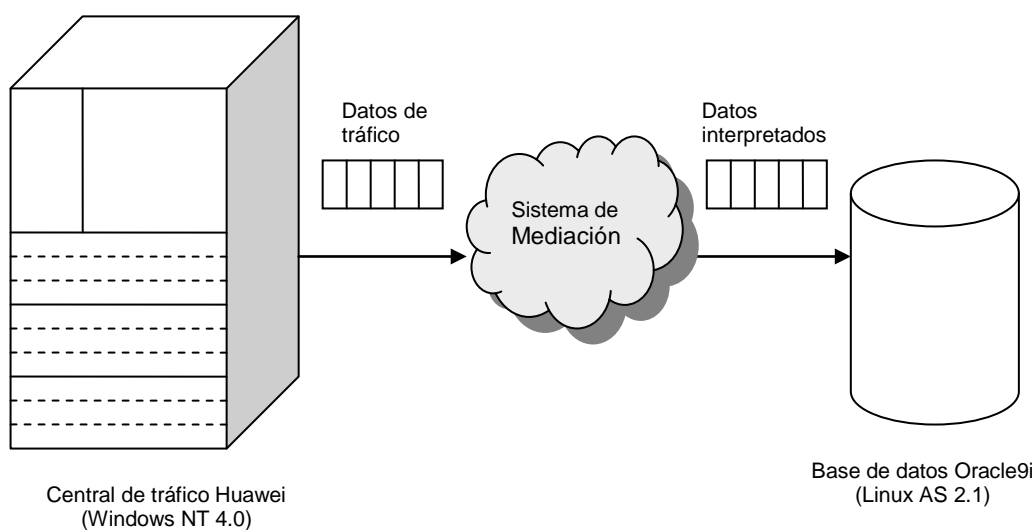


Figura 3.10: Ambiente de ejecución del sistema en Linkotel S.A.

Como muestra la figura anterior, la central Huawei de Linkotel puede ser considerada muy similar a un computador con Windows NT 4.0, lo que significa que los archivos generados por esta podrán ser

compartidos y accedidos mediante los medios comunes de compartición de archivos de Windows. La base de datos de la empresa, un motor oracle9i, se ejecuta en un equipo con Linux AS 2.1.

Dada esta arquitectura, será responsabilidad del sistema de mediación el obtener los datos generados en la central Huawei, interpretarlos y transformarlos en información útil y legible, para que posteriormente la misma pueda ser cargada en la base de datos de la empresa, asegurando de esta manera su disponibilidad a otras aplicaciones.

3.3 USUARIOS DEL SISTEMA

Se han definido dos perfiles de usuarios que accederán al sistema, estos son:

- Usuarios administradores.
- Usuarios operadores.

En el resto de esta sección se realizará una descripción de las tareas que deberán llevar a cabo estos usuarios con la ayuda del sistema a implementarse.

3.3.1 USUARIOS ADMINISTRADORES

Los usuarios administradores son responsables del mantenimiento en general de los parámetros de configuración del sistema, deberán realizar tareas tales como:

- **Definición de nuevos formatos.-** Mediante esta actividad se define la estructura y significado de los archivos generados por una central de tráfico adquirida por la operadora telefónica.
- **Mantenimiento de formatos ya definidos.-** Se realizan modificaciones o correcciones a previas definiciones de formatos de datos.
- **Definición de procesos de carga.-** Se definen procesos responsables de realizar la transformación de datos, indicando parámetros tales como: archivo de origen, archivo destino y el

formato adecuado para la interpretación de los campos presentes en el archivo.

- **Revisión y mantenimiento de los logs generales del sistema.-**
La revisión periódica de estos archivos permitirá evaluar el funcionamiento del sistema, así como determinar si es necesario realizar un ajuste en cuanto a los parámetros de ejecución de los procesos para acelerar el rendimiento global.
- **Configuración de los parámetros generales del sistema.-** En esta actividad se definen los parámetros administrativos del sistema, tales como: valores por defecto para los procesos de transformación, nivel de advertencia del log de errores, directorios a utilizar, etc.

3.3.2 USUARIOS OPERADORES

Los usuarios operadores son aquellos que llevarán a cabo las operaciones cotidianas del sistema, tales como:

- **Ejecución de procesos de carga.-** Los usuarios operadores del sistema son los encargados de iniciar los procesos de carga diseñados por el administrador cuando sea necesario hacerlo, debido a la llegada de nuevos datos de uso producidos por las centrales de tráfico.
- **Revisión de los logs de ejecución.-** Por cada proceso de interpretación ejecutado se genera un log asociado, el cual deberá ser revisado por el usuario operador del sistema para verificar si hubieron advertencias o errores en el proceso de carga. Es también su responsabilidad el reportar cualquier novedad o problema indicado en los mismos.

3.4 ESPECIFICACIÓN DE LOS CASOS DE USO DEL SISTEMA

En esta sección se tratan los casos de uso del sistema. Considerando que la implementación del sistema será realizada en una herramienta orientada a objetos, se va a usar para gran parte del análisis y diseño, el Lenguaje de Modelamiento Unificado o UML (por sus siglas en inglés). Mediante este lenguaje podremos especificar, construir y documentar sistemas implementados con lenguajes orientados a objetos.

En UML, el modelo de casos de Usos es una herramienta que describe la funcionalidad de un sistema. Definimos un caso de uso como la representación de la interacción entre un usuario y el sistema.

Los actores son definidos como los usuarios del sistema y pueden ser personas, máquinas u otros sistemas. Los actores participan en los casos de uso cuando realizan un trabajo significativo para el sistema.

Definimos a los escenarios como descripciones formales del posible flujo de eventos que pueden ocurrir durante un caso de uso en particular.

A continuación, las siguientes figuras identifican a los actores primarios y los secundarios del sistema, así como las metas de los actores primarios:

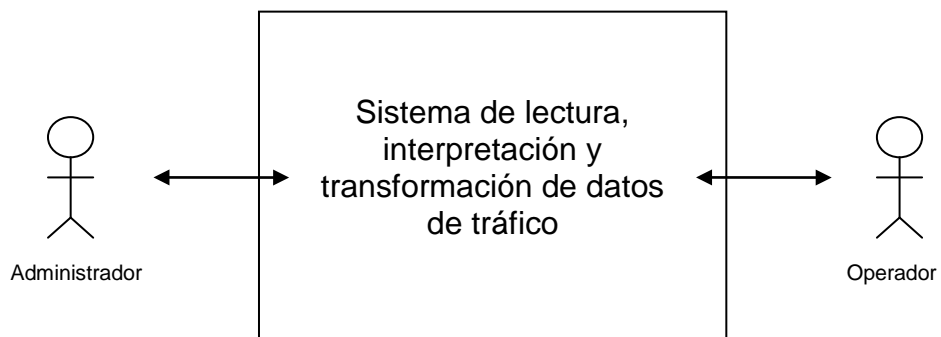


Figura 3.11: Identificación de los actores primarios del sistema

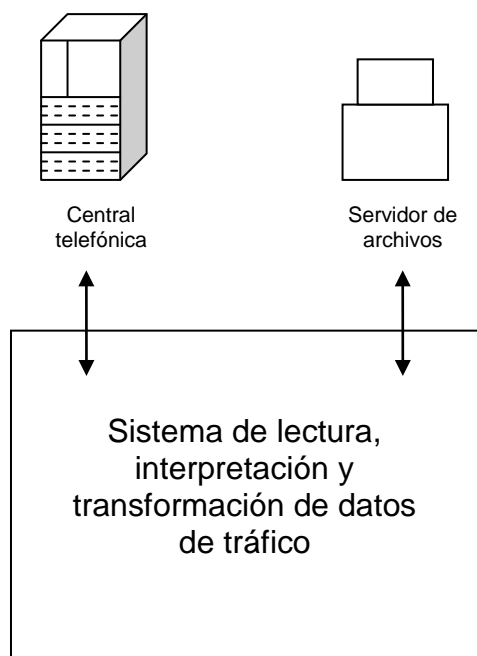


Figura 3.12: Identificación de los actores secundarios del sistema

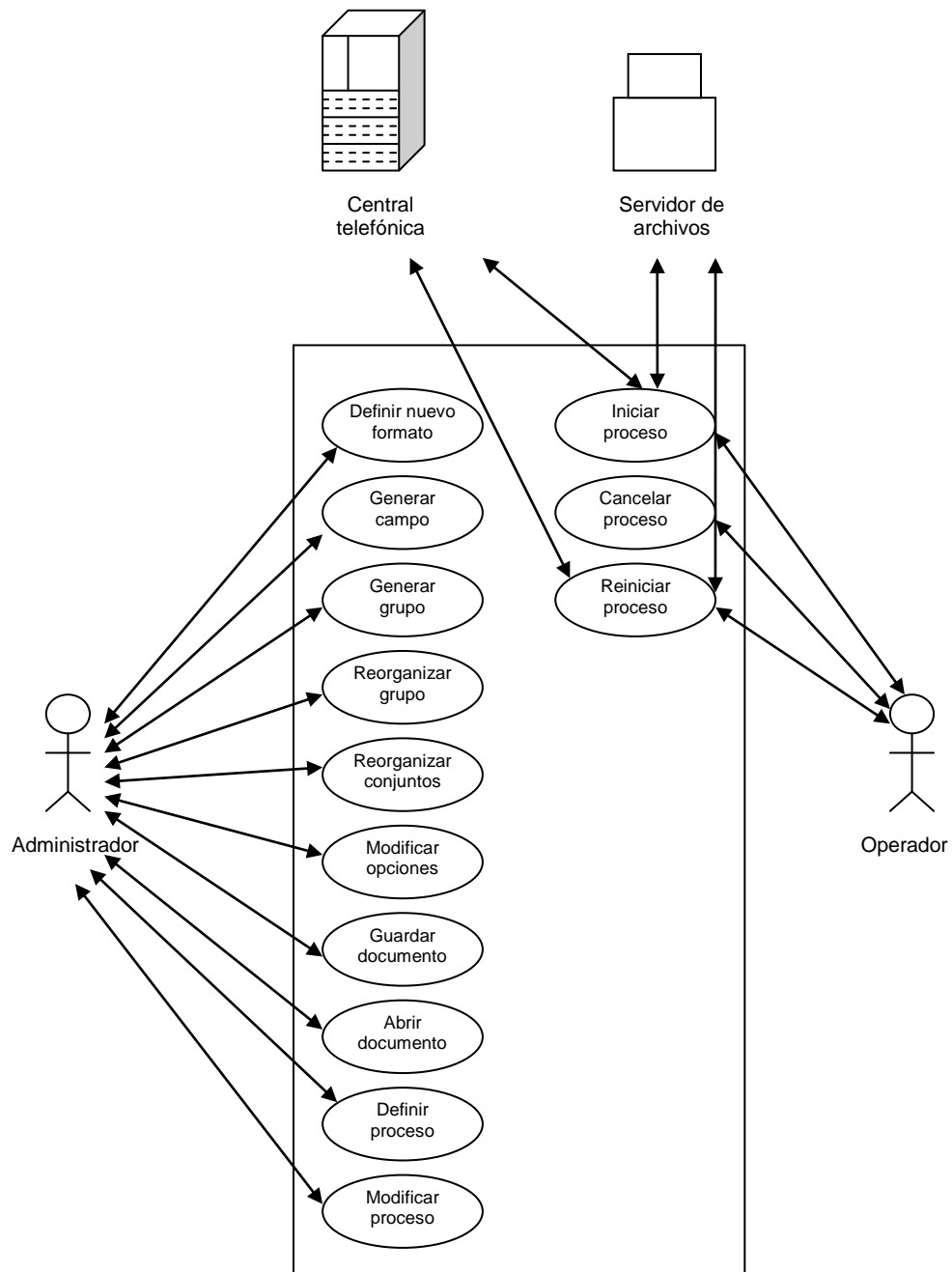


Figura 3.13: Identificación de las metas de los actores primarios

A continuación se listan los casos de uso que se han considerado en el sistema.

- Usuario administrador define un nuevo formato.
- Usuario administrador genera un campo.
- Usuario administrador genera un grupo.
- Usuario administrador reorganiza un grupo de campos.
- Usuario administrador reorganiza un conjunto de grupos.
- Usuario administrador modifica las opciones del formato actual.
- Usuario administrador guarda un documento.
- Usuario administrador abre un documento.
- Usuario administrador define un proceso.
- Usuario administrador modifica un proceso.
- Usuario operador inicia un proceso.
- Usuario operador cancela un proceso.
- Usuario operador reinicia un proceso.

En el resto de esta sección se describirán brevemente los casos de uso listados, para ver la descripción completa de los casos de uso y todos sus escenarios referirse al apéndice B:

Nombre:**1. Usuario administrador define un nuevo formato.****Descripción:**

El usuario genera un nuevo formato de entre los formatos disponibles. Puede escoger entre los tipos de formatos reconocidos por el sistema (de estructura fija, campos identificados por IDs, o de texto utilizando separadores).

Nombre:**2. Usuario administrador genera un campo.****Descripción:**

El usuario genera y llena un nuevo campo dentro del descriptor de formatos actualmente abierto. Los campos se crean con la mínima información posible ya llenada.

Nombre:**3. Usuario administrador genera un Grupo.****Descripción:**

Se genera un nuevo grupo de campos dentro de la definición de datos existente.

Nombre:**4. Usuario administrador reorganiza un grupo de campos.****Descripción:**

El usuario reorganiza el orden de algún campo dentro del grupo de campos actuales.

Nombre:**5. Usuario administrador reorganiza un conjunto de grupos.****Descripción:**

El usuario decide mover uno de los grupos entre los demás grupos del mismo nivel.

Nombre:**6. Usuario administrador modifica las opciones del formato actual.****Descripción:**

El usuario modifica las opciones generales del documento actual.

Nombre:**7. Usuario administrador guarda un documento.****Descripción:**

El usuario decide guardar el documento actual. No se permite guardarlo cuando este tiene algún campo o grupo marcado como inválido.

Nombre:**8. Usuario administrador abre un documento.****Descripción:**

El usuario intenta abrir un archivo de extensión XML que aparentemente es un archivo de definición de formatos. La estructura e integridad del archivo es verificado usando el XSD correspondiente.

Nombre:**9. Usuario administrador define un proceso****Descripción:**

El usuario define un nuevo proceso para ser ejecutado posteriormente. Este tendrá los valores por omisión que han sido configurados para el sistema.

Nombre:**10. Usuario administrador modifica un proceso****Descripción:**

El usuario abre las propiedades de un formato y lo modifica.

Nombre:**11. Usuario Operador inicia un proceso****Descripción:**

El usuario intenta iniciar uno o más procesos que tenga seleccionados.

Nombre:**12. Usuario Operador cancela un proceso****Descripción:**

El usuario intenta cancelar uno o más de los procesos en la tabla de ejecución.

Nombre:**13. Usuario Operador reinicia un proceso.****Descripción:**

El usuario intenta reiniciar uno o más procesos de los que se están ejecutando.

CAPÍTULO 4

4. DISEÑO

4.1 DISEÑO ARQUITECTÓNICO DEL SISTEMA

En esta sección se describe el diseño y la arquitectura del sistema de lectura, interpretación y transformación de datos generados por centrales telefónicas de tráfico implementado como parte de este proyecto de tesis.

4.1.1 ELECCIÓN DE LAS TECNOLOGÍAS Y LENGUAJES DE PROGRAMACIÓN ADECUADOS PARA SATISFACER LOS REQUERIMIENTOS

Para la implementación de este proyecto de tesis, se necesitaba escoger entre las tecnologías que permitieran desarrollar una solución que cumpliera con los requerimientos especificados en el capítulo 3. Entre estos requerimientos, los que más influyen en el momento de elegir las tecnologías a usarse son los de:

- Flexibilidad en la representación de formatos complejos de archivos de tráfico.
- Capacidad de ejecutar múltiples procesos de transformación en paralelo.
- Portabilidad de la aplicación entre plataformas.
- Eficiencia en la transformación de formatos.

Para cumplir con el primer requerimiento, que planteaba la elección de una tecnología de representación y definición de información compleja, se determinó que la opción adecuada es el lenguaje de marcas extendido, o XML. Para una revisión completa sobre esta

tecnología, su potencialidad y objetivos, referirse al capítulo 2 de este documento.

Para cumplir con los demás requerimientos, que plantean en cambio la necesidad de resolver problemas relativos a concurrencia, portabilidad y eficiencia de procesamiento, se determinó que los principales lenguajes candidatos, por las características de los problemas que debían resolver, eran Java y C++.

A continuación, la siguiente tabla describe como afrontan estos lenguajes los problemas que deberán ser resueltos en el proyecto:

	Java	C++
Concurrencia	Sencilla, implementada como parte inherente del lenguaje.	Compleja, implementada con librerías y semáforos.
Portabilidad	Alta, no es necesario la recompilación de código.	Baja, es necesario la recompilación de código entre plataformas distintas.
Eficiencia (velocidad de ejecución)	Menos eficiente que C++.	Altamente eficiente.
Administración de memoria	Automática. Existe un proceso recolector de basura.	Manual, más eficiente, pero pueden deslizarse bugs con facilidad.
Ciclo de desarrollo y mantenimiento	Más corto que el de C++.	Más largo que la mayor parte de otros lenguajes de tercera generación.

Tabla 4.1: Cuadro comparativo entre los lenguajes Java y C++

Java maneja mucho mejor los temas relacionados con concurrencia, portabilidad, administración automática de memoria del sistema y lo

referente al ciclo de desarrollo y mantenimiento de software. C++ en cambio es uno de los lenguajes de tercera generación más eficientes que existen en cuanto a velocidad de ejecución. Ambos son herramientas orientadas a objetos sumamente potentes.

Para realizar este proyecto de tesis se ha elegido trabajar con Java, ya que se ha determinado que la mayor parte de los requerimientos son resueltos de mejor manera por este lenguaje. En cuanto a la eficiencia, la velocidad de ejecución solicitada en los requerimientos operativos para los procesos de transformación de datos, está en un rango que permite que sea alcanzada con facilidad por este lenguaje (ver sección 3.1.2.1).

4.1.2 DISEÑO DE LA IMPLANTACIÓN DEL SISTEMA EN LINKOTEL S.A.

A continuación se muestra un diagrama que representa una solución aplicable a Linkotel S.A. para los procesos de interpretación de los datos generados por su central telefónica (referirse también a la figura 3.10: *Ambiente de ejecución del sistema en Linkotel S.A.*):

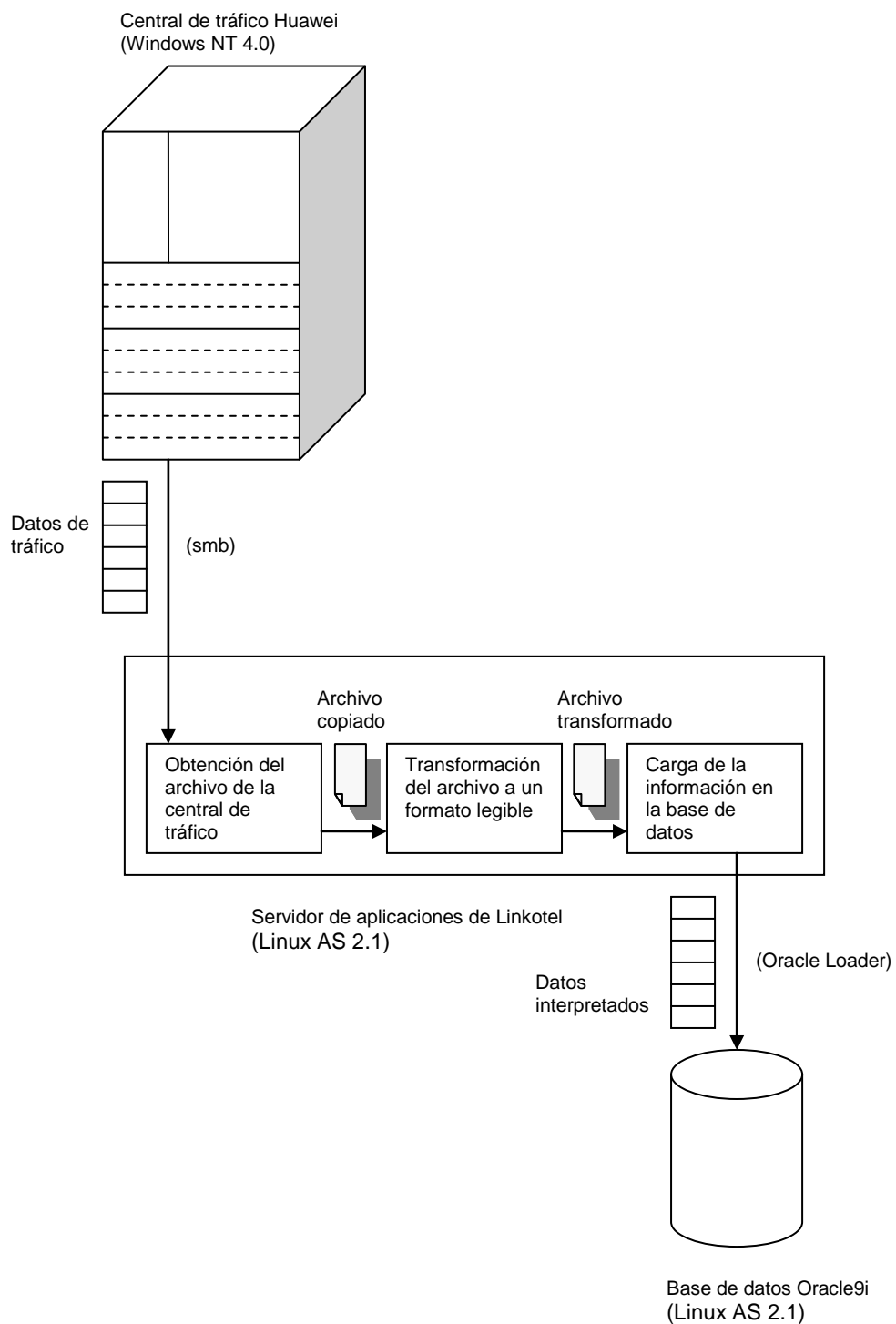


Figura 4.1: Implantación del sistema en Linkotel S.A.

Como se explicó en el capítulo 3, la central de tráfico Huawei, dada su arquitectura, puede ser considerada de manera similar a un computador con el sistema operativo Windows NT 4.0. El equipo en donde se ejecutarán los procesos de mediación en Linkotel, es un servidor con el sistema operativo Linux AS 2.1, el cual actúa principalmente como servidor de aplicaciones de la empresa.

La figura anterior muestra el siguiente flujo de información:

- La central de tráfico genera un archivo de consumos, en un directorio al cual se le da acceso compartido mediante los mecanismos tradicionales de compartición de archivos de Windows NT.
- Un proceso configurado en el sistema de mediación para realizar la transformación del archivo generado por la central en la etapa anterior, lo copia al servidor local. Debido a que este es un sistema operativo Linux, habrá que verificar que el kernel del mismo esté configurado para soportar el “montaje” de directorios compartidos ubicados en otra máquina con Windows, para que de esta manera pueda accederse de manera transparente al archivo generado por la central. En este caso el protocolo involucrado en el transporte de

archivos sobre la red es smb, el cual es el protocolo de compartición de recursos usado por Windows.

- El proceso de mediación, una vez verificada la copia de la información desde la central, realiza una invocación al módulo encargado de realizar la transformación de formatos sobre un archivo, de esta manera se generará un nuevo archivo fácilmente interpretable por los módulos o subsistemas siguientes.
- Una vez notificado que la información ha sido transformada al formato elegido, el operador ejecuta sobre el archivo generado un comando del Oracle Loader, el cual es una herramienta altamente eficiente para realizar la carga de información a bases de datos Oracle. De esta manera la información está ya fácilmente disponible para las aplicaciones que la requieran.

4.2 DISEÑO DE LOS MÓDULOS DEL SISTEMA

Esta sección describe el diseño de los distintos módulos que componen el sistema implementado como parte de este proyecto de tesis, estos son los siguientes:

- **Módulo de definición de formatos.-** Responsable de permitir la creación de las definiciones de los formatos empleados por las centrales de tráfico.
- **Módulo de definición y ejecución de procesos.-** Responsable de permitir definir los distintos procesos que deberán ser ejecutados por el sistema, así como de la interacción que el usuario realiza sobre estos procesos.
- **Módulo de interpretación y transformación de datos.-** Responsable de realizar la lectura, interpretación y transformación de datos de tráfico, en base a los procesos definidos con el módulo anterior.

A lo largo de las siguientes subsecciones se realizará una descripción detallada del diseño seguido por cada uno de estos módulos. Se utilizarán como recursos diagramas de clases así como la descripción de los diagramas de secuencia más importantes y representativos. Para examinar todos los diagramas de clases y los diagramas de secuencia de cada uno de los escenarios del sistema, referirse al apéndice C y D respectivamente.

4.2.1 MÓDULO DE DEFINICIÓN DE FORMATOS

Uno de los aspectos más importantes del sistema, es su capacidad de especificar formatos complejos que representen las estructuras de los archivos generados por las centrales telefónicas de tráfico. Es necesario definir alguna forma de almacenamiento para estas “especificaciones de formatos”, de manera que pueda dárseles mantenimiento cuando sea necesario hacerlo.

Los principales puntos que deben poder ser especificados en este documento de definición de formatos, son los siguientes:

- **Propiedades generales del documento.-** Estas propiedades representan los metadatos de todo el documento. Incluyen parámetros tales como el tipo de codificación utilizada para la lectura de campos de texto, existencia de separadores entre registros, comentarios que describen el documento, etc.
- **Distribución de los campos en los registros.-** Estos datos proporcionan información sobre cómo se encuentran distribuidos los campos en los archivos generados por la central de tráfico, es decir,

si se apegan o no a algún orden específico, y la lógica subyacente a ese orden cuando este existe.

- **Información sobre la estructura interna de los campos.-**

Representan información sobre cómo deben ser accedidos cada uno de los campos presentes en el archivo generado por la central, qué interpretación debe hacerse de la lectura y cómo debe ser esta traducida a una representación fácilmente comprensible.

Como se explicó en el inicio de este capítulo, se ha empleado XML para especificar los archivos de descripción de formatos producidos por el sistema. Para describir la estructura de estos archivos XML se ha utilizado XSD, la cual es una tecnología de especificación de esquemas de documentos XML para su validación. Para una revisión completa de la tecnología XSD y sus alcances, referirse al capítulo 2 de este documento.

En el resto de esta sección se detallará la información que deben comprender cada uno de los elementos del documento de definición de formatos. Se emplearán sucesivamente ilustraciones que representan gráficamente fragmentos de la especificación XSD que está siendo descrita. Para revisar la descripción de la representación

gráfica utilizada para describir documentos XSD en este proyecto de tesis, referirse al apéndice G.

4.2.1.1 ARQUITECTURA GENERAL DEL DOCUMENTO DE DESCRIPCIÓN DE FORMATOS

La siguiente figura representa gráficamente y a un nivel general la especificación XSD de un documento descriptor de formatos. Para revisar la especificación XSD formal de este documento, referirse al apéndice E:

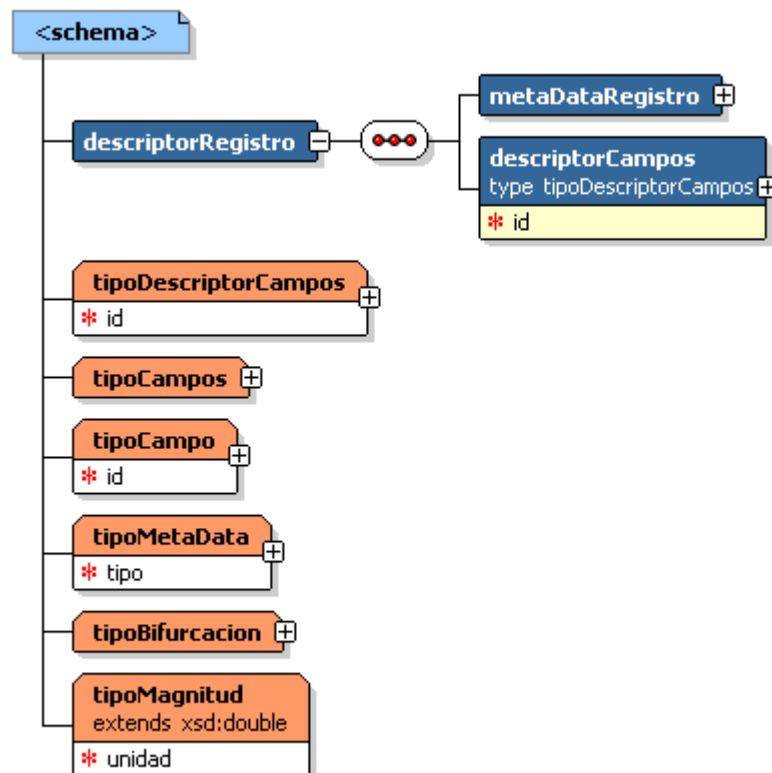



Figura 4.2: Esquema general del documento de definición de formatos

El elemento principal o raíz es un objeto llamado “*descriptorRegistro*”, el símbolo  indica que este elemento está constituido por una secuencia de otros dos elementos, los cuales son: “*metaDataRegistro*” y “*descriptorCampos*”.

A continuación se proporciona una descripción de cada uno de los elementos mostrados en el gráfico:

- **metaDataRegistro.-** Contiene la información sobre las propiedades globales y generales del documento, esto es, los metadatos del mismo.
- **descriptorCampos.-** Describe la estructura física de los campos que integran el formato que está siendo descrito. En términos generales, define tanto las estructuras secuenciales como las condicionales de distribución de campos. Posee también un atributo “*id*” el cual proporciona un código lógico de identificación para este elemento.
- **tipoDescriptorCampos, tipoCampos, tipoCampo, tipoMetaData, tipoBifurcacion y tipoMagnitud.-** Representan definiciones modulares de tipos de datos que son utilizados a lo largo de todo el documento.

4.2.1.2 PROPIEDADES DEL DOCUMENTO

La siguiente figura muestra una lista de elementos englobados por el elemento “*metaDataRegistro*”, el cual como vimos en la sección

anterior representa a los metadatos del documento de descripción de formatos.

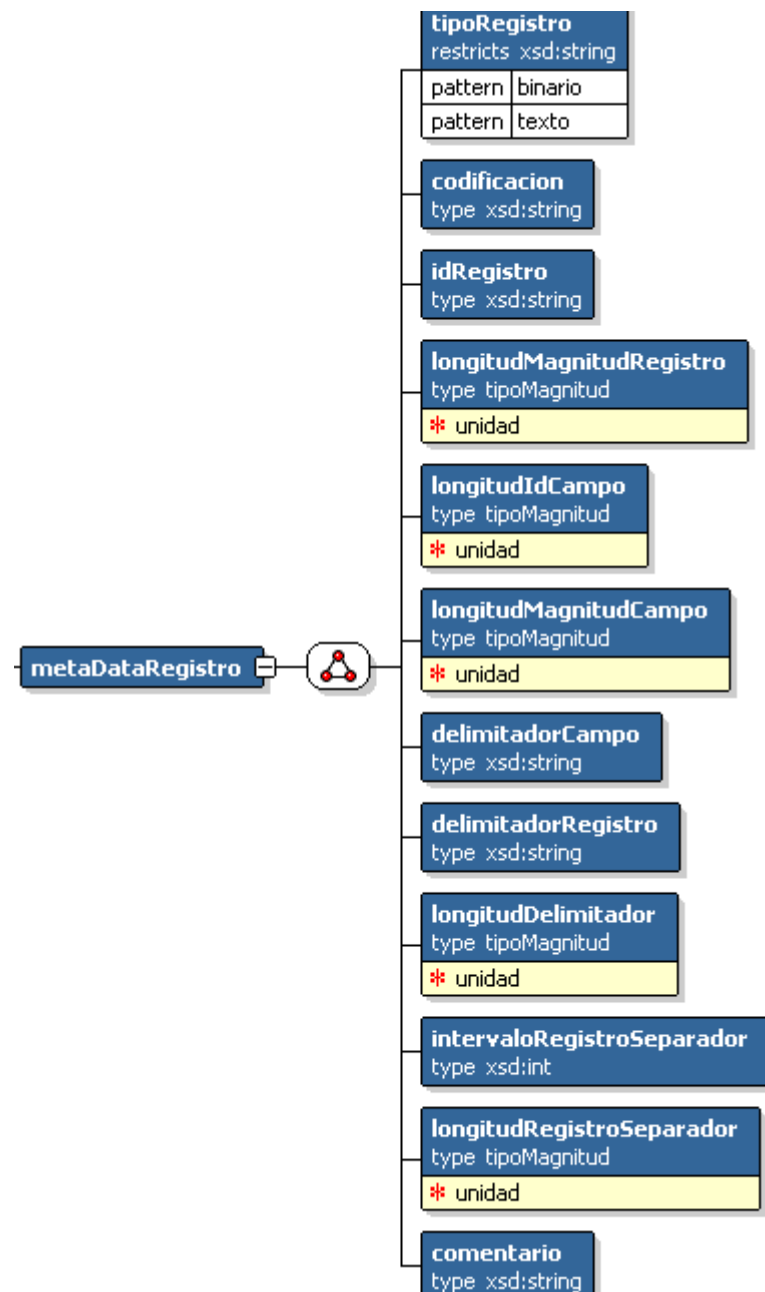



Figura 4.3: Propiedades del documento

El símbolo  indica que “*metaDataRegistro*” está constituido por cierto número de elementos, los mismos que pueden aparecer en cualquier orden dentro del mismo.

A continuación se realizará una descripción de la información que representan cada uno de estos elementos:

- **tipoRegistro.-** Tipo de archivo que será interpretado con este descriptor de formatos. Puede tomar los valores de “binario” o “texto”.
- **codificacion.-** Codificación a la que se adhieren los campos de texto presentes en el archivo generado por la central, por ej: UTF8, UTF16, ascii, etc.
- **idRegistro.-** Secuencia de caracteres que al ser leídos indican el inicio de un nuevo registro.
- **longitudMagnitudRegistro.-** Representan el número de unidades que deben ser leídas para obtener la magnitud del registro que se está procesando.

- **longitudIdCampo.-** Representan el número de unidades que deben ser leídas para obtener el ID del campo que se está procesando.
- **longitudMagnitudCampo.-** Representan el número de unidades que deben ser leídas para obtener la magnitud del campo que se está procesando.
- **delimitadorCampo.-** Indica la existencia de separadores entre los campos.
- **delimitadorRegistro.-** Indica la existencia de separadores entre los registros.
- **longitudDelimitador.-** Indica la magnitud de los separadores, en caso de que estos existan.
- **intervaloRegistroSeparador.-** Indica la existencia de un relleno de bytes cada cierto número de registros leídos.
- **longitudRegistroSeparador.-** Indica la magnitud de los rellenos de bytes en caso de que estos existan.

- **comentario.-** Representa una descripción breve sobre el tipo de formato a ser descrito y su central de tráfico asociada.

4.2.1.3 DISTRIBUCIÓN DE LOS CAMPOS EN LOS REGISTROS

Esta sección tiene como objetivo el describir los elementos que definen la distribución de los campos dentro del registro de datos al cual pertenecen.

La siguiente figura ilustra la composición de un elemento “*descriptorCampos*”:

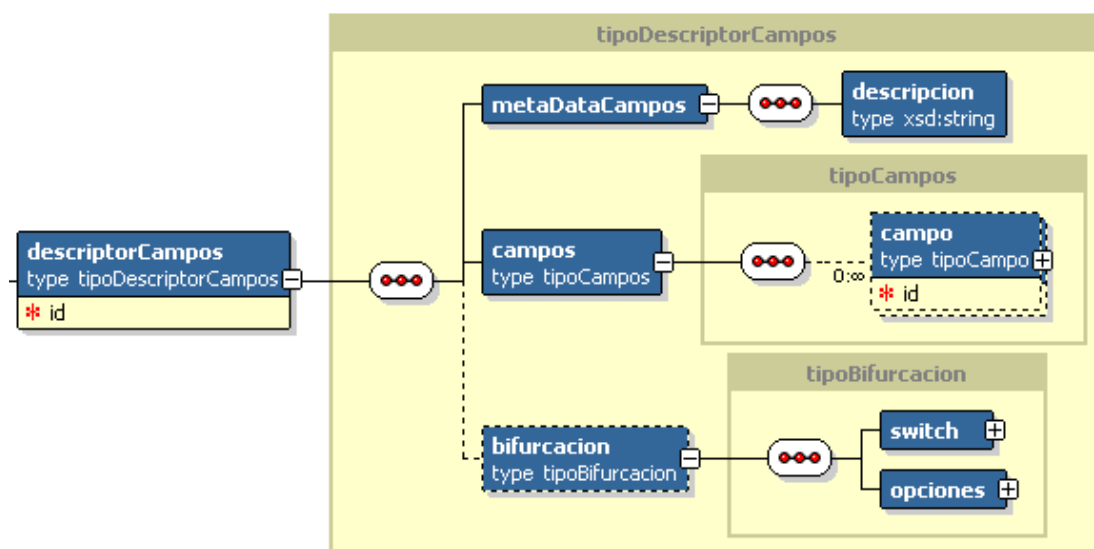


Figura 4.4: Estructura de los registros generados por la central

El elemento “*descriptorCampos*” es una secuencia de tres elementos, los cuales son: “*metaDataCampos*”, “*campos*” y “*bifurcación*”.

A continuación se proporciona una descripción de cada uno de los elementos mostrados en el gráfico:

- **metaDataCampos.-** Representa los metadatos del descriptor de campos. Dentro de él se incluye información relevante a este grupo de campos descritos.
- **descripción.-** Provee de un nombre lógico al objeto *descriptorCampos*.
- **campos.-** Representa una lista de elementos “campo” asociados con este *descriptorCampos*, los cuales deben ser leídos de forma secuencial.
- **bifurcación.-** Este elemento es opcional y permite la implementación de estructuras condicionales, esto es, un tipo de estructura que no es conocida de antemano, sino que debe ser definida a medida que se va realizando la lectura de los campos del archivo generado por la central.

Como acabamos de ver, las estructuras condicionales están implementadas con los elementos “*switch*” y “*opciones*”, la siguiente figura es una representación gráfica de dichos elementos y sus hijos:

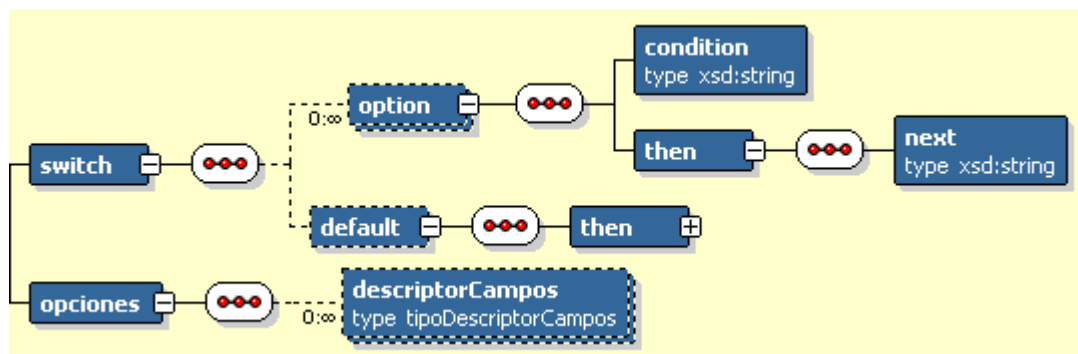


Figura 4.5: Manejo de estructuras condicionales

A continuación se dan descripciones de todos los elementos mostrados en la figura:

- **switch.-** Define estructuras condicionales mediante la especificación de varios elementos “*opciones*”, los cuales a su vez definen “*condiciones*” y “*caminos*”, en el caso de que una de las opciones cumpla con su condición, deberá ejecutarse el camino asociado a esta condición.

- **option.-** Representa exactamente una condición y su camino asociado. La condición es representada por el elemento "*condition*" y el camino a seguir por el elemento "*then*".
- **condition.-** Representa una expresión booleana que deberá ser evaluada en el contexto de los datos leídos de la información proporcionada por la central. El resultado de esta evaluación determinará si la opción asociada es válida.
- **then.-** Representa el grupo de elementos que deben ser examinados en el caso de que la condición asociada sea cierta.
- **next.-** Representa un salto a un nodo definido dentro del elemento "*opciones*".
- **default.-** Se define como una opción sin condición. Es el camino por omisión si ninguna de las condiciones definidas en las opciones previas a este elemento se cumplen.
- **opciones.-** Define los distintos caminos posibles que puede seguir la estructura física del archivo a interpretarse. Esto lo hace especificando múltiples elementos de tipo "*descriptorCampos*", cada

uno de los cuales representa un posible camino de distribución de campos a seguir.

4.2.1.4 INFORMACIÓN SOBRE LA ESTRUCTURA INTERNA DE LOS CAMPOS

Esta sección tiene como objetivo el describir los elementos que definen la estructura interna de cada campo, es decir, si son leídos o bien calculados en base a los valores de otros campos. Si son leídos cuáles son sus parámetros de lectura, cómo la información obtenida debe ser interpretada y finalmente cómo debe ser presentada al operador del sistema.

La siguiente figura ilustra la composición de un elemento “*campo*”:

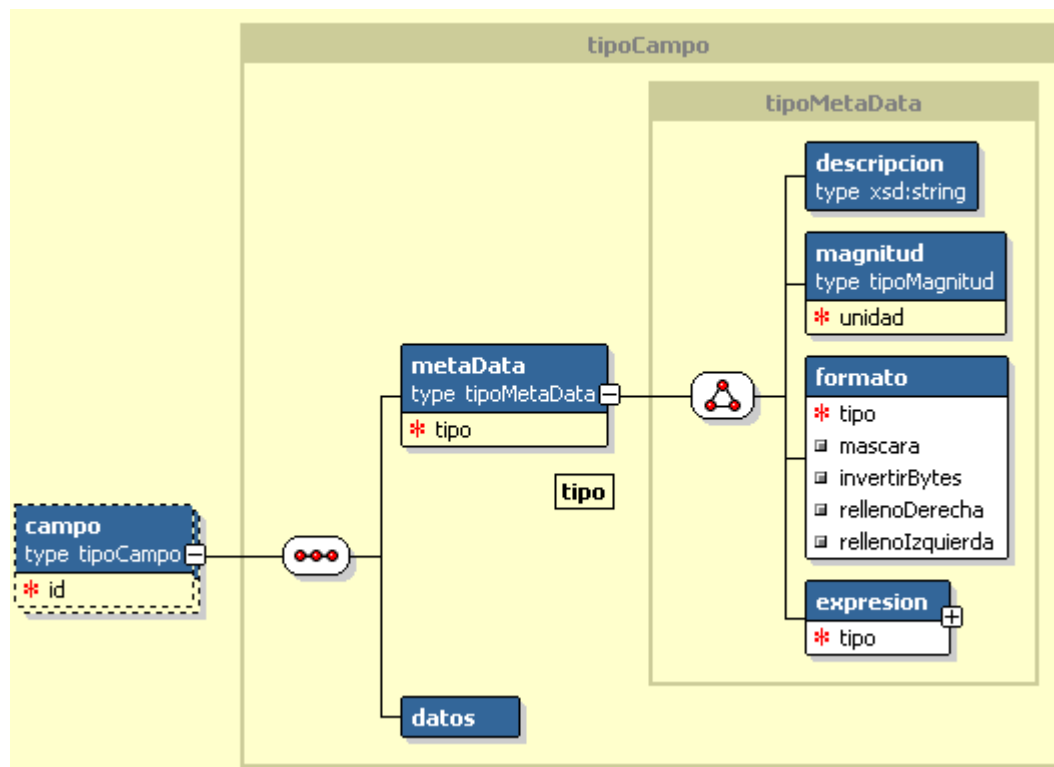


Figura 4.6: Estructura interna de los campos

El elemento “*campo*” es una secuencia de dos elementos, los cuales son: “*metaData*”, y “*datos*”. Adicionalmente este elemento posee un atributo “*id*”, el cual representa un identificador único para el campo dentro del documento de definición de formatos. Cualquier referencia a este campo se hará en base a su ID. A continuación se proporciona una descripción de cada uno de los elementos mostrados en el gráfico:

- **metaData.-** Este elemento encapsula toda la información valiosa para realizar una interpretación adecuada del campo en base a los

datos generados por la central de tráfico. Dentro de él se definen cuatro elementos: “*descripcion*”, “*magnitud*”, “*formato*” y “*expresion*”. Adicionalmente este elemento posee un atributo “*tipo*”, el cual determina si el campo debe ser leído o bien si debe ser calculado en base a valores de otros campos previamente procesados.

- **descripcion.-** Este elemento proporciona una breve descripción sobre qué datos están siendo representados por el campo.
- **magnitud.-** Representa la magnitud del campo, es decir cuantas unidades deben ser leídas para que pueda ser procesado. Las unidades están definidas por el atributo “*unidad*”, el cual puede tomar los valores de: byte, nibble, bit y char.
- **formato.-** Sus atributos proporcionan información sobre cómo debe ser interpretado el campo una vez ya leído de su origen. El atributo “*tipo*” es el nombre que identifica al formato en el sistema, por ejemplo: “date” o “int”. El atributo “*mascara*” determina como debe ser “enmascarada” la lectura una vez realizada, esto es útil para expresiones que representan fechas. Los atributos “*rellenoDerecha*” y “*rellenoIzquierda*” se aplican a todo campo que haya sido definido como poseedor de relleno o “padding”.

- **expresión.-** Este elemento encapsula la información necesaria para interpretar el campo en el caso de que no sea leído, sino calculado en base a los valores de otros campos. También puede representar expresiones que sirvan como valor por omisión del campo en el caso de que la lectura de datos no arroje ningún valor válido.
- **datos.-** Este elemento se encuentra reservado para uso posterior.

A continuación, la siguiente figura ilustra la definición de expresiones con las que pueden estar asociados los campos:

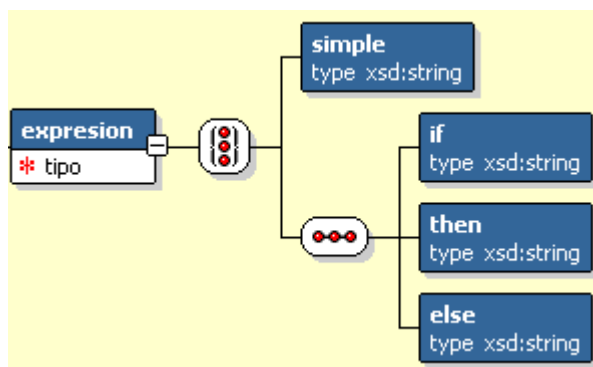



Figura 4.7: Definición de expresiones

El símbolo  indica que un elemento expresión posee o bien un elemento “*simple*”, o bien una secuencia de los siguientes tres elementos: “*if*”, “*then*”, “*else*”. Adicionalmente un atributo “*tipo*”, indica

que la expresión debe ser considerada una expresión simple, o bien una condicional.

A continuación se muestra una descripción de todos los elementos comprendidos en el elemento “*expresión*”:

- **simple.**- Representa una expresión sencilla que debe evaluarse en el contexto de los datos leídos de la central.
- **if.**- Representa una condición booleana que debe ser evaluada en el contexto de los datos leídos de la central.
- **then.**- Si la expresión representada por el elemento “if” es verdadera, entonces el valor del campo es la evaluación de la expresión definida en este elemento, en el contexto de los datos leídos de la central.
- **else.**- Si la expresión representada por el elemento “if” es falsa, entonces el valor del campo es la evaluación de la expresión definida en este elemento, en el contexto de los datos leídos de la central.

4.2.1.5 CREACIÓN Y EDICIÓN DEL DOCUMENTO DESCRIPTOR DE FORMATOS

Como se dijo anteriormente, el objetivo principal de este módulo es el de proveer una interfaz de usuario que permita la creación y edición de los elementos hasta aquí listados, como parte de un documento de definición de formatos. El siguiente diagrama de clases muestra las relaciones estáticas entre los objetos que permiten llevar a cabo esta tarea:

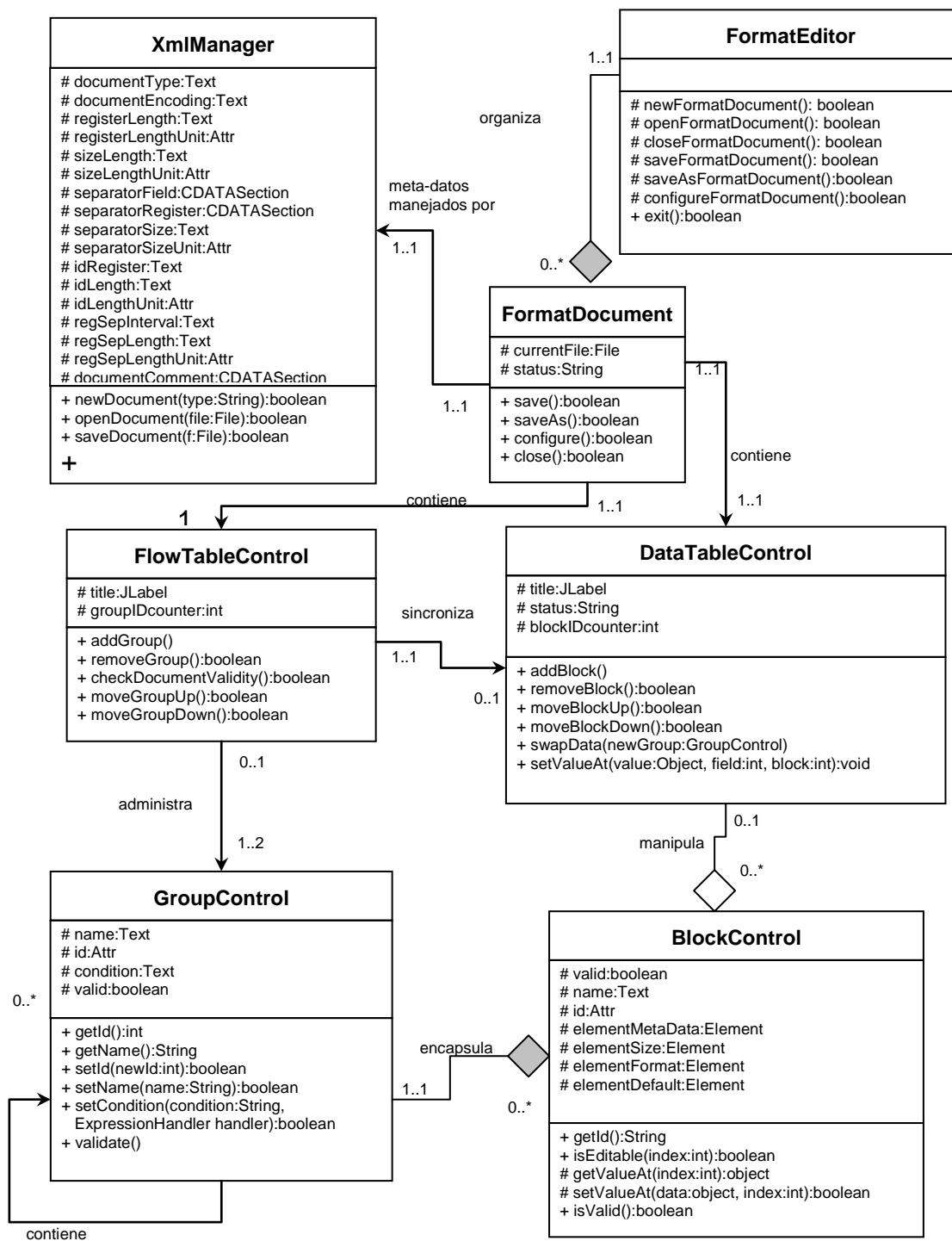


Figura 4.8: Diagrama de clases del módulo de Definición de Formatos

FormatEditor es la clase encargada de administrar los diferentes formatos que se van a editar. Se encarga de pasar a los diferentes formatos los comandos del usuario de alto nivel como guardar, deshacer (y rehacer), cerrar, etc. *FormatDocument* es el manejador de cada una de las instancias de definición de formatos abiertos. Cada instancia de esta clase encapsula a un documento único, de tal manera que en el constructor del mismo se define si va a representar a un documento nuevo o va a abrir uno ya existente. *FormatDocument* en sí no hace mucho trabajo, básicamente delega tareas a las otras clases componentes del mismo. *FormatDocument* también contiene a los elementos de interfaz de la región del documento, incluyendo a las barras de herramientas para controlar a los campos y grupos; *FormatDocument* simplemente redirecciona dichas peticiones a las clases correspondientes.

FormatDocument administra a tres clases principales:

- **XmlManager.**- Maneja el acceso al archivo XML asociado al documento, así como también contiene a las propiedades de ámbito “global” del documento. Esta es la clase encargada de hacer la conversión de un XML a DOM y viceversa (parte del proceso de abrir y guardar documentos). Los parámetros globales del

documento sólo están contenidos en esta clase, la presentación de dichos parámetros para que el usuario los pueda modificar queda delegado otra clase (llamada *Options*).

- **FlowTableControl.-** Clase que administra el árbol de grupos. Se encarga de realizar las operaciones como reordenamiento de nodos, insertar y remover nodos, cambiar los datos del grupo (nombre, expresión), etc. Cada uno de los grupos del formato está encapsulado en la clase *GroupControl*.
- **DataTableControl.-** Esta clase muestra los contenidos del grupo seleccionado. El *GroupControl* seleccionado contiene varios *BlockControl*, cada uno de ellos representa a un campo del formato, y *DataTableControl* los presenta como columnas en una tabla. Esta clase se encarga de las operaciones que manipulan a los campos tales como su reordenamiento, eliminación o creación. Las operaciones de modificación de los campos en sí son pasadas al *BlockControl* en cuestión. Como la presentación de la tabla depende del tipo de formato con el que se está trabajando, existen diferentes clases que heredan a *DataTableControl* para representar los datos bajo cada uno de los tipos de formato.

En la descripción de las dos clases anteriores, se mencionaron a *GroupControl* y *BlockControl*. *GroupControl* simplemente representa a un grupo de campos, el cual contiene los datos relevantes al mismo junto con un vector de clases *BlockControl*. El *GroupControl* también es un nodo para árboles Swing, e implementa toda la funcionalidad del mismo. *BlockControl* es la super-clase para los campos. Cada diferente subclase de *BlockControl* se encarga de administrar los datos individuales de su tipo. Como la presentación al usuario es por medio de tablas, la identificación de los campos es realizada por el índice de la fila en la cual aparece el campo, de esta manera el paso de información se vuelve transparente para las demás clases que simplemente pasan los mensajes entre la interfaz y los campos.

A continuación se muestra como las clases relacionadas con grupos interaccionan entre ellas cuando se desea mover un grupo después del siguiente:

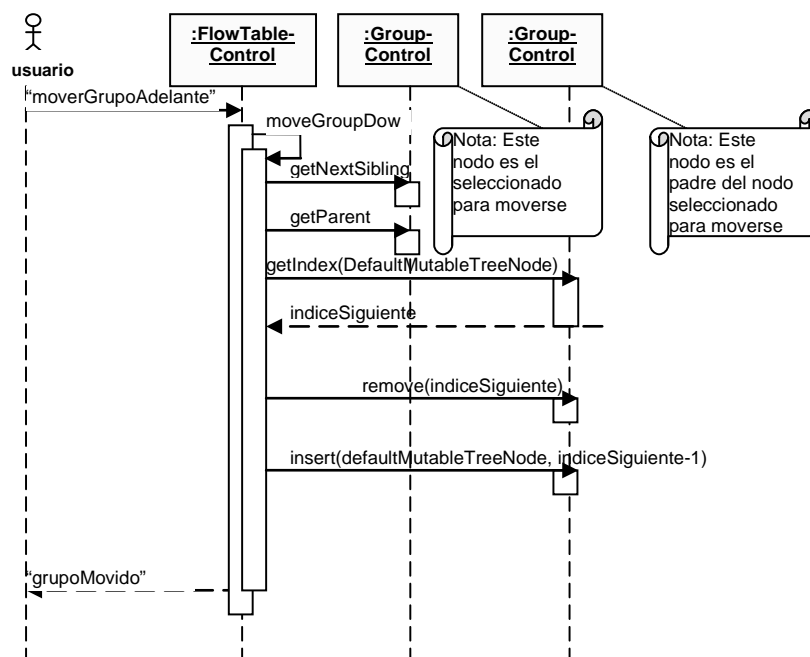


Figura 4.9: Usuario mueve grupo después del siguiente

Aquí se muestra como la instrucción llega a *FlowTableControl*, quien localiza al nodo actual y a su padre para hacer el intercambio. Por medio de los métodos para remover e insertar nodos es que se reordenan los grupos en cuestión.

En el siguiente diagrama se muestra la relación de las clases involucradas en la modificación directa de campos:

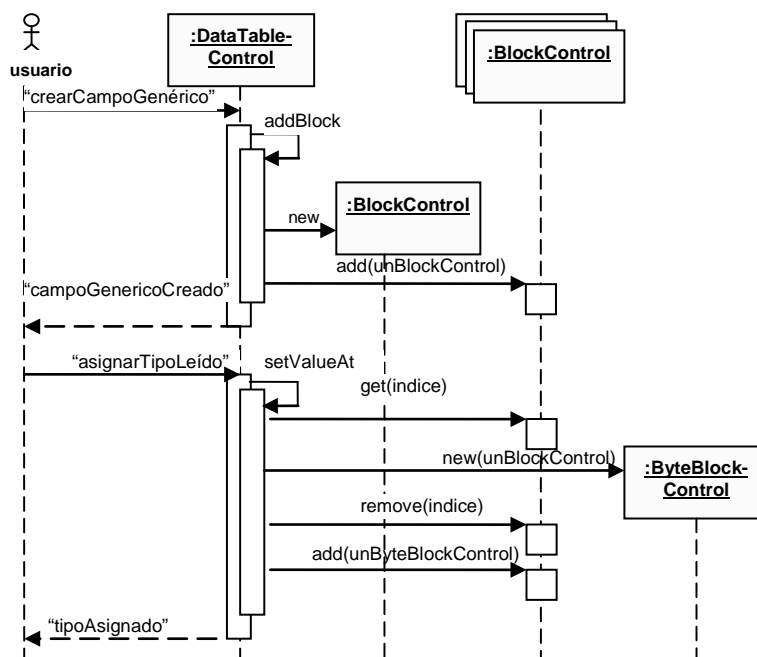


Figura 4.10: Usuario crea un campo de tipo leído

Se puede ver como el mensaje del usuario llega directamente a *DataTableControl* (ya que esta clase contiene a la barra de herramientas correspondiente), como esta clase genera un nuevo bloque y lo agrega al arreglo. Posteriormente, el usuario cambia el tipo del campo generado, ya que existe una clase heredera de cada tipo de campo, es necesario instanciar al nuevo tipo de campo, y en cuyo constructor se pasa el campo anterior, para lograr preservar la mayor cantidad de campos posibles.

4.2.2 MÓDULO DE DEFINICIÓN Y EJECUCIÓN DE PROCESOS

Una vez definida la estructura del documento de definición de formatos, se procedió a realizar el diseño de un módulo que permita la especificación y administración de procesos de interpretación de datos.

Mientras la especificación del documento descriptor de formatos responde preguntas tales como de qué manera la información sobre un formato será almacenada de modo que pueda ser utilizada y editada posteriormente, el diseño de los procesos de interpretación responde las preguntas sobre cuándo iniciará el proceso, qué archivos serán interpretados, cuáles archivos deben ser generados y otros parámetros adicionales. La siguiente tabla lista los parámetros requeridos para definir un proceso de interpretación de datos de tráfico:

Nombre del parámetro	Descripción
<i>Nombre</i>	El nombre del proceso puede ser cualquier cadena de texto, su finalidad es proporcionar una breve descripción sobre el proceso configurado.
<i>Momento de Inicio</i>	Establece un intervalo de tiempo antes de iniciar la ejecución (por ejemplo, se puede fijar un proceso para que empiece una hora después de que el usuario solicita su arranque).
<i>Intervalo entre Ejecuciones</i>	Se puede definir que un proceso se ejecute varias veces. Este valor indica cuánto tiempo esperar antes de continuar con la siguiente iteración de la ejecución.
<i>Número de Ejecuciones</i>	Representa el número de veces que el proceso debe ejecutarse antes de ser nominado como "terminado".
<i>Offset Inicial</i>	No siempre se desea que el procesamiento del archivo sea desde el principio (por ejemplo cuando ya se sabe que cierta parte del archivo ya fue procesado con anterioridad), aquí se especifica desde donde se debe empezar a leer el archivo de origen.
<i>Número de Registros</i>	Especifica cuantos registros se desean leer del archivo de origen. El valor -1 indica que hay que leer todos los registros hasta el final del archivo.
<i>Buffer de Lectura</i>	En casos en que el archivo sea muy grande como para tenerlo todo en memoria, es recomendable procesar el archivo en partes, el buffer de lectura especifica que tan grande son los trozos de archivo que se procesan a la vez. Mientras menor el tamaño del buffer de lectura, más tiempo demorará el procesamiento de los archivos.
<i>Formato</i>	Especifica el archivo descriptor de formatos a usarse para este proceso.
<i>Reporteador</i>	Aquí se escoge de entre los reporteadores disponibles para definir el formato de salida de los archivos.
<i>Archivo de Origen</i>	Especifica el archivo del cual extraer los datos al momento de ejecutarse.
<i>Archivo Destino</i>	El nombre del archivo que se generará con este proceso.
<i>Política a usar si el archivo destino existe</i>	Determina como el proceso reaccionará en el caso de que el archivo destino ya exista, su valores pueden ser: <i>Sobreescribirlo:</i> El archivo destino será eliminado y se creará uno nuevo. <i>Anexar:</i> Los nuevos datos serán agregados al final del archivo existente. <i>Abortar:</i> El proceso será cancelado.
<i>Directorio de Trabajo</i>	Si se define un directorio de trabajo, el archivo de origen será copiado a este antes de procesarlo. Esta opción es útil para casos en que la disponibilidad al archivo de origen no sea garantizada (como el acceso al archivo a través de una red).

Tabla 4.2: Parámetros de definición de procesos

Es responsabilidad de este módulo proveer de una interfaz de usuario que permita la especificación de los parámetros listados, así como la ejecución de ciertas acciones sobre los procesos de interpretación una vez que hayan sido definidos. La siguiente tabla lista dichas acciones:

Nombre de la acción	Descripción
<i>Iniciar</i>	El proceso se pone en marcha. Si está definido para esperar cierta cantidad de tiempo lo hará antes de buscar al archivo de tráfico y empezar a interpretarlo.
<i>Detener</i>	El proceso es detenido y presenta un reporte de los registros procesados o cualquier evento ocurrido durante el procesamiento mientras se encontraba activo.
<i>Reiniciar</i>	El proceso es detenido, una vez hecho esto es iniciado nuevamente de manera automática.

Tabla 4.3: Acciones a realizar sobre los procesos definidos por el sistema

Este módulo es responsable de interpretar los comandos emitidos por el usuario cuando este desea realizar una de las acciones descritas en la tabla anterior. La ejecución de las acciones requeridas será delegada al módulo de interpretación y transformación de datos y sus respuestas serán mostradas como retroalimentación al usuario. El siguiente diagrama ilustra este proceso:

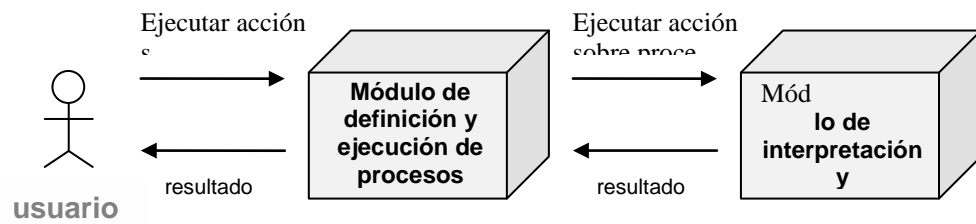


Figura 4.11: Ejecución de acciones sobre procesos

A continuación, el siguiente diagrama de clases muestra las relaciones estáticas entre los objetos que conforman a este módulo:

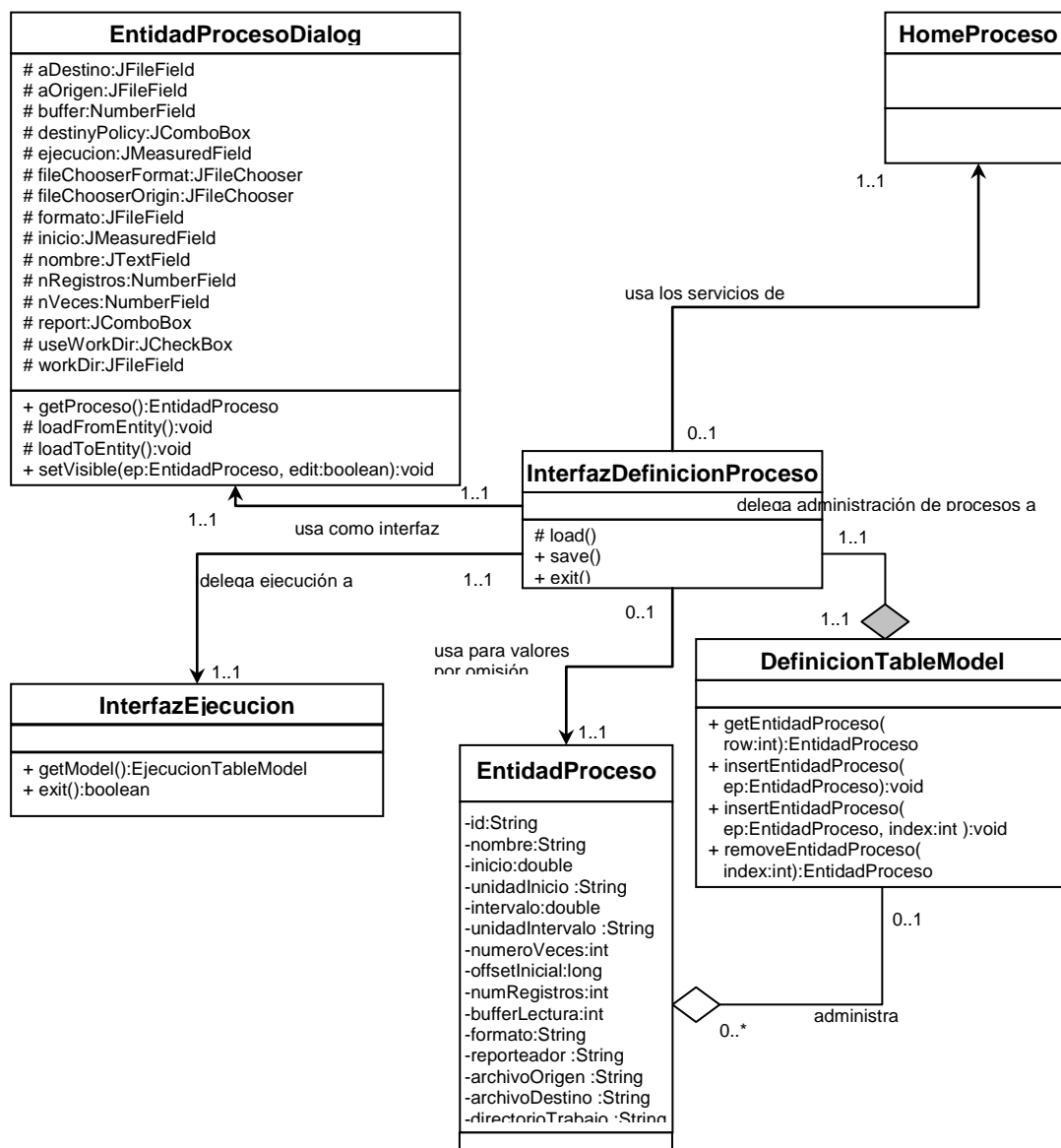


Figura 4.12: Diagrama de clases del módulo de definición y ejecución de procesos – Sección Definición

Este diagrama muestra la relación existente entre las clases al momento de definir procesos. *InterfazDefinicionProceso* es la clase principal, la cual encapsula a la interfaz del usuario para dicha definición de procesos. *DefinicionTableModel* es la clase encargada de

manejar el modelo de datos de la tabla donde aparecen todos los procesos, y es esta clase la que contiene al arreglo de procesos definidos así como los métodos para acceder y manipularlos. *EntidadProcesoDialog* es una clase que encapsula a un diálogo Swing, por medio del cual se puede modificar a un proceso a la vez.

HomeProceso permite al acceso de funciones de sistema relacionados con procesos, como la función para cargar los procesos, y la función para removerlos o actualizarlos. *EntidadProceso* es quien encapsula a cada proceso junto con todos sus campos. Finalmente, se mantiene un enlace con *InterfazEjecucion*, a quién se delega la ejecución de procesos cuando así lo pide un usuario.

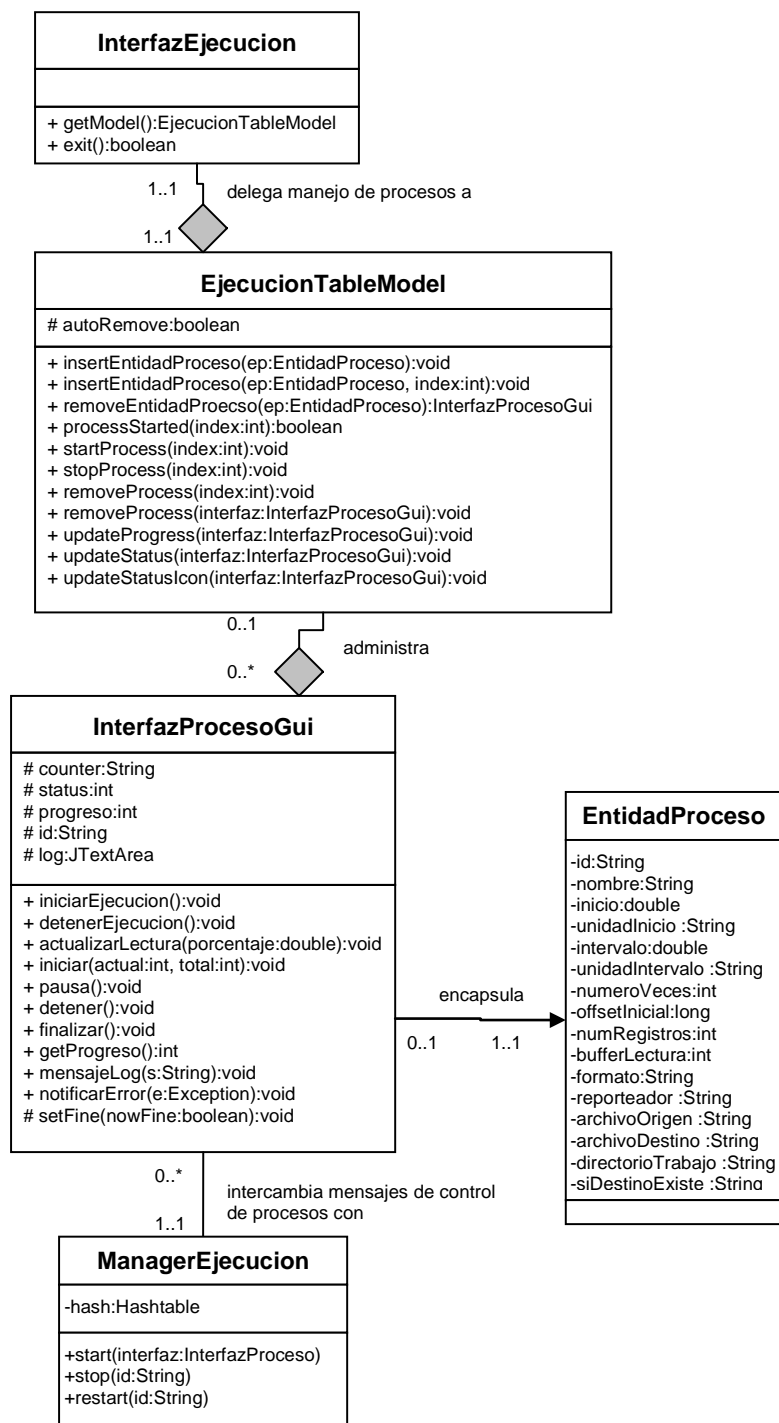


Figura 4.13: Diagrama de clases del módulo de definición y ejecución de procesos – Sección Ejecución

Este diagrama muestra las clases involucradas en la sección gráfica de ejecución de procesos. Cuando un proceso se manda a ejecutar, el mensaje llega a *InterfazEjecucion* (quien contiene la interfaz gráfica), quien se lo delega a *EjecucionTableModel*. *EjecucionTableModel* contiene a un conjunto de instancias de la clase *InterfazProcesoGui*, clase encargada de mantener el enlace de información entre el proceso en cuestión y la interfaz del usuario. *InterfazProcesoGui* básicamente contiene las variables que indican el estado de ejecución (junto con el log asociado al proceso). *ManagerEjecucion* es la clase que administra y maneja a los procesos que se están ejecutando, esta clase interactúa constantemente con *InterfazProcesoGui* para notificar los cambios (y eventos) sucedidos durante la ejecución del proceso. La *EntidadProceso* no hace más que ser un contenedor de información para las otras clases durante la etapa de ejecución. También existe una instancia de *EntidadProceso* que sirve de molde para la creación de los demás procesos, pues esta instancia contiene los valores por omisión del proceso.

A continuación se presenta el diagrama que indica la interacción de objetos que ocurre al generar un nuevo proceso:

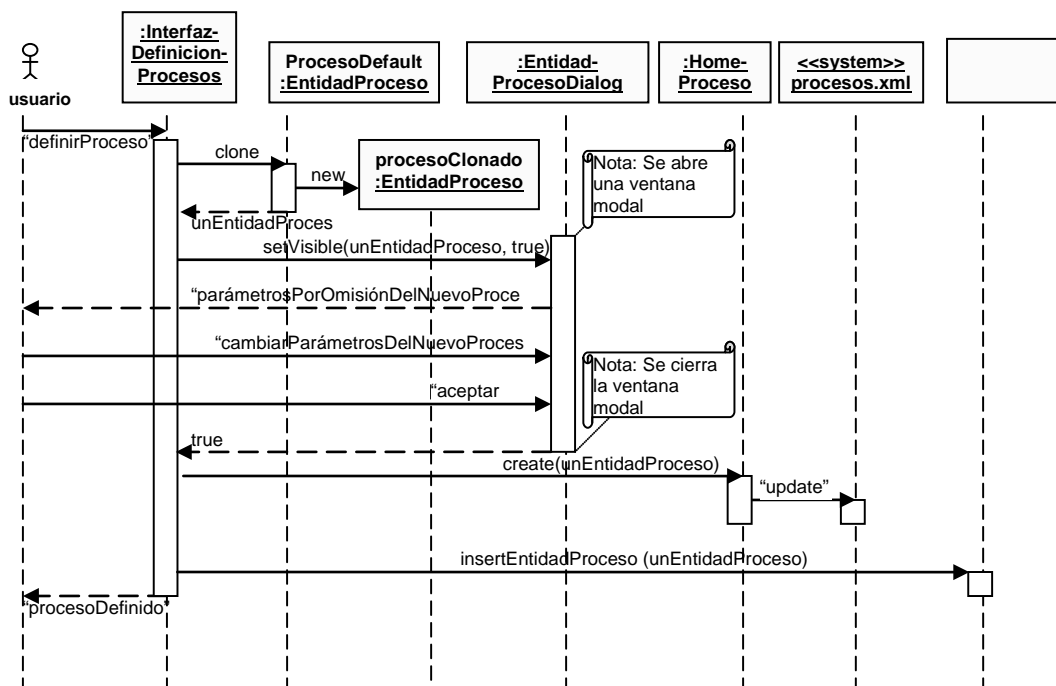


Figura 4.14: Usuario define un nuevo proceso

Aquí se puede ver como el comando del usuario pasa a *InterfazDefinicionProceso*, quien usa a la *EntidadProceso* modelo para generar un nuevo proceso. Luego este proceso se pasa a la clase *EntidadProcesoDialog* para presentar al usuario el nuevo proceso. Una vez que el nuevo proceso tenga sus campos definidos, este es pasado a *HomeProceso* para que registre la creación de dicho proceso (para su almacenamiento en un DOM y posteriormente a un archivo XML). Finalmente, el nuevo proceso es agregado a la lista de procesos que contiene *DefinicionTableModel*.

A continuación se presenta un diagrama de interacción de objetos que muestra el inicio de un proceso:

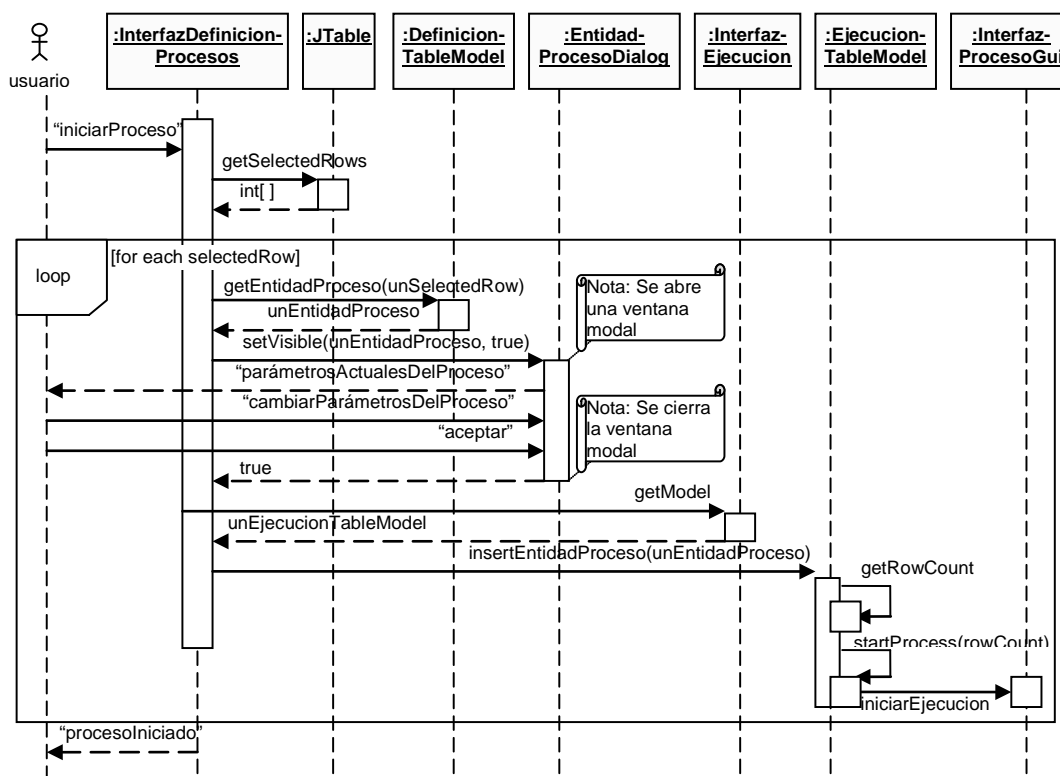


Figura 4.15: Usuario inicia un proceso

Este diagrama muestra como las clases se encargan de iniciar todos los procesos que estén seleccionados en la tabla de *DefinicionTableModel*. Se recorren todos los procesos seleccionados, y cada uno es presentado al usuario por medio de *EntidadProcesoDialog*, para que el usuario pueda hacer cualquier cambio o modificación de último minuto al proceso (notar que dichos cambios no afectan al proceso que se conserva en

DefinicionTableModel, sólo afectan a una nueva instancia de *EntidadProceso* que será usada en la sección de ejecución). Por cada proceso que el usuario vaya aceptando, se hace una llamada a ingresar un nuevo proceso en la sección de ejecución. *InterfazEjecucion* provee acceso directo a *EjecucionTableModel*, al cual se le indica que hay un nuevo proceso que agregar. *EjecucionTableModel* toma el proceso, y crea un nuevo *InterfazProcesoGui* que toma al proceso en cuestión y inmediatamente arranca la ejecución del mismo.

4.2.3 MÓDULO DE INTERPRETACIÓN Y TRANSFORMACIÓN DE DATOS

Una de las consideraciones más importantes en este módulo es la determinación de la naturaleza del código que llevará a cabo la interpretación y transformación de los datos de tráfico. Es posible considerar al documento descriptor de formatos como un conjunto de instrucciones que permitan la generación dinámica de código en un lenguaje de programación meta. Este código, una vez compilado, será capaz de realizar el proceso de transformación sobre los archivos que se adhieran al formato descrito.

Otra solución es considerar al mismo documento descriptor de formatos como un lenguaje de programación de alto nivel, el cual debe simplemente ser interpretado por un módulo implementado en un lenguaje de programación cualquiera, responsable a bajo nivel de la creación e interacción de objetos en memoria y de la interpretación de las estructuras de control embebidas en el documento de descripción de formatos.

En el resto de esta sección se explicará como se ha llevado a cabo el proceso de interpretación del “código” embebido en un documento descriptor de formatos generado por el sistema.

El siguiente diagrama de clases representa las relaciones estáticas entre los principales objetos responsables de realizar la interpretación y transformación de los archivos generados por centrales de tráfico:

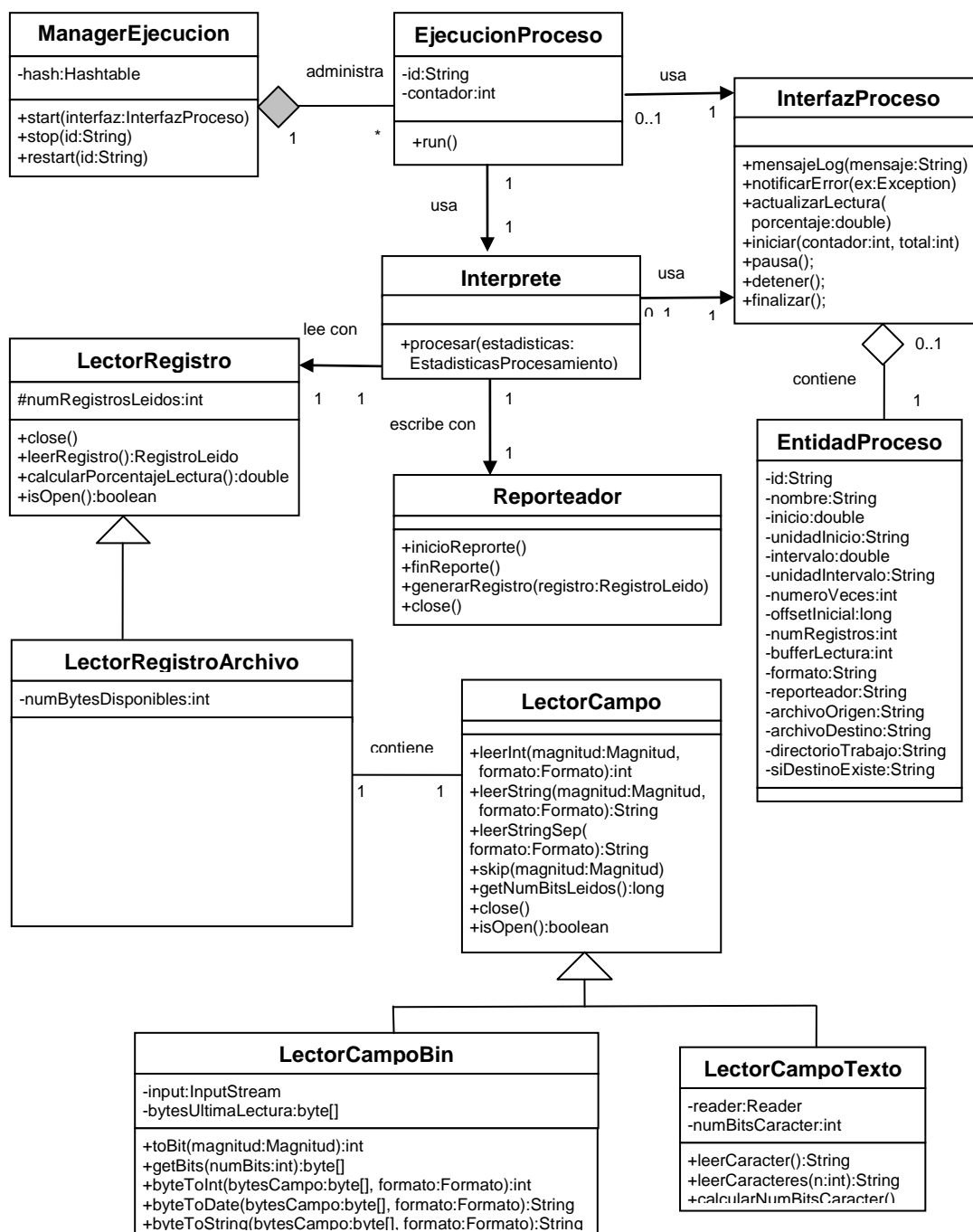


Figura 4.16: Diagrama de clases del módulo de interpretación de datos (parte 1)

A continuación se da una breve descripción de los objetos descritos en el diagrama anterior:

El objeto *ManagerEjecucion* es el punto de partida para la ejecución y de procesos de interpretación de datos, administra a todas las instancias de *EjecucionProceso*, que como su nombre lo indica, representan a un proceso en ejecución.

El objeto *InterfazProceso* abstrae a la interfaz gráfica con la que interactúa el usuario, su finalidad es permitir enviarle a este retroalimentación de los eventos ocurridos durante el procesamiento de los archivos generados por la central telefónica. Este objeto está relacionado con una instancia de *EntidadProceso*, la cual representa la definición de un proceso de transformación con todos los datos necesarios para llevarlo a cabo adecuadamente.

El objeto *Interprete* tiene como responsabilidades realizar a bajo nivel la transformación de los datos y proporcionar ciertas estadísticas que resumen los eventos ocurridos durante el procesamiento del archivo. El funcionamiento de la clase *Interprete* depende principalmente de dos objetos, estos son instancias de: *Reporteador* y *LectorRegistro*.

El objeto *Reporteador* utilizado es elegido por el usuario en el momento que este define los parámetros del proceso. Este objeto es

el responsable de formatear adecuadamente y escribir los datos a un archivo destino una vez que han sido procesados.

El objeto *LectorRegistro* utilizado es escogido por el sistema y depende de las características del formato de archivo que se desea interpretar. Este objeto es el responsable de interpretar el archivo con los datos sin procesar generados por la central.

Un objeto *LectorRegistro* emplea un objeto de tipo *LectorCampos*, el cual le permite leer uno por uno los campos presentes en el registro interpretado. El tipo de *LectorCampo* utilizado depende del tipo de archivo a ser interpretado, se emplean: *LectorCampoBin* para archivos binarios y *LectorCampoTexto* para archivos basados en texto.

Todos los objetos persistentes del sistema son almacenados en archivos XML, se han definido diferentes clases que encapsulan cada uno de estas entidades persistentes, así como clases que representan a managers que permiten administrarlos. El siguiente diagrama de clases resume este diseño:

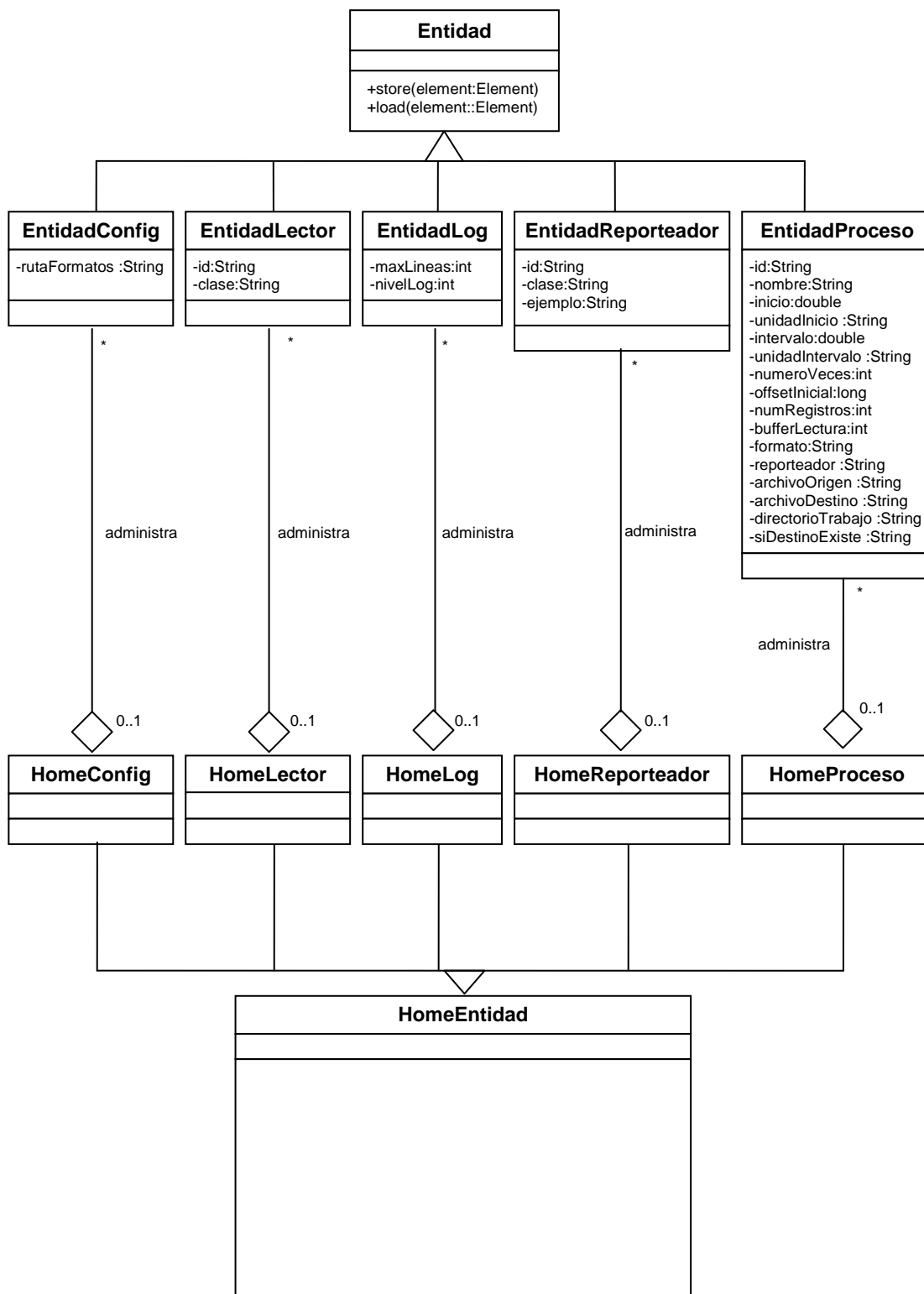


Figura 4.17: Diagrama de clases del módulo de interpretación de datos (parte 2)

A continuación se describen brevemente los objetos descritos en la figura anterior:

El objeto *EntidadConfig* representa a una entidad que encapsula los datos de la configuración general del sistema, almacena todos los parámetros persistentes que la aplicación requiere para ejecutarse correctamente.

El objeto *EntidadLector* encapsula los parámetros que describen a un “lector”. Esto es, un objeto capaz de realizar la lectura e interpretación de un formato de archivo de tráfico.

El objeto *EntidadLog* permite almacenar una representación de los datos persistentes importantes para el correcto funcionamiento del log del sistema.

El objeto *EntidadReporteador* encapsula los parámetros que describen a un “reporteador”. Esto es, un tipo de objeto que formatea y muestra los datos una vez ya procesados.

Las propiedades y métodos comunes de estas clases son resumidos en el objeto *Entidad*, el cual provee de la funcionalidad general necesaria para almacenar y cargar información de archivos XML.

Las clases *HomeConfig*, *HomeLector*, *HomeLog*, *HomeReporteador* y *HomeProceso*, tal como muestra el diagrama, permiten acceder a instancias de las clases que representan a los objetos persistentes en el sistema. La funcionalidad común de estas clases está encapsulada en la clase *HomeEntidad*.

Todos los objetos del sistema que requieran acceso a una de los objetos managers (estas son las clases que heredan de *HomeEntidad*), pueden hacerlo mediante métodos de la clase *Configuracion*, cuyas relaciones estáticas son descritas en el siguiente diagrama:

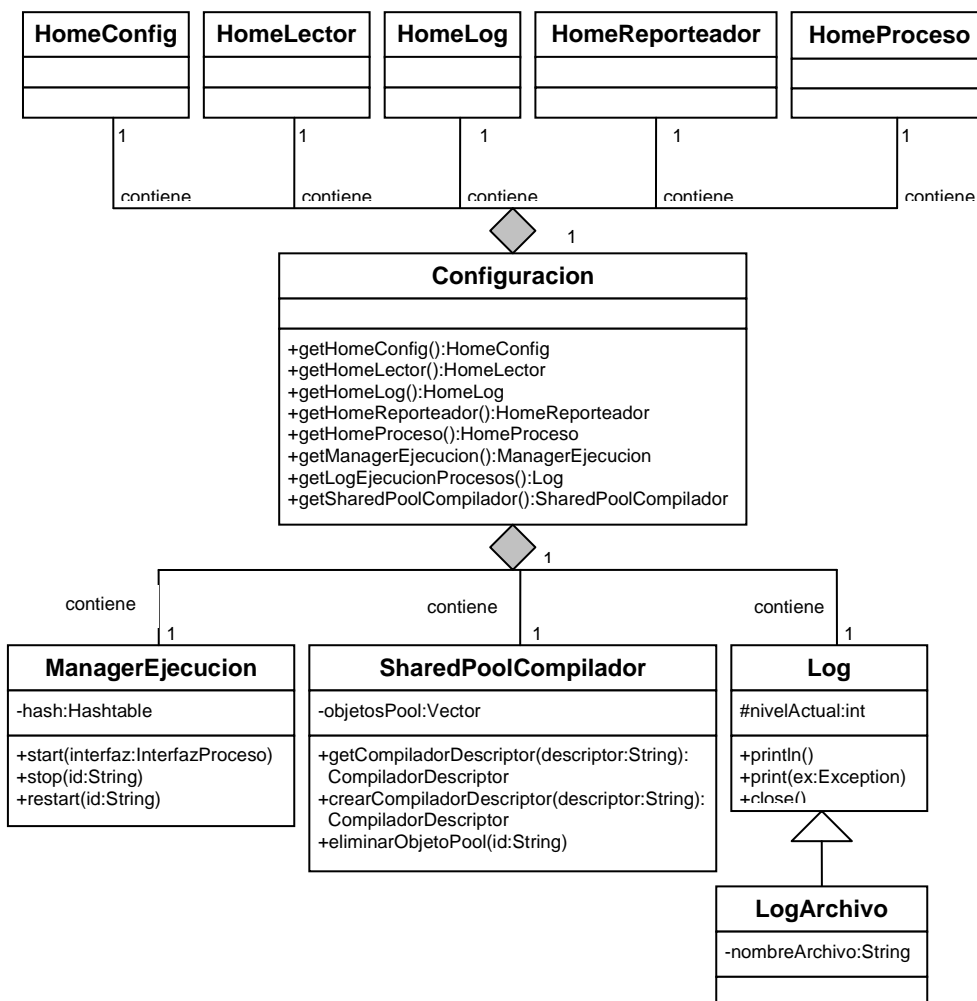


Figura 4.18: Diagrama de clases del módulo de interpretación de datos (parte 3)

Como muestra el diagrama anterior, la clase *Configuracion* proporciona acceso a objetos que existen a lo largo de toda la ejecución de la aplicación. Estos solo son desalojados de la memoria cuando el sistema termina su ejecución.

La siguiente figura representa la primera parte de un diagrama de secuencia que representa como interactúan la mayor parte de los

objetos descritos hasta aquí para llevar a cabo la ejecución de un proceso de interpretación de datos definido por el sistema:

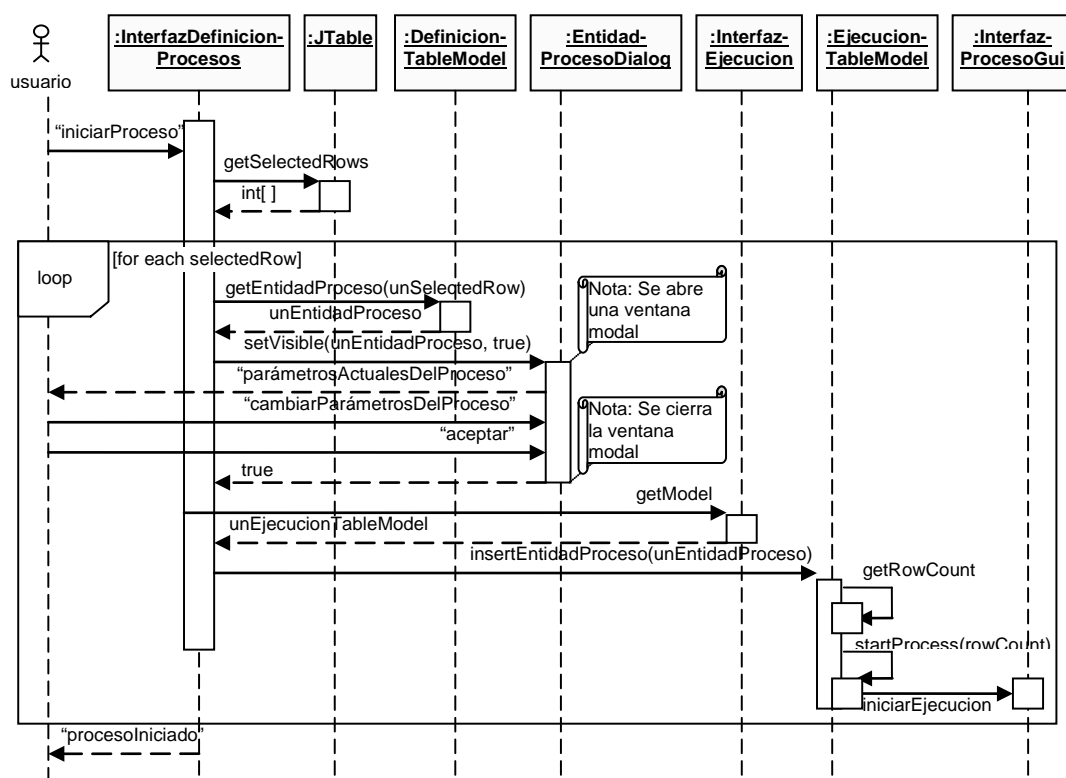


Figura 4.19: Usuario inicia un proceso (a)

Los objetos mostrados en esta primera parte del diagrama forman parte del módulo de definición y ejecución de procesos. Como se explicó en la sección anterior, el usuario del sistema no interactúa directamente con el módulo de interpretación y transformación de datos, sino que cada requerimiento de realizar una acción sobre uno de los procesos definidos previamente (tal como “ejecutar el proceso”) es recibido por el módulo de definición y ejecución de

procesos, este a su vez examina el requerimiento y lo delega al módulo de interpretación y transformación de datos.

La siguiente figura muestra la continuación del diagrama de secuencia anterior:

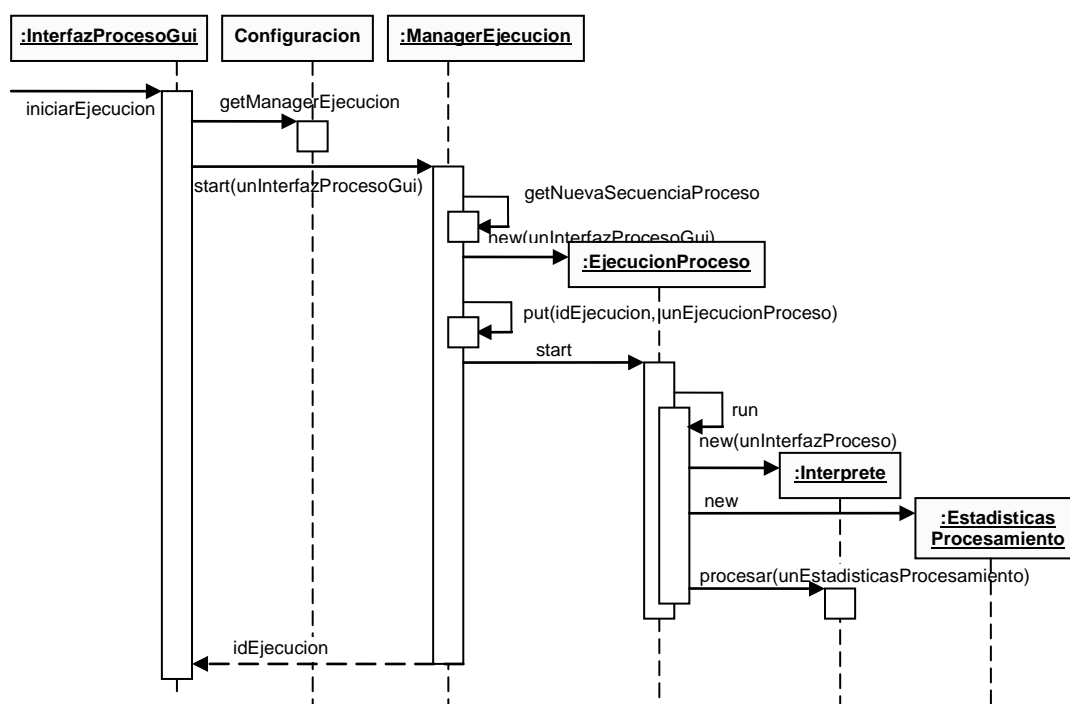


Figura 4.20: Usuario inicia un proceso (b)

En esta figura puede verse como el objeto *ManagerEjecucion* es el punto de partida para la ejecución de procesos. Para llevar a cabo la ejecución de un proceso, la clase *ManagerEjecucion* instancia a un objeto *EjecucionProceso* y lo deposita en un repositorio de objetos (un hashtable, de acuerdo a la implementación). Luego se invoca al

método “start” de *EjecucionProceso*, el cual a su vez instancia a un objeto *Interprete* y llama a su método “procesar”, el cual llevará a cabo la interpretación de los archivos generados por la central. La siguiente figura muestra la última parte del diagrama de secuencia descrito, mostrando el diseño de cómo la entidad *Interprete* ejecuta el método “procesar”:

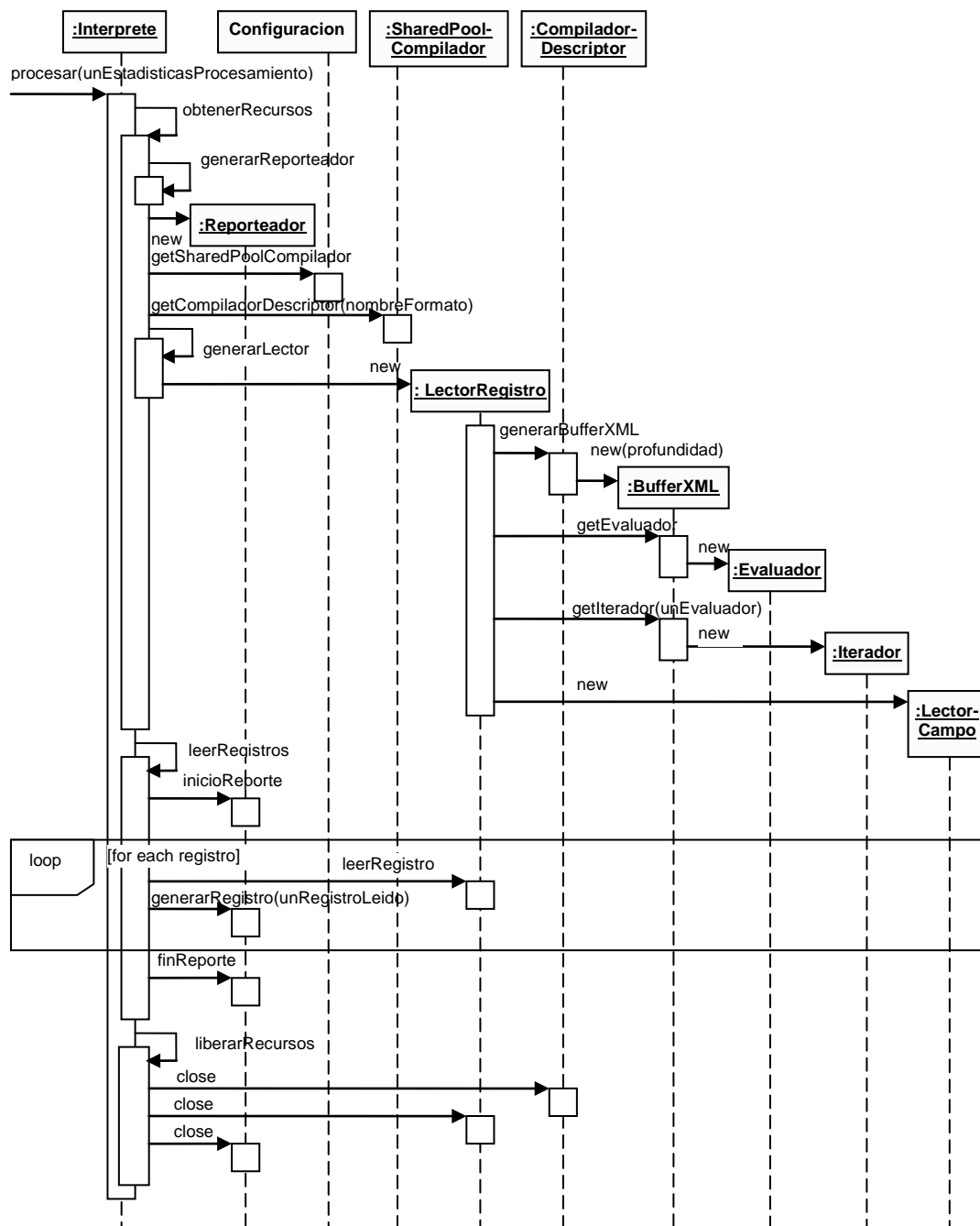


Figura 4.21 Usuario inicia un proceso (c)

La figura anterior muestra los objetos que intervienen en la última etapa del diagrama de secuencia que representa la ejecución de un proceso.

Uno de los principales problemas de diseño que debían ser resueltos por este módulo, es el de crear un grupo de entidades que abstrayeran a un documento descriptor de formatos. La siguiente figura da una representación gráfica de cómo este documento descriptor de formatos representa la estructura de los archivos generados por las centrales de tráfico:

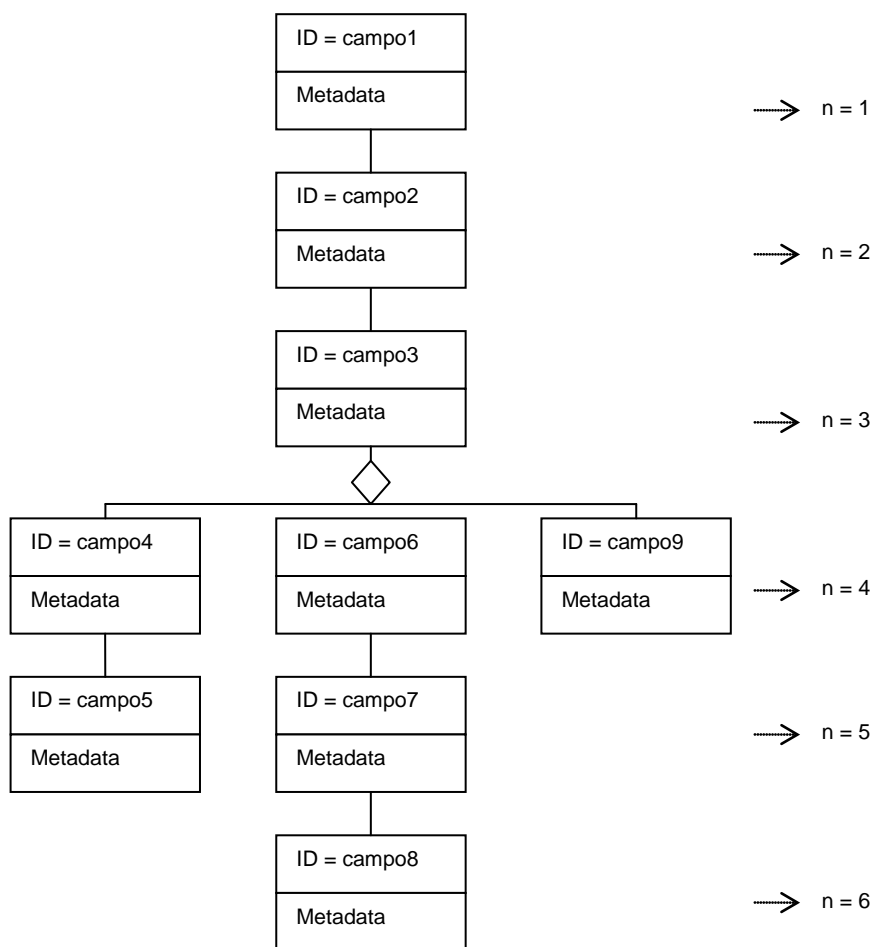


Figura 4.22: Representación de un descriptor de formatos a manera de un árbol

El gráfico muestra que la estructura de un registro no es estática, sino que está condicionada a los valores de ciertos campos que son conocidos solamente en tiempo de ejecución. Decimos entonces que la estructura del registro es dinámica.

A medida que se va realizando la interpretación de los campos de un registro en particular, pueden encontrarse ciertos tipos de campos

que no deben ser leídos del archivo de datos proporcionado por la central, sino más bien calculados en función de campos que han sido ya interpretados en la lectura del registro actual.

El siguiente gráfico compara el comportamiento de los campos leídos y de los campos calculados como funciones de otros:

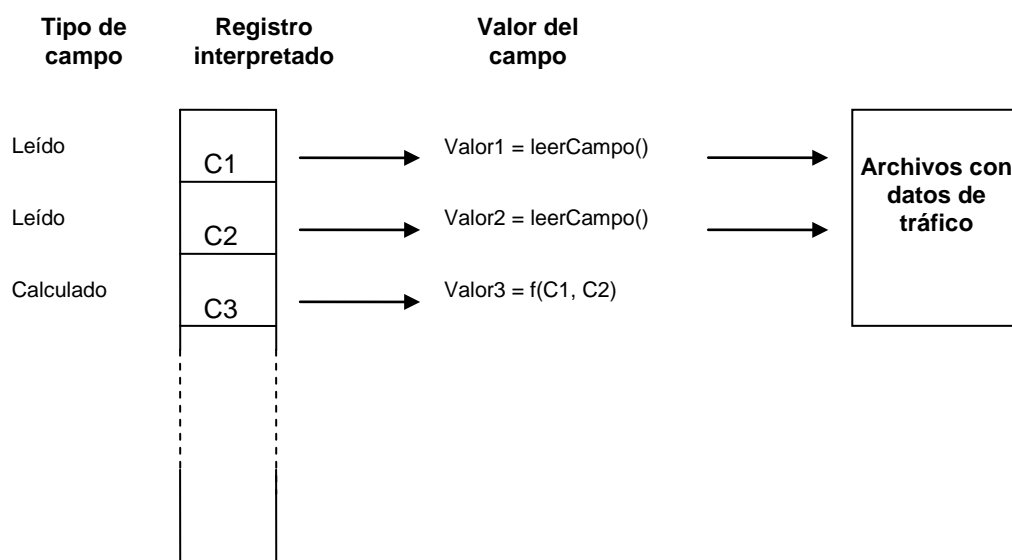


Figura 4.23: Diferencias entre campos leídos y calculados

La figura anterior muestra a los tres primeros campos de un registro, los dos iniciales están definidos como campos de lectura, por lo que el valor de los mismos debe extraerse del archivo con los datos de tráfico. El tercero en cambio está definido como un campo calculado,

por lo que su valor está en función de otros campos del mismo registro, en el caso de este ejemplo, los dos primeros.

El siguiente diagrama de clases ilustra las relaciones entre los objetos que definen un formato de datos de tráfico:

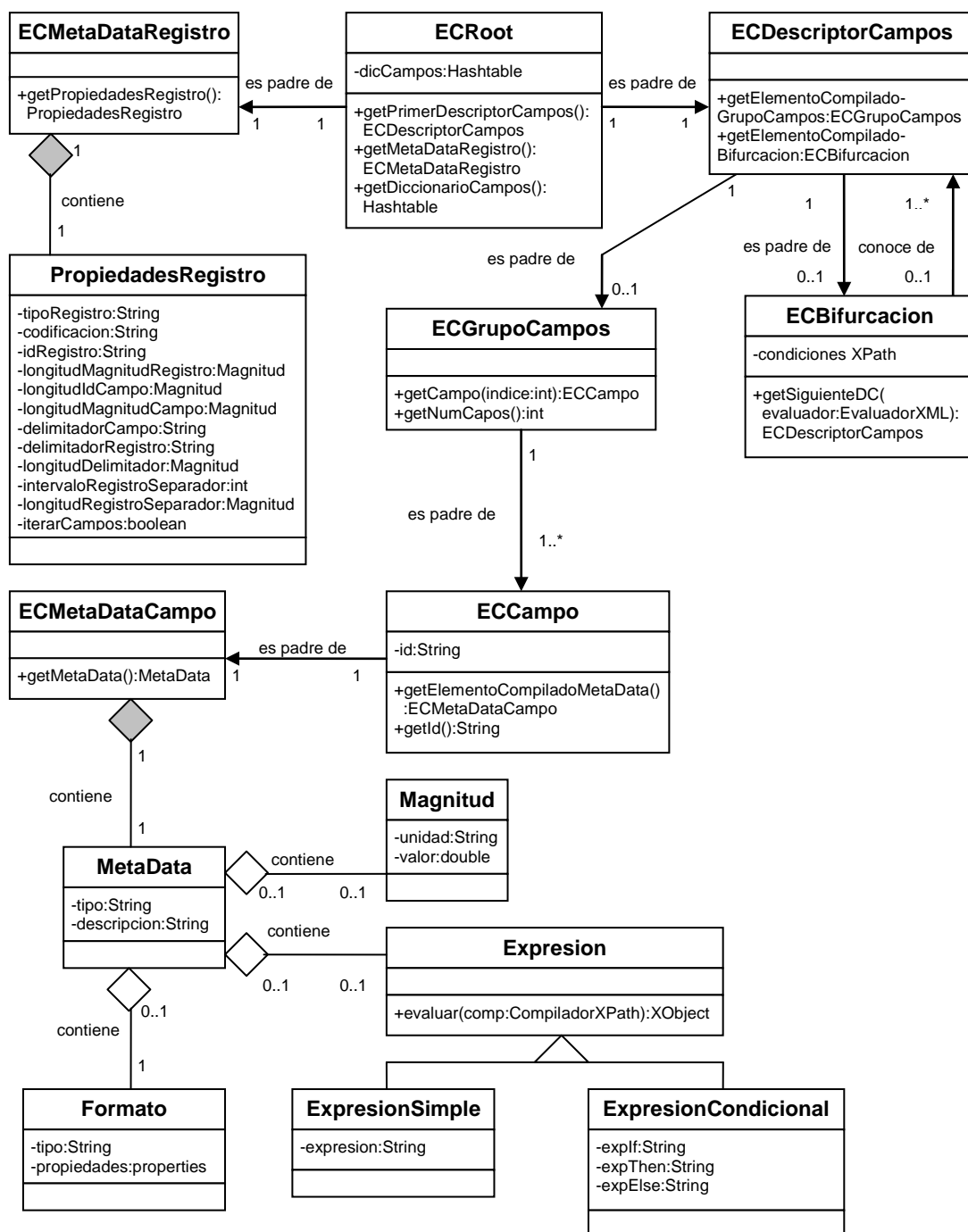


Figura 4.24: Diagrama de clases del módulo de interpretación de datos (parte 4)

La figura muestra entre otros un objeto denominado *ECRoot*. Este objeto encapsula a todos los demás objetos que definen una representación de un formato. Es padre de un objeto *ECMetaDataRegistro* y de un objeto *ECDescriptorCampos*. El primero de estos objetos representan los metadatos del formato y contiene información general sobre este almacenada en un objeto *PropiedadesRegistro*.

El objeto *ECDescriptorCampos* permite definir la estructura de un cierto número de campos de datos. Es padre de dos objetos: *ECGrupoCampos* y *ECBifurcacion*. El primero de estos posee una lista de objetos tipo *ECCampo*, estos representan una definición de campos que van a ser encontrados siempre de manera secuencial en un archivo de datos generado por una central de tráfico. Cada uno de los objetos *ECCampo*, son padres de un objeto *ECMetaData*, que a su vez es padre de un objeto *MetaData*, el cual define los metadatos de un campo de datos. El objeto *MetaData* contiene tres tipos de objetos: *Formato*, *Magnitud* y *Expresion*. El objeto *Formato* encapsula a un grupo de propiedades que permiten determinar cómo será interpretado el dato, es decir si debe tratárselo como a una fecha, un número, como una secuencia simple de caracteres, etc. El objeto *Magnitud* indica cuantas unidades deben ser leídas del archivo

generado por la central para procesar el campo. El objeto *Expresion* determina si el campo debe ser leído del archivo de datos, o bien calculado en función de otros campos ya interpretados. Como se muestra en el diagrama puede ser de dos tipos: *ExpresionSimple* y *ExpresionCondicional*. El primer tipo encapsula a expresiones simples, es decir a aquellas que simplemente son evaluadas en funciones de otros campos. El objeto *ExpresionCondicional* en cambio representa una condición lógica a evaluar y a dos posibles expresiones adicionales, la expresión escogida dependerá si la condición lógica, una vez evaluada, fue verdadera o falsa.

El objeto *ECBifurcacion* contiene referencias a varios objetos *ECDescriptorCampos*, cada uno de estos asociados con una expresión booleana que actúa como una “condición lógica de entrada”.

Cuando se terminan de interpretar los campos listados en *GrupoCampos*, se procede a decidir cual nuevo elemento *ECDescriptorCampos*, presente en *ECBifurcacion*, es con el cual se deberá continuar. La elección es llevada a cabo examinando cada una de las condiciones de entrada, hasta encontrar una que tome el valor lógico de verdadero.

De esta manera el objeto *ECBifurcacion* implementa las estructuras de formatos dinámicas, al permitir decidir en tiempo de ejecución que nuevo objeto *ECDescriptorCampos* utilizar.

La mayor parte de los objetos descritos en el diagrama anterior, representan elementos que han sido almacenados en un documento descriptor de formatos generado por el sistema. Es por eso que comparten cierta funcionalidad común. El siguiente diagrama de clases muestra qué características comunes han sido abstraídas en un objeto *ElementoCompilado*, del cual heredan los demás:

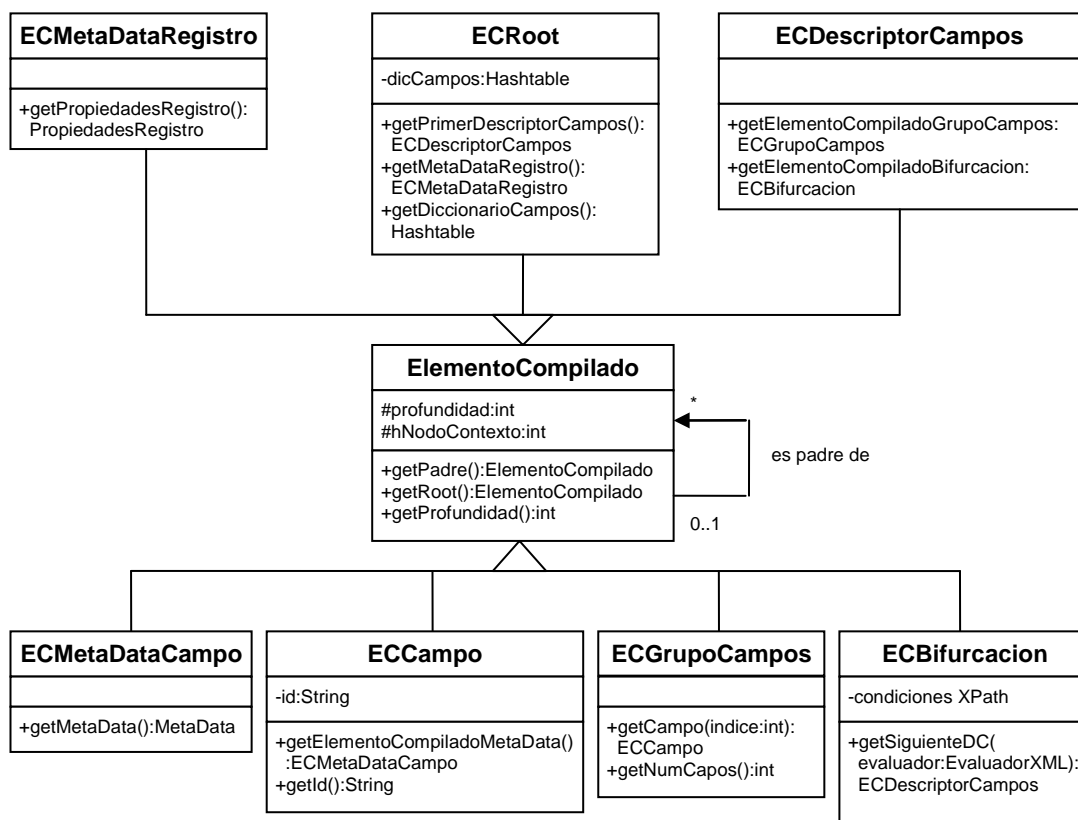


Figura 4.25: Diagrama de clases del módulo de interpretación de datos (parte 5)

Para poder evaluar los distintos tipos de expresiones (funciones) presentes en un documento descriptor de formatos, es adecuado la definición de estructuras que sirvan como repositorio para los datos de los campos que han sido ya debidamente interpretados, de esta manera, dichos campos podrán ser referenciados mediante su ID en caso de ser requeridos. El tamaño de las estructuras usadas para el procesamiento de los registros es determinada por la profundidad del

árbol de descripción de formatos, esto es debido a que el número de campos a ser leídos es solamente conocido en tiempo de ejecución.

La siguiente figura representa la estructura utilizada para evaluar expresiones que referencien a otros campos, en un árbol descriptor de formatos con una profundidad seis niveles:

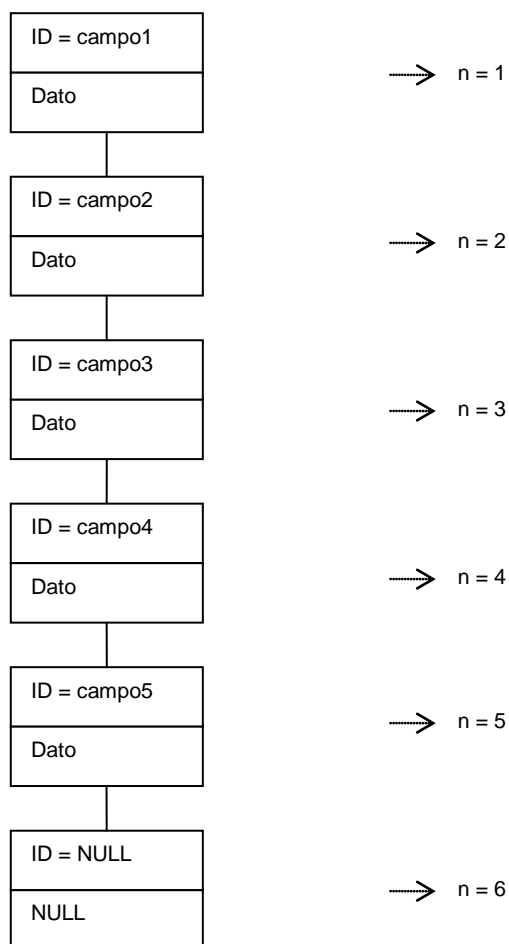


Figura 4.26: Estructura para la evaluación de expresiones

El ejemplo mostrado en la figura ilustra la lectura de un registro compuesto por 5 campos. Dado que para este ejemplo la profundidad del árbol que representa al documento descriptor de formatos es de 6, es esta la capacidad asignada a la estructura responsable de la evaluación de expresiones. Debido a que al llevar a cabo la lectura hubo un campo sobrante (el último), este tiene valores indefinidos tanto para el ID como para sus datos.

El siguiente diagrama de clases muestra las relaciones entre los objetos que implementan este comportamiento:

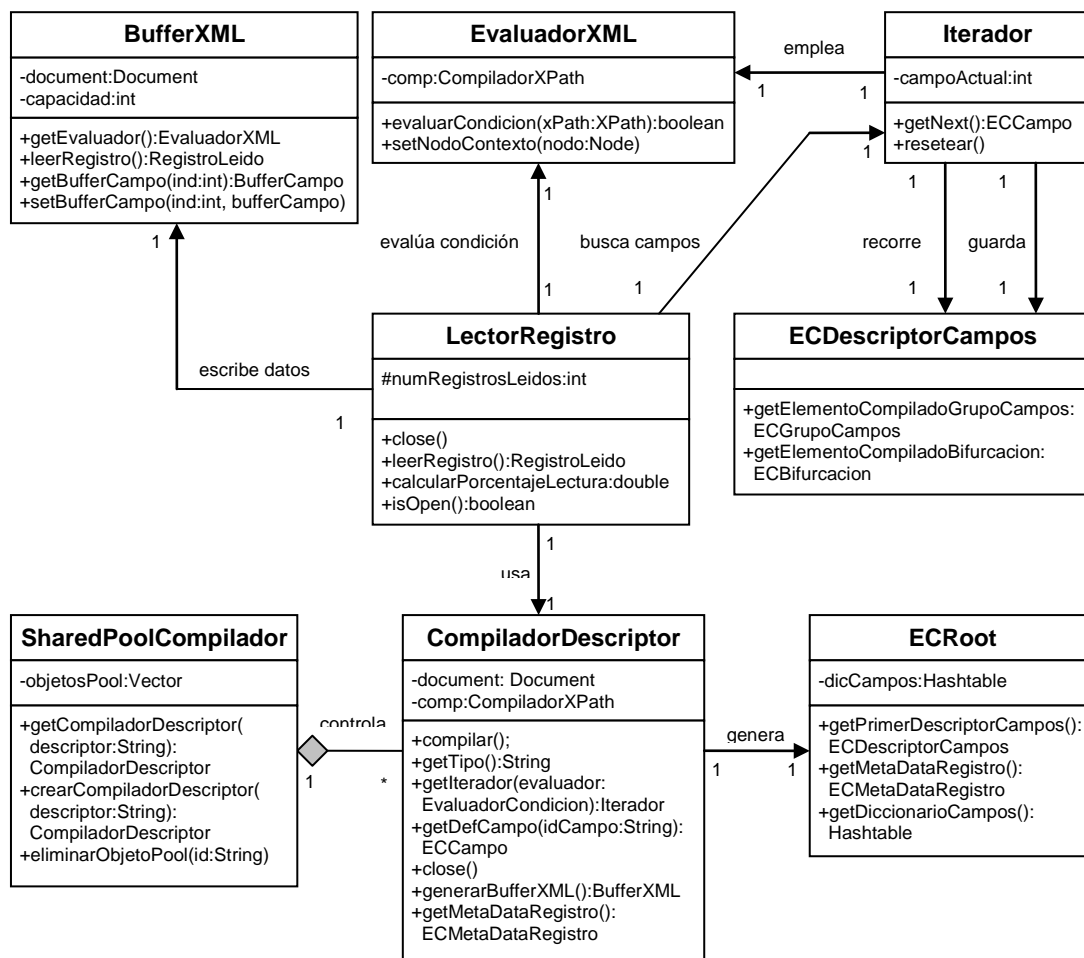


Figura 4.27: Diagrama de clases del módulo de interpretación de datos (parte 6)

A continuación se describen brevemente los objetos mostrados en la figura:

Un objeto *LectorRegistro* es responsable de realizar la lectura de los registros de datos. Cada registro está compuesto por cierto número de campos. Cuando se realiza la lectura de un nuevo registro, cada campo obtenido es almacenado en una instancia de *BufferXML*, el

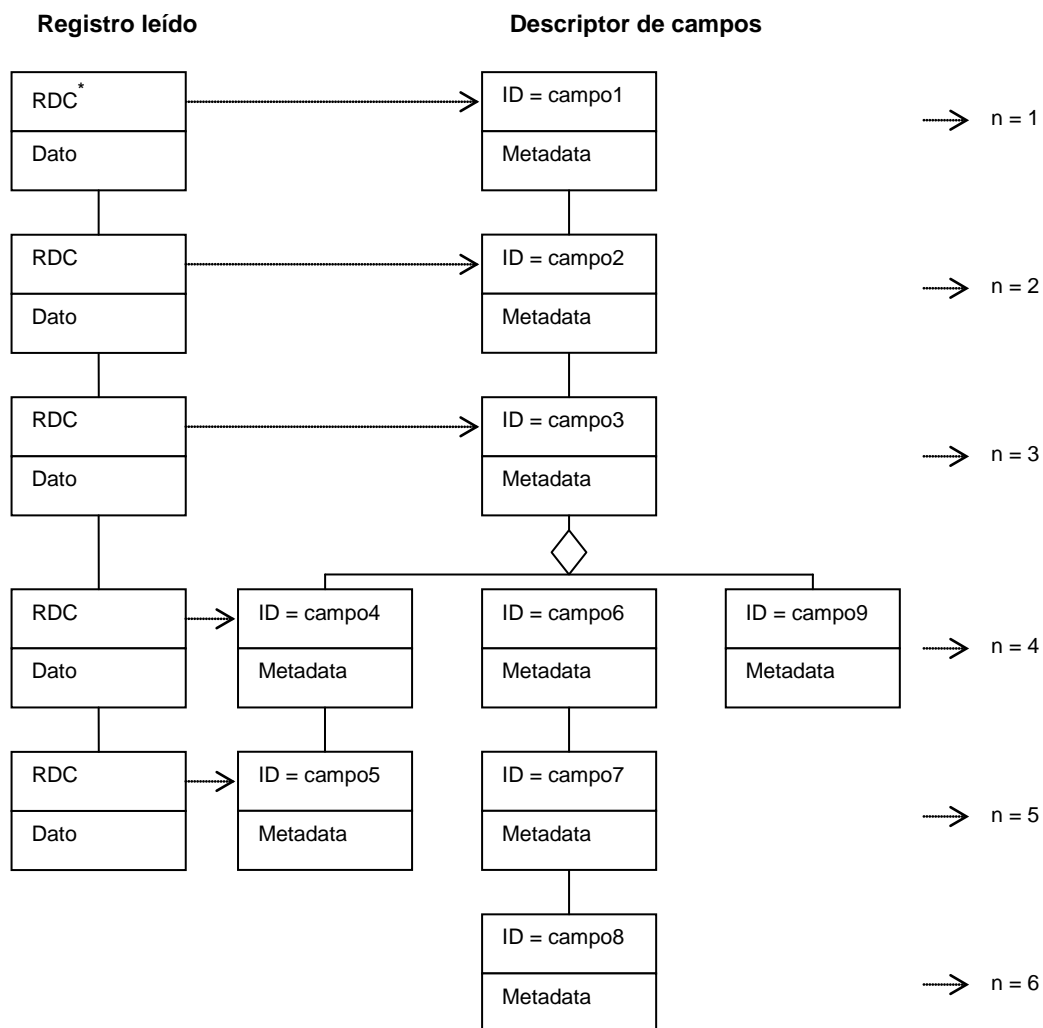
cual actúa como un repositorio temporal de los campos interpretados del registro actual. El objeto *BufferXML* permite también obtener instancias de objetos *EvaluadorXML*, los cuales permiten la evaluación de expresiones en el contexto de los campos presentes en el objeto *BufferXML*.

El objeto *CompiladorDescriptor* es el responsable de interpretar documentos XML que representan a un “documento descriptor de formatos”, el cual ha sido generado previamente con el módulo de definición de formatos del sistema. Este objeto *CompiladorDescriptor* permite generar a las estructuras en memoria que representan a este documento descriptor de formatos. La cantidad de memoria requerida para representar a estos datos depende de la extensión y complejidad de los formatos descritos. Con la finalidad de minimizar el consumo de memoria se utiliza el objeto *SharedPoolCompilador*, objeto responsable de controlar a todas las instancias de clases *CompiladorDescriptor*, asegurando que no exista más de una de estas instancias por cada formato siendo utilizado en un momento determinado. Si la aplicación necesita una instancia de un *CompiladorDescriptor* para un formato que ya ha sido requerido con anterioridad, se recicla la ya existente. En caso de que no exista una instancia se generará una nueva.

Las instancias de *CompiladorDescriptor* permiten generar objetos de tipo *Iterador*, los cuales permiten recorrer secuencialmente las estructuras en memoria que representan a un documento descriptor de formatos.

El objeto *Iterador* está asociado con un objeto *EvaluadorXML* proporcionado por una instancia de *BufferXML*. Esta instancia de *EvaluadorXML* es utilizada por *Iterador* para evaluar las estructuras condicionales del registro, es decir, para resolver las expresiones booleanas que determinarán cual es la secuencia de campos a la que se apegan los registros.

Una vez que las expresiones que posee el campo son evaluadas, el resultado es almacenado en su repositorio final, el cual es una colección de todos los campos del registro que ya han sido leídos e interpretados. La siguiente figura representa esta estructura:



* RDC.-Referencia a un descriptor de campo

Figura 4.28: Representación de un registro ya leído

En la figura mostrada hay dos estructuras importantes: a la derecha tenemos a una estructura que representa un “*Descriptor de campos*”, como ya se ha dicho, esta describe el “mapa de caminos” en donde están representadas todas las descripciones de los campos que

pueden potencialmente constituir un registro para una central de tráfico en particular.

La estructura de la izquierda representa un registro completamente leído e interpretado. Cada uno de los elementos de esta estructura encapsulan a dos objetos: el valor interpretado del campo y una referencia a su correspondiente descriptor de campos ubicado en la estructura de la derecha. Es esta estructura la que deberá ser comunicada a los módulos encargados de distribuir o mostrar la información obtenida.

Para observar todos los diagramas de interacción de objetos con sus posibles escenarios, refiérase al apéndice D.

CAPÍTULO 5

5. IMPLEMENTACIÓN Y PRUEBAS

5.1 PROCESO DE IMPLEMENTACIÓN

En esta sección se describe el proceso de implementación del sistema, el cual se llevó a cabo una vez terminadas las etapas de análisis y diseño.

5.1.1 CAPACITACIÓN EN LA TECNOLOGÍA INVOLUCRADA

Antes de proceder a la implementación del sistema se realizó una capacitación en las siguientes tecnologías:

- **XML.-** Se realizó una investigación exhaustiva sobre las opciones ofrecidas por XML para la definición y estructuración de información.
- **XSD.-** Después de repasar las ventajas y desventajas de diversas tecnologías de definición de esquemas para documentos XML, se optó por dedicar tiempo de investigación al lenguaje XSD como herramienta para definir la estructura de los documentos XML del sistema. Esta tecnología proporcionó una gran flexibilidad para la definición de toda clase de estructuras de datos.
- **XPath.-** Al inicio del proyecto no se tenían conocimientos sobre esta tecnología, pero las investigaciones realizadas durante la etapa de análisis y diseño determinaron que era lo más adecuado para la evaluación de expresiones de diversos tipos en el contexto de documentos XML.
- **Xalan.-** Se investigó esta implementación de XPath realizada por Apache por ser una de las más adecuadas por su facilidad de uso y eficiencia.

- **DOM y SAX.-** Aunque se comprendían los conceptos fundamentales que involucraban estas dos tecnologías de representación de documentos XML, fue necesario una revisión a fondo de las capacidades del API de cada una de estas.
- **JAXP.-** Se dedicó tiempo a la investigación de este estándar para en la medida de lo posible usarlo para la interacción con el API de DOM y SAX.

Para una descripción completa de las tecnologías descritas en esta sección, referirse al capítulo 2 de este proyecto de tesis.

5.1.2 SIMULACIÓN DEL AMBIENTE DE PRODUCCIÓN DEL PRODUCTO

Antes de proceder a la implementación de la solución, se realizó un modelo a pequeña escala del escenario físico en el que debería ejecutarse la aplicación. Para esto se instaló Linux Advanced Server 2.1 en un PC y Windows XP en otro, de manera que estos equipos simularan el servidor de aplicaciones de Linkotel (Linux) y su central de tráfico Huawei (Windows) respectivamente. Luego se procedió a

configurar el sistema operativo Linux para que pueda montar en su sistema de archivos recursos compartidos en la red mediante smb (el protocolo de compartición de archivos usado por Windows). Debido a que el kernel del Linux AS 2.1 no tiene incluido por omisión el módulo encargado de realizar esta tarea, fue necesario añadirlo y configurarlo.

5.1.3 DEFINICIÓN DE ESTÁNDARES DE DESARROLLO

Antes de llevar a cabo la implementación del sistema, se definieron la mayor parte de los estándares de programación que regirían en el desarrollo del proyecto. De esta manera se facilitará la interacción entre los módulos del programa y el futuro mantenimiento de los mismos. Ciertos acuerdos, como el tipo de codificación de texto (UTF8, UTF16, ascii, etc...) a utilizar en el código fuente, fueron resueltos en la etapa de implementación cuando se vio la necesidad de establecer también un estándar al respecto.

Los principales estándares de programación fueron los siguientes:

- Todo los nombres de clase comenzarán con letras mayúsculas.

- Todos los nombres de variables y métodos empezarán con letras minúsculas.
- Las constantes se escribirían completamente en mayúsculas.
- La indentación para estructuras de código anidadas será de 4 espacios.
- Se estableció que los archivos de programa seguirían la codificación UTF8.

5.2 PROBLEMAS PRESENTADOS EN LA IMPLEMENTACIÓN

Esta sección describe los problemas que ocurrieron en la etapa de implementación del sistema.

Un problema que surgió durante las pruebas de portabilidad fue que se observó que los archivos de configuración que contenían los nombres de las carpetas en donde se distribuyen los recursos del sistema, usaban como separador de directorios el símbolo de windows: “\”, esto ocasionaba errores de ejecución en la plataforma Linux. Se lo solucionó definiendo siempre a los separadores de directorios con el estilo de Linux: “/”, ya que Windows en cambio no

tiene problemas para comprender a cualquiera de estos dos símbolos.

Respecto a la escritura y compilación de código, pronto se cayó en cuenta de que era necesario establecer un estándar para la codificación a utilizar en los archivos de clases. Esto se debió a que los ambientes de desarrollo eran diferentes en cuanto a sistemas operativos, lo que provocaba que los archivos de código fuente solo podían ser editados correctamente en la plataforma en la cual habían sido escritos, ya que tenían por defecto diferentes codificaciones, lo que limitaba al trabajo en equipo. Por esta razón se llegó al acuerdo de utilizar UTF8 como codificación para todos los archivos del programa.

Otro problema que ocurrió durante el desarrollo fue que originalmente la velocidad de operación del sistema estaba muy por debajo de lo solicitado en los requerimientos iniciales. Esto se debió a que se estaba utilizando para la evaluación de expresiones XPath el API de alto nivel propuesta por XALAN, la cual era la implementación de XPath escogida. Aunque esta API de alto nivel era sumamente sencilla de utilizar, era ineficiente para la evaluación de grandes cantidades de expresiones complejas. Para solucionar

este problema fue necesario una investigación sobre el API de bajo nivel de XALAN, que aunque era algo más compleja de utilizar, proveía de una velocidad de ejecución más eficiente. Se desarrollaron clases que envolvieran un poco a esta API de bajo nivel, de acuerdo a las necesidades del sistema.

Sobre a la interfaz gráfica, el mayor problema fue el de ubicar adecuadamente la gran cantidad de opciones que podían ser configuradas en las definiciones de formatos. Fue necesario el uso de barras de desplazamiento para poder mostrar todas las alternativas. También hubieron arduas discusiones de cómo debía distribuirse el espacio de la aplicación, ya que el ordenamiento de elementos más intuitivo era el que más espacio desperdiciaba. La sección de desarrollo de formatos en particular tuvo que ser rediseñada un par de veces.

La implementación del módulo de definición de formatos presentó varias dificultades por la gran complejidad que se requería en su interfaz. La funcionalidad de las clases base de Swing no bastaban, en varias situaciones fue necesario localizar clases personalizadas en el Internet y en otros casos dichas clases especializadas de interfaz tuvieron que ser construidas.

Referente a los sistemas operativos, uno de los problemas que tuvieron que solucionarse fue el aprender a cambiar los parámetros del kernel del Linux Advanced Server 2.1, ya que este por omisión no traía activada la opción de poder montar en el sistema de archivos recursos que se encuentren compartidos en la red mediante el protocolo smb.

5.3 PRUEBAS REALIZADAS

Una vez que el sistema estaba cerca de terminar su implementación, se determinó que podían ya realizarse las pruebas que determinarían si los requerimientos funcionales y operativos habían sido cumplidos, la descripción de las pruebas realizadas se examinarán en lo que resta de esta sección.

5.3.1 PRUEBAS DE FUNCIONALIDAD

Para determinar si los objetivos establecidos en cuanto a funcionalidad podían ser satisfechos por el sistema, se procedió a obtener las especificaciones de los formatos usados por las centrales

telefónicas descritas en la sección 1.3.2 (objetivos y alcance del proyecto), estas son centrales de tipo Huawei, Ericsson y Alcatel.

Estas especificaciones son documentos emitidos por el vendedor de la central telefónica, que describen a un nivel granular como están estructurados los datos generados por la central de tráfico. Se indican detalles sobre cuántos tipos distintos de registros pueden ser generados por la central y cuáles son las condiciones en la lectura que permiten determinar con cuál de estos tipos se está trabajando.

Se procedió luego a describir con la interfaz gráfica del sistema los formatos definidos en las especificaciones de las centrales de tráfico analizadas. Para ver la definición de los formatos generados por el sistema para cada una de estas centrales, referirse al apéndice F.

Una vez hecho esto se procedió a interpretar, usando la descripción de formatos generada por el sistema, distintos datos de muestra de las centrales telefónicas y se compararon los resultados con los datos binarios presentes en los archivos sin procesar, esto se hizo con la ayuda de un editor hexadecimal, el cual permitió verificar que los datos obtenidos concordaban con su representación binaria.

Se comprobó entonces que los requerimientos, en cuanto a la transformación adecuada entre formatos, estaban siendo cumplidos exitosamente. Luego se procedió a evaluar la definición de procesos de transformación, aquí se programaron varios procesos para ejecutarse concurrentemente y a intervalos de tiempo definidos. Luego de verificar la correcta ejecución de los mismos se dieron por cumplidos los requerimientos funcionales y empezó la verificación de los requerimientos operativos. Las pruebas realizadas en esta etapa están descritas en las siguientes secciones.

5.3.2 PRUEBAS DE EFICIENCIA

Los requerimientos de eficiencia emitidos al inicio del proyecto de tesis (ver sección 3.1.2.1), determinaron que el sistema para ser operativo necesitaba trabajar a una velocidad de procesamiento tal que permita transformar 2.000.000 de registros de llamadas generados por la central Huawei de Linkotel, en un lapso menor a tres horas.

Esto significa que la velocidad de interpretación y transformación debe ser mayor a:

$$\frac{2000000}{(60 * 60 * 3)} = 185,2 \text{ registros/segundo}$$

Las pruebas del sistema para datos generados por la central Huawei, determinaron que el tiempo medio empleado en el análisis de cada registro era inferior a los 0,00055 segundos. Por lo tanto la velocidad del proceso de transformación es superior a:

$$\frac{1}{0,00055} = 1818,2 \text{ registros/segundo}$$

Esta es una cifra casi 10 veces mayor a la solicitada en los requerimientos operativos de eficiencia.

5.3.3 PRUEBAS DE PORTABILIDAD

Las pruebas de portabilidad del sistema consistieron en evaluar si este podía ejecutarse sin problemas tanto en un sistema operativo Linux como en otro sistema operativo con Windows, en base a los requerimientos operativos descritos en la sección 3.1.2.2.

Fue importante realizar la evaluación en ambos sistemas operativos, debido a que es en un Linux AS 2.1 en donde será ejecutado a corto plazo el sistema en Linkotel. Sin embargo la compatibilidad con Windows es importante debido a que la central Huawei puede ser considerada un sistema con Windows NT 4.0, por lo que si existe esta compatibilidad es posible ejecutar el sistema en este equipo si llegara a determinarse que es más conveniente. Esto puede eventualmente ocurrir debido a que el equipo con Linux tiene ya muchas responsabilidades asignadas, tales como servir de base de datos de la empresa y servidor de aplicaciones.

CAPÍTULO 6

6. CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

Una vez terminado el presente proyecto de tesis y llevados a cabo los objetivos establecidos al inicio del mismo, se emitieron las siguientes conclusiones:

1. Se cumplieron los objetivos y alcances descritos en la sección 1.3.2 de este documento, al realizar la implementación de un módulo capaz de llevar a cabo la lectura, interpretación y transformación de distintos formatos de datos de tráfico a un formato común y estandarizado, permitiendo la especificación de

la estructura de los formatos de entrada por parte de un operador que no posee necesariamente conocimientos de programación.

2. En base al análisis de los patrones más comunes de los formatos de archivos generados por centrales telefónicas de tráfico, pueden definirse reglas para especificar una gran cantidad de nuevos formatos de archivos.
3. Las empresas de telefonía, y en general cualquier empresa que administre una red de servicios, puede mejorar la eficiencia de sus procesos alimentados por los datos de consumos, mediante la implementación de un sistema automatizado de mediación entre los elementos de su red de servicios y los distintos sistemas de soporte de negocios.
4. XML puede utilizarse no solamente para estructurar y organizar datos complejos, sino que puede ser empleado también para crear lenguajes de programación de alto nivel que incluyan los componentes principales de cualquier lenguaje de programación tradicional, a saber: estructuras de control y estructuras de datos.

5. El lenguaje Java, pese a ser interpretado, es una opción que puede ser tomada en cuenta para el desarrollo de aplicaciones que demanden un alto rendimiento en términos de velocidad de procesamiento. Esto es debido a las características mejoradas de las nuevas implementaciones de máquinas virtuales de Java.

6. Las aplicaciones basadas en Java como el sistema desarrollado para Linkotel S.A. como parte de este proyecto de tesis, pueden ser portadas a un gran número de plataformas distintas. Esto permite el poder decidir de manera flexible cuál es el equipo y sistema operativo más conveniente para llevar a cabo la ejecución de la aplicación.

7. Fue posible encontrar una forma de acceder de manera automatizada a la información de tráfico generada por la central telefónica de Linkotel S.A., al considerar a este equipo como un computador común con Windows NT. Esto se debe a que la central Huawei utilizada por Linkotel tiene embebido un computador con el sistema operativo mencionado, y los archivos generados por esta son depositados en el disco duro de este computador.

6.2 RECOMENDACIONES

A continuación se listan las recomendaciones emitidas para este proyecto de tesis:

1. Se recomienda configurar cuidadosamente los procesos de transformación en lo que concierne a la asignación de memoria para los buffers de lectura, ya que esto tiene una incidencia directa en la eficiencia con la cual se llevará a cabo el proceso, así como en el consumo de los recursos disponibles del equipo. Idealmente el búffer de lectura debería ser igual al tamaño medio de los archivos generados por la central, pero este número está limitado a la cantidad de memoria física que posea el equipo, al número de procesos concurrentes de carga que deseen ejecutarse y a la cantidad y tipo de sistemas que estén ejecutándose en el mismo equipo de hardware.
2. Dado que la especificación actual de Java no incluye una implementación de XPath como parte del JDK, se ha utilizado para este proyecto la implementación de XPath proporcionada por Apache mediante Xalan. Se recomienda migrar el código hacia

una implementación estándar de XPath cuando esta exista como parte del JDK en futuras versiones de Java.

3. Para procesar grandes volúmenes de información, se recomienda trabajar en la medida de lo posible con formatos de salida de datos sencillos, ya que el formateo complejo de los datos interpretados provenientes de la central, afectará en algo la velocidad de ejecución de la aplicación.
4. La implementación actual del sistema permite escoger formatos de salida de datos basados en plugins, es decir implementaciones de clases que realizan el formateo de la información ya procesada de una manera determinada. Debido a esto el sistema puede producir tantos formatos de salida como plugins tenga añadidos. Se recomienda implementar a mediano plazo una interfaz gráfica que permita la definición de los formatos de salida de datos, de manera similar a como se lo hace con los formatos de entrada.
5. Se recomienda realizar como siguiente paso para la construcción de un sistema de mediación telefónica, el diseño e implementación de los módulos de Agregación y Correlación descritos en la sección 1.1.3. De esta manera el sistema será capaz de filtrar

información redundante e incompleta. Una vez realizados estos módulos proceder al diseño e implementación del módulo de tarificación que permitirá asociar un valor monetario a los datos de consumos y finalmente el módulo de entrega de datos que será responsable de distribuir la información ya procesada a las distintas aplicaciones de soporte de negocios que la requieran.

APÉNDICE A: MANUAL DE USUARIO

En este apéndice se muestra el manual de usuario del sistema desarrollado como parte de este proyecto de tesis.

Sección 1: Instalación

1. Requerimientos del Sistema

- Sistema operativo soportado por Java (prácticamente todos los Sistemas Operativos modernos)
- Tener el “Java Runtime Environment” (JRE) versión 1.4.2 o superior instalado. Alternativamente se puede usar el Java Standard Development Kit (JDK) con los mismos requerimientos de versión.

- La aplicación requiere un mínimo de 64MB de memoria libre para correr. Pero hay que tomar en cuenta que para mejorar el rendimiento durante el procesamiento, se requiere de más memoria, la cantidad requerida está directamente relacionada con el tamaño de los archivos de entrada y el formato usado. La cantidad de memoria recomendada es 256MB.

2. Instalación de Java

Este capítulo esta escrito para aquellos que no tengan Java instalado. Al instalar Java se puede escoger entre dos paquetes: El Java Runtime Enviroment (JRE) y el Java Standard Development Kit (SDK); el SDK es orientado a programadores de Java, mientras el JRE es el requerimiento único para lograr correr aplicaciones hechas en Java. Por tanto, esta guía se centra en la instalación del JRE.

1. Obtener el JRE mas reciente disponible de Sun Microsystems (<http://java.sun.com>). El archivo obtenido tendrá un nombre dependiente del sistema Operativo en cuestión (los nombres de archivo son ejemplos basados en la versión estable mas reciente disponible al momento de escritura de este documento):

1. Para sistemas Windows: `j2re-1_4_2_05-windows-i586-p.exe`

2. Para sistemas Unix/Linux: j2re-1_4_2_05-linux-i586.bin

2. Instalación del JRE:

1. Bajo Windows:

1. Ejecutar el instalador, aceptar la licencia y escoger un directorio destino para el JRE. (Ejemplo: c:/java/jre-1.4.2_05)
2. Fijar las variables de entorno JAVA_HOME y CLASSPATH

2. Bajo Windows Nt/XP:

1. Ingresar a las propiedades del Sistema (Accesible desde “Mi PC” -> propiedades)
2. Ingresar a la sección que dice “avanzado” y luego “variables de entorno”
3. Encontrar la entrada JAVA_HOME dentro de las variables del sistema, si no existe, crearla.
4. Cambiar el valor de la entrada al directorio en el cual el JRE fue instalado (Ejemplo: c:/java/jre-1.4.2_05)
5. Revisar que la variable de entorno CLASSPATH tenga el valor “\$JAVA_HOME/classes” (si no existe, crear la variable).

3. Bajo Windows 95/98/ME:

1. Editar el archivo autoexec.bat (ubicado en C:/)
2. Agregar (o editar) la entrada para el JAVA_HOME (Ejemplo: set JAVA_HOME=c:/java/jre-1.4.2_05)

3. Revisar que exista una entrada dentro del PATH para el directorio 'bin' del JAVA_HOME (Ejemplo: set PATH=\$PATH:JAVA_HOME/bin)
4. Agregar (o editar) la entrada para el CLASSPATH (set CLASSPATH=\$JAVA_HOME/classes)
4. Bajo Linux (instrucciones de línea de comando, se requiere acceso de administrador para la instalación global del JRE)

1. Mover el archivo de instalación al lugar donde se va a instalar Java (se recomienda /opt) y cambiar a ese directorio:

```
mv j2re-1_4_2_05-linux-i586.bin /opt/  
cd /opt
```

2. Agregar la bandera de ejecución al archivo obtenido (ya que por omisión los archivos obtenidos del Internet no son ejecutables):

```
chmod +x j2re-1_4_2_05-linux-i586.bin
```

3. Ejecutar el instalador (el JRE sera instalado en un subdirectorio de nombre basado en la versión como /opt/sun-jdk-1.4.2_05)

```
./j2re-1_4_2_05-linux-i586.bin
```

4. Fijar las variables de entorno JAVA_HOME y CLASSPATH.

Para fijar estas variables a nivel del sistema, agregar las líneas a `/etc/profile`, para fijar las variables a nivel de un usuario, agregar las líneas a `~/.bash_profile`:

```
export JAVA_HOME=/opt/ sun-jdk-1.4.2_05
export CLASSPATH=$JAVA_HOME/classes
```

3. Instalación de paquetes externos

Hay ciertos paquetes de Java necesarios para la ejecución del sistema que no vienen incluidos en el JRE ni el JDK oficiales de Java. Esta sección discute como obtener e instalar estos paquetes adicionales. Estos paquetes son parte del Java Web Services Developers Pack y eventualmente serán parte oficial del JRE.

Los paquetes (en formato `.jar`) están incluidos dentro del directorio “endorsed” (incluido en la aplicación), el directorio endorsed debe ser movido al directorio de instalación de Java, dentro del subdirectorio “lib” del mismo. Por ejemplo, si el JRE fue instalado en `C:/java/jre-1.4.2_05`, el directorio endorsed debe ser movido a `C:/java/jre-1.4.2_05/lib/endorsed`. Los archivos contenidos dentro de endorsed (para referencia) son:

- dom.jar
- sax.jar
- xalan.jar
- xercesImpl.jar
- xslt.jar

4. Instalación del Programa

La instalación del programa es bastante simple. Sólo hay que copiar el directorio de la aplicación al lugar donde uno desea instalarlo. Como el programa es orientado a mono-usuario, no existe una ruta recomendada, queda a criterio del usuario decidir donde colocar el programa.

Para lanzar el programa se ejecuta el “lanzador” de la aplicación desde el directorio del mismo.

- Bajo Windows ejecutar mediacion.bat
- Bajo Sistemas Operativos basados en Unix ejecutar mediacion.sh

Sección 2: Uso del Sistema

Capítulo 1. Términos Usados y Definiciones

Durante el resto del manual se harán referencias a los siguientes términos frecuentemente:

Formato: Un formato es un conjunto de “campos” y “grupos” los cuales en conjunto describen la estructura de un archivo; usando el Formato correspondiente, la aplicación será capaz de leer o trabajar sobre un archivo cualquiera (cuya estructura interna se adhiera al formato en cuestión). La sección de “Definición de Formatos” trata exclusivamente sobre la creación y modificación de archivos XML que describen a un formato en particular. Los formatos actualmente se dividen en tres categorías según sus características mas generales:

- **Formatos Binario** (también conocidos como formatos binario de estructura fija): Son formatos almacenados de manera binaria que siguen un ordenamiento rígido de sus campos. Los campos leídos del archivo deben aparecer en el mismo orden que se encuentran en la definición del formato (con una ligera excepción: ver la definición de “grupos”).

- **Formatos Binario estructurados por ID:** El orden de los campos no es fijo, y es posible que un campo aparezca varias veces dentro del mismo registro. La identificación de los campos es hecho por el ID del mismo. Se lee el ID del archivo, se compara contra la definición de formatos y se interpreta los datos leídos según la estructura del campo con el ID correspondiente.
- **Formatos de Texto por Separadores:** El archivo es de tipo texto (al contrario de los dos formatos anteriores que son binarios) y se identifican a los campos y registros por medio de separadores.

Registro: Conjunto de información leída de un archivo que satisface las especificaciones de un Formato en particular. Se puede decir que el formato describe la estructura de un registro en particular, y los archivos que se leen contienen una serie de registros.

Campo: En el contexto de Formatos, un campo se refiere a una entidad atómica dentro del formato. En el fondo, un formato esta constituido exclusivamente de campos. Un campo es un dato simple con propiedades propias del tipo de campo. Por ejemplo, un campo puede ser una fecha, de tipo binario, longitud 3 bytes, de máscara “YYYYMMDD”. De manera general, existen dos tipos de campos: aquellos que son leídos de un archivo (como el

ejemplo anterior) y aquellos que son “generados”, es decir, tienen un valor que se calcula en base a una ecuación cualquiera (que puede contener el valor de campos previamente definidos).

Grupo: Colección ordenada de campos. La idea de los grupos es proveer flexibilidad a la definición de formatos cuando la estructura de los registros es dinámica. Por ejemplo, puede que la definición de los primeros cinco campos siempre sea la misma, pero la definición de los campos siguientes varía dependiendo del valor de los campos anteriores (Si el campo cinco es cero, los siguientes campos usan reglas distintas a si el campo hubiera sido diferente de cero). Los grupos se pueden ver como nodos de un árbol, una vez que se termina de recorrer el grupo actual, hay que decidir cual de los sub-grupos hay que elegir después; para este propósito es que cada grupo tiene asociado una expresión que define (si se evalúa a verdadero) si el grupo debe ser seleccionado o no.

Proceso: Es el enlace que existe entre archivos de origen, archivos destino, formatos y reportadores. Un proceso define que archivos usar como origen, que nombre ponerle a los archivos generados, que formato usar para interpretar los archivos de entrada, y que reporteador usar para generar los archivos de salida. Los procesos también definen utilidades extras como:

- Repetición de ejecuciones: Se puede configurar un proceso para que se ejecute varias veces con una cantidad de tiempo intermedio (por ejemplo, ejecutar la transformación de datos 24 veces con un lapso de una hora entre ejecuciones)
- Usar directorio temporal: Copia los archivos de origen a otro lugar antes de procesarlos. Útil en situaciones donde el archivo de origen se puede volver inaccesible durante la transformación de los datos.
- Control sobre la lectura: Se puede definir si se desea leer todo el archivo origen, o sólo una parte de él; se especifican offsets de inicio y cantidad de registros a leer.
- Políticas sobre archivos existentes: Si el archivo destino ya existe, se lo puede sobrescribir, se puede abortar el proceso, o se puede anexar los datos al archivo existente.

El sistema tiene dos áreas designadas para el manejo de Procesos: “Definición de Procesos” es el área donde se administra la generación de los procesos y se definen sus valores; “Ejecución de Procesos” en cambio es el área donde se ve el estado de procesos que se encuentran en ejecución actualmente.

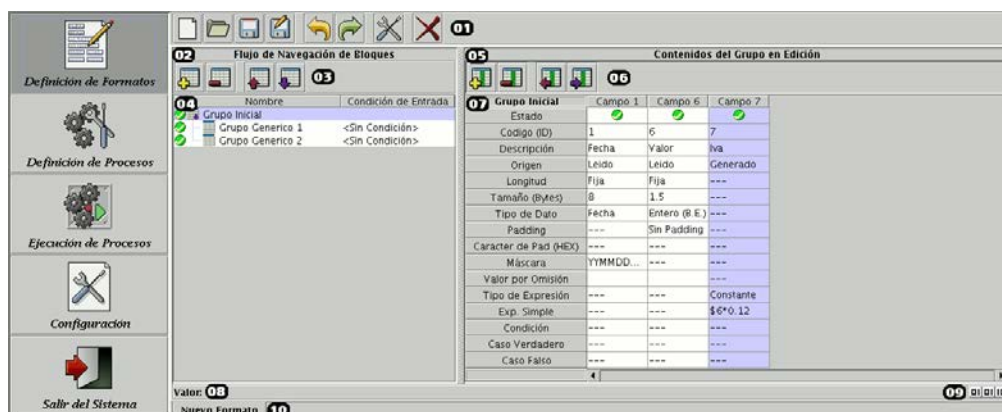
Reporteador: El Formato es quien define como leer la información del archivo origen, pero el reporteador es quien define qué hacer con la

información leída. Los reportadores son el proceso inverso de los formatos, definen como manipular la información leída y como debe ser almacenada en el archivo destino.

Log: Los “logs” de información son imprescindibles en este tipo de sistemas. Un log es un archivo en el cual se van guardando los detalles de lo que está pasando durante la ejecución de los procesos, es una útil fuente de información histórica sobre el uso de los procesos.

Capítulo 2. La Sección “Definición de Formatos”

Al iniciar el sistema, inicialmente se muestra la sección de definición de formatos. La ventana aparece relativamente vacía al principio ya que no hay ningún formato cargado. Pero bajo uso normal la ventana tiene el siguiente aspecto:



A continuación se explican las diferentes secciones de esta interfaz:

1) Barra de Herramientas principal: Aquí se acceden a opciones globales que afectan al documento. Los botones y su funcionalidad se explican a continuación:

- **Nuevo:** Genera una plantilla para describir un nuevo formato. Antes de generar el formato se debe escoger de entre una pequeña lista de “tipos” de formatos disponibles. Estos “tipos de formato” existen debido a cambios fundamentales entre los formatos (por ejemplo, formatos almacenados de tipo binario contra formatos almacenados en formato de texto).
- **Abrir:** Abre un formato existente. Las definiciones de los formatos son guardados en XML, y el diálogo permite seleccionar cualquier documento de extensión XML, pero este es revisado internamente

para asegurarse que el archivo sea compatible con la aplicación (de no ser así, no se podrá abrir el archivo seleccionado).

- **Guardar / Guardar Como:** Permiten almacenar el descriptor del formato actual. Hay que notar que si la definición del formato esta incompleta/ es inválida no se permitirá guardarlo por motivos de integridad (ya que formatos incompletos no deben ser usados en la sección de ejecución de procesos). Hacer que la definición de formatos sea válida no es difícil, solo requiere que cada uno de los campos sea válido (cada campo tiene su icono de estado) y que las condiciones de los grupos sean válidos también.
- **Undo/Redo:** Para facilitar la corrección de errores cometidos durante la definición de procesos esta sección provee la conocida funcionalidad de undo/rede implementada en todo programa moderno de edición.
- **Configuración:** Abre el diálogo de configuración para la definición de documento actual. Los contenidos de este diálogo varían dependiendo en el tipo de formato que se está especificando. Ver el capítulo 5 para mas detalles.
- **Cerrar:** Cierra la definición de formato actual. Si el documento fue modificado, primero se pregunta al usuario si desea grabar los cambios.

2) Área de Grupos: En esta sección se muestra una representación visual de los grupos que conforman al formato actual. Esta sección sólo tiene sentido para formatos de estructura fija (en la cual el orden de los campos en los archivos de lectura es el mismo que el especificado en la definición del formato).

3) Barra de Herramientas para el área de grupos: Ofrece las funcionalidad necesaria para la manipulación de grupos completos: Creación de grupos, eliminación de grupos, y desplazamiento (re-ordenamiento) de grupos. Hay que notar que el re-ordenamiento sólo puede ser usado entre grupos que pertenecen al mismo nivel del árbol (entre nodos hermanos).

4) El árbol de grupos: Aquí se muestra la estructuración de los grupos, y se permite acceso a la edición de los datos generales de cada grupo:

Estado: El estado indica la validación de la condición del grupo (mas no de los contenidos del mismo)

- Nombre: Usado para identificar a los grupos y su propósito
- Condición de Entrada: Es la expresión booleana que se evalúa para decidir si el grupo en cuestión es el grupo a seguir. Al terminar el procesamiento del grupo padre, se evalúa la expresión de cada hijo

en el orden que aparecen en el formato, la primera expresión que retorne “verdadero” definirá el camino a seguir. (Ver la sección sobre el diálogo de Expresiones para obtener más detalles sobre expresiones)

5) Sección de Campos: Aquí se muestran y manipula la información referente a los campos contenidos dentro del grupo actual.

6) Barra de herramientas para los campos: Ofrece acceso a las operaciones básicas de manipulación de campos: Creación y eliminación de campos, así como reorganización de campos.

7) En esta tabla se presentan los contenidos del grupo actual. Cada columna representa un campo, y las filas indican las diferentes propiedades de cada campo. La cantidad de filas disponibles dependen del tipo de campo. Si hay cierto tipo de información que no se usa en un cierto formato, la fila correspondiente no existirá, en cambio, si cierta información no es necesaria (dadas las circunstancias del campo actual) esta no será editable y tendrá un valor de “---” (Ver el capítulo 5.4 para más detalles sobre los atributos de los campos).

8) Barra de Estado: A través de esta barra se notifica al usuario de eventos que no requieren la presencia de un diálogo modal de información (el cual interrumpe el ritmo de trabajo del usuario), esta barra también es usada para mostrar el valor del atributo que tiene el enfoque (útil ya que el espacio dentro de la columna es relativamente pequeño como para mostrar el contenido de los atributos).

9) Aquí se muestra una representación gráfica del tipo de formato en que se está trabajando actualmente (la imagen contiene como texto de ayuda la descripción del tipo de formato que se está definiendo), este es el único lugar donde se puede ver cual es el tipo de formato que se está generando actualmente.

10) Aquí se muestran al usuario todos los formatos actualmente abiertos en distintas pestañas. Cada pestaña tiene el nombre del documento correspondiente.

2.1 El diálogo de opciones para Formatos

Los diálogos de configuración de formatos varían de formato a formato; a continuación se explican las opciones más comunes en estos diálogos:

Formato Binario de Campos por ID

ID de los registros (en decimal): 1 **01**

Tamaño del campo del ID: 8 bytes **02**

Longitud de la magnitud de los Campos: 8 bytes **03**

Unidad de la magnitud de los Campos: byte **04**

Leer la magnitud del Registro **05**

Longitud de la magnitud del Registro: 0 bytes **06**

Unidad de la magnitud del Registro: byte **07**

Usar un separador cada X registros **08**

Número de Registros agrupados: 0 **09**

Longitud del Separador: 0 bytes **10**

Codificación del Archivo: UTF-8 **11**

Comentario: **12**

OK Cancel

Formato de Texto con Separador

Codificación del Archivo: UTF-8

Caracter Separador entre Campos: | **13**

Caracter Separador entre Registros: [enter] **14**

Comentario:

Otro formato de ejemplo generado para el manual de Usuario.

OK Cancel

1) *ID de los Registros*: Usado en formatos basados por ID. Ya que los campos no tienen orden fijo, es imposible definir donde empiezan y terminan los registros, a menos que estos también sean un “campo” cuyo único propósito es identificar los límites de los campos. Aquí se ingresa el ID para tal caso (ver la sección 2.4 para mas detalles).

2) *Tamaño del Campo del ID*: En un formato basado por ID, es necesario definir cual es el tamaño que el ID usa dentro del mismo (debemos leer 1 byte para saber cual es el ID? Dos bytes? Solo 3 bits ya que existen sólo 8 campos?), este campo sirve para definir dicho valor.

3) *Longitud de la Magnitud de los campos*: Al igual que en el punto 2, cuando se trata de un formato de campos de longitud variable, es necesario especificar cuantos bits/bytes son necesarios leer para identificar la longitud del campo (Si este campo tiene valor de 3 bits, eso quiere decir que la máxima longitud de campos sera de 111b, es decir, 7 bytes).

4) *Unidad de la Magnitud de los campos*: Una vez leída la magnitud del campo, se tiene sólo un número, aquí se define en que unidad se debe interpretar la magnitud previamente leída.

5) *Leer la Magnitud del Registro*: En ciertos formatos es deseable especificar cual es la longitud total del registro. Habilitar dicha lectura habilita los siguientes dos campos que funcionan de manera similar a los campos tres y cuatro.

6) *Longitud de la Magnitud del Registro*: Especifica el tamaño dentro del archivo binario en el cual está almacenado la magnitud del registro a leer.

7) *Unidad de la Magnitud del Registro*: Especifica la unidad en la viene el campo atributo anterior.

8, 9, 10) *Usar un Separador cada X registros*: En ciertos formatos el archivo origen no contiene todos los registros pegados uno después del otro, sino que cada cierto número de registros existe un “separador” (o delimitador, probablemente usado para comprobar la integridad del archivo o porque se requiere que cada cierto número de registros ocupe una cantidad fija). En esta sección se fija los parámetros para dicho separador. Se especifica la frecuencia con la que aparece el separador y su tamaño.

11) *Codificación*: Este campo es usado para especificar como esta almacenado el texto. En formatos basados en texto, este valor afecta a todo el archivo de origen, en los formatos binarios, este valor afecta a aquellos campos marcados como “tipo texto”. El rango de codificaciones permitidas no cubre todas las existentes, pero si cubren la base fundamental que debe ser soportada en todo el mundo (ISO8859-1 es la codificación latino-americana).

12) *Comentario*: Aquí se puede anexar cualquier tipo de información que se desee, por ejemplo, se puede ingresar el rango de fechas para el cual el formato es válido.

13) *Separador entre campos*: En los formatos basados en texto, aquí va el carácter separador usado para identificar donde empiezan/terminan los campos de un registro.

14) *Separador entre registros*: Igual que en el campo 8, este separador identifica la terminación de los registros y el inicio del siguiente.

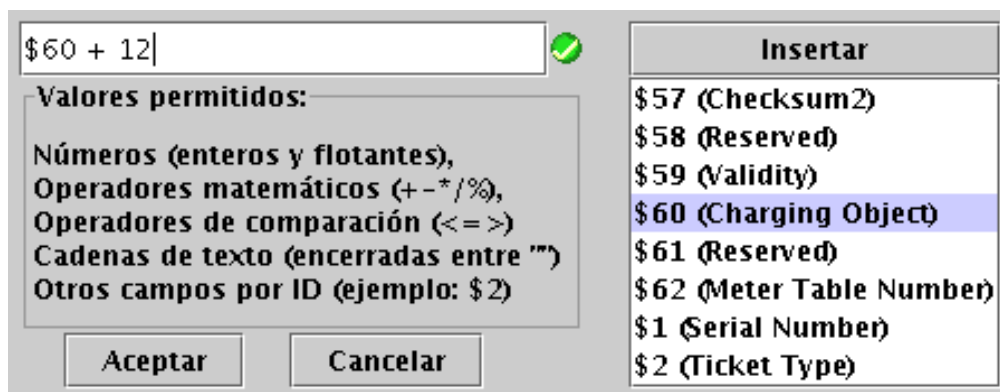
2.2 El diálogo de definición de Expresiones:

En varias partes de la sección de definición de Formatos es necesario especificar una expresión. Las expresiones son generalmente de dos tipos:

Booleanos: Donde el resultado de la expresión debe ser “verdadero” o “falso”. Expresiones condicionales son las usadas en la definición de grupos.

Genérico: La expresión es más como una ecuación, y el resultado de esta en lo general será un valor numérico, aunque también puede ser un valor booleano, o una constante. Este es el tipo de expresión que se permite dentro de ciertos atributos de los campos.

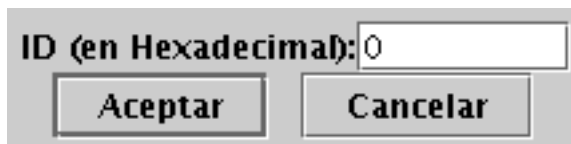
El diálogo de expresiones se muestra a continuación:



El diálogo no es complejo, tiene un área donde se ingresa la ecuación. El icono a la derecha indica que la expresión ingresada es válida, y se actualiza cada vez que el cursor cambia de foco a otra área (o cuando el usuario presiona ENTER mientras se escribe la ecuación, en tal caso, si la expresión ingresada es correcta, el foco se mueve al botón de Aceptar).

En la parte derecha existe una lista de campos, esta lista contiene a todos los campos válidos en el contexto del campo actual (por ejemplo, un campo no puede hacer referencia a otros campos que se definen después). En las ecuaciones los campos son referidos por medio de su ID, pero en la lista se provee además el nombre del campo ya que este es más fácil de identificar.

2.3 El diálogo de definición de Ids



Cuando se crean nuevos campos en el formato de campos definidos por ID, lo primero que hay que hacer es ingresar un ID para estos campos. Como los IDs son valores numéricos cuyo único propósito es identificar a los campos, estos son almacenados en hexadecimal.

2.4 Explicación de los atributos de los Campos

Aquí se explican la mayoría de los atributos de definen a un campo en particular:

- *Código (ID)*: Cada campo debe tener un identificador único. El propósito de este es dar un “nombre único” a cada campo. Este es el nombre que se usa en las expresiones para hacer referencia al campo. En la mayoría de formatos este valor se asigna automáticamente al crear un nuevo campo, la excepción es en el formato de campos por ID, ya que el ID de cada campo es también parte de las especificación del formato.

- *Descripción:* Permite la asignación de un nombre arbitrario al campo. Útil para definir cual es el propósito del campo en cuestión.
- *Origen:* Se refiere al modo de generación del campo. Si el campo es especificado como “leído”, entonces se indica que el campo se lee directamente del archivo de origen, mientras campos especificados como “definidos” son aquellos que se generan a partir de expresiones (las cuales pueden ser constantes o depender del valor de otros campos)

Los siguientes son atributos propios de los campos de origen leído:

- *Longitud:* Aquí se define si el campo es de longitud variable o fija. Cuando el campo es de longitud fija, el siguiente atributo se vuelve editable, el cual define el tamaño del campo.
- *Tipo de Dato:* Definir el tipo de dato ayuda a definir como el contenido leído debe ser interpretado. Por ejemplo, el tipo “hexadecimal” especifica que los datos no deben pasar ninguna transformación. El dato de tipo entero tiene dos variaciones: “big endian” y “little endian”, estos son términos técnicos que especifican cual es el bit más significativo al momento de leer la información.

- *Padding*: Para los casos en que la información leída no use todo su espacio asignado (por ejemplo, un campo de tamaño fijo a 24 bytes que contenga una cadena de texto en su interior, no siempre usará todos los 24 bytes disponibles), el pad es el valor hexadecimal que se usará para identificar la finalización de la sección de datos (si el padding es "F", se sobreentiende que después de los datos vendrá una cadena de 111... hasta terminar el tamaño designado al campo).
- *Máscara*: Aplicable solo a campos tipo fecha, la máscara especifica como se interpreta el valor leído del archivo (ejemplos de máscaras comunes son YYYYMMDD y MMDDYYYY)
- *Valor por Omisión*: Es posible especificar un valor que deben tomar los campos cuando no tienen un valor en el archivo de origen (por ejemplo, si el campo tiene padding y el contenido de todo el campo son puros caracteres de padding)

Los siguientes son atributos propios de los campos generados:

- *Tipo de Expresión:* Los campos generados se clasifican en dos categorías: Aquellos generados por expresiones simples, y aquellos generados por expresiones condicionales.
- *Expresión Simple:* En el caso de una expresión simple, se aplica una expresión única y el resultado de esta es directamente el valor del campo generado. Se usan expresiones simples también para generar campos con valores constantes.
- *Condición:* Si la expresión es de tipo condicional, es necesario especificar tres expresiones para completar la definición de un campo definido. La “condición” es una expresión de tipo booleano, cuya evaluación define cual de las dos siguientes expresiones será ejecutada.
- *Caso Verdadero:* Expresión que será ejecutada en caso de el la evaluación de “condición” resulte verdadera. Esta expresión sigue las mismas características de la “expresión simple”. Si la condición resulta falso, entonces la expresión de “Caso falso” es la que se ejecuta.

Capítulo 3. La sección “Definición de Procesos”

Esta es la sección en la que se definen procesos. La interfaz está diseñada de tal manera que se pueden manipular varios procesos a la vez. Los procesos definidos son guardados automáticamente cuando se cierra la aplicación. La interfaz se describe a continuación:

	Nombre	Momento d.	Intervalo de	Nú.	Nú.	Of.	Buffer	Formato	Reportea	Archivo de Origen	Archivo de Destino
1	HUAWEI_SEP	0 segundos	0 horas	1	-1	0	2000	huawei.xml	separador	archivosLeidos/huawei/20040123.BIL	archivosGenerados/20040
2	HUAWEI_XML	0 horas	0 horas	1	-1	0	2000	huawei.xml	xml	archivosLeidos/huawei/20040123.BIL	archivosGenerados/20040
3	HUAWEI_SEP_2	0 horas	0 horas	1	-1	0	20000	huawei.xml	separador	archivosLeidos/huawei/20040419.BIL	archivosGenerados/20040
4	HUAWEI_XML_2	0 horas	0 horas	1	-1	0	20000	huawei.xml	xml	archivosLeidos/huawei/20040419.BIL	archivosGenerados/20040
5	HUAWEI_TEXT	0 horas	0 horas	1	-1	0	2000	huawei_text.xml	separador	archivosLeidos/huawei_text/20040	archivosGenerados/20040
6	HUAWEI_TEXT...	0 horas	0 horas	25	-1	0	20000	huawei_text.xml	xml	/home/wallex/workspace/Mediacion...	archivosGenerados/20040
7	ERICSSON_TXT	0 horas	10 segundos	3	-1	0	500000	ericsson.xml	separador	archivosLeidos/ericsson/TFILE00-4...	archivosGenerados/ericss
8	ERICSSON_XML	0 horas	0 horas	1	-1	0	500000	ericsson.xml	xml	archivosLeidos/ericsson/TFILE00-4...	archivosGenerados/ericss
9	ALCATEL_TXT	0 horas	0 horas	1	-1	0	500000	alcatel.xml	separador	archivosLeidos/alcatel/000DCTSA.M...	archivosGenerados/alcatel
10	ALCATEL_XML	0 horas	0 horas	1	-1	0	500000	alcatel.xml	xml	archivosLeidos/alcatel/000DCTSA.M...	archivosGenerados/alcatel

En la parte superior se encuentra la barra de herramientas para controlar los procesos. Los botones sirven para:

- Seleccionar todos los procesos
- Invertir la selección actual
- Agregar (generar) nuevos procesos
- Remover (eliminar) procesos
- Editar los datos de los procesos seleccionados

- Ejecutar los procesos

La tabla muestra todos los procesos y sus correspondientes valores. Estos son editables directamente desde la tabla, pero también existe un diálogo que muestra todas las propiedades de un proceso en particular de una manera más accesible (para cuando se quieran editar múltiples partes del proceso a la vez, como durante su creación).

Antes de ejecutar un proceso, también se presenta el diálogo con las propiedades del proceso. Cualquier cambio que se realice en este diálogo sólo afectará al proceso que se ejecuta, pero no afectará a la instancia del proceso que se encuentra en la sección de “definición” de procesos.

3.1 El diálogo de Procesos

Permite un acceso rápido a la edición de los parámetros de un proceso en particular. A continuación se explican sus diferentes campos:

Nombre:	01	HUAWEI (Ejemplo por Separador)
Momento de Inicio:	02	10.00 minutos
Intervalo entre Ejecuciones:	03	1.00 horas
Numero de Ejecuciones:	04	24
Offset Inicial (Bytes):	05	0
Numero de Registros:	06	-1
Buffer de Lectura (Bytes):	07	20,000
Formato:	08	huawei.xml ...
Reporteador:	09	separador
Archivo de Origen:	10	archivosLeidos/huawei/20040419.BIL ...
Archivo de Destino:	11	archivosGenerados/20040419.txt ...
En caso de que el archivo destino exista:	12	ANEXAR
<input checked="" type="checkbox"/> Usar directorio de trabajo:	13	ex/workspace/Mediacion/temp/temp ...
<input type="button" value="Aceptar"/> <input type="button" value="Cancelar"/>		

1) *Nombre*: El nombre del proceso puede ser cualquier cadena de texto, su uso es lograr facilitar la identificación de los aspectos involucrados en el proceso.

2) *Momento de Inicio*: Establece un intervalo de tiempo “antes” de iniciar la ejecución (por ejemplo, se puede fijar un proceso para que empiece una hora después de que el usuario solicita que se arranque)

3) *Intervalo entre Ejecuciones*: También se puede definir que un proceso se ejecute varias veces después de marcado como “iniciado”. Este valor indica cuanto tiempo esperar antes de continuar con la siguiente iteración de la ejecución.

4) *Número de Ejecuciones*: Como su nombre lo indica, el número de ejecuciones que el proceso debe realizar antes de ser nominado como “terminado”.

5) *Offset Inicial*: No siempre se desea que el procesamiento del archivo sea desde el principio (por ejemplo cuando ya se sabe que cierta parte del archivo ya fue procesado con anterioridad), aquí se especifica desde donde se debe empezar a leer el archivo de origen. Hay que notar que este campo es usado automáticamente cuando el proceso tiene fijo un número de iteraciones (para no volver a procesar las mismas partes del archivo ya procesadas con anterioridad).

6) *Número de Registros*: Especifica cuantos registros se desean leer del archivo de origen. El valor -1 indica que hay que leer todos los registros hasta el final del archivo.

7) *Buffer de Lectura*: En casos en que el archivo sea muy grande como para tenerlo todo en memoria, es recomendable procesar el archivo en partes, el buffer de lectura especifica que tan grande son los trozos de archivo que se procesan a la vez. Recordar que a menor el tamaño del buffer de lectura, más tiempo demorará el procesamiento de los archivos.

8) *Formato*: Aquí se especifica el archivo descriptor de formatos a usarse para este proceso.

9) *Reporteador*: Aquí se escoge de entre los reporteadores disponibles para definir el formato de salida de los archivos.

10) *Archivo de Origen*: Especificación del archivo del cual extraer los datos al momento de ejecución.

11) *Archivo Destino*: El nombre del archivo que se generará con este proceso.

12) *Política a usar cuando el archivo destino ya existe*: Existen algunas opciones de cómo el proceso reaccionará ante el caso en que el archivo destino ya exista:

- *Sobreescribirlo*: El archivo destino será eliminado.
- *Anexar*: Los nuevos datos serán agregados al final del archivo existente.
- *Abortar*: El proceso será cancelado.

13) *Usar Directorio de Trabajo*: Si se define un directorio de trabajo, el archivo de origen será copiado a este antes de procesarlo. Esta opción es útil para casos en que la disponibilidad al archivo de origen no sea garantizada (como al acceder a archivos a través de la red).

Capítulo 4. Sección de “Ejecución de Procesos”:

En esta sección se presenta la información sobre procesos que se han ejecutado (o se están ejecutando). La información presentada en esta sección no es persistente entre ejecuciones del sistema (es decir, no se guarda al cerrar el sistema).

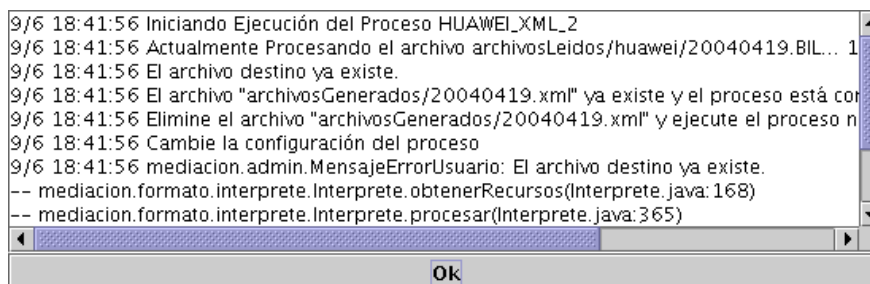
	Nombre	Estado	Log	Archivo de Origen	Archivo de Destino	Formato de Archivo	Status Carga
1	HUAWEI_SEP	Finalizado	...	archivosLeidos/huawei/200...	archivosGenerados/20040...	huawei.xml	0%
2	HUAWEI_X...	Finalizado	...	archivosLeidos/huawei/200...	archivosGenerados/20040...	huawei.xml	0%
3	HUAWEI_S...	Finalizado	...	archivosLeidos/huawei/200...	archivosGenerados/20040...	huawei.xml	0%
4	HUAWEI_X...	Finalizado	...	archivosLeidos/huawei/200...	archivosGenerados/20040...	huawei.xml	0%
5	HUAWEI_T...	Finalizado	...	archivosLeidos/huawei/text...	archivosGenerados/20040...	huawei_texto.xml	0%
6	HUAWEI_T...	Procesando 1...	...	/home/wallex/workspace/...	archivosGenerados/20040...	/home/wallex/workspace/...	3%
7	ERICSSON...	Finalizado	...	archivosLeidos/ericsson/TT...	archivosGenerados/ericcso...	/home/wallex/workspace/...	0%
8	ERICSSON...	Finalizado	...	archivosLeidos/ericsson/TT...	archivosGenerados/ericcso...	/home/wallex/workspace/...	0%
9	ALCATEL...	Finalizado	...	archivosLeidos/alcatel/000...	archivosGenerados/alcatel.txt	/home/wallex/workspace/...	100%
10	ALCATEL...	Procesando 1/1	...	archivosLeidos/alcatel/000...	archivosGenerados/alcatel...	alcatel.xml	6.1%

La barra de herramientas ofrece las siguientes operaciones sobre los procesos en ejecución:

- Seleccionar todos los Procesos
- Invertir selección actual
- Remover un proceso (lo detiene primero de ser necesario)
- Detener un proceso (no lo remueve, permite ser reiniciado posteriormente)
- Arrancar/Reiniciar un proceso (equivalente a detenerlo, removerlo, y empezarlo de nuevo con los mismos parámetros desde la ventana de definición de procesos)

A continuación se explican cada una de las columnas en la tabla de información:

- *Nombre*: El nombre del proceso en cuestión.
- *Estado*: Muestra el estado del proceso. El icono de estado indica si ha sucedido algún evento que valga la pena investigar (en cuyo caso sería una buena idea revisar el log del proceso); junto al icono aparece una descripción escrita de lo que está pasando con el proceso actualmente.
- *Log*: Este botón permite acceso al log del proceso. Desde ahí se puede uno informar sobre varios acontecimientos que pueden suceder mientras se ejecutan los procesos.



```

9/6 18:41:56 Iniciando Ejecución del Proceso HUAWEI_XML_2
9/6 18:41:56 Actualmente Procesando el archivo archivosLeidos/huawei/20040419.BIL... 1
9/6 18:41:56 El archivo destino ya existe.
9/6 18:41:56 El archivo "archivosGenerados/20040419.xml" ya existe y el proceso está cor
9/6 18:41:56 Elimine el archivo "archivosGenerados/20040419.xml" y ejecute el proceso n
9/6 18:41:56 Cambie la configuración del proceso
9/6 18:41:56 mediacion.admin.MensajeErrorUsuario: El archivo destino ya existe.
-- mediacion.formato.interprete.Interprete.obtenerRecursos(Interprete.java:168)
-- mediacion.formato.interprete.Interprete.procesar(Interprete.java:365)

```

- *Archivo de Origen:* El archivo que se está procesando.
- *Archivo Destino:* El archivo al cual se escriben los resultados del procesamiento.
- *Formato del Archivo:* El descriptor de formato usado en el procesamiento.
- *Estado de la Carga:* El Muestra el progreso actual del proceso. Hay que notar que esta barra de progreso corresponde a cada ejecución individual (es decir, si el proceso se ejecuta dos veces, la barra se llenará dos veces durante la ejecución del mismo).

Capítulo 5. Configuración de la Aplicación

En esta sección se configuran las opciones globales del sistema. Existen secciones para las opciones relacionadas con la generación de procesos y el log de eventos entre otras.

01 Valores por Omisión en la Definición de Procesos

Tamaño del Buffer de Lectura: 1001016 bytes

Formato: huawei.xml **Elegir**

Reporteador: separador Ejemplo: campo1|campo2|campo3|campo4|campo5| ...

En caso de que el archivo destino del proceso ya exista: ANEXAR

Usar Directorio de Trabajo: temp **Elegir**

Opciones Generales

02 Directorio por Omisión para los Formatos: /home/wallex/workspace/Mediacion/descriptores/formatos **Elegir**

03 Tamaño del Log: 3000 líneas

04 Nivel de Detalles del Log: Depuración

Guardar Cambios

1) *Valores por Omisión en la Definición de Procesos:* Aquí se definen varias características para los nuevos procesos. Cada vez que se defina un nuevo proceso, los valores de esta sección serán usados para rellenar los campos correspondientes.

2) *Directorio por omisión para los formatos:* En varias partes del sistema se hace referencia a los archivos que describen los múltiples formatos. Aquí se define el directorio en el cual se asume que se almacenan dichos descriptores:

- En la definición de Formatos, los diálogos para abrir/guardar formatos aparecerán inicialmente en ese directorio.

- En los atributos de los procesos, es posible no especificar la ruta completa al archivo que define el formato; si se ingresa sólo el nombre del archivo, la aplicación buscará al formato en este directorio.

3) *Tamaño del Log*: Para evitar que los log ocupen demasiado espacio, se provee la funcionalidad de limitar el tamaño del log. El log es de tipo rotatorio, de tal manera que cuando el archivo se llena, se comienzan a eliminar líneas del “principio” del archivo, de tal manera que los contenidos del log siempre serán presentados en orden cronológico.

4) *Nivel de Detalles del Log*: Especifica que tanta información almacenar en los archivos de log. Mientras más información se registre, mayor será el nivel de detalle, pero el log se llenará mucho más rápido. También se puede deshabilitar el log de ser necesario.

APÉNDICE B: ESPECIFICACIÓN DE CASOS DE USO Y ESCENARIOS DEL SISTEMA

En este apéndice se especifican todos los casos de uso y escenarios del sistema listados en el capítulo 3.

Nombre:

1. **Usuario administrador define un nuevo formato.**

Descripción:

- El usuario genera un nuevo formato de entre los formatos disponibles.

Notas:

- El usuario administrador puede escoger entre los tipos de formatos disponibles (de estructura fija, campos identificados por IDs, o de texto por separadores). No hay restricciones en los archivos previamente abiertos, es decir, se pueden tener tantos archivos abiertos a la vez como la memoria del equipo lo permita.

Escenarios:

- 1.1 Se crea un formato tipo binario de estructura fija.
- 1.2 Se crea un formato tipo binario de campos identificados por ID.
- 1.3 Se crea un formato tipo texto basado en separadores.

Valor Medible:

- Nuevo formato generado con éxito.

Nombre:

2. **Usuario administrador genera un campo.**

Descripción:

- El usuario genera y llena un nuevo campo dentro del

descriptor de formatos actualmente abierto.

Notas:

- Los campos generados vienen con la mínima información posible ya llenada.
- El nuevo campo es agregado al final de la lista de campos del grupo seleccionado.
- El ID del campo es generado automáticamente, a menos que el tipo del formato sea orientado por IDs.
- El usuario puede llenar los atributos como sea conveniente, es posible dejar varios atributos sin llenar al principio.

Escenarios:

- 2.1 Se genera un campo de tipo leído.
- 2.2 Se genera un campo de tipo leído con formato de fecha y máscara correcta.
- 2.3 Se genera un campo de tipo leído con formato de fecha y máscara incorrecta.
- 2.4 Se genera un campo de tipo definido, expresión simple, de manera correcta.
- 2.5 Se genera un campo de tipo definido, expresión simple, con la expresión incorrectamente estructurada.
- 2.6 Se genera un campo de tipo definido, expresión compuesta, pero incompleta.

2.7 Se genera un campo de tipo definido, expresión compuesta, correctamente.

2.8 Se genera un campo de tipo definido, expresión compuesta, que referencia a un campo no existente.

Valor Medible:

- El campo generado mostrando los datos ingresado y el ícono de estado indicando su validez.

Nombre:

3. Usuario administrador genera un Grupo.

Descripción:

- Se genera un nuevo grupo de campos dentro de la definición de datos existente.

Notas:

- El grupo actualmente seleccionado será el padre del nuevo grupo, esta decisión no es cambiabile.
- La condición inicial del grupo será la de “por omisión”, y esta expresión es válida aún si existen otros grupos después (al mismo nivel). Los grupos existentes no son afectados de ninguna manera.

Escenarios:

3.1 Se genera un grupo de condición “por omisión”.

3.2 Se genera un grupo con condición basada en campos anteriores existentes.

3.3 Se genera un grupo con condición inválida.

Valor Medible:

- El nuevo grupo es generado, con su estado de validez indicado por su ícono correspondiente.

Nombre:

4. Usuario administrador reorganiza un grupo de campos.

Descripción:

- El usuario reorganiza el orden de algún campo dentro del grupo de campos actuales.

Notas:

- La reorganización sucede de una posición a la vez, es decir, para mover un campo varias posiciones (en cualquier sentido) es necesario realizar varios movimientos individuales.
- A medida que los campos se reorganizan, hay que revisar si la validez de los campos en cuestión es afectada.

Escenarios:

4.1 El campo se mueve satisfactoriamente a la derecha (después del campo siguiente).

- 4.2 El campo se mueve satisfactoriamente a la izquierda (antes del campo anterior).
- 4.3 El campo no se mueve a la posición siguiente por ser el último en la lista del grupo.
- 4.4 El campo se mueve a la posición anterior, pero la validez del campo actual se invalida.
- 4.5 El campo se mueve a la posición siguiente, pero la validez del campo siguiente se invalida.

Valor Medible:

- La reorganización de los campos en la tabla, cambios en el ícono de estado de los campos correspondientes.

Nombre:

5. **Usuario administrador reorganiza un conjunto de grupos.**

Descripción:

- El usuario decide mover uno de los grupos entre los demás grupos del mismo nivel.

Notas:

- La reorganización de grupos no afecta para nada la validez de los grupos involucrados (ya que cada uno no tiene relaciones de ningún tipo con los demás).

Escenarios:

- 5.1 Se mueve el grupo después del siguiente exitosamente.
- 5.2 No se mueve el grupo ya que no existen otro grupo con el cual intercambiar de lugar.

Valor Medible:

- La reorganización visible de los grupos.

Nombre:

6. **Usuario administrador modifica las opciones del formato actual.**

Descripción:

- El usuario modifica las opciones generales del documento actual.

Notas:**Escenarios:**

- 6.1 Las opciones son alteradas exitosamente.
- 6.2 Las opciones son alteradas exitosamente, corrigiendo los valores inválidos ingresados.

Nombre:**7. Usuario administrador guarda un documento.****Descripción:**

- El usuario decide guardar el documento actual.

Notas:

- No se permite guardar el documento cuando este tiene algún campo o grupo marcado como inválido.

Escenarios:

- 7.1 Documento guardado exitosamente.
- 7.2 No se puede guardar: El documento describe al menos un campo no válido.
- 7.3 No se puede guardar. el documento describe al menos un grupo no válido.

Valor Medible:

- El documento es guardado exitosamente.

Nombre:**8. Usuario administrador abre un documento.****Descripción:**

- El usuario intenta abrir un archivo de extensión XML que aparentemente es un archivo de definición de formatos.

Notas:

- La estructura e integridad del archivo es verificado usando el XSD correspondiente, sólo si el archivo pasa esa prueba se lo intenta abrir como un descriptor de formatos.

Escenarios:

- 8.1 El archivo es abierto satisfactoriamente.
- 8.2 El archivo no cumple con la especificación de formatos esperada.

Valor Medible:

- El archivo recientemente abierto se agrega a la lista de archivos abiertos.

Nombre:**9. Usuario administrador define un proceso**

Descripción:

- El usuario define un nuevo proceso para ser ejecutado posteriormente.

Notas:

- El nuevo proceso tendrá los valores por omisión que han sido configurados para el sistema.

Escenarios:

- 9.1 Se genera un proceso con los datos necesarios.
- 9.2 No se puede generar un proceso porque faltan datos.

Valor Medible:

- El proceso es agregado a la tabla de procesos.

Nombre:**10. Usuario administrador modifica un proceso**

Descripción:

- El usuario abre las propiedades de un formato y lo modifica.

Notas:

- Si no hay procesos seleccionados, se muestra un mensaje de advertencia.
- Si se seleccionan varios procesos, se los edita en orden.
- Si el usuario presiona cancelar, se cancela la edición de los demás procesos que aun no han sido editados.
- La edición de procesos no afectan a aquellos procesos que ya se están ejecutando.

Escenarios:

- 10.1 El proceso es modificado exitosamente.
- 10.2 Los cambios invalidan al proceso y no se lo puede modificar.

Valor Medible:

- Los cambios al proceso se reflejan en las celdas de las tablas.

Nombre:

- 11. **Usuario Operador inicia un proceso**

Descripción:

- El usuario intenta iniciar uno o más procesos que tenga seleccionados.

Notas:

- Si no hay procesos seleccionados se le informa al usuario con un mensaje.
- Antes de ejecutar el proceso, se abre el diálogo con las propiedades del mismo dando una oportunidad para modificar los valores, estas modificaciones no afectaran al proceso en la sección de definición.
- Si se seleccionan varios procesos, se los trata de uno en uno.

Escenarios:

- 11.1 El proceso se ejecuta exitosamente.
- 11.2 El proceso no tiene todos los datos completos, no se puede ejecutar.
- 11.3 El proceso falla en la ejecución por algún parámetro incorrecto en la definición del proceso.
- 11.4 El proceso falla en la ejecución por algún problema de acceso a los archivos especificados en el proceso.

Valor Medible:

- El progreso y estado del proceso se reflejan en las columnas

respectivas de la tabla de ejecución de procesos.

Nombre:

12. Usuario Operador cancela un proceso

Descripción:

- El usuario intenta cancelar uno o más de los procesos en la tabla de ejecución.

Notas:

- Si no hay procesos seleccionados, se informa al usuario con un diálogo.
- Sólo se detienen los procesos que se encuentran en ejecución, aquellos terminados o cancelados no son tocados.

Escenarios:

12.1 Los procesos se detienen exitosamente.

12.2 Los procesos no se detienen por no estar ejecutándose.

Valor Medible:

- El estado de los procesos es cambiado a “cancelado”.

Nombre:

13. Usuario Operador reinicia un proceso.

Descripción:

- El usuario intenta reiniciar uno o más procesos de los que se

están ejecutando.

Notas:

- Los procesos que estén en ejecución primero deben ser detenidos.

Escenarios:

13.1 El proceso es reiniciado.

13.2 El proceso es reiniciado luego de detenido.

Valor Medible:

- El estado de los procesos es cambiado según es apropiado.

A continuación se especifican los escenarios para cada uno de los casos de uso.

Caso de Uso 1: Usuario administrador define un nuevo formato.

Escenario 1.1: Se crea un formato binario de estructura fija.

Asunciones:

- Se selecciona dicho formato de la lista de formatos disponibles.

Resultados:

- Se genera un nuevo formato de tipo binario de estructura fija.
- Al nuevo formato se le genera automáticamente un grupo raíz

de nombre "Grupo Inicial".

- Al nuevo grupo se le agrega automáticamente un campo genérico de nombre "Campo 1".

Escenario 1.2: Se crea un formato tipo binario de campos identificados por ID.

Asunciones:

- Se selecciona el formato "Tipo binario de Campos identificados por ID" desde el menú de nuevos formatos.

Resultados:

- Se genera un nuevo formato de descriptores por ID.
- El nuevo grupo empieza vacío (sin campos).

Escenario 1.3: Se crea un formato tipo texto basado en separadores.

Asunciones:

- Se selecciona el formato "Tipo texto basado en separadores."

Resultados:

- Se genera un nuevo formato de descriptores de formatos de texto por separadores.
- Se genera un grupo raíz "Grupo Inicial"
- Se genera un campo inicial genérico llamado "Campo 1".

Caso de Uso 2: Usuario administrador genera un campo.**Escenario 2.1: Se genera un campo de tipo leído.****Asunciones:**

- Existe un formato abierto con un grupo existente y seleccionado.
- Se fija que el origen del campo es “leído”.

Resultados:

- Se genera un nuevo campo.
- El tipo de dato es fijado a “Entero (Big Endian) por omisión.
- El valor por omisión queda sin especificar.
- El nuevo campo queda en estado válido.

Escenario 2.2: Se genera un campo de tipo leído con formato de fecha y máscara correcta.**Asunciones:**

- Existe un formato abierto con un grupo seleccionado.
- Al nuevo campo se le asigna el tipo “fecha”.
- La máscara ingresada sigue un formato válido.

Resultados:

- Se genera un nuevo campo de tipo generado y tipo fecha.
- Se valida la máscara ingresada y se marca al campo como

válido.

Escenario 2.3: Se genera un campo de tipo leído con formato de fecha y máscara incorrecta.

Asunciones:

- Existe un formato abierto con un grupo seleccionado.
- Al nuevo campo se le asigna el tipo “fecha”.
- Al ingresar la máscara, se ingresa un patrón no reconocido.

Resultados:

- Se genera un nuevo campo de origen leído y tipo fecha.
- Al identificar que la máscara ingresada es incorrecta se presenta un mensaje de advertencia al usuario.
- El campo es marcado como inválido.

Escenario 2.4: Se genera un campo de tipo definido, expresión simple, de manera correcta.

Asunciones:

- Existe un formato abierto con al menos un grupo seleccionado.
- Al nuevo campo se asigna de origen “definido”.
- Se define al campo como de “expresión simple”.
- Se ingresa una expresión válida.

Resultados:

- Se genera un nuevo campo de tipo generado.
- Luego de validada la expresión se fija el campo al estado válido.

Escenario 2.5: Se genera un campo de tipo definido, expresión simple, con la expresión incorrectamente estructurada.

Asunciones:

- Existe un formato abierto con al menos un grupo seleccionado.
- El nuevo campo es definido de tipo “definido”.
- El campo es definido como expresión simple.
- La expresión ingresada tiene algún error de sintaxis.

Resultados:

- Se genera un nuevo campo de tipo generado.
- Cuando la expresión falla en ser verificada como válida, se presenta un mensaje al usuario.
- El campo es marcado como inválido.

Escenario 2.6: Se genera un campo de tipo definido, expresión compuesta, pero incompleta.

Asunciones:

- Existe un formato abierto con un grupo seleccionado.

- El campo a generar se define con origen “definido”.
- El campo es definido de tipo “expresión condicional”.
- No se ingresan todas las expresiones necesarias.

Resultados:

- El campo es generado con los atributos de “generado”, y y “expresión condicional”.
- El campo queda marcado como inválido.

Escenario 2.7: Se genera un campo de tipo definido, expresión compuesta, correctamente.**Asunciones:**

- Existe un formato abierto con un grupo seleccionado.
- El campo a generar se define con origen “definido”.
- El campo es definido de tipo “expresión condicional”.
- Se ingresan todas las expresiones necesarias y tienen la sintaxis correcta.

Resultados:

- El campo es generado con los atributos de “generado”, y y “expresión condicional”.
- El campo se marca como válido luego de validar todas las expresiones ingresadas.

Escenario 2.8: Se genera un campo de tipo definido, expresión compuesta, que referencia a un campo no existente.

Asunciones:

- Existe un formato abierto con un grupo seleccionado.
- El campo a generar se define con origen “definido”.
- El campo es definido de tipo “expresión condicional”.
- Se ingresan todas las expresiones necesarias y tienen la sintaxis correcta.
- Al menos una de las expresiones hace referencia a un ID de campo que no existe en el alcance del campo actual.

Resultados:

- El campo es generado con los atributos de “generado”, y “expresión condicional”.
- Al no encontrar el ID ingresado, se notifica al usuario del problema.
- Se marca al campo como inválido.

Caso de Uso 3: Usuario administrador genera un grupo.**Escenario 3.1: Se genera un grupo de condición “por omisión”.****Asunciones:**

- Se tiene seleccionado el grupo el cual se quiere especificar como previo al nuevo grupo.
- El tipo de formato seleccionado permite generación de grupos.

Resultados:

- Se genera un nuevo grupo con nombre de “Nuevo Grupo #”
Donde # es un valor auto generado dependiente del documento actual.
- El nuevo grupo es ubicado como hijo del grupo actual, y como último en la lista de los hijos de este.
- La condición es fijada a una expresión vacía, que se evalúa como “siempre cierto” que se muestra pantalla como “<Sin Condición>”.

Escenario 3.2: Se genera un grupo con condición basada en campos anteriores existentes.**Asunciones:**

- Se tiene seleccionado el grupo el cual se quiere especificar como previo al nuevo grupo.

- El tipo de formato seleccionado permite generación de grupos.

Resultados:

- Se genera un nuevo grupo con nombre de “Nuevo Grupo #”
Donde # es un valor auto generado dependiente del documento actual.
- El nuevo grupo es ubicado como hijo del grupo actual, y como último en la lista de los hijos de este.
- La condición es fijada a una expresión booleana, la cual referencia a campos previamente definidos.

Escenario 3.3: Se genera un grupo con condición inválida.**Asunciones:**

- Existe un grupo seleccionado al cual se le quiere agregar un grupo siguiente.
- El documento existente permite la generación de nuevos grupos.
- Al nuevo grupo se le asigna una expresión que no puede ser evaluada.

Resultados:

- Se genera un nuevo grupo con las características por omisión de los grupos.
- El grupo es colocado al final de los demás grupos que el sigan

al mismo grupo antecedente.

- Se cambia el ícono de estado del grupo indicando la invalidez de su expresión.

Caso de Uso 4: Usuario administrador reorganiza un grupo de campos.

Escenario 4.1: El campo se mueve satisfactoriamente a la derecha (después del campo siguiente).

Asunciones:

- Existe un grupo actual el cual contiene mas de un campo.
- Debe existir entre estos un campo seleccionado.
- Existe al menos un campo en el grupo actual ubicado después del campo actual.
- El campo siguiente al seleccionado no tiene referencias hacia el campo actual.

Resultados:

- El campo se mueve a la derecha satisfactoriamente.
- El estado de los dos campos permanece inalterado.

Escenario 4.2: El campo se mueve satisfactoriamente a la izquierda (antes del campo anterior).

Asunciones:

- Existe un grupo actual el cual contiene mas de un campo.
- Debe existir entre estos un campo seleccionado.
- Existe al menos un campo en el grupo actual ubicado antes del campo actual.
- El campo seleccionado no tiene referencias hacia el campo anterior.

Resultados:

- El campo seleccionado intercambia posición con el campo anterior.
- El estado de ambos campos es preservado.

Escenario 4.3: El campo no se mueve a la posición siguiente por ser el último en la lista del grupo.

Asunciones:

- Existe un grupo actual que contiene al menos un campo.
- Existe un campo seleccionado.
- Se solicita mover el campo seleccionado a la derecha.
- No hay más campos a la derecha del campo seleccionado.

Resultados:

- El campo seleccionado no es afectado para nada.

Escenario 4.4: El campo se mueve a la posición anterior, pero la validez del campo actual se invalida.

Asunciones:

- Existe un grupo actual el cual contiene mas de un campo.
- Debe existir entre estos un campo seleccionado.
- Existe al menos un campo en el grupo actual ubicado antes del campo actual.
- El campo seleccionado contiene una referencia al campo anterior.

Resultados:

- Los campos en cuestión cambian de posición.
- El campo seleccionado se marca como inválido.

Escenario 4.5: El campo se mueve a la posición siguiente, pero la validez del campo siguiente se invalida.

Asunciones:

- Existe un grupo actual el cual contiene mas de un campo.
- Debe existir entre estos un campo seleccionado.
- Existe al menos un campo en el grupo actual ubicado después

del campo actual.

- El campo siguiente contiene una referencia al campo seleccionado.

Resultados:

- Los campos en cuestión cambian de posición.
- El campo que estaba después del seleccionado se marca como inválido.

Caso de Uso 5: Usuario administrador reorganiza un conjunto de Grupos.

Escenario 5.1: Se mueve el grupo después del siguiente exitosamente.

Asunciones:

- El formato actual permite la creación de grupos.
- El grupo seleccionado tiene otros grupos al mismo nivel (con el mismo ancestro).
- Existe un grupo ubicado después del grupo seleccionado.

Resultados:

- Los grupos son intercambiados de posición.
- El estado de los grupos permanecen inalterados.

Escenario 5.2: No se mueve el grupo ya que no existen otro grupo con el cual intercambiar de lugar.

Asunciones:

- El formato actual permite la creación de grupos.
- El grupo seleccionado no tiene otro grupo en la dirección de movimiento deseada.

Resultados:

- El grupo seleccionado no es alterado.

Caso de Uso 6: Usuario administrador modifica las opciones del formato actual.

Escenario 6.1: Las opciones son alteradas exitosamente.

Asunciones:

- Existe un documento abierto con opciones disponibles a modificar.
- Los cambios realizados en las opciones son válidos.

Resultados:

- Las opciones del documento son actualizadas a sus nuevos valores.

Escenario 6.2: Las opciones son alteradas exitosamente, corrigiendo valores inválidos.

Asunciones:

- Existe un documento abierto con opciones disponibles a modificar.
- Algunos de los nuevos campos ingresados tienen valores inválidos.

Resultados:

- Los valores inválidos son interpretados como los valores válidos más cercanos (si no se pueden validar para nada, son revertidos a los valores originales)
- Se actualizan las propiedades del documento con los nuevos valores.

Caso de Uso 7: Usuario administrador guarda un documento.

Escenario 7.1: Documento guardado exitosamente.

Asunciones:

- El documento ya tiene un nombre de archivo asociado.
- Se tiene acceso de escritura al archivo en cuestión.
- Todos los campos del documento están marcados como válidos.

- Todos los grupos del documento están marcados como válidos.

Resultados:

- El documento es guardado exitosamente.
- Se notifica al usuario en la barra de estado sobre dicha acción.

Escenario 7.2: No se puede guardar: El documento describe al menos un campo no válido.**Asunciones:**

- El documento tiene un nombre de archivo asociado.
- Algún campo está marcado como no válido.

Resultados:

- Se cambia el grupo seleccionado al grupo que contiene el campo no válido.
- Se enfoca el campo no válido y se presenta un mensaje de error al usuario.

Caso de Uso 8: Usuario administrador abre un documento.**Escenario 8.1: El archivo es abierto satisfactoriamente.****Asunciones:**

- Al archivo especificado a abrir existe y se tienen permisos de lectura sobre este.

- El archivo cumple con la especificación dado por el “XML Schema Definition” de los archivos de definición de formatos.

Resultados:

- El archivo es leído y el documento armado y presentado listo para edición.

Escenario 8.2: El archivo no cumple con la especificación de formatos esperada.**Asunciones:**

- Al archivo especificado a abrir existe y se tienen permisos de lectura sobre este.
- El archivo no cumple con la especificación del XSD correspondiente a los archivos de definición de formatos.

Resultados:

- No se abre el archivo.
- Se muestra al usuario un mensaje indicando que el archivo no cumple con el formato esperado.

Caso de Uso 9: Usuario administrador define un proceso.**Escenario 9.1: Se genera un proceso con los datos necesarios.****Asunciones:**

- Se ingresan al menos la mínima cantidad de datos necesarios para definir un proceso.

Resultados:

- Se genera un nuevo proceso con los datos por omisión en la mayoría de los campos (a excepción de aquellos modificados explícitamente por el usuario).
- El proceso es agregado al final de la lista de procesos existentes.

Escenario 9.2: No se puede generar un proceso porque faltan datos.**Asunciones:**

- No se ingresan los suficientes datos para generar un nuevo proceso.

Resultados:

- Se muestra un diálogo especificando los datos que faltan.
- No se genera un nuevo proceso todavía.

Caso de Uso 10: Usuario administrador modifica un proceso.**Escenario 10.1: El proceso es modificado exitosamente.****Asunciones:**

- Se selecciona un proceso existente a editar.
- Los nuevos valores ingresados son validos.

Resultados:

- El proceso es actualizado.

Escenario 10.2: Los cambios invalidan al proceso y no se lo puede modificar.**Asunciones:**

- Se selecciona un proceso existente a editar.
- Los cambios hechos invalidan al proceso.

Resultados:

- Se notifica al usuario que faltan datos (o que hay datos inválidos).
- No se modifica al proceso todavía.

Caso de Uso 11: Usuario operador inicia un proceso.**Escenario 11.1: El proceso se ejecuta exitosamente.****Asunciones:**

- El archivo de formatos especificado existe y se tiene permiso de lectura sobre este.
- El archivo origen de datos existe y se tiene permiso de lectura sobre este.
- La ubicación del archivo destino existe y se tiene permisos de escritura sobre la localización.
- Si el archivo destino existe, el proceso está especificado para continuar igualmente (ya sea por sobre-escritura o anexando)
- Si se usa un directorio de trabajo, se tiene permisos de escritura sobre este.
- No existen interrupciones durante el procesamiento del archivo.
- La definición de formatos escogida corresponde al del archivo de origen.
- El archivo de origen esta correctamente estructurado (no presenta corrupciones, todos los registros están completos, etc)

Resultados:

- El archivo destino es generado exitosamente.
- El estado del proceso es cambiado a “finalizado”.

Escenario 11.2: El proceso no tiene todos los datos completos, no se puede ejecutar.

Asunciones:

- Falta algún campo del proceso que se quiere iniciar (como los nombre de archivos involucrados)

Resultados:

- Se notifica del error al usuario.
- El proceso no se inicia.

Escenario 11.3: El proceso falla en la ejecución por algún parámetro incorrecto en la definición del proceso.

Asunciones:

- Alguno de los archivos especificados no existe.

Resultados:

- Se presenta al usuario el mensaje de error.
- No se inicia el proceso.

Escenario 11.4: El proceso falla en la ejecución por algún problema de acceso a los archivos especificados en el proceso.

Asunciones:

- El archivo de formatos especificado existe y se tiene permiso de

lectura sobre este.

- El archivo origen de datos existe y se tiene permiso de lectura sobre este.
- La ubicación del archivo destino existe y se tiene permisos de escritura sobre la localización.
- Si el archivo destino existe, el proceso está especificado para continuar igualmente (ya sea por sobre-escritura o anexando)
- Si se usa un directorio de trabajo, se tiene permisos de escritura sobre este.
- Durante la ejecución del proceso inesperadamente se pierde la conexión con el archivo de origen.

Resultados:

- Se detiene la ejecución.
- Se eliminan los datos leídos del registro actual.
- Se registran el evento en el log.
- Se notifica visualmente al usuario que hubo un problema.
- El archivo destino conserva todos los registros que fueron leídos previamente con éxito.

Caso de Uso 12: Usuario operador cancela un proceso.**Escenario 12.1: Los procesos se detienen exitosamente.****Asunciones:**

- Al menos uno de los procesos seleccionados a detener se está ejecutando.

Resultados:

- Aquellos procesos seleccionados que se estén ejecutando serán detenidos.
- Los archivos destino contendrán la información de los registros que fueron leídos hasta el momento de la cancelación.

Escenario 12.2: Los procesos no se detienen por no estar ejecutándose.**Asunciones:**

- Ninguno de los procesos seleccionados está en estado de ejecución.

Resultados:

- Ningún proceso es afectado.

Caso de Uso 13: *Usuario operador reinicia un proceso.***Escenario 13.1: El proceso es reiniciado exitosamente.****Asunciones:**

- Al menos uno de los procesos seleccionados estaba en estado detenido.
- Ninguno de los procesos seleccionados esta en estado de ejecución.

Resultados:

- Todos los procesos seleccionados son reiniciados.

Escenario 13.2: El proceso es reiniciado luego de detenido.**Asunciones:**

- Al menos uno de los procesos seleccionados está en estado de ejecución.

Resultados:

- Aquellos procesos en estado de ejecución son detenidos.
- Todos los procesos seleccionados son reiniciados.

APÉNDICE C: DIAGRAMAS DE CLASES DEL SISTEMA

En este apéndice se mostrarán los diagramas de clases del sistema. En los diagramas de clase se describen gráficamente las clases de una aplicación y las relaciones estáticas entre las mismas.

Las clases pueden tener tanto atributos como operaciones. Los atributos son utilizados para guardar información en el objeto (los objetos son instancias de las clases). Las operaciones o métodos son las funciones que puede realizar el objeto.

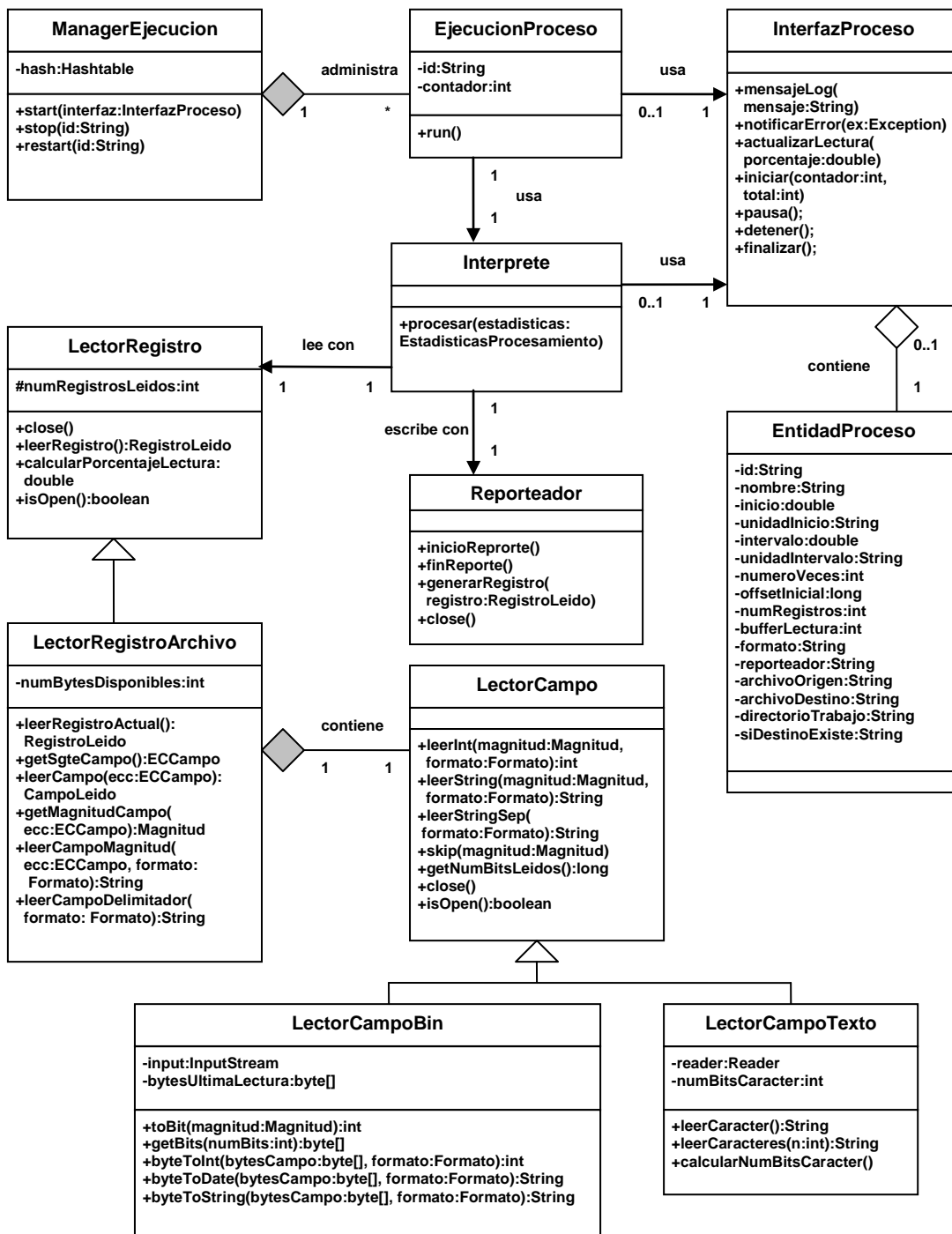


Figura C.1: Diagrama de clases del sistema (parte 1)

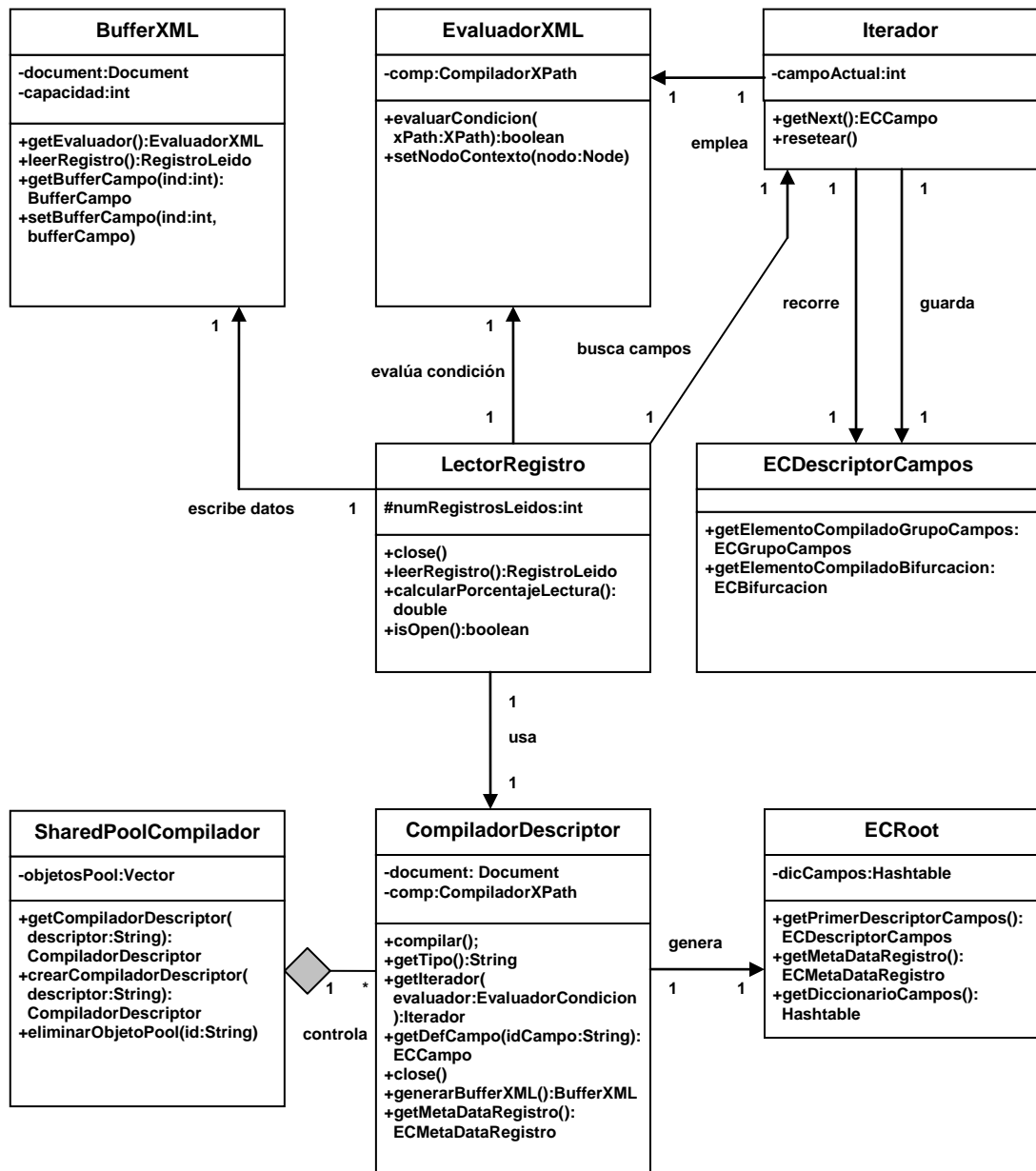


Figura C.2: Diagrama de clases del sistema (parte 2)

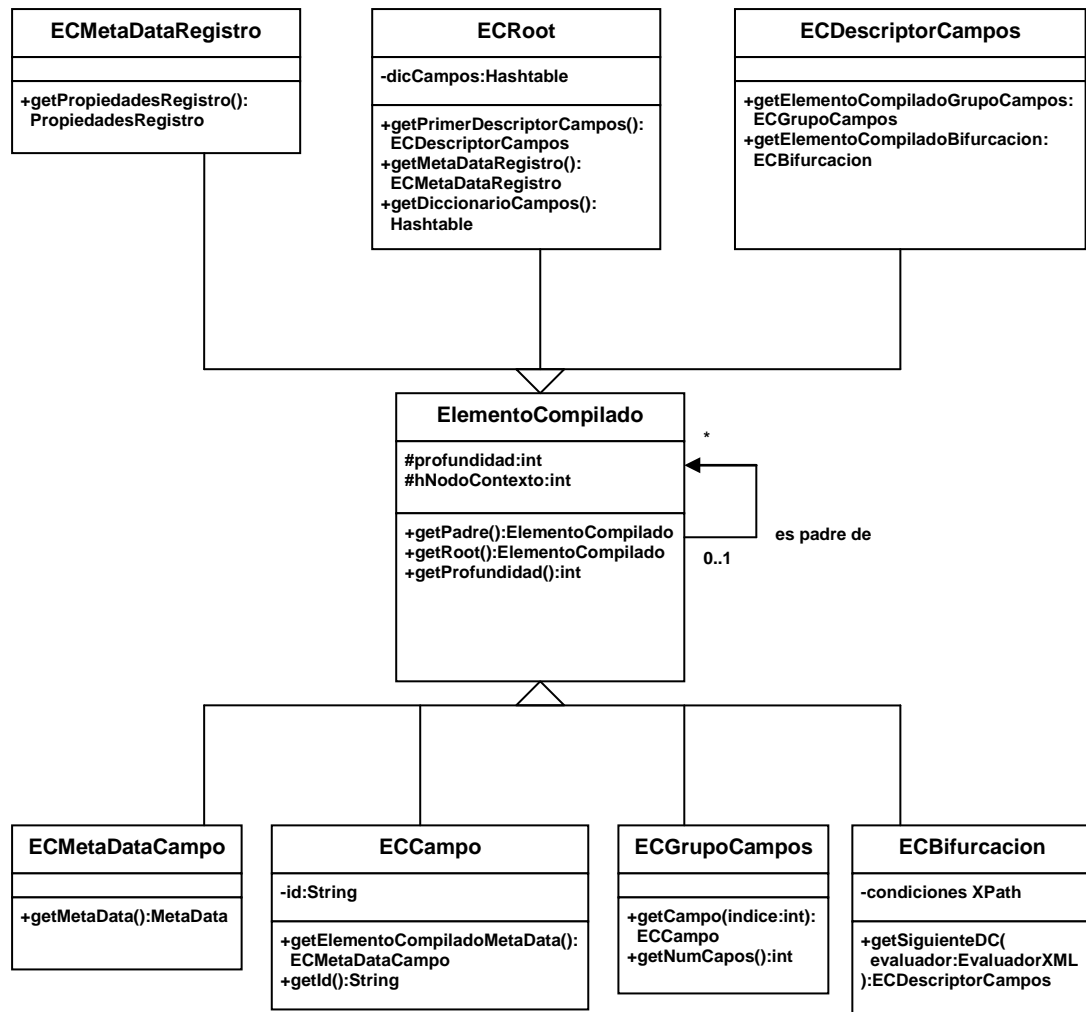


Figura C.3: Diagrama de clases del sistema (parte 3)

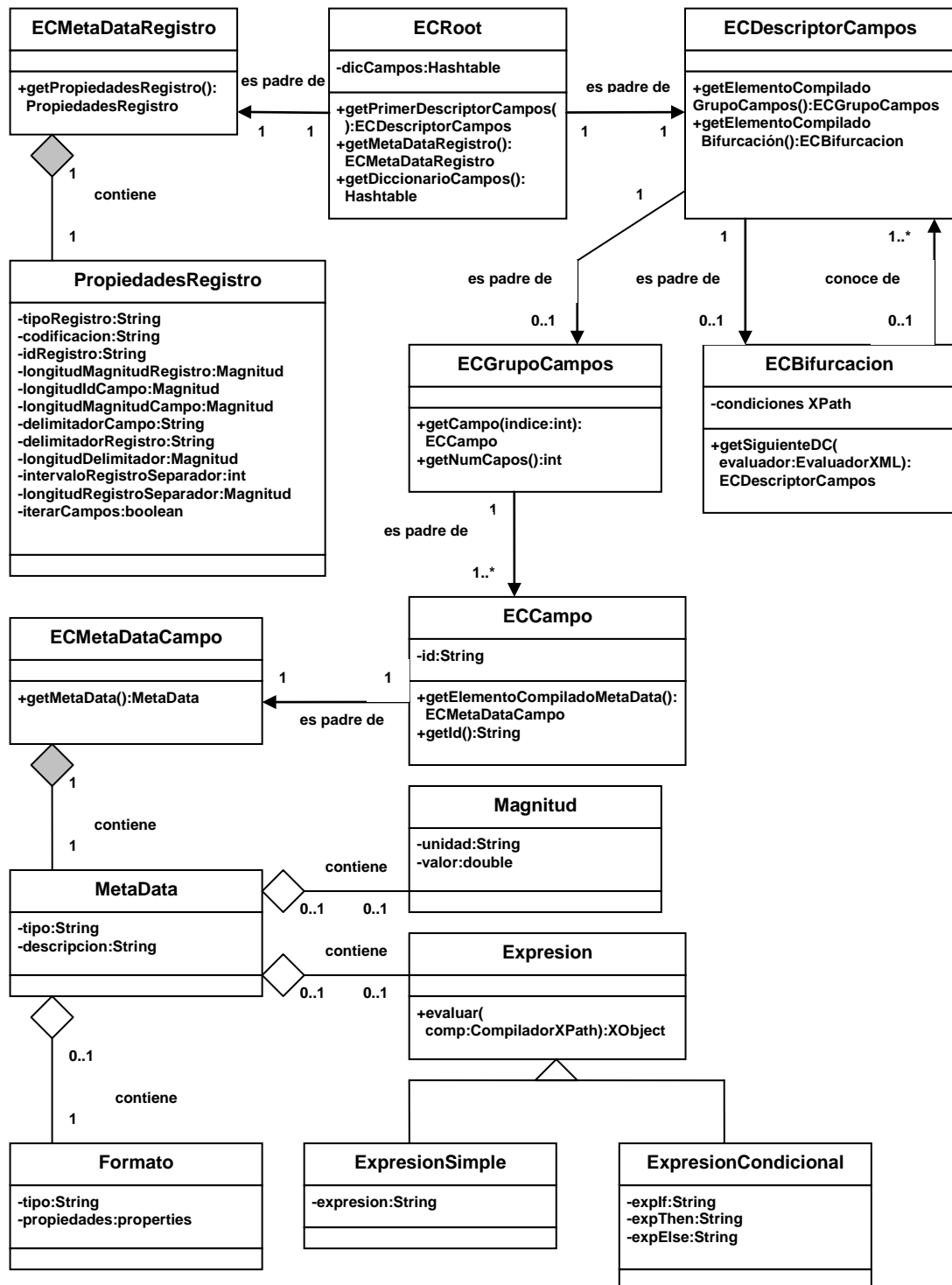


Figura C.4: Diagrama de clases del sistema (parte 4)

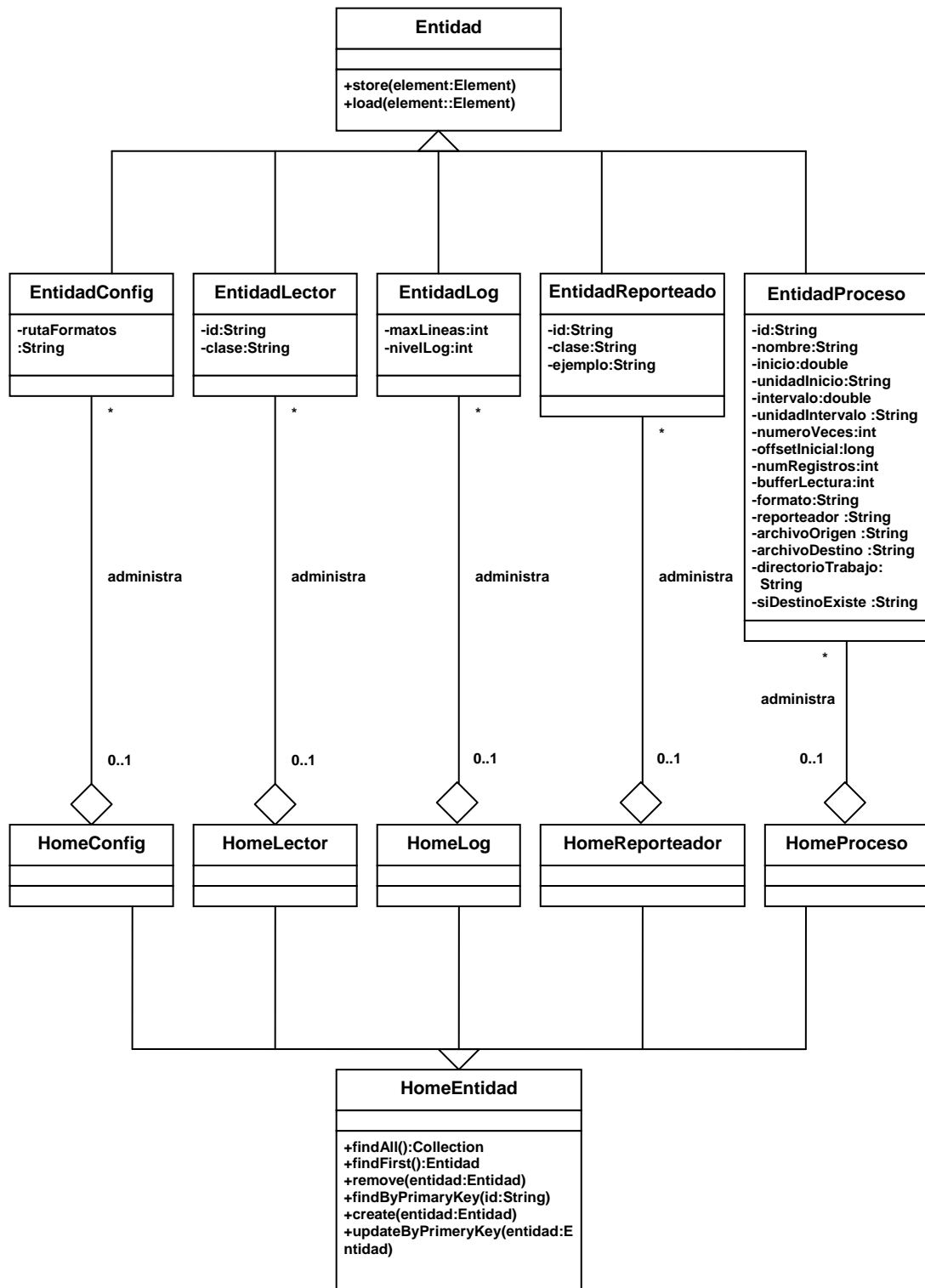


Figura C.5: Diagrama de clases del sistema (parte 5)

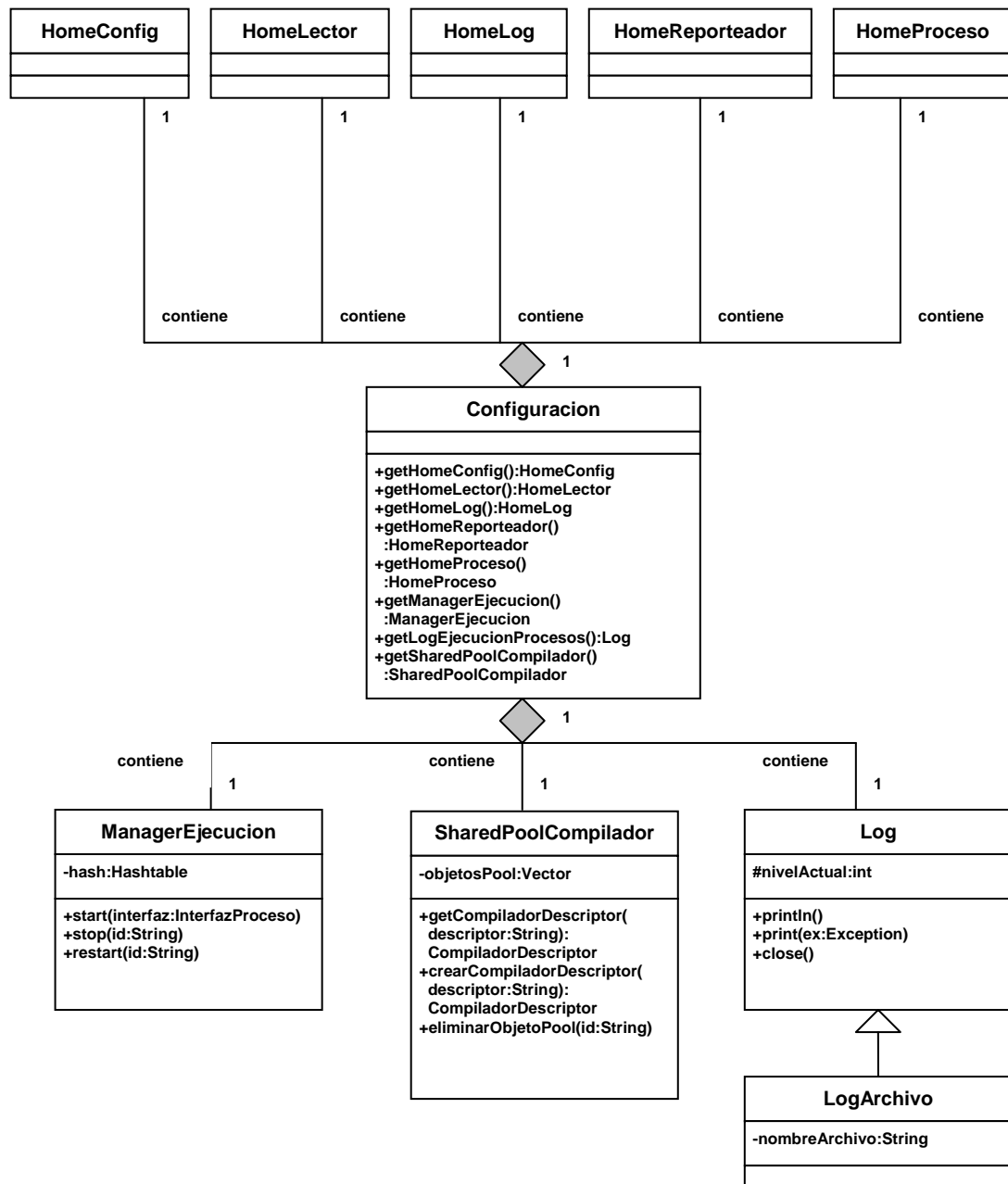


Figura C.6: Diagrama de clases del sistema (parte 6)

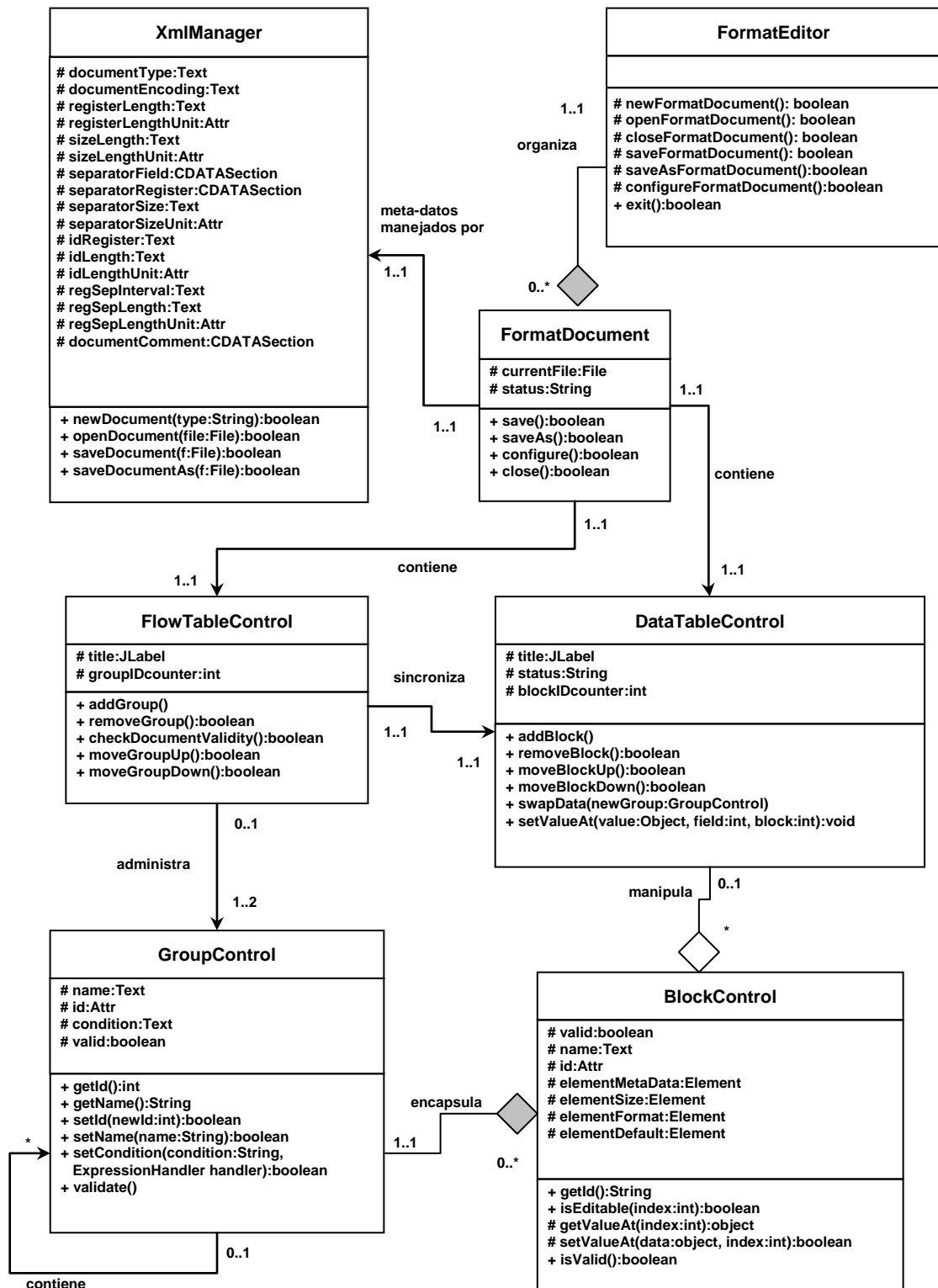


Figura C.7: Diagrama de clases del sistema (parte 7)

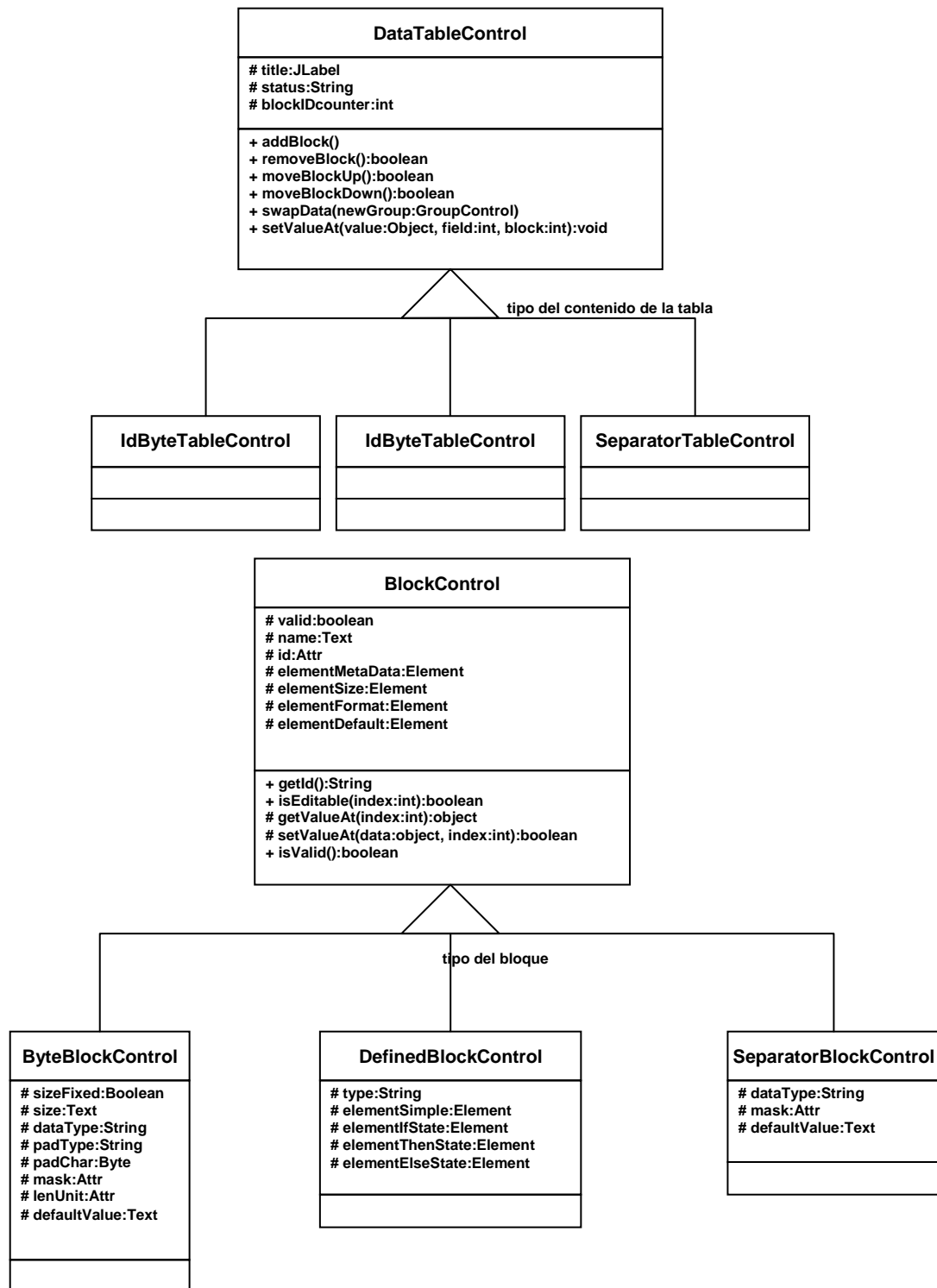


Figura C.8: Diagrama de clases del sistema (parte 8)

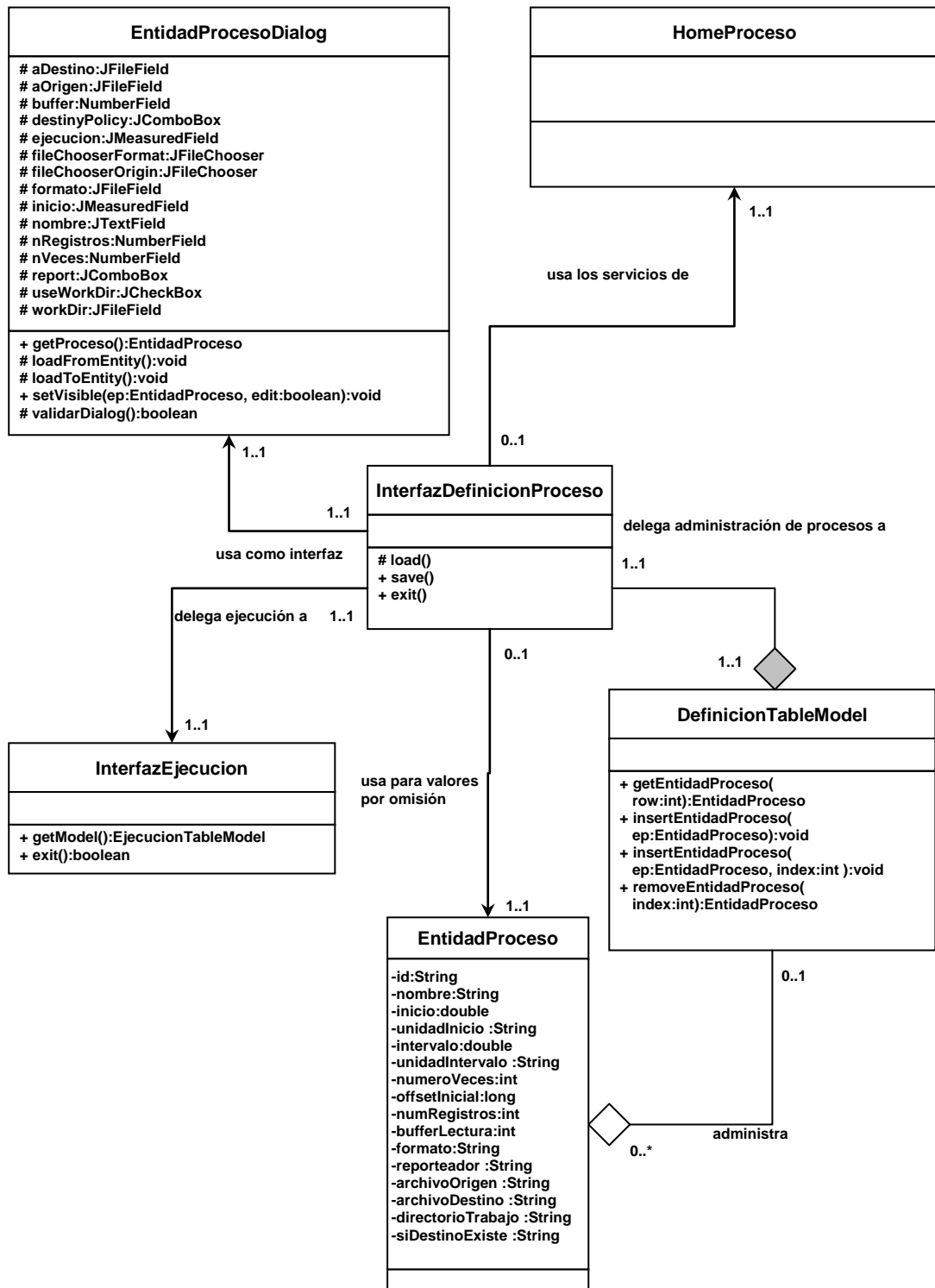


Figura C.9: Diagrama de clases del sistema (parte 9)

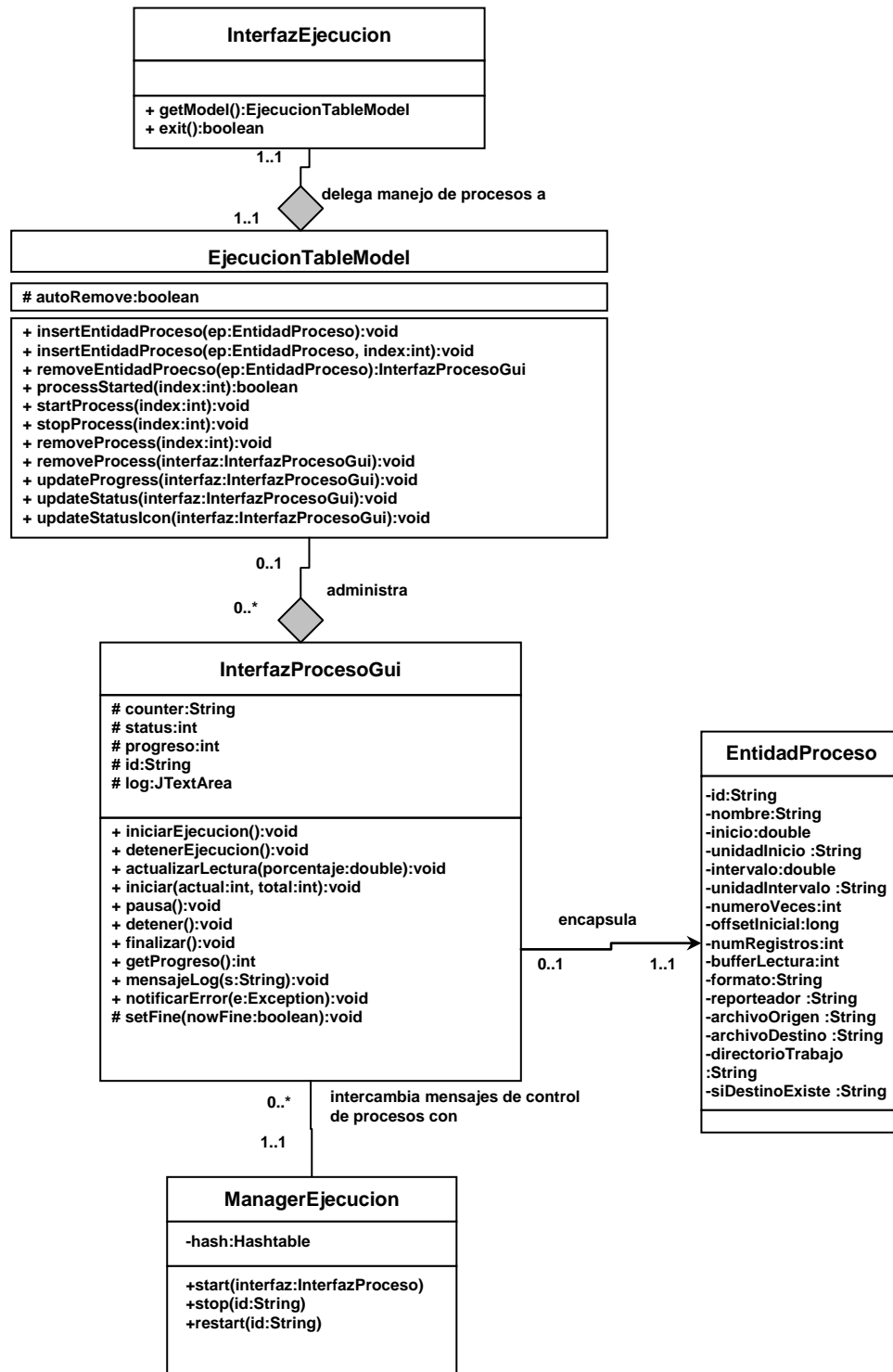


Figura C.10: Diagrama de clases del sistema (parte 10)

APÉNDICE D: DIAGRAMAS DE SECUENCIA DEL SISTEMA

En este apéndice se mostrarán los diagramas de secuencia del sistema. Los diagramas de secuencia describen gráficamente la interacción entre los objetos del sistema. En estos esquemas los objetos son ubicados a lo largo del eje horizontal superior, y el eje vertical representa una escala de tiempo. Estos diagramas se harán para cada escenario especificado en la etapa de análisis.

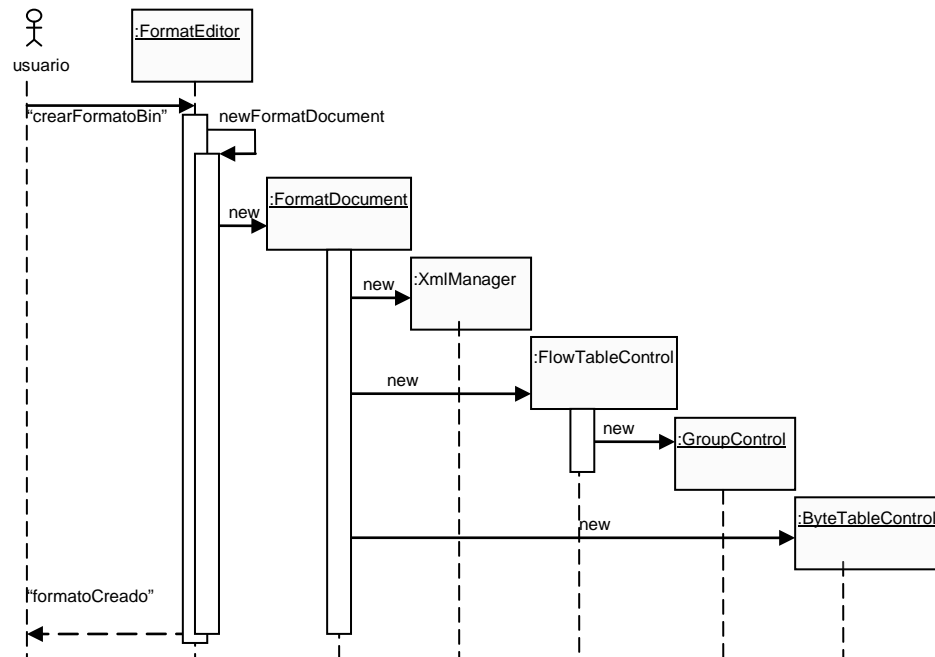


Figura D.1: Usuario crea formato binario de estructura fija

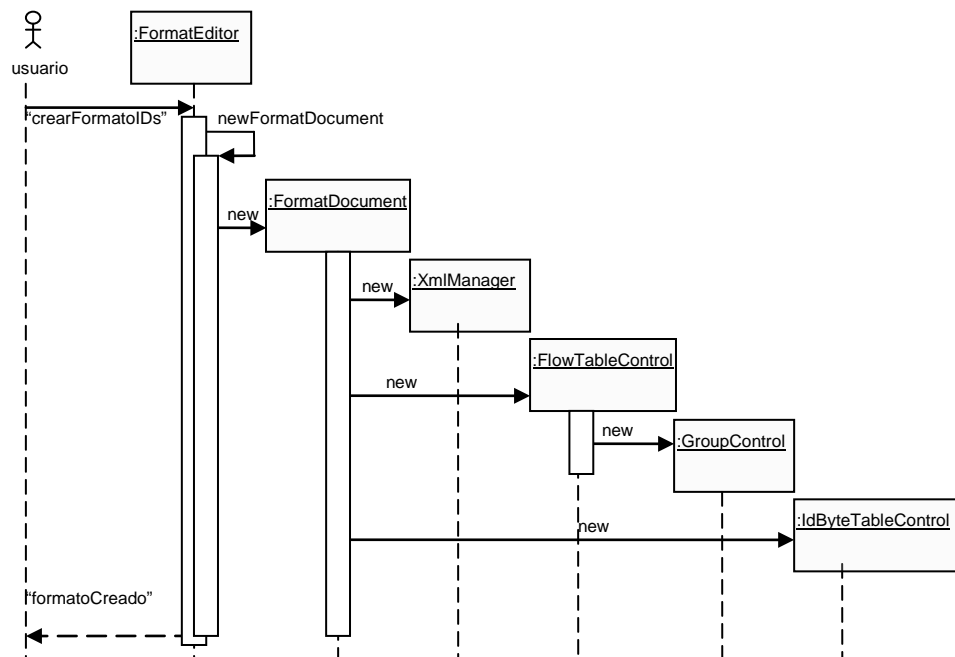


Figura D.2: Usuario crea formato binario de campos identificados por IDs

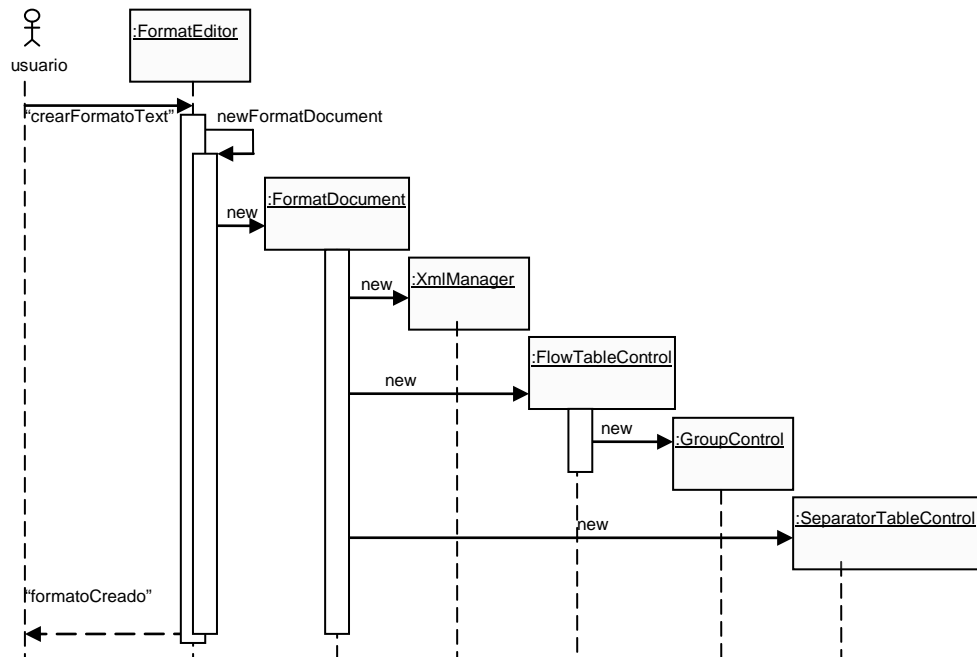


Figura D.3: Usuario crea formato de texto basado en separadores

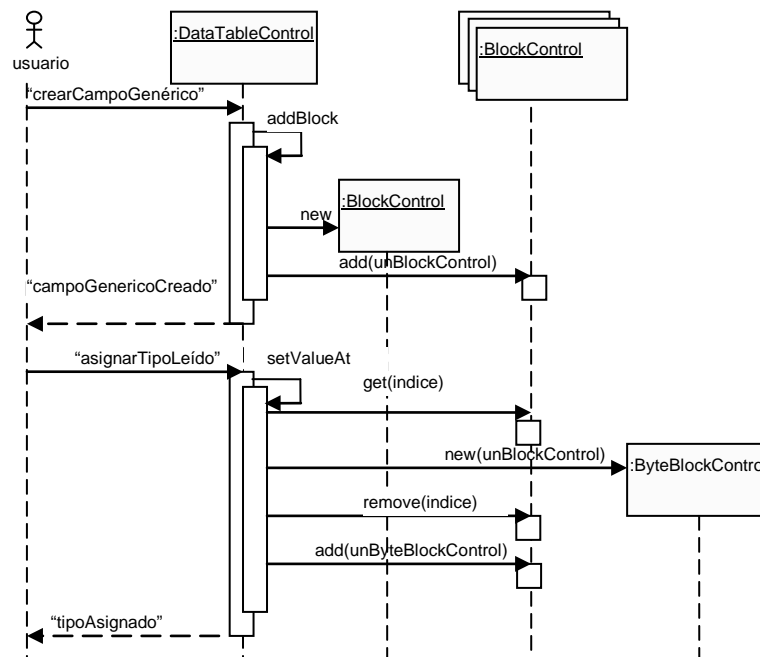


Figura D.4: Usuario crea un campo de tipo leído

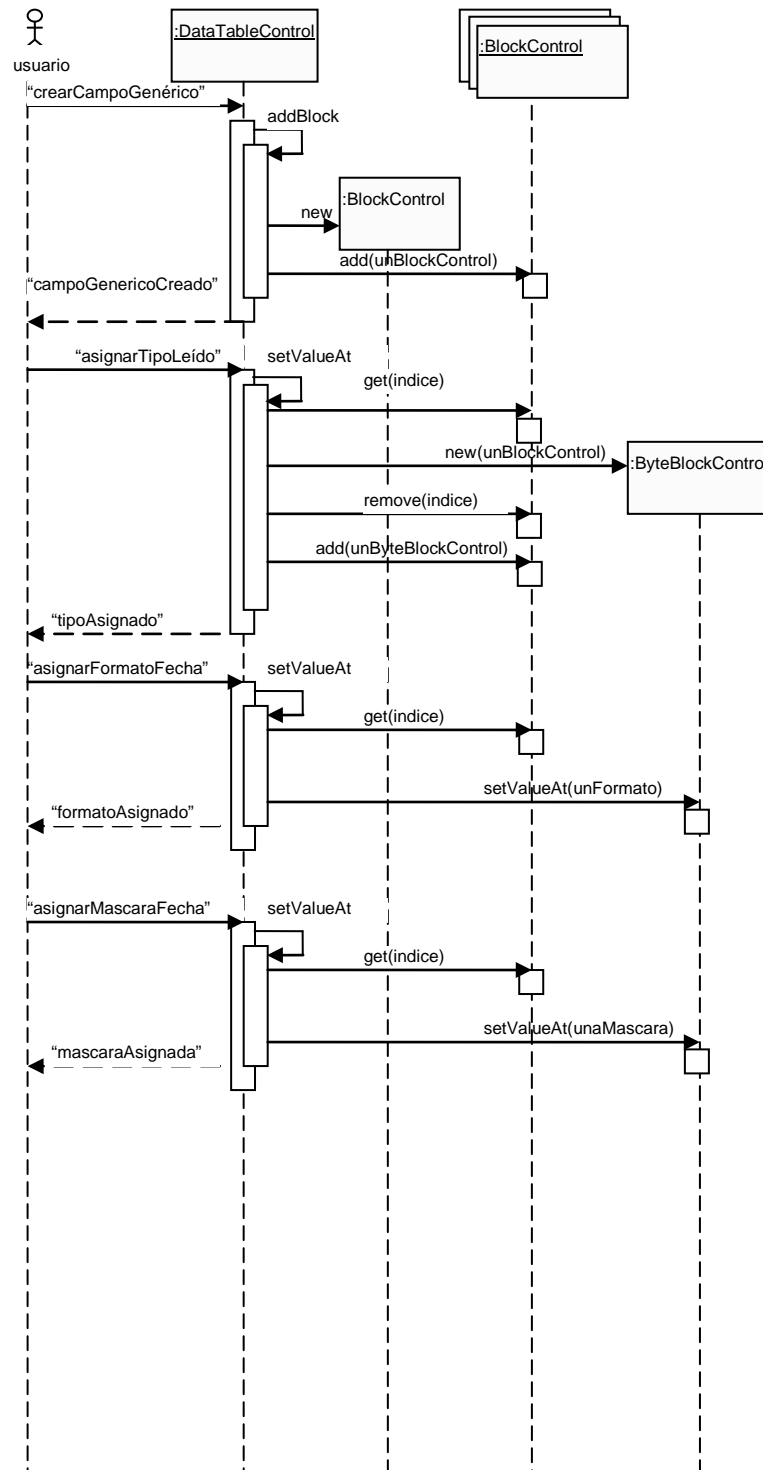


Figura D.5: Usuario genera un campo con formato fecha y máscara correcta

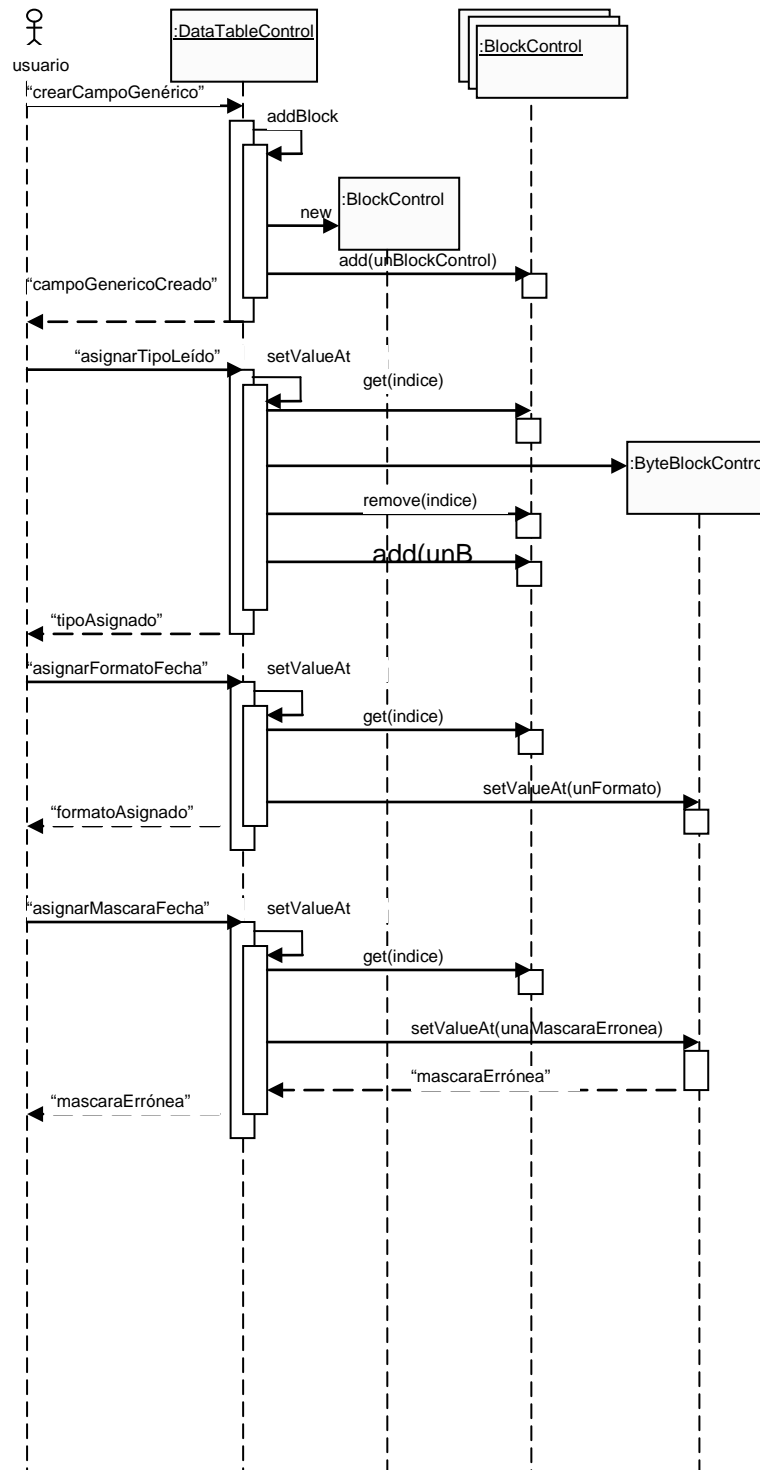


Figura D.6: Usuario genera un campo con formato fecha y máscara incorrecta

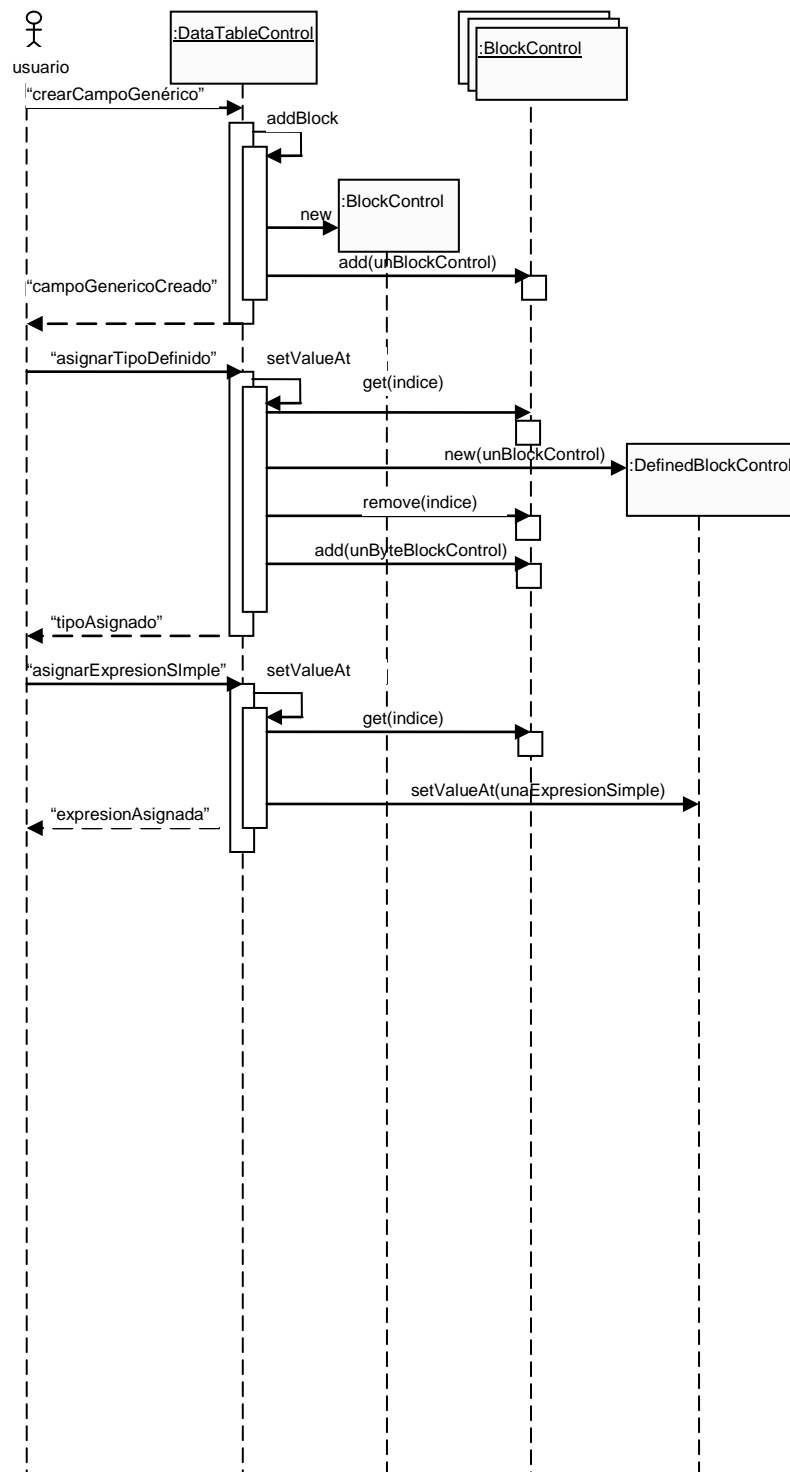


Figura D.7: Usuario genera un campo de definición simple correctamente

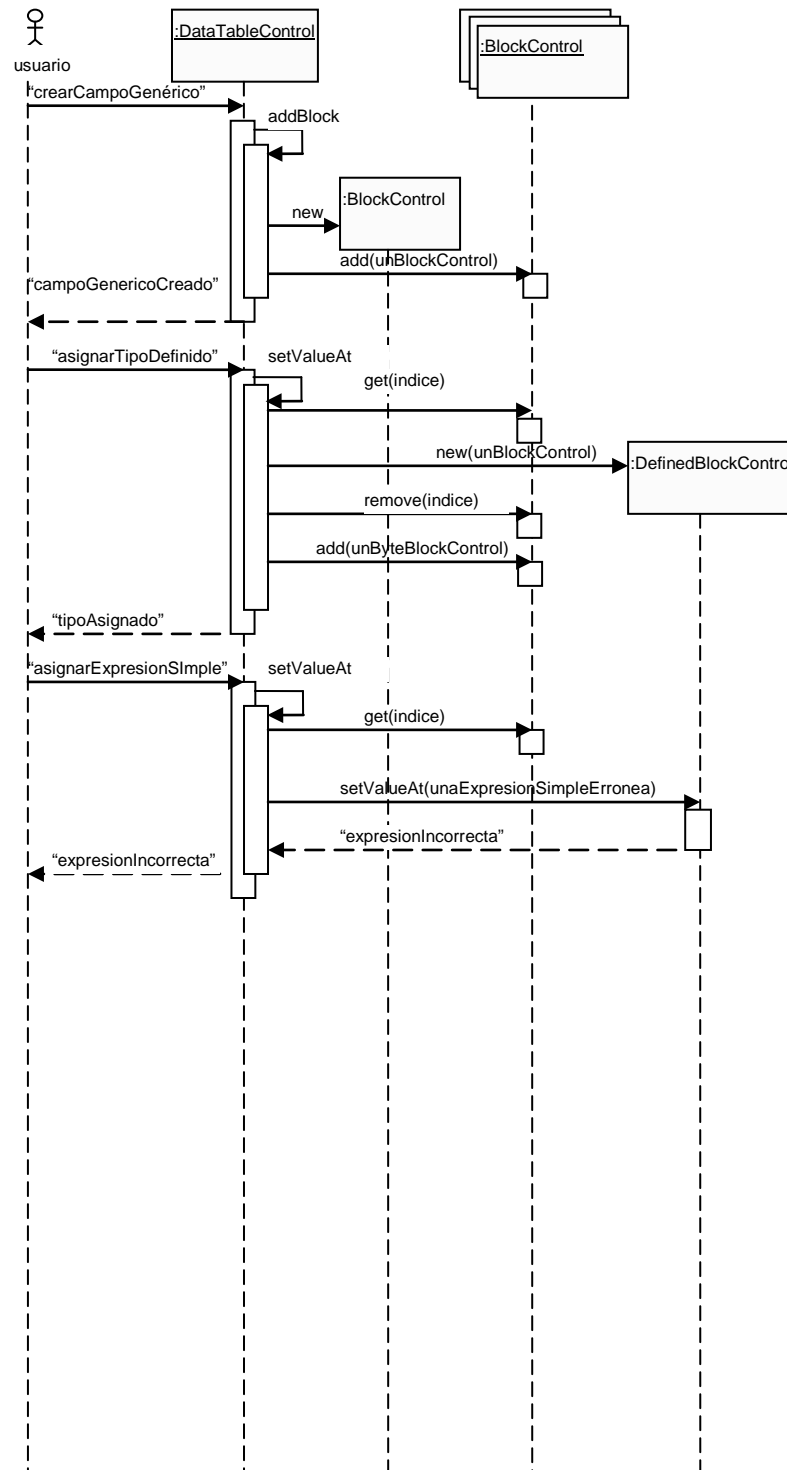


Figura D.8: Usuario genera un campo de definición simple incorrectamente

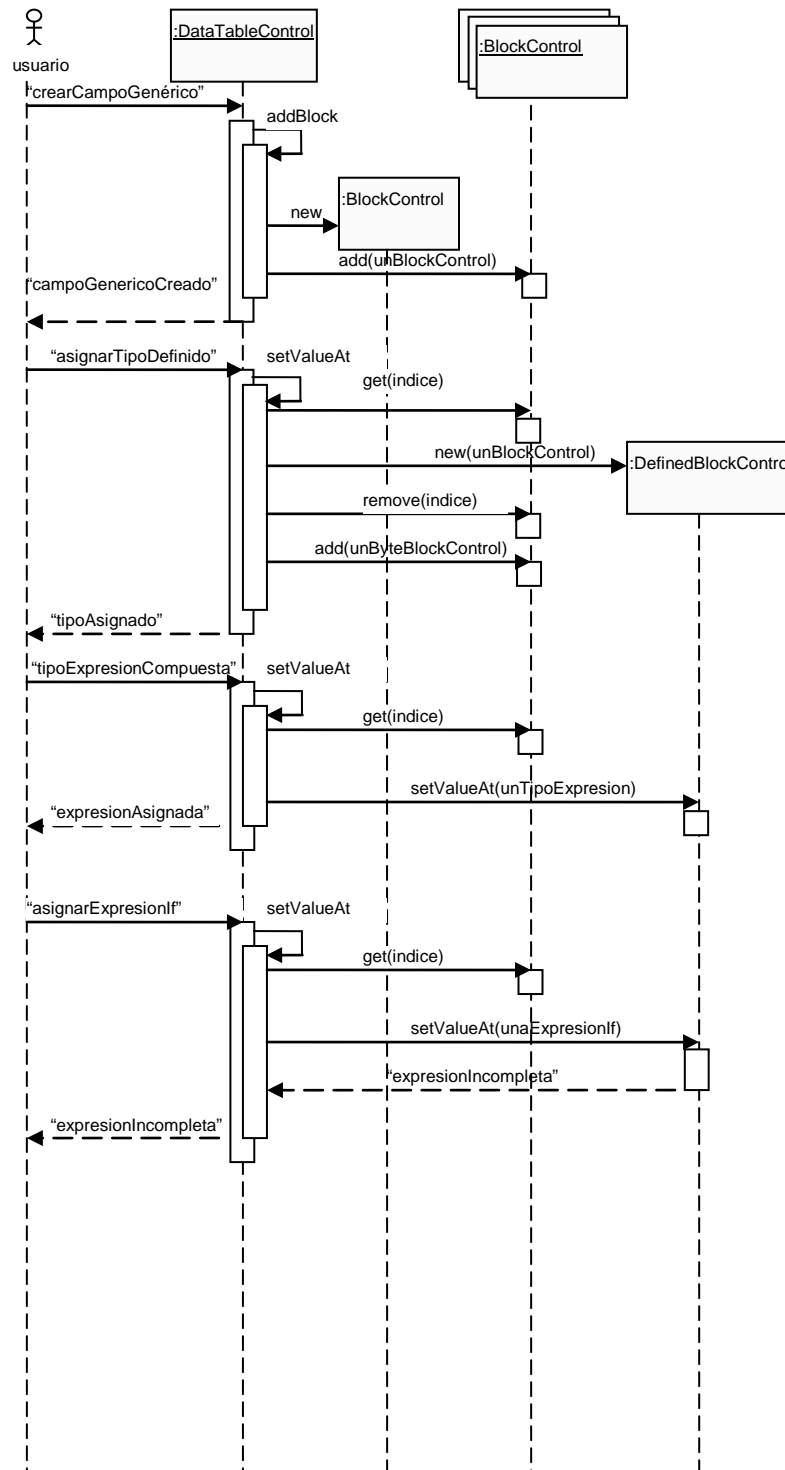


Figura D.9: Usuario genera un campo de definición compuesta incompleto

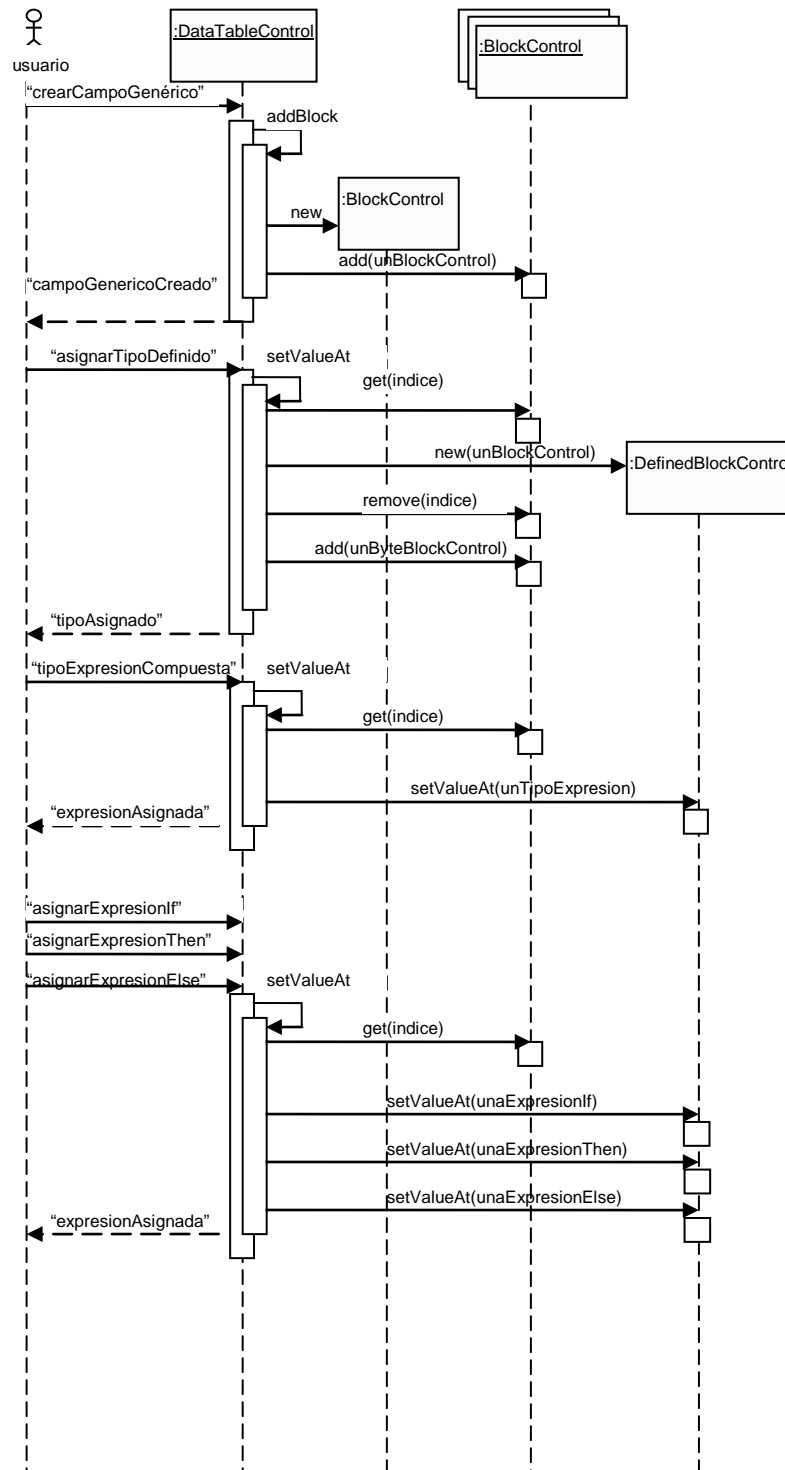


Figura D.10: Usuario genera un campo de definición compuesta correctamente

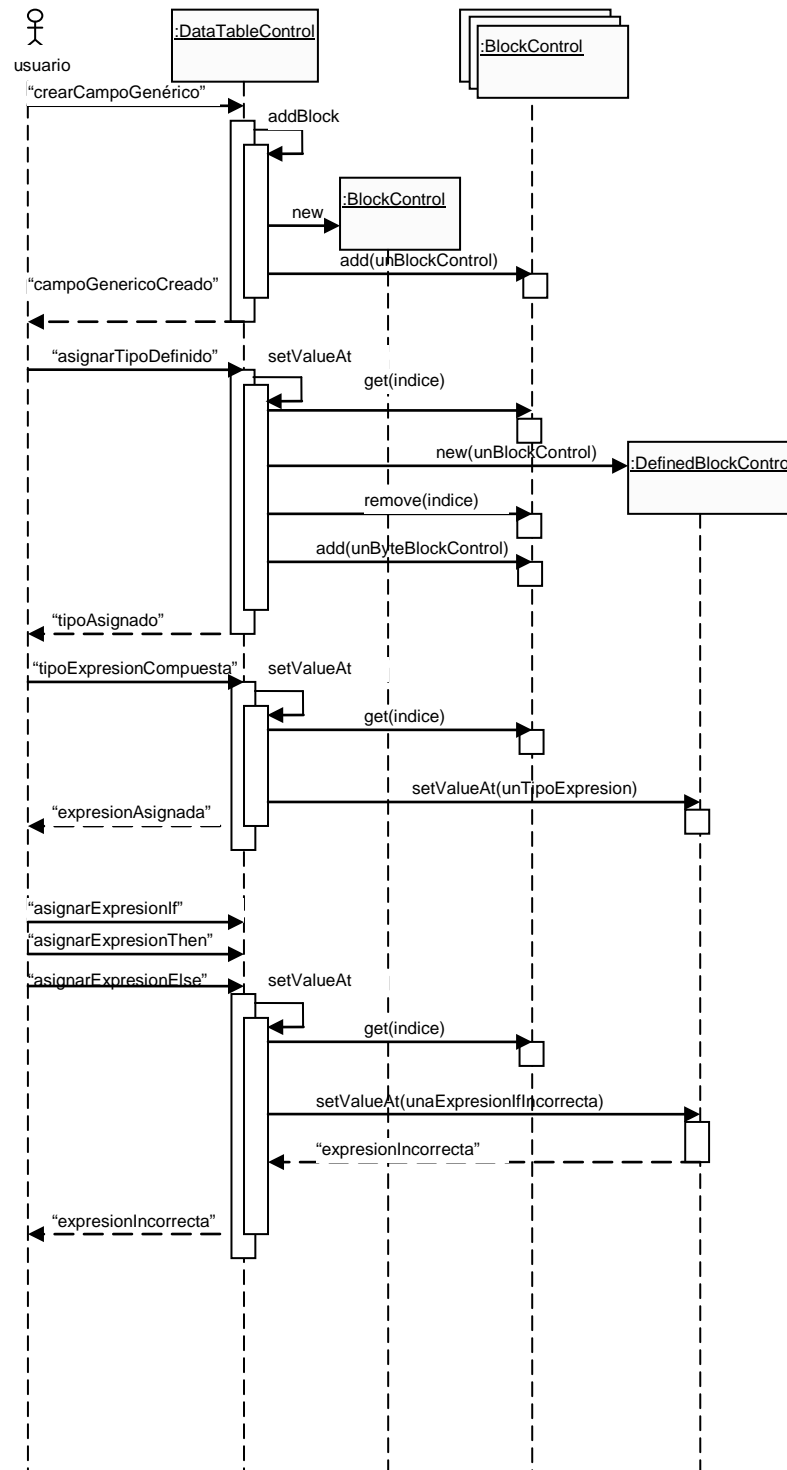


Figura D.11: Usuario genera un campo de definición compuesta incorrectamente

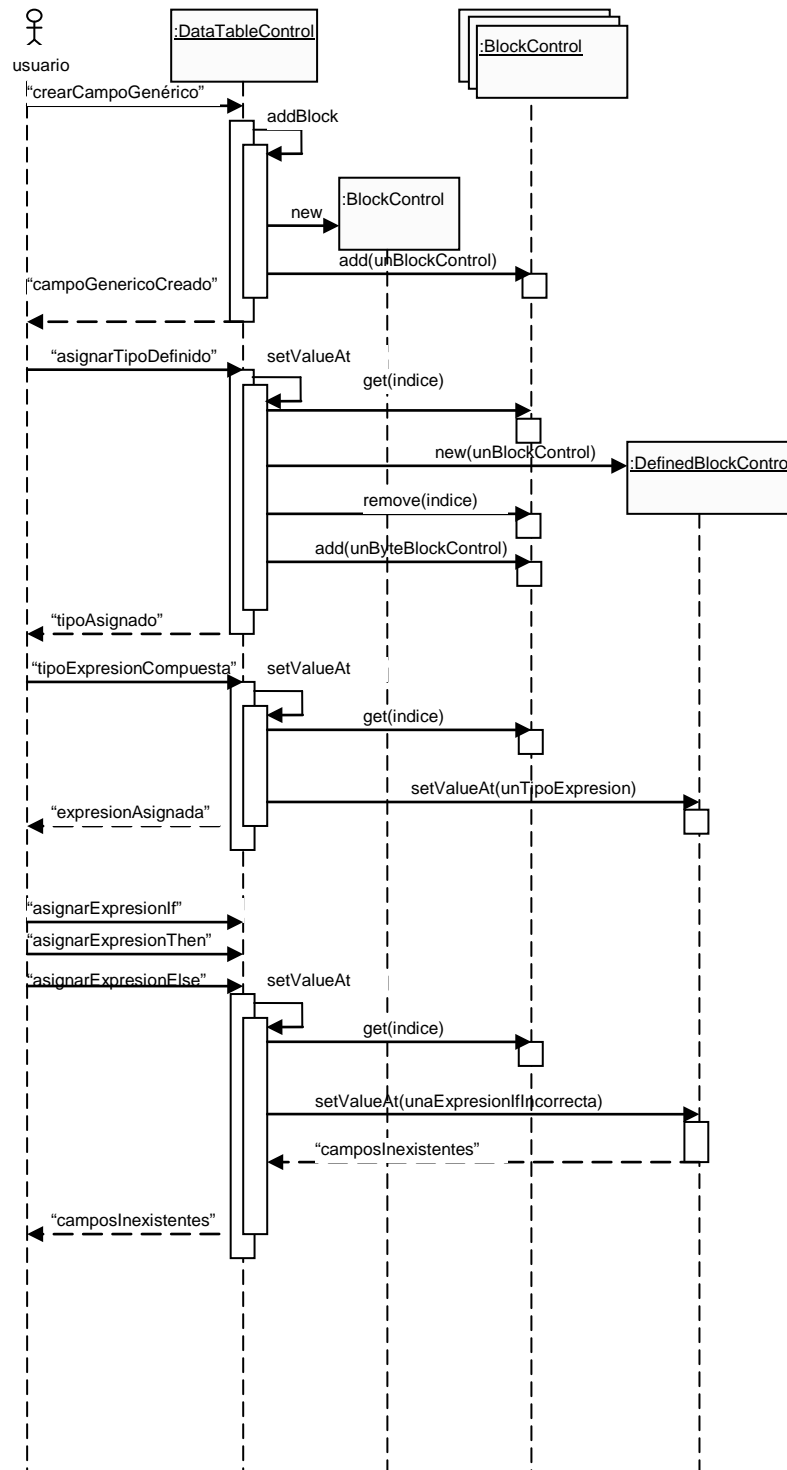


Figura D.12: Usuario referencia campos inexistentes en expresión compuesta

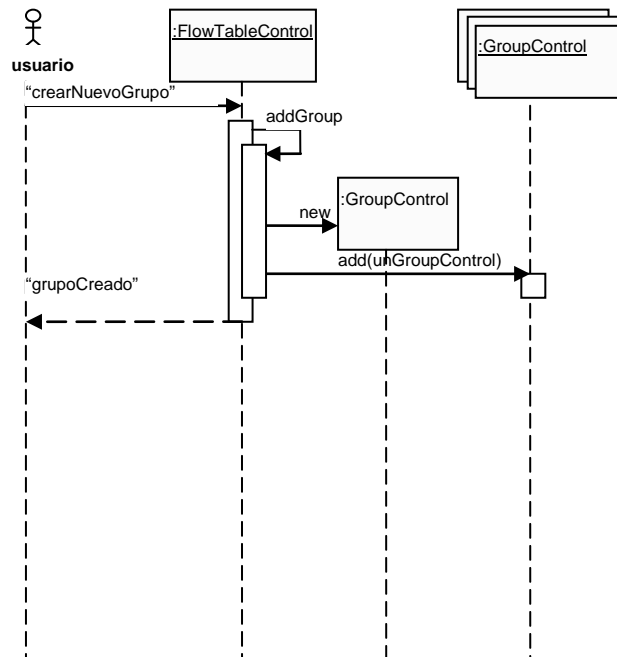


Figura D.13: Usuario crea un grupo con condición por omisión

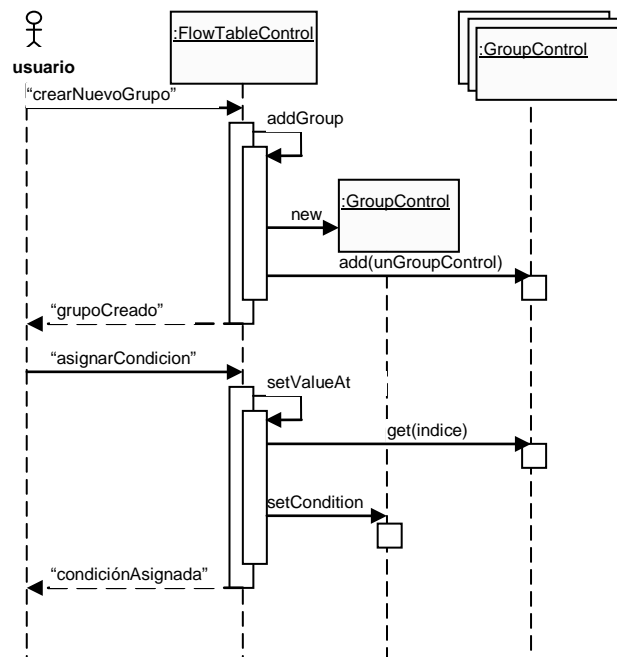


Figura D.14: Usuario crea un grupo con condición basada en campos

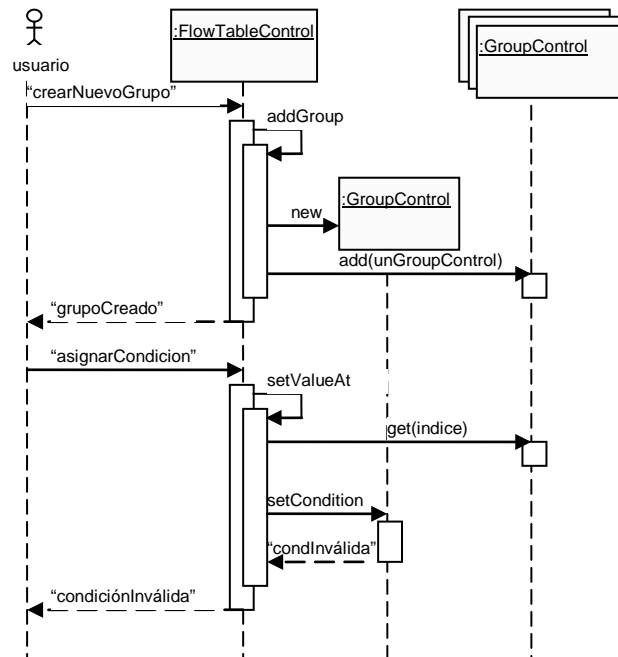


Figura D.15: Usuario crea un grupo con condición inválida

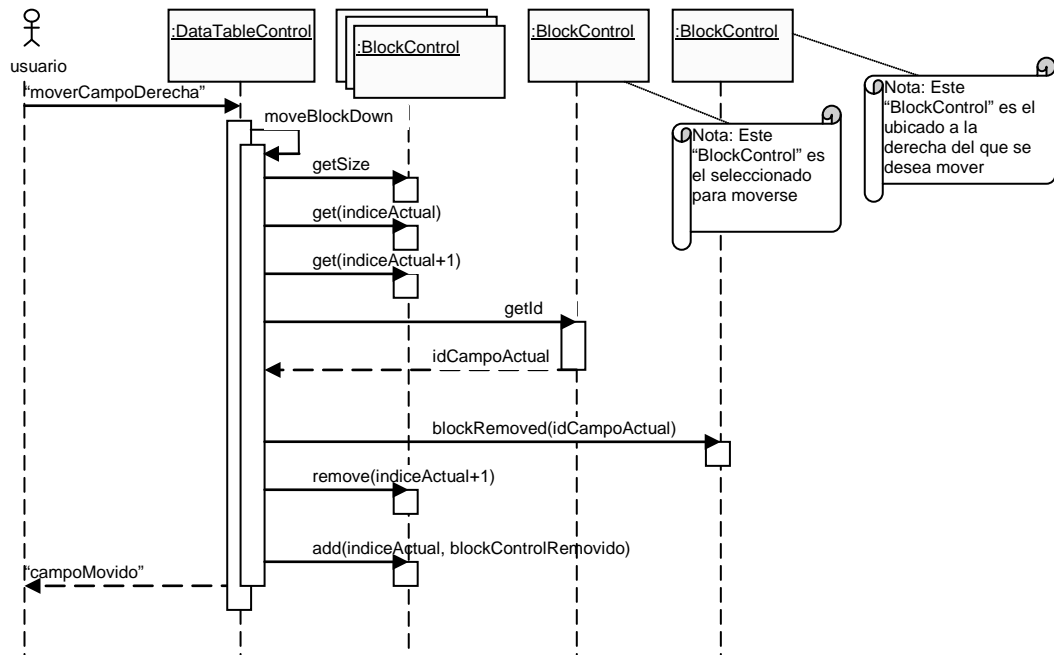


Figura D.16: Usuario mueve campo a la derecha

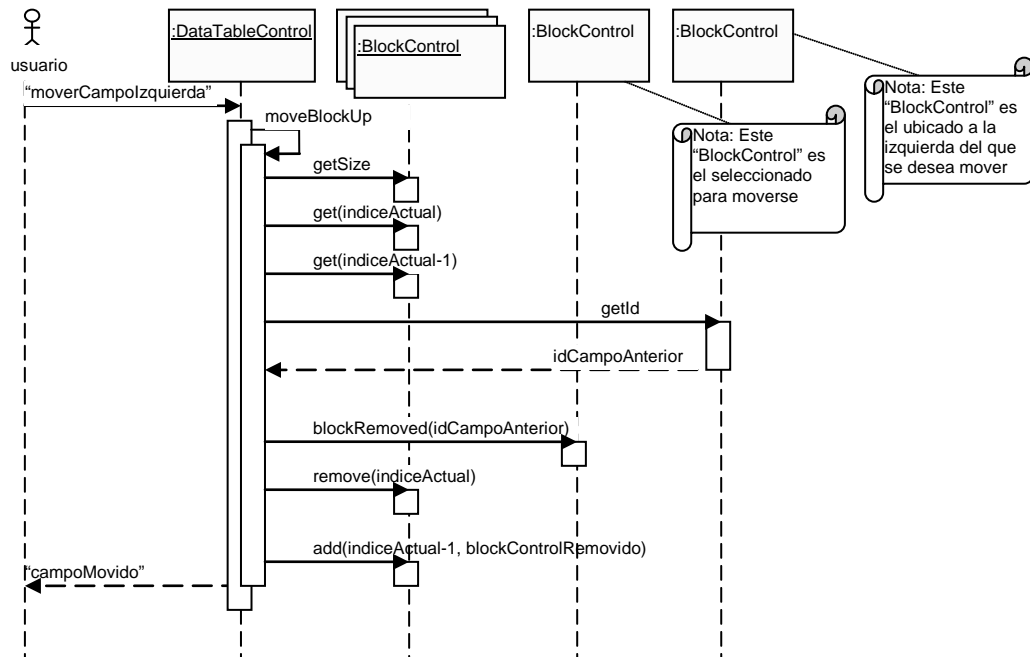


Figura D.17: Usuario mueve campo a la izquierda

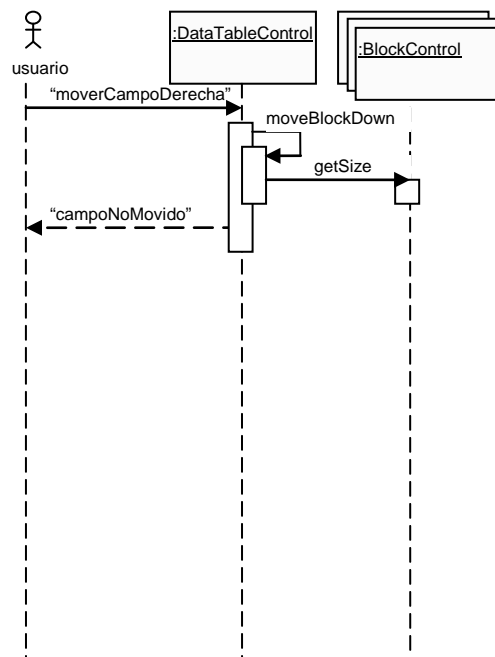


Figura D.18: Usuario no puede mover campo seleccionado porque es el último

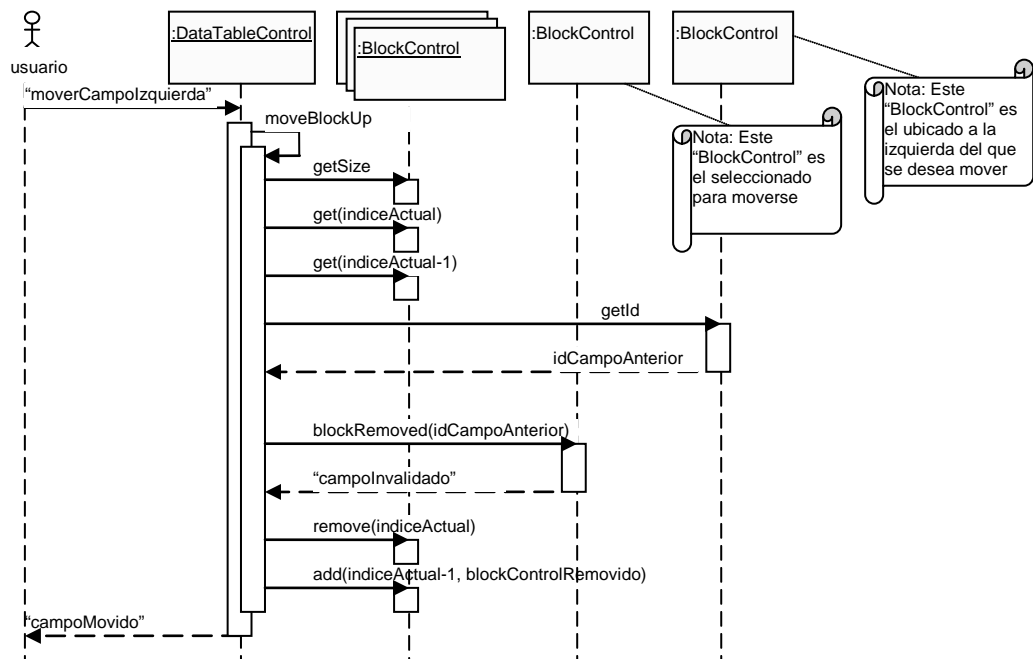


Figura D.19: Usuario mueve campo a la izquierda, pero invalida el campo actual

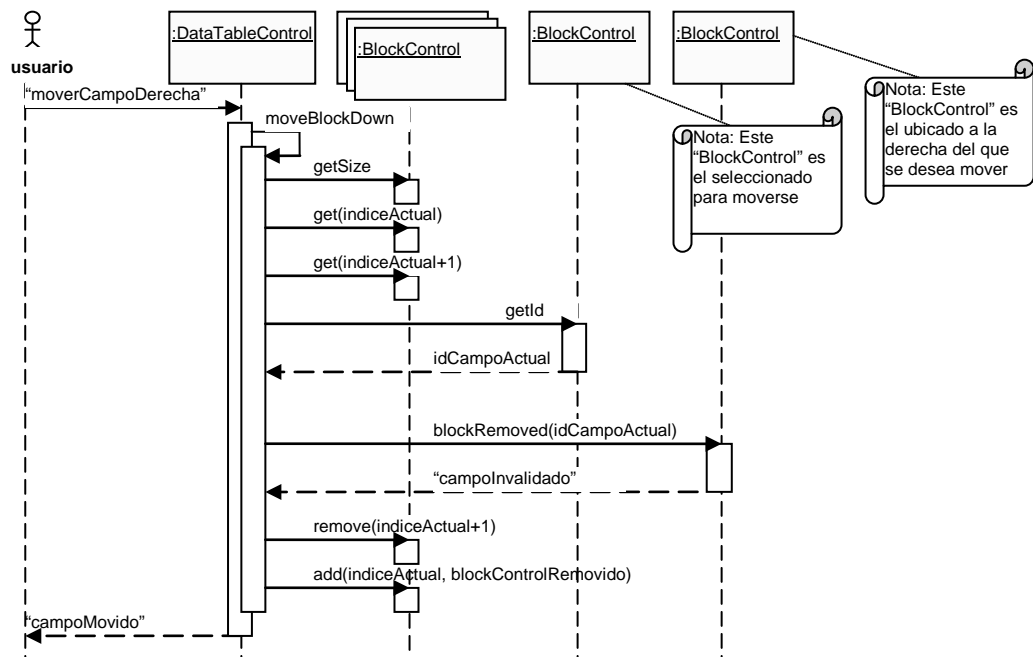


Figura D.20: Usuario mueve campo a la derecha, pero invalida el campo siguiente

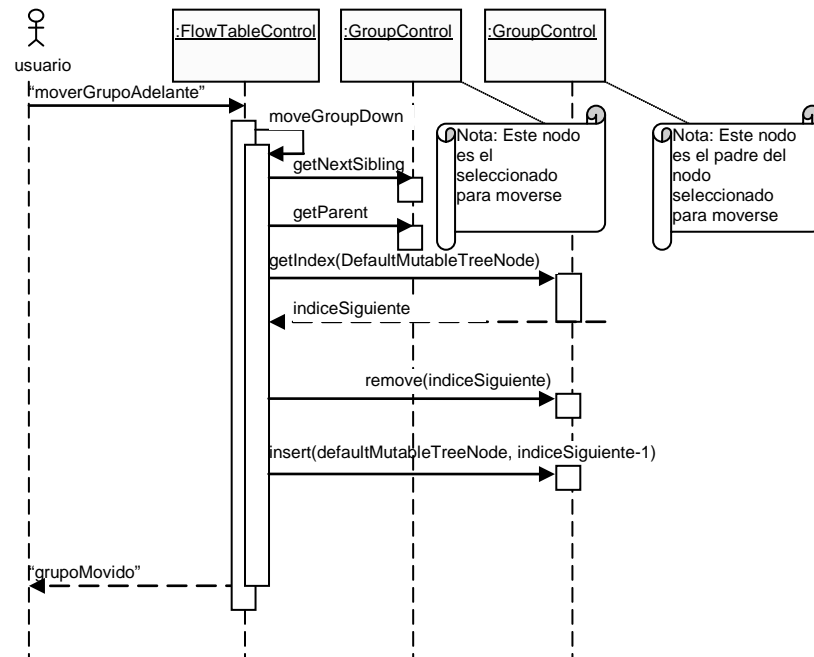


Figura D.21: Usuario mueve grupo después del siguiente

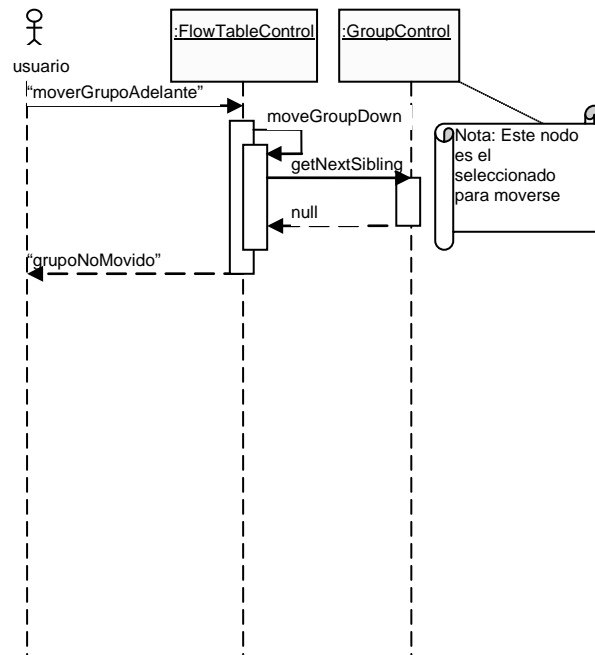


Figura D.22: Usuario no puede mover grupo porque no existen otros

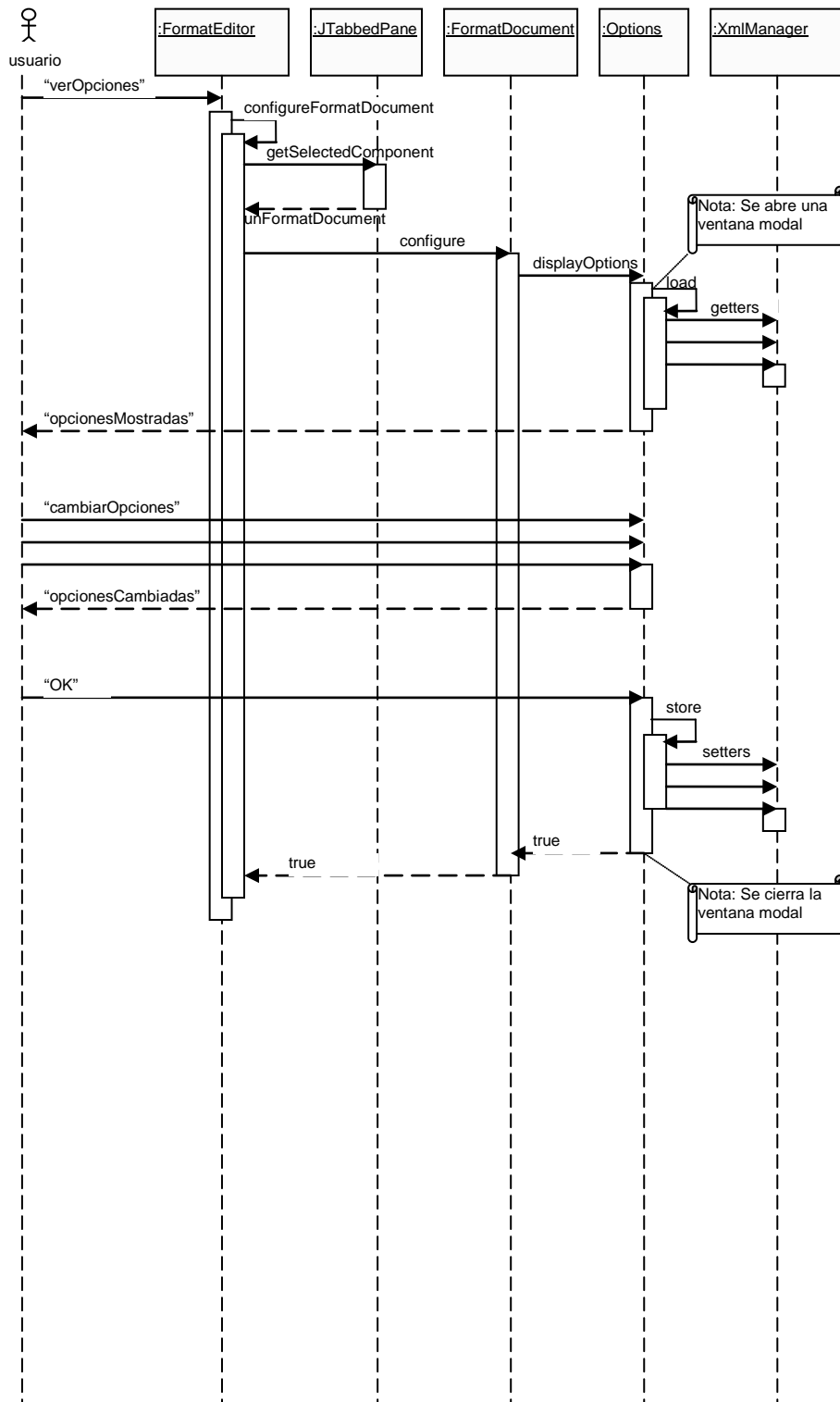


Figura D.23: Usuario modifica las opciones generales del documento

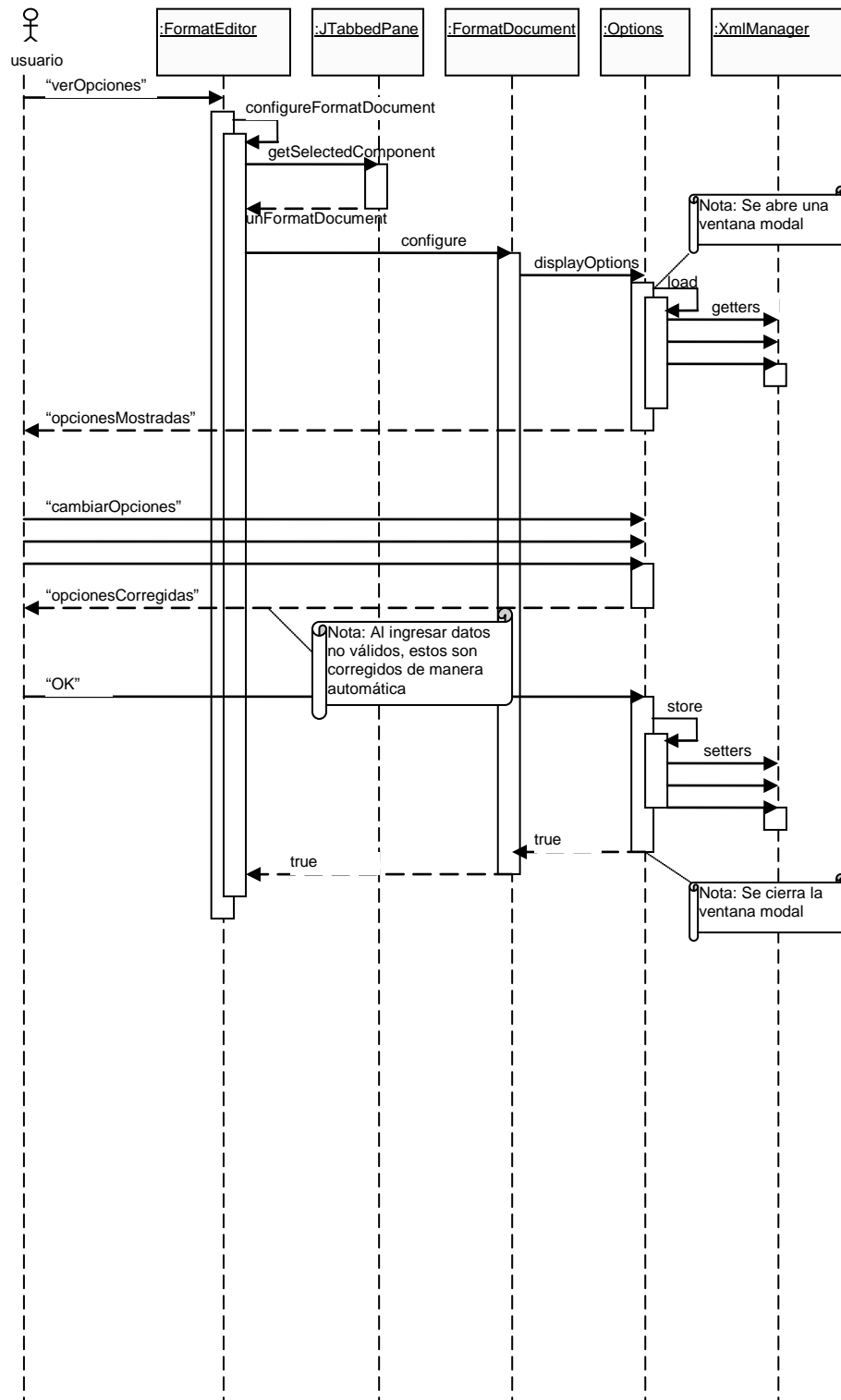


Figura D.24: Usuario modifica con errores las opciones generales del documento

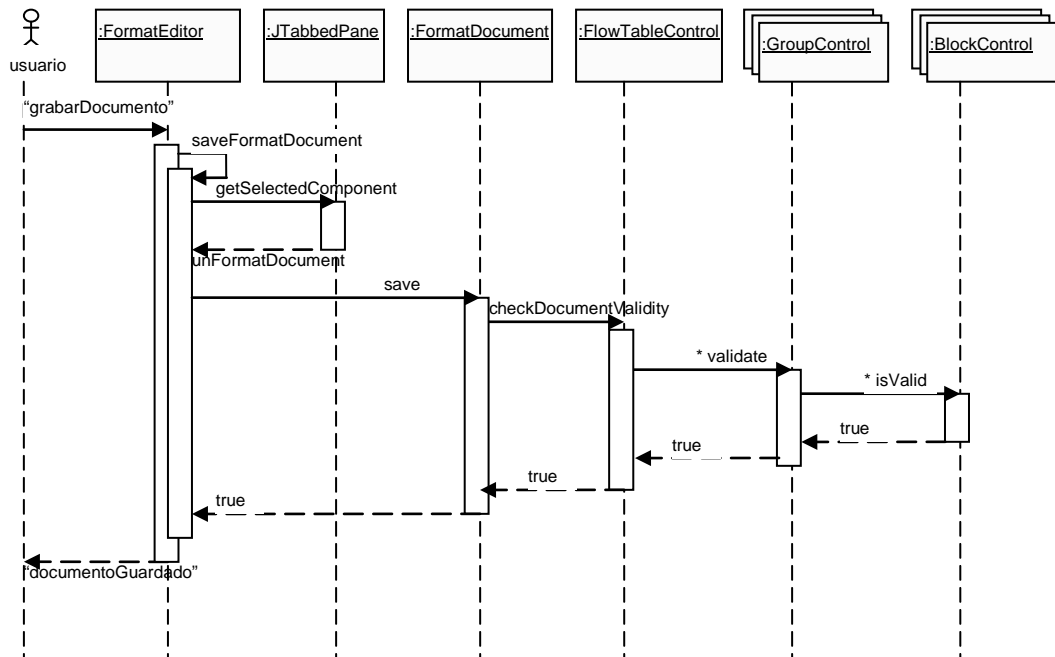


Figura D.25: Usuario guarda un documento exitosamente

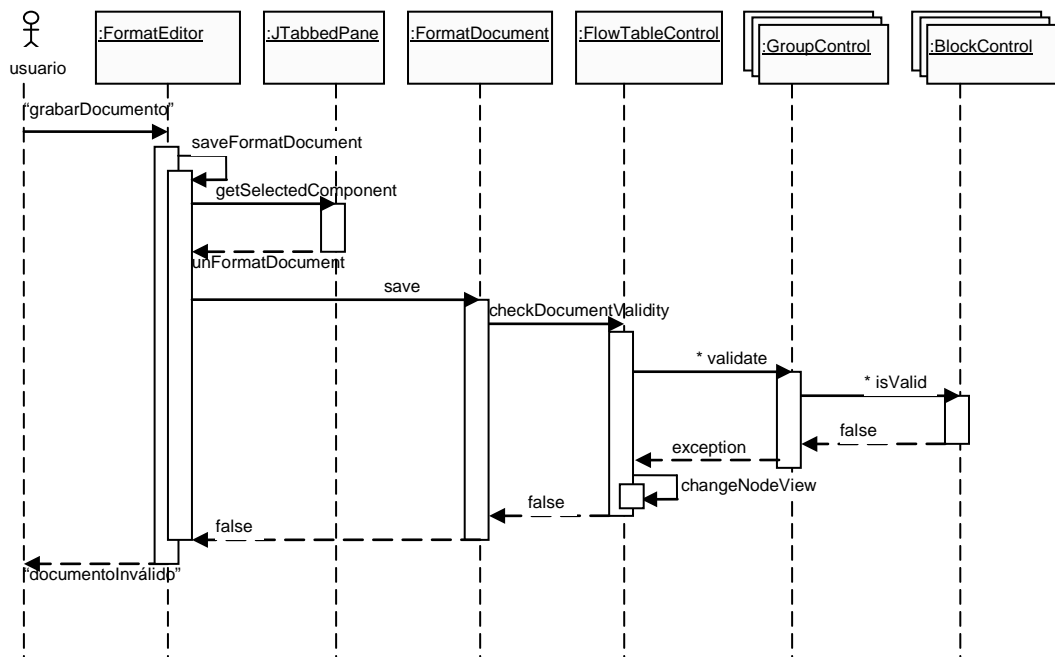


Figura D.26: Usuario intenta guardar un documento inválido

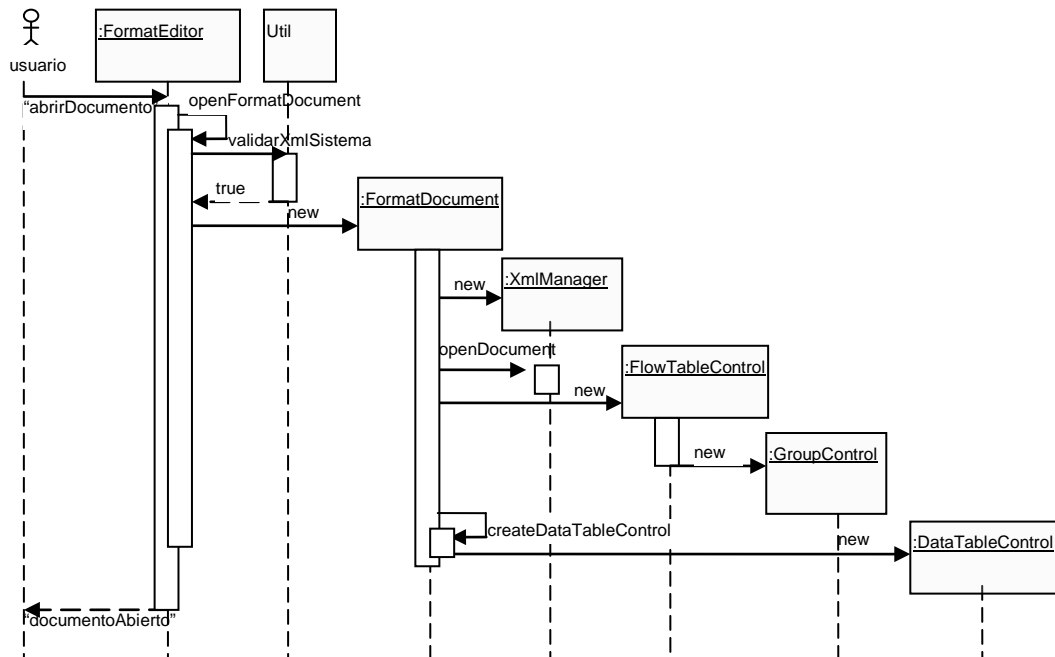


Figura D.27: Usuario abre un documento

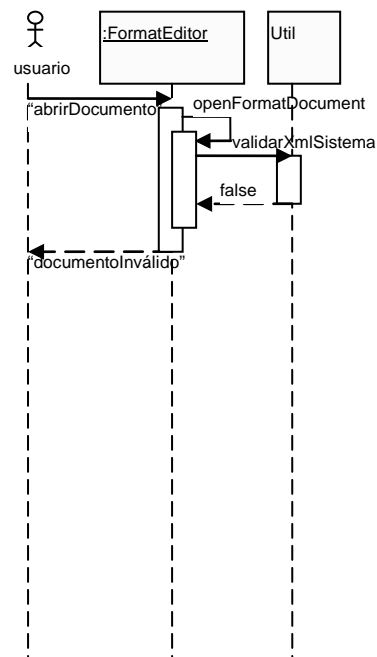


Figura D.28: Usuario intenta abrir un documento inválido

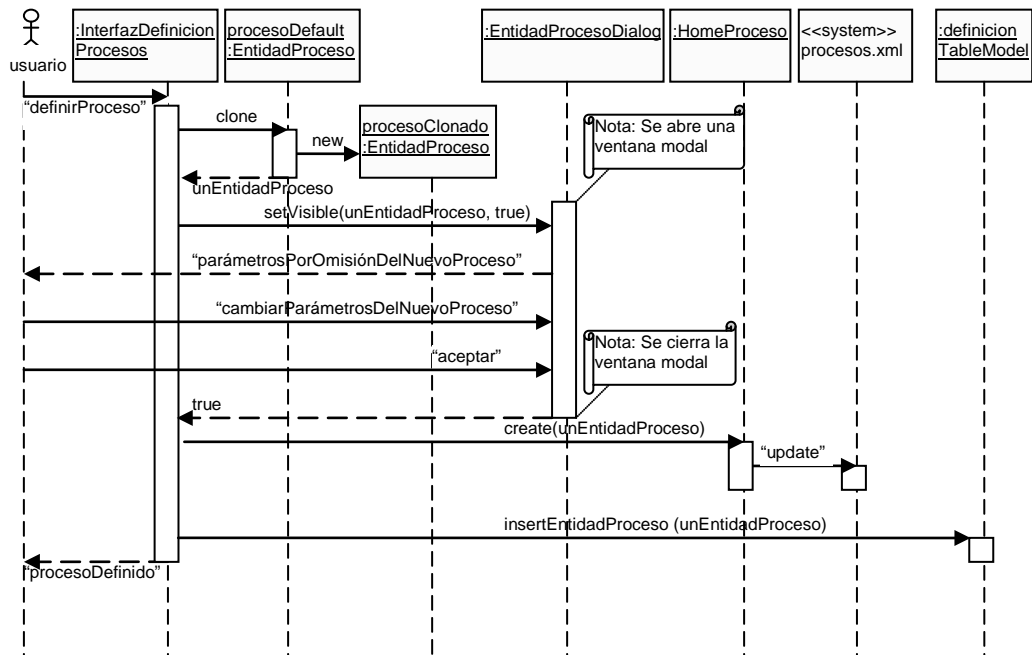


Figura D.29: Usuario define un nuevo proceso

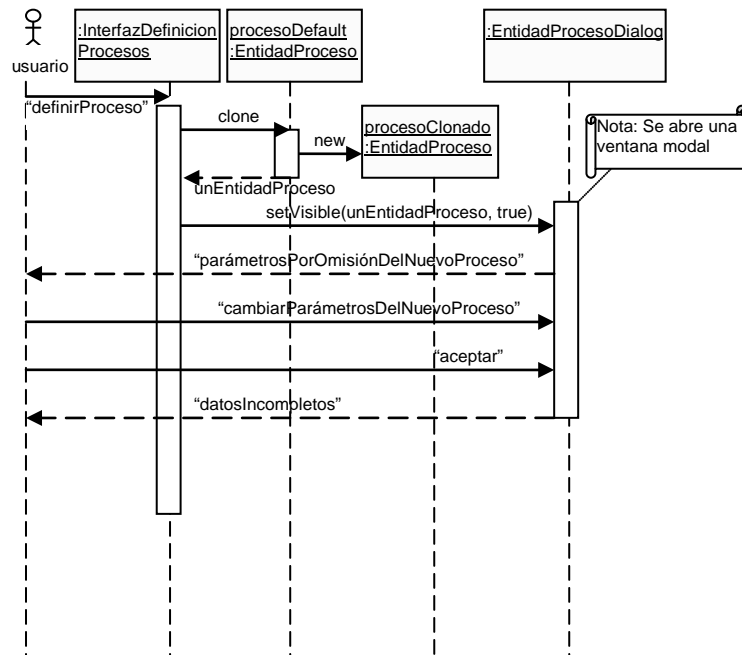


Figura D.30: Usuario define un nuevo proceso con datos incompletos

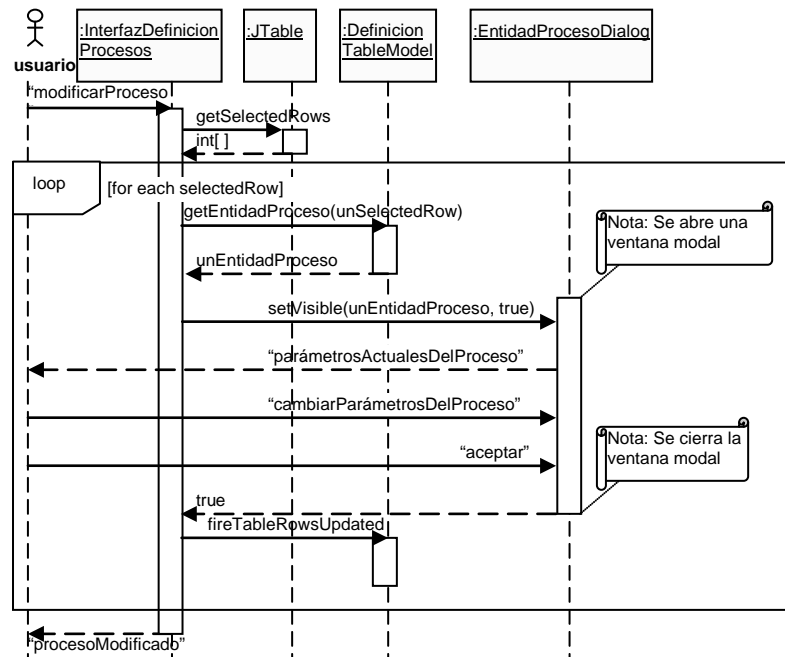


Figura D.31: Usuario modifica un proceso

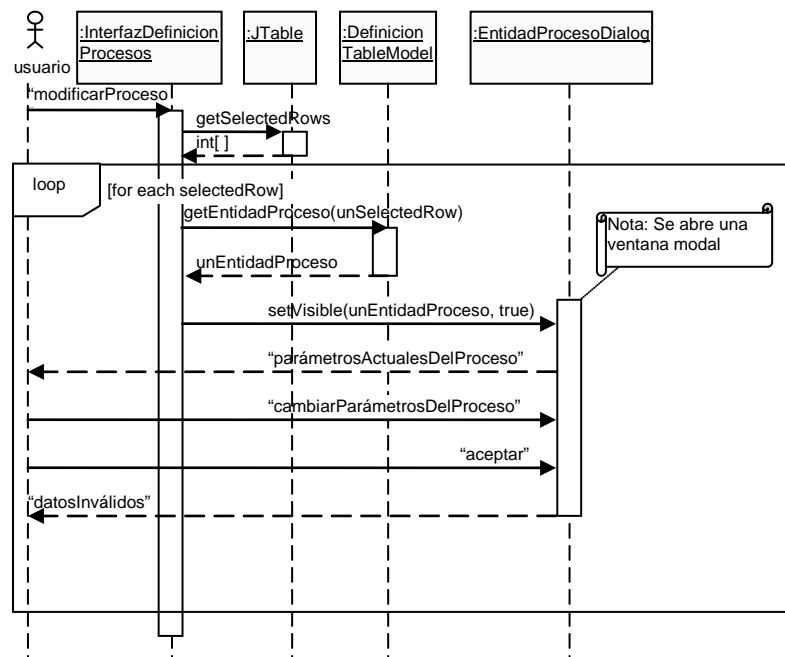


Figura D.32: Usuario modifica un proceso con errores

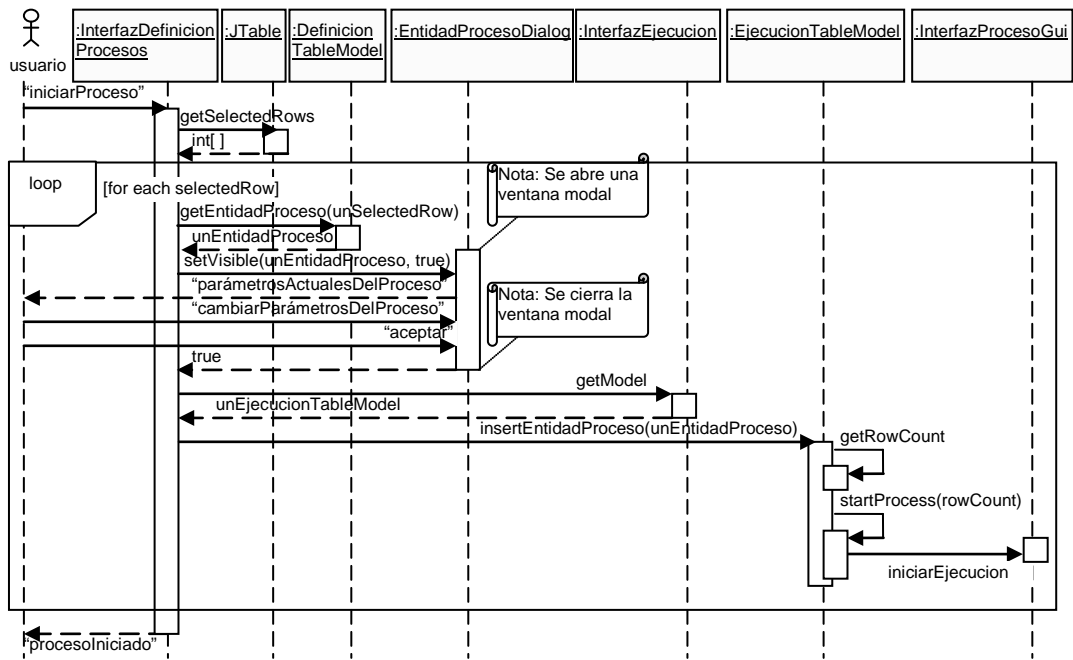


Figura D.33 a: Usuario inicia un proceso

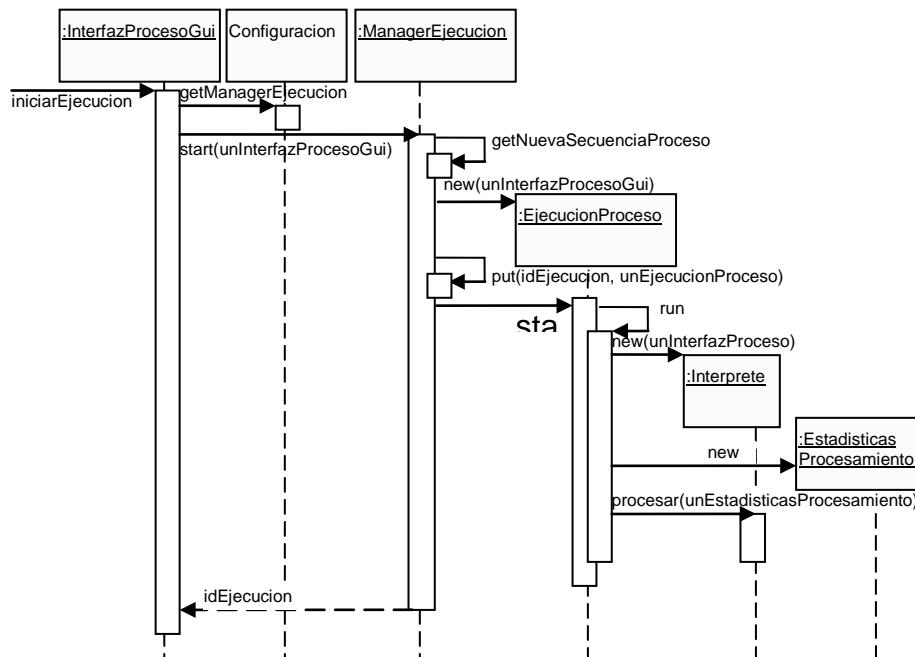


Figura D.33 b: Usuario inicia un proceso

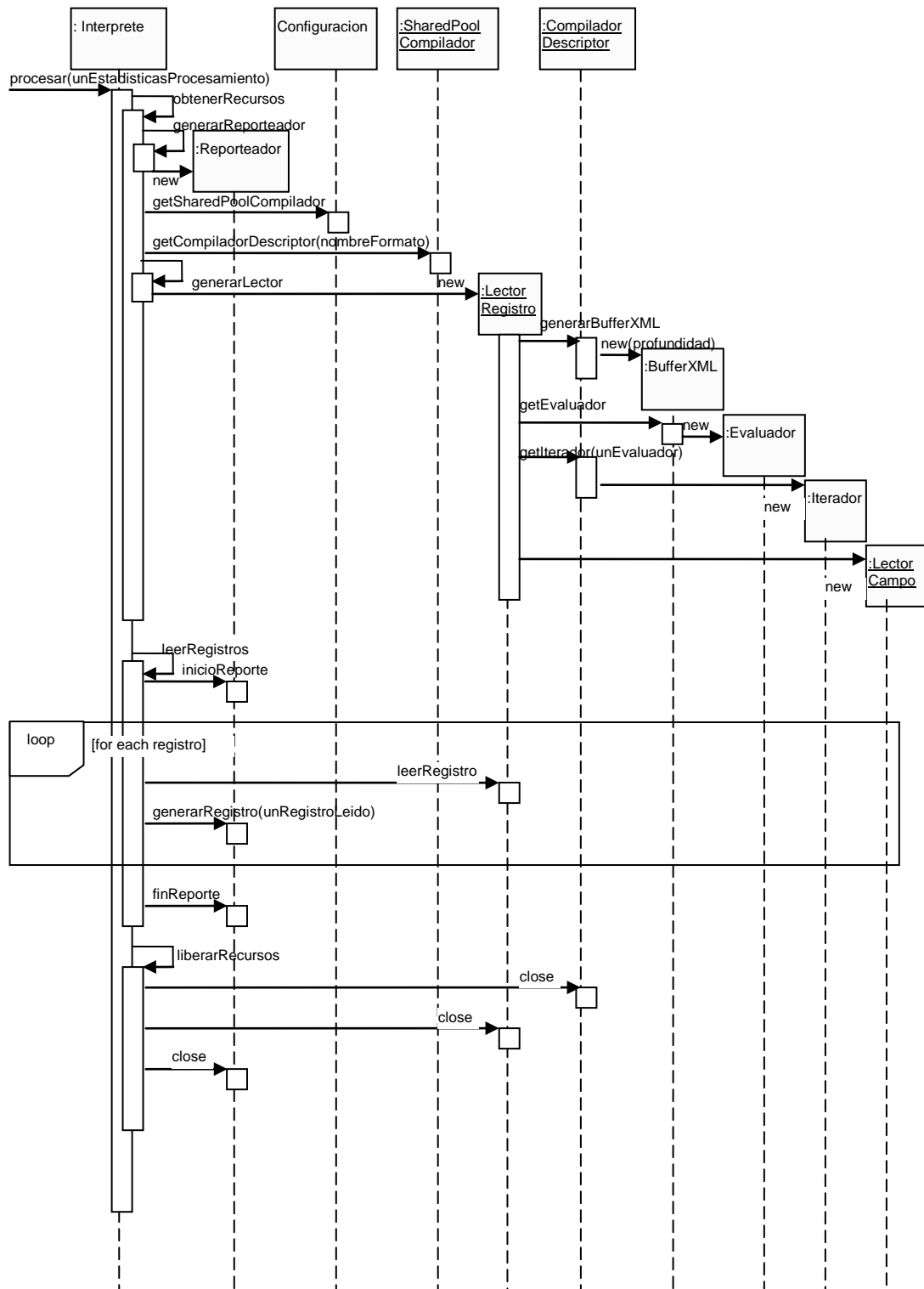


Figura D.33 c: Usuario inicia un proceso

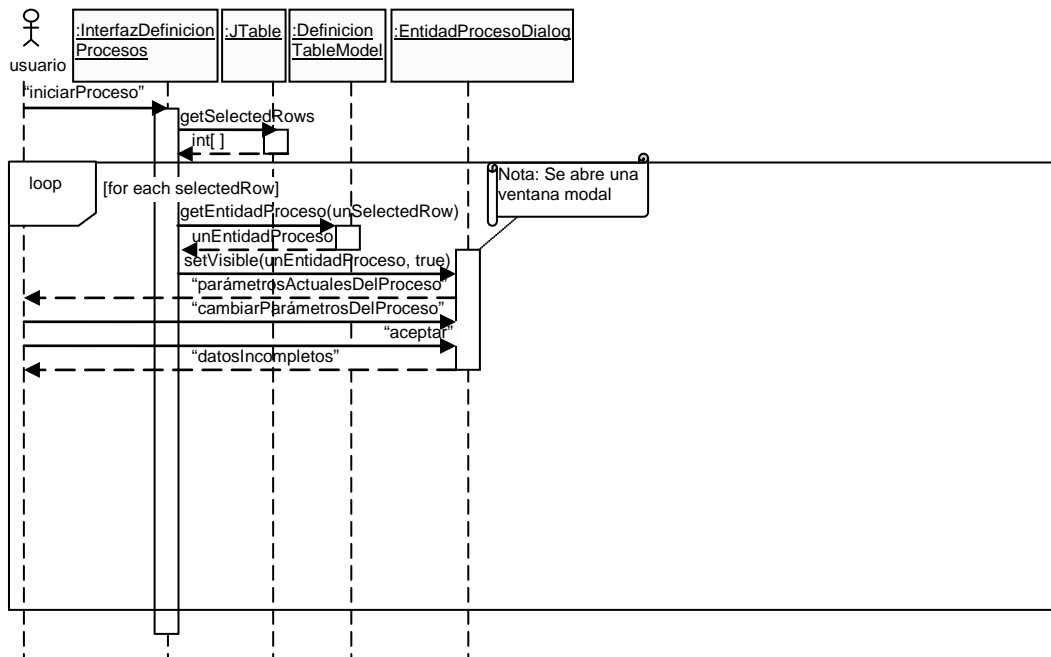


Figura D.34: Usuario inicia un proceso con datos incompletos

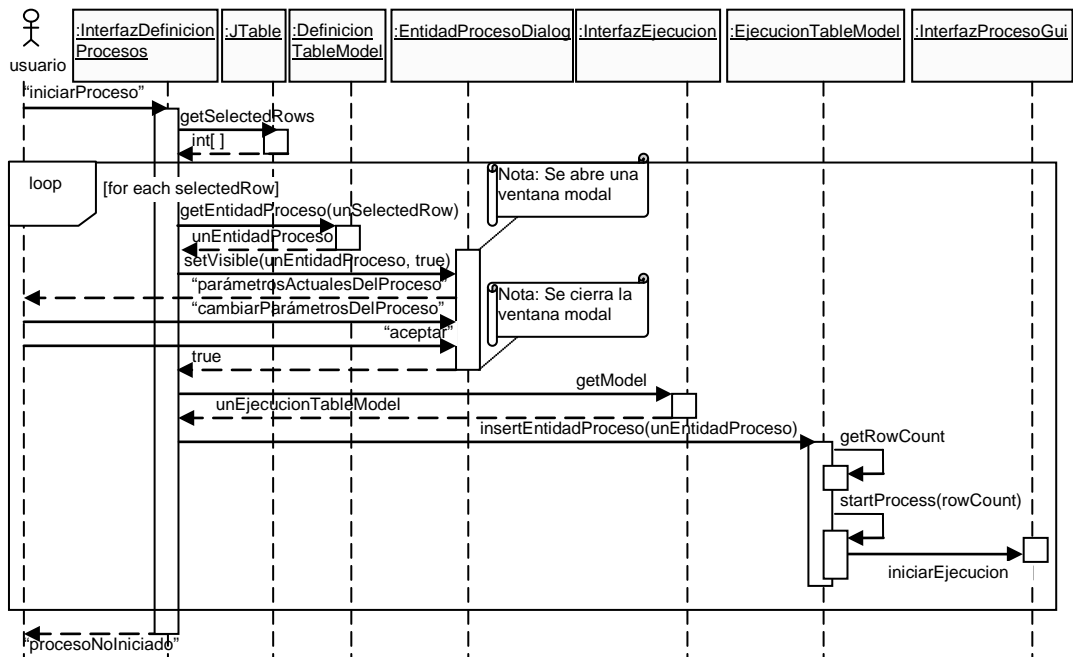


Figura D.35 a: Usuario inicia un proceso que falla por parámetros incorrectos

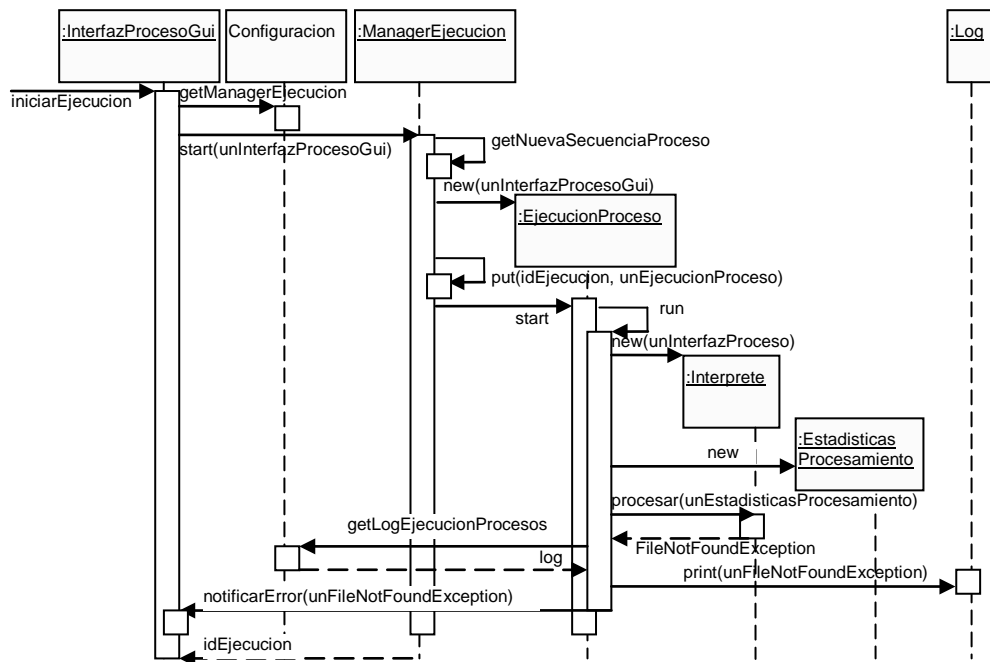


Figura D.35 b: Usuario inicia un proceso que falla por parámetros incorrectos

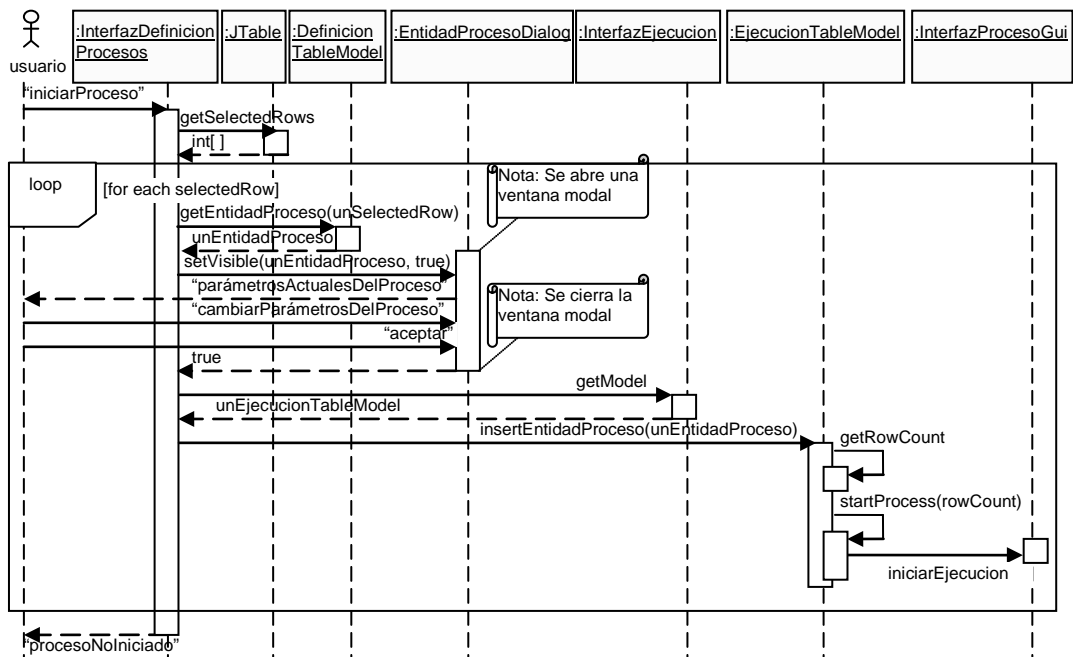


Figura D.36 a: Usuario inicia un proceso que falla por problemas de lectura

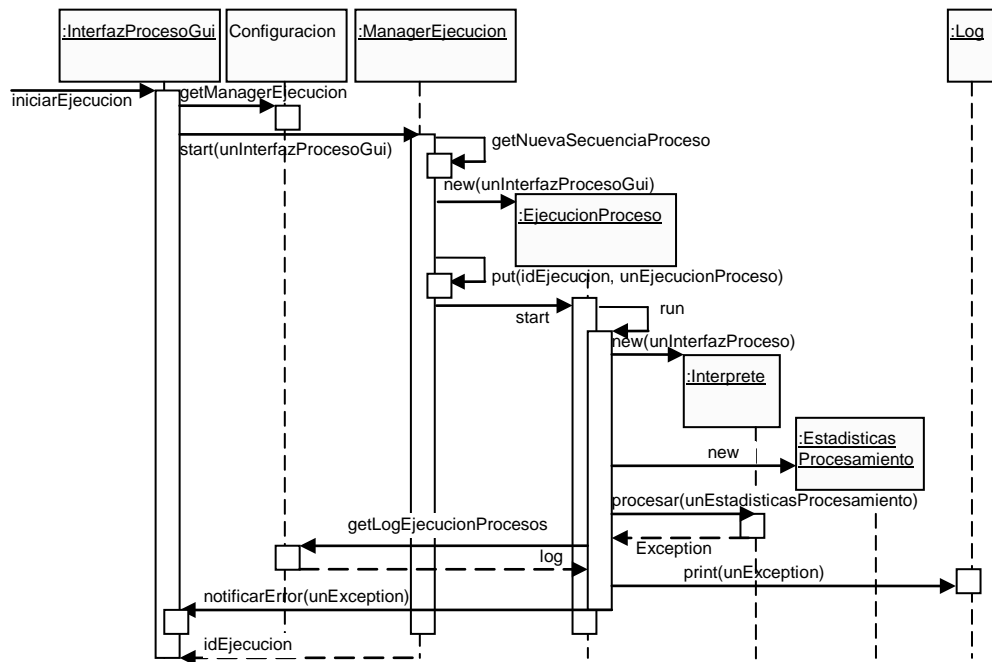


Figura D.36 b: Usuario inicia un proceso que falla por problemas de lectura

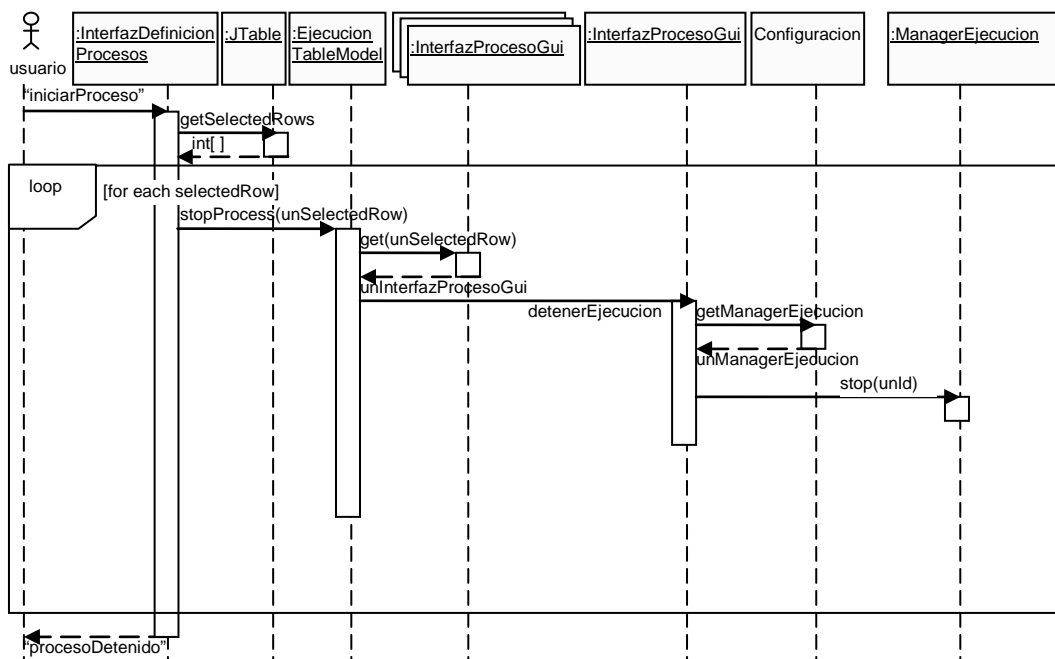


Figura D.37 a: Usuario detiene un proceso

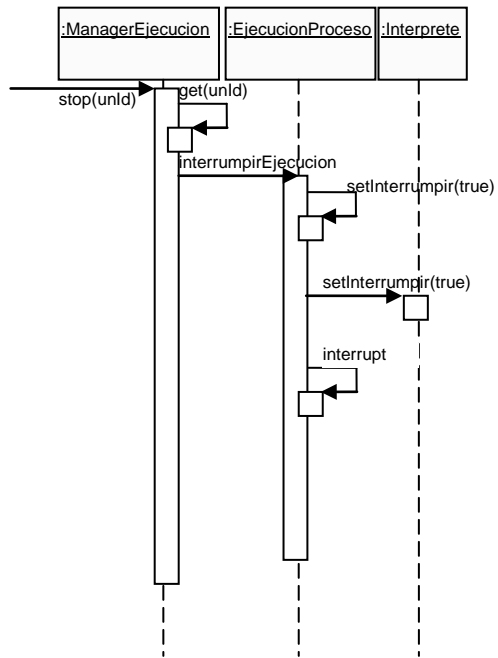


Figura D.37 b: Usuario detiene un proceso

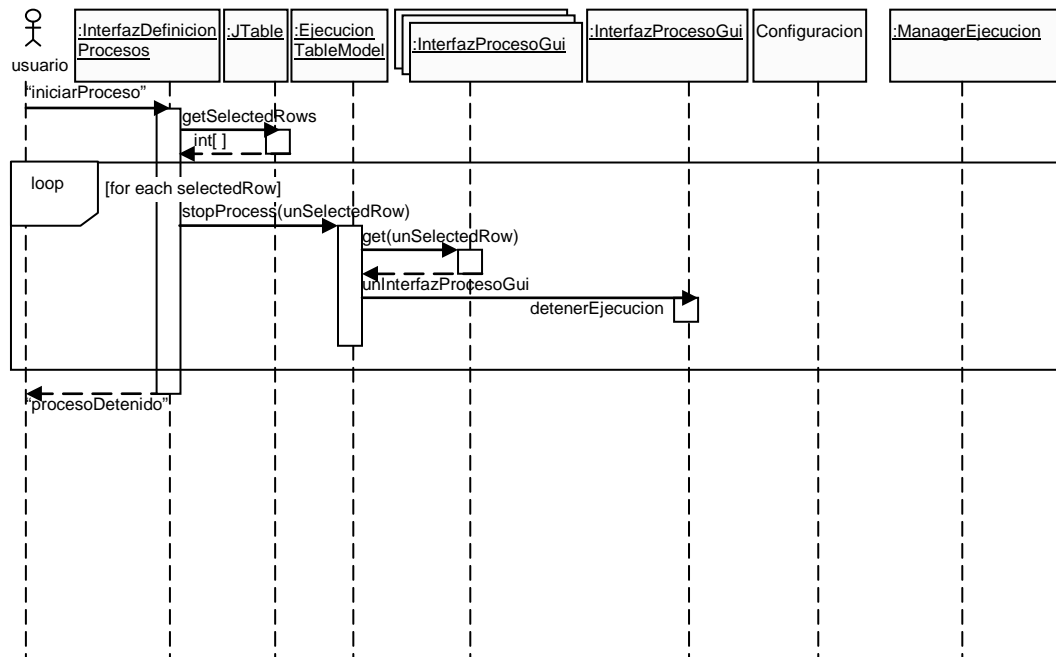


Figura D.38: Usuario intenta detener procesos que no se están ejecutando

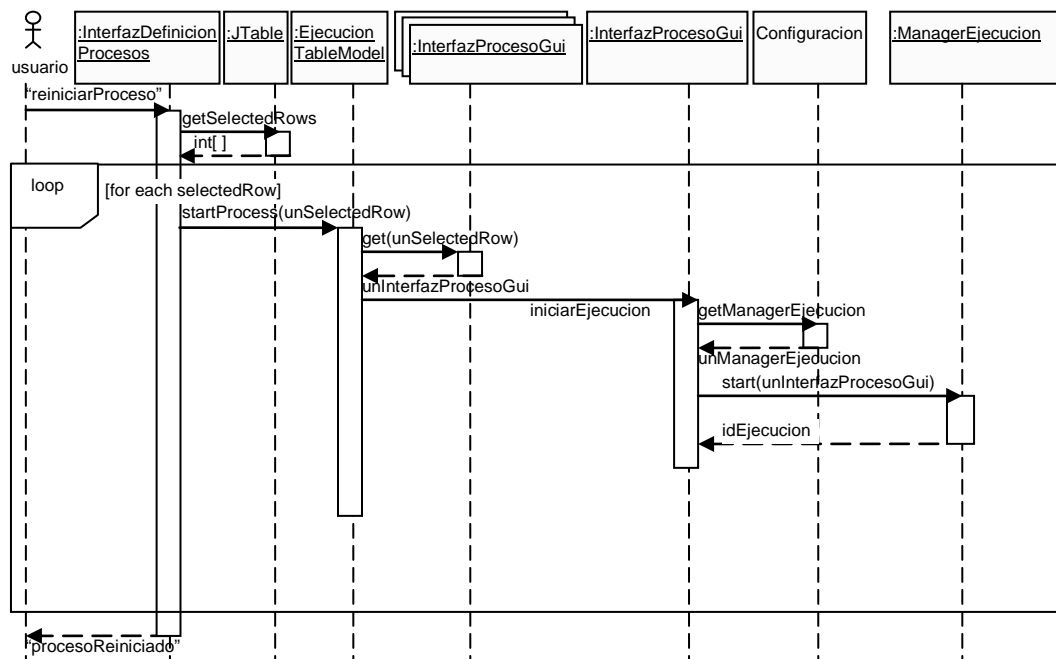


Figura D.39 a: Usuario reinicia procesos detenidos

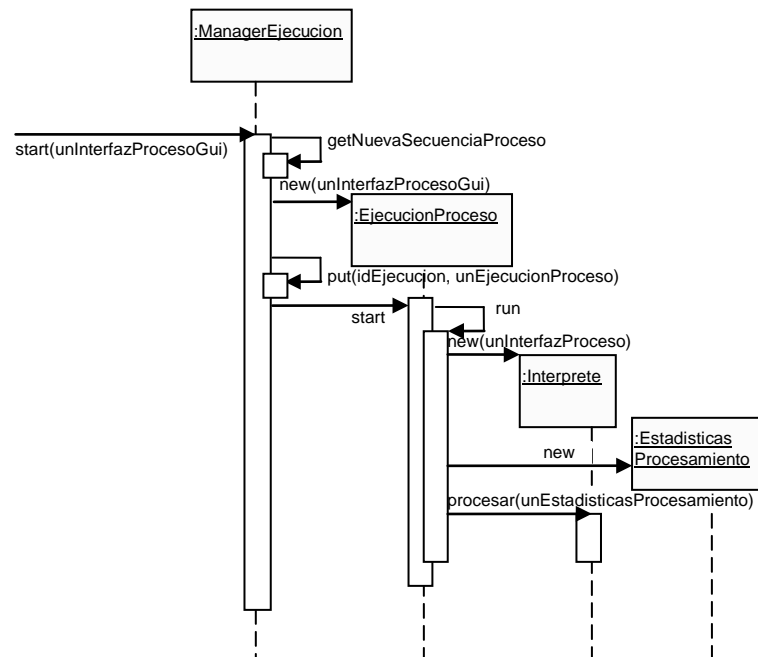


Figura D.39 b: Usuario reinicia procesos detenidos

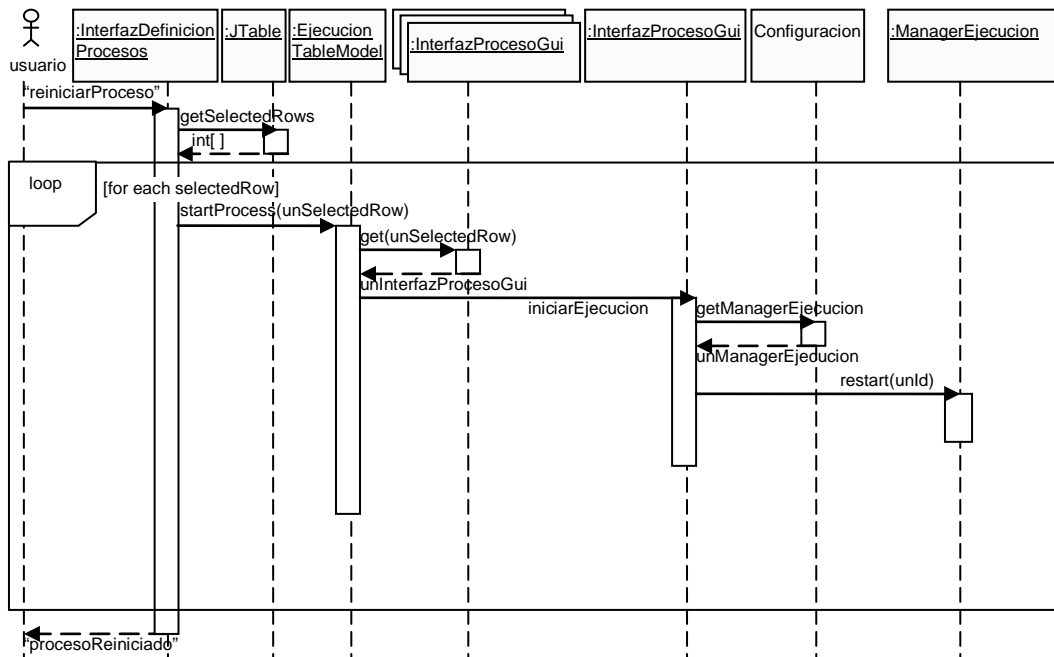


Figura D.40 a: Usuario reinicia procesos en ejecución

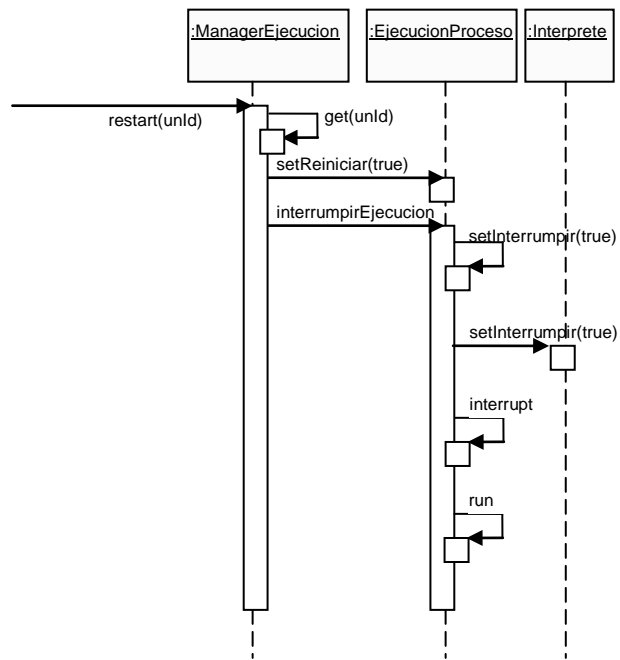


Figura D.40 b: Usuario reinicia procesos en ejecución

APÉNDICE E: DOCUMENTOS XSD

DEL SISTEMA

A continuación se muestra el documento XSD empleado para especificar un documento descriptor de formatos generado por el sistema:

```
<?xml version="1.0"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="descriptorRegistro">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="metaDataRegistro">
          <xsd:complexType>
            <xsd:all>
              <xsd:element name="tipoRegistro">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
```

```

        <xsd:pattern value="binario"/>
        <xsd:pattern value="texto"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="codificacion" type="xsd:string"/>
<xsd:element name="idRegistro" type="xsd:string"/>
<xsd:element name="longitudMagnitudRegistro" type="tipoMagnitud"/>
<xsd:element name="longitudIdCampo" type="tipoMagnitud"/>
<xsd:element name="longitudMagnitudCampo" type="tipoMagnitud"/>
<xsd:element name="delimitadorCampo" type="xsd:string"/>
<xsd:element name="delimitadorRegistro" type="xsd:string"/>
<xsd:element name="longitudDelimitador" type="tipoMagnitud"/>
<xsd:element name="intervaloRegistroSeparador" type="xsd:int"/>
<xsd:element name="longitudRegistroSeparador" type="tipoMagnitud"/>
<xsd:element name="comentario" type="xsd:string"/>

    </xsd:all>
</xsd:complexType>
</xsd:element>
<xsd:element name="descriptorCampos" type="tipoDescriptorCampos"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:complexType name="tipoDescriptorCampos">
    <xsd:sequence>
        <xsd:element name="metaDataCampos">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="descripcion" type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>

```

```

    </xsd:element>

    <xsd:element name="campos" type="tipoCampos" minOccurs="1"
                maxOccurs="1"/>

    <xsd:element name="bifurcacion" type="tipoBifurcacion" minOccurs="0"/>
</xsd:sequence>

<xsd:attribute name="id" type="xsd:integer" use="required"/>
</xsd:complexType>
<xsd:complexType name="tipoCampos">
  <xsd:sequence>
    <xsd:element name="campo" type="tipoCampo" minOccurs="0"
                maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="tipoCampo">
  <xsd:sequence>
    <xsd:element name="metaData" type="tipoMetaData"/>
    <xsd:element name="datos"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:integer" use="required"/>
</xsd:complexType>
<xsd:complexType name="tipoMetaData">
  <xsd:all>
    <xsd:element name="descripcion" type="xsd:string"/>
    <xsd:element name="magnitud" type="tipoMagnitud"/>
    <xsd:element name="formato">
      <xsd:complexType>
        <xsd:attribute name="tipo" type="xsd:string" use="required"/>
        <xsd:attribute name="mascara" type="xsd:string"/>
        <xsd:attribute name="invertirBytes" type="xsd:string"/>
        <xsd:attribute name="rellenoDerecha" type="xsd:string"/>
        <xsd:attribute name="rellenoIzquierda" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:all>
</xsd:complexType>

```

```

</xsd:element>

<xsd:element name="expresion">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="simple" type="xsd:string"/>
      <xsd:sequence>
        <xsd:element name="if" type="xsd:string"/>
        <xsd:element name="then" type="xsd:string"/>
        <xsd:element name="else" type="xsd:string"/>
      </xsd:sequence>
    </xsd:choice>
    <xsd:attribute name="tipo" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
</xsd:all>
<xsd:attribute name="tipo" use="required">
  <!-- use="required"-->
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="leido"/>
      <xsd:enumeration value="expresion"/>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:attribute>
</xsd:complexType>
<xsd:complexType name="tipoBifurcacion">
  <xsd:sequence>
    <xsd:element name="switch">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="option" minOccurs="0"

```

```

        maxOccurs="unbounded">
<xsd:complexType>
  <xsd:sequence>
    <xsd:element name="condition" type="xsd:string"/>
    <xsd:element name="then">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="next"
            type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="default" minOccurs="0" maxOccurs="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="then">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="next"
              type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

```

```

<xsd:element name="opciones">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="descriptorCampos" minOccurs="0"
        maxOccurs="unbounded" type="tipoDescriptorCampos">
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="tipoMagnitud">
  <xsd:simpleContent>
    <xsd:extension base="xsd:double">
      <xsd:attribute name="unidad" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="byte"/>
            <xsd:enumeration value="nibble"/>
            <xsd:enumeration value="bit"/>
            <xsd:enumeration value="char"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
</xsd:schema>

```

APÉNDICE F: DESCRIPTORES DE FORMATOS PARA CENTRALES HUAWEI, ERICSSON Y ALCATEL

El siguiente apéndice muestra los descriptores de formatos generados por el sistema, para especificar la estructura de los archivos generados por distintos tipos de centrales telefónicas de tráfico.

Documento descriptor de formatos para una central Huawei:

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<descriptorRegistro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:noNamespaceSchemaLocation="lectorArchivo.xsd">  
  
  <metaDataRegistro>
```



```

<tipoRegistro>binario</tipoRegistro>
<codificacion>UTF-8</codificacion>
<idRegistro/>
<longitudMagnitudRegistro unidad="bit">0</longitudMagnitudRegistro>
<longitudIdCampo unidad="bit">0</longitudIdCampo>
<longitudMagnitudCampo unidad="bit">0</longitudMagnitudCampo>
<delimitadorCampo><![CDATA[]]></delimitadorCampo>
<delimitadorRegistro><![CDATA[]]></delimitadorRegistro>
<longitudDelimitador unidad="byte">0</longitudDelimitador>
<intervaloRegistroSeparador>0</intervaloRegistroSeparador>
<longitudRegistroSeparador unidad="byte">0</longitudRegistroSeparador>
<comentario><![CDATA[]]></comentario>
</metaDataRegistro>
<descriptorCampos id="1">
  <metaDataCampos>
    <descripcion>OVS0038</descripcion>
  </metaDataCampos>
  <campos>
    <campo id="1">
      <metaData tipo="leido">
        <descripcion>Serial Number</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato invertirBytes="true" tipo="int"/>
        <expresion tipo="simple">
          <simple/>
        </expresion>
      </metaData>
      <datos>&quot;&quot;</datos>
    </campo>
    <campo id="2">
      <metaData tipo="leido">
        <descripcion>Ticket Type</descripcion>

```

```

        <magnitud unidad="byte">1</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
</campos>
<bifurcacion>
    <switch>
        <option>
            <condition>campo[@id = 2]/datos=&quot;01&quot;</condition>
            <then>
                <next>opciones/descriptorCampos[@id=&quot;2&quot;]</next>
            </then>
        </option>
        <option>
            <condition>campo[@id = 2]/datos=&quot;F0&quot;</condition>
            <then>
                <next>opciones/descriptorCampos[@id=&quot;3&quot;]</next>
            </then>
        </option>
        <option>
            <condition>campo[@id = 2]/datos=&quot;F1&quot;</condition>
            <then>
                <next>opciones/descriptorCampos[@id=&quot;4&quot;]</next>
            </then>
        </option>
        <option>
            <condition>campo[@id = 2]/datos=&quot;F2&quot;</condition>
            <then>

```

```

        <next>opciones/descriptorCampos[@id=&quot;5&quot; ]</next>
    </then>
</option>
<option>
    <condition>campo[@id = 2]/datos=&quot;F3&quot;</condition>
    <then>
        <next>opciones/descriptorCampos[@id=&quot;6&quot; ]</next>
    </then>
</option>
</switch>
<opciones>
    <descriptorCampos id="2">
        <metaDataCampos>
            <descripcion>Detailed Ticket</descripcion>
        </metaDataCampos>
        <campos>
            <campo id="3">
                <metaData tipo="leido">
                    <descripcion>Checksum</descripcion>
                    <magnitud unidad="byte">1</magnitud>
                    <formato tipo="hex"/>
                    <expresion tipo="simple">
                        <simple/>
                    </expresion>
                </metaData>
                <datos>&quot;&quot;</datos>
            </campo>
            <campo id="4">
                <metaData tipo="leido">
                    <descripcion>Partial Record Indicator</descripcion>
                    <magnitud unidad="byte">0.5</magnitud>
                    <formato tipo="int"/>

```

```
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="5">
    <metaData tipo="leido">
        <descripcion>Validity</descripcion>
        <magnitud unidad="byte">0.125</magnitud>
        <formato tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="6">
    <metaData tipo="leido">
        <descripcion>Clock Changed Flag</descripcion>
        <magnitud unidad="byte">0.125</magnitud>
        <formato tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="7">
    <metaData tipo="leido">
        <descripcion>Free Flag</descripcion>
        <magnitud unidad="byte">0.125</magnitud>
```

```
<formato tipo="int"/>
  <expresion tipo="simple">
    <simple/>
  </expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="8">
  <metaData tipo="leido">
    <descripcion>Call Attempt Flag</descripcion>
    <magnitud unidad="byte">0.125</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="9">
  <metaData tipo="leido">
    <descripcion>Complaint Flag</descripcion>
    <magnitud unidad="byte">0.125</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="10">
  <metaData tipo="leido">
    <descripcion>Centralized Charging Flag</descripcion>
```

```
<magnitud unidad="byte">0.125</magnitud>
<formato tipo="int"/>
<expresion tipo="simple">
  <simple/>
</expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="11">
  <metaData tipo="leido">
    <descripcion>Credit Card Flag</descripcion>
    <magnitud unidad="byte">0.125</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="12">
  <metaData tipo="leido">
    <descripcion>Charging Method</descripcion>
    <magnitud unidad="byte">0.125</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="13">
  <metaData tipo="leido">
```

```
<descripcion>Payer</descripcion>
<magnitud unidad="byte">0.5</magnitud>
<formato tipo="int"/>
<expresion tipo="simple">
  <simple/>
</expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="14">
  <metaData tipo="leido">
    <descripcion>Conversation End Time</descripcion>
    <magnitud unidad="byte">6</magnitud>
    <formato mascara="YYMMDDHHmmSS" tipo="date"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="15">
  <metaData tipo="leido">
    <descripcion>Conversation Duration</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato invertirBytes="true" tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="16">
```

```
<metaData tipo="leido">
  <descripcion>Caller Seize Time</descripcion>
  <magnitud unidad="byte">4</magnitud>
  <formato invertirBytes="true" tipo="int"/>
  <expresion tipo="simple">
    <simple/>
  </expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="17">
  <metaData tipo="leido">
    <descripcion>Caller seize time</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato invertirBytes="true" tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="18">
  <metaData tipo="leido">
    <descripcion>Incomplete Call Watch</descripcion>
    <magnitud unidad="byte">0.25</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
```



```
<campo id="19">
  <metaData tipo="leido">
    <descripcion>Spare</descripcion>
    <magnitud unidad="byte">0.25</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="20">
  <metaData tipo="leido">
    <descripcion>Charging Adress Nature</descripcion>
    <magnitud unidad="byte">0.5</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="21">
  <metaData tipo="leido">
    <descripcion>Charging DnSet</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
```

```
</campo>
<campo id="22">
  <metaData tipo="leido">
    <descripcion>Charging Number</descripcion>
    <magnitud unidad="byte">10</magnitud>
    <formato rellenoDerecha="F" tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="23">
  <metaData tipo="leido">
    <descripcion>Caller DnSet</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="24">
  <metaData tipo="leido">
    <descripcion>Connecting DnSet</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
```

```
<datos>&quot;&quot;;</datos>
</campo>
<campo id="25">
  <metaData tipo="leido">
    <descripcion>Caller Address Nature</descripcion>
    <magnitud unidad="byte">0.5</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;;</datos>
</campo>
<campo id="26">
  <metaData tipo="leido">
    <descripcion>Connecting Address Nature</descripcion>
    <magnitud unidad="byte">0.5</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;;</datos>
</campo>
<campo id="27">
  <metaData tipo="leido">
    <descripcion>Caller Number</descripcion>
    <magnitud unidad="byte">10</magnitud>
    <formato rellenoDerecha="F" tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
```

```
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="28">
  <metaData tipo="leido">
    <descripcion>Connecting Number</descripcion>
    <magnitud unidad="byte">10</magnitud>
    <formato rellenoDerecha="F" tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="29">
  <metaData tipo="leido">
    <descripcion>Dial Number</descripcion>
    <magnitud unidad="byte">12</magnitud>
    <formato rellenoDerecha="F" tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="30">
  <metaData tipo="leido">
    <descripcion>CENTREX Group Number</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
```

```
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="31">
    <metaData tipo="leido">
        <descripcion>Caller CENTREX Short Number</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato rellenoDerecha="F" tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="32">
    <metaData tipo="leido">
        <descripcion>Called CENTREX Short Number</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato rellenoDerecha="F" tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="33">
    <metaData tipo="leido">
        <descripcion>Caller Module Number</descripcion>
        <magnitud unidad="byte">1</magnitud>
        <formato tipo="int"/>
        <expresion tipo="simple">
```

```
        <simple/>
    </expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="34">
    <metaData tipo="leido">
        <descripcion>Called Module Number</descripcion>
        <magnitud unidad="byte">1</magnitud>
        <formato tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="35">
    <metaData tipo="leido">
        <descripcion>Incoming Trunk Group Number</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato invertirBytes="true" tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="36">
    <metaData tipo="leido">
        <descripcion>Outgoing Trunk Group Number</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato invertirBytes="true" tipo="int"/>
```

```
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="37">
    <metaData tipo="leido">
        <descripcion>Caller Device Type</descripcion>
        <magnitud unidad="byte">1</magnitud>
        <formato tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="38">
    <metaData tipo="leido">
        <descripcion>Called Device Type</descripcion>
        <magnitud unidad="byte">1</magnitud>
        <formato tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="39">
    <metaData tipo="leido">
        <descripcion>Caller Category</descripcion>
        <magnitud unidad="byte">1</magnitud>
```

```

        <formato tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="40">
    <metaData tipo="leido">
        <descripcion>Called Category</descripcion>
        <magnitud unidad="byte">1</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="41">
    <metaData tipo="leido">
        <descripcion>Call Type</descripcion>
        <magnitud unidad="byte">0.5</magnitud>
        <formato tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="42">
    <metaData tipo="leido">
        <descripcion>Service Type</descripcion>

```



```

        <magnitud unidad="byte">0.5</magnitud>
        <formato tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="43">
    <metaData tipo="leido">
        <descripcion>Supplementary Service Type</descripcion>
        <magnitud unidad="byte">1</magnitud>
        <formato tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="44">
    <metaData tipo="leido">
        <descripcion>Charging Case</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="45">
    <metaData tipo="leido">

```

```
<descripcion>Tariff</descripcion>
<magnitud unidad="byte">2</magnitud>
<formato tipo="int"/>
<expresion tipo="simple">
  <simple/>
</expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="46">
  <metaData tipo="leido">
    <descripcion>Charging Pulse</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato invertirBytes="true" tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="47">
  <metaData tipo="leido">
    <descripcion>Fee</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato invertirBytes="true" tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="48">
```

```
<metaData tipo="leido">
  <descripcion>Bearer Service</descripcion>
  <magnitud unidad="byte">1</magnitud>
  <formato tipo="int"/>
  <expresion tipo="simple">
    <simple/>
  </expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="49">
  <metaData tipo="leido">
    <descripcion>Teleservice</descripcion>
    <magnitud unidad="byte">0.5</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="50">
  <metaData tipo="leido">
    <descripcion>Conversation End Reason</descripcion>
    <magnitud unidad="byte">0.5</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
```

```
<campo id="51">
  <metaData tipo="leido">
    <descripcion>Reason Number</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="52">
  <metaData tipo="leido">
    <descripcion>Release Number</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="53">
  <metaData tipo="leido">
    <descripcion>UUS1 Count</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
```

```
</campo>
<campo id="54">
  <metaData tipo="leido">
    <descripcion>UUS2 Count</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="55">
  <metaData tipo="leido">
    <descripcion>UUS3 Count</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="56">
  <metaData tipo="leido">
    <descripcion>Reserved</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
```

```

        <datos>&quot;&quot;;</datos>
    </campo>
</campos>
</descriptorCampos>
<descriptorCampos id="3">
    <metaDataCampos>
        <descripcion>Meter Table Ticket</descripcion>
    </metaDataCampos>
    <campos>
        <campo id="57">
            <metaData tipo="leido">
                <descripcion>Checksum2</descripcion>
                <magnitud unidad="byte">1</magnitud>
                <formato tipo="hex"/>
                <expresion tipo="simple">
                    <simple/>
                </expresion>
            </metaData>
            <datos>&quot;&quot;;</datos>
        </campo>
        <campo id="58">
            <metaData tipo="leido">
                <descripcion>Reserved</descripcion>
                <magnitud unidad="byte">0.5</magnitud>
                <formato tipo="hex"/>
                <expresion tipo="simple">
                    <simple/>
                </expresion>
            </metaData>
            <datos>&quot;&quot;;</datos>
        </campo>
        <campo id="59">

```

```
<metaData tipo="leido">
  <descripcion>Validity</descripcion>
  <magnitud unidad="byte">0.125</magnitud>
  <formato tipo="hex"/>
  <expresion tipo="simple">
    <simple/>
  </expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="60">
  <metaData tipo="leido">
    <descripcion>Charging Object</descripcion>
    <magnitud unidad="byte">0.25</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="61">
  <metaData tipo="leido">
    <descripcion>Reserved</descripcion>
    <magnitud unidad="byte">0.125</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
```

```
<campo id="62">
  <metaData tipo="leido">
    <descripcion>Meter Table Number</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="63">
  <metaData tipo="leido">
    <descripcion>Meter Table Generate Time</descripcion>
    <magnitud unidad="byte">6</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="64">
  <metaData tipo="leido">
    <descripcion>DnSet</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
```



```
</campo>
<campo id="65">
  <metaData tipo="leido">
    <descripcion>Caller Address Nature</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="66">
  <metaData tipo="leido">
    <descripcion>Telephone Number</descripcion>
    <magnitud unidad="byte">10</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="67">
  <metaData tipo="leido">
    <descripcion>Trunk Group Number</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
```

```
<datos>&quot;&quot;;</datos>
</campo>
<campo id="68">
  <metaData tipo="leido">
    <descripcion>Module Number</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;;</datos>
</campo>
<campo id="69">
  <metaData tipo="leido">
    <descripcion>Device Type</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;;</datos>
</campo>
<campo id="70">
  <metaData tipo="leido">
    <descripcion>PSN</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
```

```

    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="71">
    <metaData tipo="leido">
        <descripcion>Meter Table 1 Value</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="72">
    <metaData tipo="leido">
        <descripcion>Meter Table 1 Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="73">
    <metaData tipo="leido">
        <descripcion>Meter Table 2 Value</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>

```

```
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="74">
    <metaData tipo="leido">
        <descripcion>Meter Table 2 Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="75">
    <metaData tipo="leido">
        <descripcion>Meter Table 3 Value</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="76">
    <metaData tipo="leido">
        <descripcion>Meter Table 3 Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
```

```
        <simple/>
    </expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="77">
    <metaData tipo="leido">
        <descripcion>Meter Table 4 Value</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="78">
    <metaData tipo="leido">
        <descripcion>Meter Table 4 Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="79">
    <metaData tipo="leido">
        <descripcion>Meter Table 5 Value</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
```

```
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="80">
    <metaData tipo="leido">
        <descripcion>Meter Table 5 Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="81">
    <metaData tipo="leido">
        <descripcion>Meter Table 6 Value</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="82">
    <metaData tipo="leido">
        <descripcion>Meter Table 6 Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
```

```

        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="83">
    <metaData tipo="leido">
        <descripcion>Meter Table 7 Value</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="84">
    <metaData tipo="leido">
        <descripcion>Meter Table 7 Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="85">
    <metaData tipo="leido">
        <descripcion>Meter Table 8 Value</descripcion>

```

```

        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="86">
    <metaData tipo="leido">
        <descripcion>Meter Table 8 Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="87">
    <metaData tipo="leido">
        <descripcion>Meter Table 9 Value</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="88">
    <metaData tipo="leido">

```



```
<descripcion>Meter Table 9 Call Times</descripcion>
<magnitud unidad="byte">2</magnitud>
<formato tipo="hex"/>
<expresion tipo="simple">
  <simple/>
</expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="89">
  <metaData tipo="leido">
    <descripcion>Meter Table 10 Value</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="90">
  <metaData tipo="leido">
    <descripcion>Meter Table 10 Call Times</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="91">
```

```

        <metaData tipo="leido">
            <descripcion/>
            <magnitud unidad="byte">26</magnitud>
            <formato tipo="hex"/>
            <expresion tipo="simple">
                <simple/>
            </expresion>
        </metaData>
        <datos>&quot;&quot;</datos>
    </campo>
</campos>
</descriptorCampos>
<descriptorCampos id="4">
    <metaDataCampos>
        <descripcion>Meter Table Statistics</descripcion>
    </metaDataCampos>
    <campos>
        <campo id="92">
            <metaData tipo="leido">
                <descripcion>Checksum</descripcion>
                <magnitud unidad="byte">1</magnitud>
                <formato tipo="hex"/>
                <expresion tipo="simple">
                    <simple/>
                </expresion>
            </metaData>
            <datos>&quot;&quot;</datos>
        </campo>
        <campo id="93">
            <metaData tipo="leido">
                <descripcion>Reserved</descripcion>
                <magnitud unidad="byte">0.5</magnitud>
            </metaData>
            <datos>&quot;&quot;</datos>
        </campo>
    </campos>
</descriptorCampos>

```

```
<formato tipo="hex"/>
  <expresion tipo="simple">
    <simple/>
  </expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="94">
  <metaData tipo="leido">
    <descripcion>Validity</descripcion>
    <magnitud unidad="byte">0.125</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="95">
  <metaData tipo="leido">
    <descripcion>Reserved</descripcion>
    <magnitud unidad="byte">0.375</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="96">
  <metaData tipo="leido">
    <descripcion>Meter Table Number</descripcion>
```

```

        <magnitud unidad="byte">1</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="97">
    <metaData tipo="leido">
        <descripcion>Meter Table Generate Time</descripcion>
        <magnitud unidad="byte">6</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="98">
    <metaData tipo="leido">
        <descripcion>Module Number</descripcion>
        <magnitud unidad="byte">1</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="99">
    <metaData tipo="leido">

```

```
<descripcion>Call Category</descripcion>
<magnitud unidad="byte">1</magnitud>
<formato tipo="hex"/>
<expresion tipo="simple">
  <simple/>
</expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="100">
  <metaData tipo="leido">
    <descripcion>Meter Table 1 Value</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="101">
  <metaData tipo="leido">
    <descripcion>Meter Table 1 Call Times</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="102">
```

```
<metaData tipo="leido">
  <descripcion>Meter Table 2 Value</descripcion>
  <magnitud unidad="byte">4</magnitud>
  <formato tipo="hex"/>
  <expresion tipo="simple">
    <simple/>
  </expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="103">
  <metaData tipo="leido">
    <descripcion>Meter Table 2 Call Times</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="104">
  <metaData tipo="leido">
    <descripcion>Meter Table 3 Value</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
```

```
<campo id="105">
  <metaData tipo="leido">
    <descripcion>Meter Table 3 Call Times</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="106">
  <metaData tipo="leido">
    <descripcion>Meter Table 4 Value</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="107">
  <metaData tipo="leido">
    <descripcion>Meter Table 4 Call Times</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
```

```
</campo>
<campo id="108">
  <metaData tipo="leido">
    <descripcion>Meter Table 5 Value</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="109">
  <metaData tipo="leido">
    <descripcion>Meter Table 5 Call Times</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="110">
  <metaData tipo="leido">
    <descripcion>Meter Table 6 Value</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
```



```
<datos>&quot;&quot;;</datos>
</campo>
<campo id="111">
  <metaData tipo="leido">
    <descripcion>Meter Table 6 Call Times</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;;</datos>
</campo>
<campo id="112">
  <metaData tipo="leido">
    <descripcion>Meter Table 7 Value</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;;</datos>
</campo>
<campo id="113">
  <metaData tipo="leido">
    <descripcion>Meter Table 7 Call Times</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
```

```
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="114">
  <metaData tipo="leido">
    <descripcion>Meter Table 8 Value</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="115">
  <metaData tipo="leido">
    <descripcion>Meter Table 8 Call Times</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="116">
  <metaData tipo="leido">
    <descripcion>Meter Table 9 Value</descripcion>
    <magnitud unidad="byte">4</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
```

```
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="117">
    <metaData tipo="leido">
        <descripcion>Meter Table 9 Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="118">
    <metaData tipo="leido">
        <descripcion>Meter Table 10 Value</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="119">
    <metaData tipo="leido">
        <descripcion>Meter Table 10 Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
```

```

        <simple/>
    </expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="120">
    <metaData tipo="leido">
        <descripcion>Reserved</descripcion>
        <magnitud unidad="byte">42</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
</campos>
</descriptorCampos>
<descriptorCampos id="5">
    <metaDataCampos>
        <descripcion>Trunk Duration Statistics</descripcion>
    </metaDataCampos>
    <campos>
        <campo id="121">
            <metaData tipo="leido">
                <descripcion>Checksum</descripcion>
                <magnitud unidad="byte">1</magnitud>
                <formato tipo="hex"/>
                <expresion tipo="simple">
                    <simple/>
                </expresion>
            </metaData>

```

```
<datos>&quot;&quot;;</datos>
</campo>
<campo id="122">
  <metaData tipo="leido">
    <descripcion>Reserved</descripcion>
    <magnitud unidad="byte">0.5</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;;</datos>
</campo>
<campo id="123">
  <metaData tipo="leido">
    <descripcion>Validity</descripcion>
    <magnitud unidad="byte">0.125</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;;</datos>
</campo>
<campo id="124">
  <metaData tipo="leido">
    <descripcion>Reserved</descripcion>
    <magnitud unidad="byte">1.375</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
```

```
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="125">
  <metaData tipo="leido">
    <descripcion>Ticket Generated Time</descripcion>
    <magnitud unidad="byte">6</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="126">
  <metaData tipo="leido">
    <descripcion>Module Number</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
<campo id="127">
  <metaData tipo="leido">
    <descripcion>Reserved</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos>&quot;&quot;</datos>
</campo>
```

```
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="128">
    <metaData tipo="leido">
        <descripcion>Trunk Group Number</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="129">
    <metaData tipo="leido">
        <descripcion>Incoming Office Call Duration</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="130">
    <metaData tipo="leido">
        <descripcion>Incoming Office Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
```

```
        <simple/>
    </expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="131">
    <metaData tipo="leido">
        <descripcion>Tandem Incoming Call Duration</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="132">
    <metaData tipo="leido">
        <descripcion>Tandem Incoming Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="133">
    <metaData tipo="leido">
        <descripcion>Outgoing Trunk Call Duration</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
```



```

        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="134">
    <metaData tipo="leido">
        <descripcion>Outgoing Trunk Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="135">
    <metaData tipo="leido">
        <descripcion>Reserved</descripcion>
        <magnitud unidad="byte">82</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
</campos>
</descriptorCampos>
<descriptorCampos id="6">
    <metaDataCampos>

```

```
<descripcion>Free Call Statistics</descripcion>
</metaDataCampos>
<campos>
  <campo id="136">
    <metaData tipo="leido">
      <descripcion>Checksum</descripcion>
      <magnitud unidad="byte">1</magnitud>
      <formato tipo="hex"/>
      <expresion tipo="simple">
        <simple/>
      </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
  </campo>
  <campo id="137">
    <metaData tipo="leido">
      <descripcion>Reserved</descripcion>
      <magnitud unidad="byte">0.5</magnitud>
      <formato tipo="hex"/>
      <expresion tipo="simple">
        <simple/>
      </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
  </campo>
  <campo id="138">
    <metaData tipo="leido">
      <descripcion>Validity</descripcion>
      <magnitud unidad="byte">0.125</magnitud>
      <formato tipo="hex"/>
      <expresion tipo="simple">
        <simple/>
      </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
  </campo>
  <campo id="139">
    <metaData tipo="leido">
      <descripcion>Free Call Statistics</descripcion>
      <magnitud unidad="byte">1</magnitud>
      <formato tipo="hex"/>
      <expresion tipo="simple">
        <simple/>
      </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
  </campo>
</campos>
```

```
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="139">
    <metaData tipo="leido">
        <descripcion>Reserved</descripcion>
        <magnitud unidad="byte">1.375</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="140">
    <metaData tipo="leido">
        <descripcion>Ticket Generated Time</descripcion>
        <magnitud unidad="byte">6</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="141">
    <metaData tipo="leido">
        <descripcion>Module Number</descripcion>
        <magnitud unidad="byte">1</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
```

```
        <simple/>
    </expresion>
</metaData>
<datos>&quot;&quot;</datos>
</campo>
<campo id="142">
    <metaData tipo="leido">
        <descripcion>Reserved</descripcion>
        <magnitud unidad="byte">1</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="143">
    <metaData tipo="leido">
        <descripcion>Free Conversation Duration</descripcion>
        <magnitud unidad="byte">4</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="144">
    <metaData tipo="leido">
        <descripcion>Free Call Times</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="hex"/>
```

```
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
<campo id="145">
    <metaData tipo="leido">
        <descripcion>Reserved</descripcion>
        <magnitud unidad="byte">96</magnitud>
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos>&quot;&quot;</datos>
</campo>
</campos>
</descriptorCampos>
</opciones>
</bifurcacion>
</descriptorCampos>
</descriptorRegistro>
```

Documento descriptor de formatos para una central Ericsson:

```

<?xml version="1.0" encoding="UTF-8"?>
<descriptorRegistro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:noNamespaceSchemaLocation="lectorArchivo.xsd">
  <metaDataRegistro>
    <tipoRegistro>binario</tipoRegistro>
    <codificacion>UTF-8</codificacion>
    <idRegistro>
    </idRegistro>
    <longitudMagnitudRegistro unidad="bit">0</longitudMagnitudRegistro>
    <longitudIdCampo unidad="bit">0</longitudIdCampo>
    <longitudMagnitudCampo unidad="bit">0</longitudMagnitudCampo>
    <delimitadorCampo><![CDATA[]]></delimitadorCampo>
    <delimitadorRegistro><![CDATA[]]></delimitadorRegistro>
    <longitudDelimitador unidad="byte">0</longitudDelimitador>
    <intervaloRegistroSeparador>19</intervaloRegistroSeparador>
    <longitudRegistroSeparador unidad="byte">72</longitudRegistroSeparador>
    <comentario></comentario>
  </metaDataRegistro>
  <descriptorCampos id="1">
    <metaDataCampos>
      <descripcion>Grupo Inicial</descripcion>
    </metaDataCampos>
    <campos>
      <campo id="1">
        <metaData tipo="leido">
          <descripcion>Tipo LLamada</descripcion>
          <magnitud unidad="byte">2</magnitud>
          <formato tipo="string" />
          <expresion tipo="simple">

```

```
        <simple/>
    </expresion>
</metaData>
<datos/>
</campo>
<campo id="2">
    <metaData tipo="leido">
        <descripcion>Area Origen</descripcion>
        <magnitud unidad="byte">3</magnitud>
        <formato tipo="string" />
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="3">
    <metaData tipo="leido">
        <descripcion>Numero Origen</descripcion>
        <magnitud unidad="byte">7</magnitud>
        <formato tipo="string" />
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="21">
    <metaData tipo="leido">
        <descripcion>espacio</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="string" />
```

```
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
</datos/>
</campo>
<campo id="4">
    <metaData tipo="leido">
        <descripcion>Area Destino</descripcion>
        <magnitud unidad="byte">2</magnitud>
        <formato tipo="string" />
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="5">
    <metaData tipo="leido">
        <descripcion>Numero Destino</descripcion>
        <magnitud unidad="byte">7</magnitud>
        <formato tipo="string" />
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="14">
    <metaData tipo="leido">
        <descripcion>espacio</descripcion>
        <magnitud unidad="byte">9</magnitud>
```



```
        <formato tipo="string" />
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="20">
    <metaData tipo="leido">
        <descripcion>Indefinido</descripcion>
        <magnitud unidad="byte">5</magnitud>
        <formato tipo="string" />
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="6">
    <metaData tipo="leido">
        <descripcion>Fecha</descripcion>
        <magnitud unidad="byte">6</magnitud>
        <formato tipo="string" />
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="7">
    <metaData tipo="leido">
        <descripcion>Hora</descripcion>
```

```
        <magnitud unidad="byte">6</magnitud>
        <formato tipo="string" />
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="8">
    <metaData tipo="leido">
        <descripcion>Duracion</descripcion>
        <magnitud unidad="byte">6</magnitud>
        <formato tipo="string" />
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="15">
    <metaData tipo="leido">
        <descripcion>espacio</descripcion>
        <magnitud unidad="byte">5</magnitud>
        <formato tipo="string" />
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="9">
    <metaData tipo="leido">
```

```
<descripcion>ID Central</descripcion>
<magnitud unidad="byte">15</magnitud>
<formato tipo="string" />
<expresion tipo="simple">
  <simple/>
</expresion>
</metaData>
<datos/>
</campo>
<campo id="16">
  <metaData tipo="leido">
    <descripcion>indefinido</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato tipo="string" />
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos/>
</campo>
<campo id="10">
  <metaData tipo="leido">
    <descripcion>Ruta Destino</descripcion>
    <magnitud unidad="byte">6</magnitud>
    <formato tipo="string" />
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos/>
</campo>
<campo id="13">
```

```
<metaData tipo="leido">
  <descripcion>espacio</descripcion>
  <magnitud unidad="byte">1</magnitud>
  <formato tipo="string" />
  <expresion tipo="simple">
    <simple/>
  </expresion>
</metaData>
<datos/>
</campo>
<campo id="11">
  <metaData tipo="leido">
    <descripcion>Ruta Origen</descripcion>
    <magnitud unidad="byte">7</magnitud>
    <formato tipo="string" />
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos/>
</campo>
<campo id="17">
  <metaData tipo="leido">
    <descripcion>entreCampos</descripcion>
    <magnitud unidad="byte">13</magnitud>
    <formato tipo="string" />
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos/>
</campo>
```

```
</campos>  
</descriptorCampos>  
</descriptorRegistro>
```

Documento descriptor de formatos para una central Alcatel:

```

<?xml version="1.0" encoding="UTF-8"?>
<descriptorRegistro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:noNamespaceSchemaLocation="lectorArchivo.xsd">
  <metaDataRegistro>
    <tipoRegistro>binario</tipoRegistro>
    <codificacion>UTF-8</codificacion>
    <idRegistro>
    </idRegistro>
    <longitudMagnitudRegistro unidad="bit">0</longitudMagnitudRegistro>
    <longitudIdCampo unidad="bit">0</longitudIdCampo>
    <longitudMagnitudCampo unidad="bit">0</longitudMagnitudCampo>
    <delimitadorCampo><![CDATA[]]></delimitadorCampo>
    <delimitadorRegistro><![CDATA[]]></delimitadorRegistro>
    <longitudDelimitador unidad="byte">0</longitudDelimitador>
    <intervaloRegistroSeparador>0</intervaloRegistroSeparador>
    <longitudRegistroSeparador unidad="byte">0</longitudRegistroSeparador>
    <comentario></comentario>
  </metaDataRegistro>
  <descriptorCampos id="1">
    <metaDataCampos>
      <descripcion>Grupo Inicial</descripcion>
    </metaDataCampos>
    <campos>
      <campo id="1">
        <metaData tipo="leido">
          <descripcion>Faturacion Detallada</descripcion>
          <magnitud unidad="byte">1</magnitud>
          <formato tipo="hex"/>
          <expresion tipo="simple">

```

```
        <simple/>
      </expresion>
    </metaData>
  <datos/>
</campo>
<campo id="2">
  <metaData tipo="leido">
    <descripcion>Categoria Abonado</descripcion>
    <magnitud unidad="byte">1</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos/>
</campo>
<campo id="3">
  <metaData tipo="leido">
    <descripcion>Fecha</descripcion>
    <magnitud unidad="byte">2</magnitud>
    <formato invertirBytes="false" tipo="int"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos/>
</campo>
<campo id="4">
  <metaData tipo="leido">
    <descripcion>Hora</descripcion>
    <magnitud unidad="byte">3</magnitud>
    <formato tipo="hex"/>
```

```
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
</datos/>
</campo>
<campo id="5">
    <metaData tipo="leido">
        <descripcion>Area Origen</descripcion>
        <magnitud unidad="byte">1</magnitud>
        <formato invertirBytes="false" tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="6">
    <metaData tipo="leido">
        <descripcion>Numero Origen</descripcion>
        <magnitud unidad="byte">3</magnitud>
        <formato invertirBytes="false" tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="7">
    <metaData tipo="leido">
        <descripcion>reservado 1</descripcion>
        <magnitud unidad="byte">1</magnitud>
```



```
        <formato tipo="hex"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="8">
    <metaData tipo="leido">
        <descripcion>Area Destino</descripcion>
        <magnitud unidad="byte">1</magnitud>
        <formato invertirBytes="false" tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="9">
    <metaData tipo="leido">
        <descripcion>Numero Destino</descripcion>
        <magnitud unidad="byte">3</magnitud>
        <formato invertirBytes="false" tipo="int"/>
        <expresion tipo="simple">
            <simple/>
        </expresion>
    </metaData>
    <datos/>
</campo>
<campo id="10">
    <metaData tipo="leido">
        <descripcion>reservado 2</descripcion>
```

```
<magnitud unidad="byte">5</magnitud>
<formato tipo="hex"/>
<expresion tipo="simple">
  <simple/>
</expresion>
</metaData>
<datos/>
</campo>
<campo id="11">
  <metaData tipo="leido">
    <descripcion>Duracion</descripcion>
    <magnitud unidad="byte">3</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos/>
</campo>
<campo id="12">
  <metaData tipo="leido">
    <descripcion>reservado 3</descripcion>
    <magnitud unidad="byte">8</magnitud>
    <formato tipo="hex"/>
    <expresion tipo="simple">
      <simple/>
    </expresion>
  </metaData>
  <datos/>
</campo>
</campos>
</descriptorCampos>
```

</descriptorRegistro>

APÉNDICE G: DEFINICIÓN DE UN XML SCHEMA DEFINITION

A continuación se explica brevemente como definir diferentes tipos de elementos usando XML Schema Definition (XSD):

ELEMENTOS Y ATRIBUTOS

Un elemento se define con la siguiente sintaxis:

```
<xsd:element name="nombre" type="tipo">
```

Esta línea define un elemento del nombre especificado y el tipo definido. Los tipos definen la estructura y contenido del elemento. Igual que en los lenguajes de programación, el tipo puede ser simple (como cadena, o numérico) o algo complejo (como una estructura de elementos).

Un atributo se declara de manera similar:

```
<xsd:attribute name="nombre" type="tipo">
```

Los atributos de la definición describen las características de los elementos y atributos, a continuación se describen las propiedades que son comunes para elementos y atributos:

Nombre	Descripción
name	Define el nombre del nodo.
default	Fija el valor por omisión.
fixed	Fija que el nodo debe tener sólo este valor (mutuamente excluyente de del atributo "default")
form	De valor "qualified" o "unqualified", especifica si el nodo debe pertenecer a un espacio de nombre.

Tabla G.1: Propiedades comunes para Elementos y Atributos de un Esquema XML

Sólo hay un campo para los atributos que no se aplica a los elementos:

Nombre	Descripción
use	Define la cardinalidad del atributo. Sus posibles valores son "required", "optional", "prohibited"

Tabla G.2: Propiedades específicas de los Atributos en un Esquema XML

A continuación se enlistan los campos únicos para los elementos:

Nombre	Descripción
abstract	Define un elemento abstracto que no puede aparecer en el documento, pero del cual otros elementos del esquema pueden heredar.

Nombre	Descripción
minOccurs	Define la mínima cardinalidad del elemento cuando es definido como hijo de otro nodo.
maxOccurs	Define la máxima cardinalidad del elemento cuando es definido como hijo de otro nodo.
nillable	Define que el elemento debe estar vacío.
ref	Usado en vez de "name" para hacer referencia a otro elemento ya definido y reusar sus características.
substitutionGroup	Permite definir al elemento como parte de un grupo de sustitución, de tal manera que el elemento actual sea permisible en lugar del elemento al que se referencia con este parámetro.

Tabla G.3: Propiedades exclusivas de los Elementos en un Esquema XML

TIPOS DE DATOS

Cada elemento definido en un XSD tiene su tipo de dato, el cual define el tipo de contenido que puede tener. Los tipos se dividen en "simple" y "compuestos".

TIPOS DE DATOS SIMPLES

Igual que en los lenguajes de programación, los tipos de datos simples son cosas como números, booleanos, fechas, etc. En XML Esquema, también se pueden definir tipos simples que extienden a los otros datos simples predefinidos por la especificación XML Esquema.

LISTA DE DATOS SIMPLES PREDEFINIDOS

- string
- normalizedString
- token
- unsignedByte
- base64Binary
- hexBinary

- integer
- positiveInteger
- negativeInteger
- nonNegativeInteger
- nonPositiveInteger
- int
- unsignedInt
- long
- unsignedLong
- short
- unsignedShort
- decimal
- float
- double
- boolean
- time
- dateTime
- duration
- date
- gMonth
- gYear
- gYearMonth
- gDay
- gMonthDay
- Name
- QName
- NCName
- anyURI
- language
- ID
- IDREF
- IDREFS
- ENTITY
- ENTITIES
- NOTATION
- NMTOKEN
- NMTOKENS

Los datos simples contiene atributos, los cuales describen características de los datos simples, y modificándolos es como se obtuvieron varios datos simples a partir de otros (la gran mayoría de los datos simples de tipo numérico extienden a “decimal”). La creación de datos simples se basa en la idea de extender a algún otro dato simple y modificar estas restricciones que se pueden aplicar sobre el dato. No toda restricción es aplicable a todo tipo de dato simple, las restricciones disponibles dependen del tipo de dato simple a extender⁽¹⁾. A continuación se presenta la lista de restricciones aplicables a los datos simples:

Nombre	Descripción
length	Define la longitud del campo cuando esta es fija.
minLength	Define la longitud mínima.
maxLength	Define la longitud máxima.
pattern	Permite usar expresiones regulares ⁽²⁾ para definir el formato esperado del valor.
enumeration	Permite enumerar los valores posibles del dato.
whitespace	Especifica como deben ser tratados los espacios en el elemento. Sus valores posibles son: “preserve”, “replace” y “collapse”.
maxInclusive	Define el máximo valor inclusive permitido.
maxExclusive	Define el máximo valor exclusive permitido.
minInclusive	Define el mínimo valor inclusive permitido.

¹ Ver el apéndice B del “Esquema XML - parte cero” para ver las tablas que tabulan que restricciones están disponibles para cada tipo.

² Expresiones Regulares: Permite por medio de una cadena explicar el formato deseado en otras cadenas. La explicación de las construcciones permitidas y su sintaxis en el ámbito de Esquemas XML se puede encontrar en <<http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#dt-regex>>

Nombre	Descripción
minExclusive	Define el mínimo valor exclusive permitido.
totalDigits	Define el total de dígitos a usar para representar números.
fracionDigits	Define el número de dígitos fraccionarios permitidos.

Tabla G.4: Propiedades de los datos simples en un Esquema XML.

A continuación se muestran algunos ejemplos de como usar estos atributos para definir nuevos tipos de datos de estructura simple:

Definir un tipo de entero cuyo rango válido es 0-0999:

```
<xsd:simpleType name="someInt">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="9999"/>
  </xsd:restriction>
</xsd:simpleType>
```

El siguiente tipo simple que extiende a las cadenas, define un tipo de código cuya regla de construcción es que debe estar constituido de tres dígitos seguido de un guión y dos caracteres en mayúscula (en el rango A-Z):

```
<xsd:simpleType name="SKU">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-[A-Z]{2}"/>
  </xsd:restriction>
</xsd:simpleType>
```

El siguiente ejemplo demuestra el uso de la restricción de enumeración (ya que cada elemento de la enumeración debe aparecer dentro de su propia declaración de restricción):

```
<xsd:simpleType name="ciudad">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Guayaquil"/>
    <xsd:enumeration value="Quito"/>
    <xsd:enumeration value="Cuenca"/>
    <!-- ingresar el resto de ciudades de la misma
manera -->
  </xsd:restriction>
</xsd:simpleType>
```

Los tipos atómicos NMTOKENS, IDREFS y ENTITIES son diseñados específicamente para la creación de listas, aunque se pueden crear listas de cualquier tipo simple; los elementos de la lista se separan por espacios. A continuación se muestra un ejemplo de una lista de ciudades:

```
<xsd:simpleType name="ciudades">
  <xsd:restriction base="ciudad">
    <xsd:minLength value="3"/>
  </xsd:restriction>
</xsd:simpleType>
```

TIPOS COMPUESTOS

Los tipos compuestos son definidos para describir la estructura de un elemento XML.

Un elemento puede ser de varios tipos

- Elemento con contenido de texto
- Elemento con otros elementos por dentro
- Elemento vacío
- Elemento con contenido mixto

Si el elemento extiende a uno de los tipos simples básicos y sólo modifica a las restricciones del tipo, es fácil definirlo:

```
<xsd:element name="miNumero" type="integer"
  minInclusive="0" maxInclusive="999" />
```

Cuando el elemento necesita agregar otros datos, como atributos, es necesario definirlo como un dato complejo que extiende a un dato simple:

```
<xsd:element name="miNumero">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="integer">
        <xsd:restriction>
          <xsd:minInclusive value="0" />
          <xsd:maxInclusive value="999" />
        </xsd:restriction>
        <xsd:attribute name="unidad"
          type="xsd:string" fixed="secs" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

La definición de elementos compuestos de otros elementos es la más común. A continuación se presenta un ejemplo en el cual se definen los contenidos de un elemento (los contenidos pueden aparecer en cualquier orden):

```
<xsd:element name="datosPersonales">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:element ref="nombre"/>
      <xsd:element ref="direccion"/>
      <xsd:element ref="telefono" minOccurs="0"/>
      <xsd:element ref="fax" minOccurs="0"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

Si en la declaración “xsd:complexType” se incluye el atributo “mixed='true'”, se está permitiendo que dentro de dicho elemento coexistan valores de texto y otros elementos a la vez. Tomando el ejemplo de una carta⁽³⁾ en la cual van los datos de una compra:

```
<letterBody>
<salutation>Dear Mr.<name>Robert
  Smith</name>.</salutation>
Your order of <quantity>1</quantity>
  <productName>Baby
Monitor</productName> shipped from our warehouse on
<shipDate>1999-05-21</shipDate>. ....
</letterBody>
```

³Tomado de la sección 2.5.2 de la Especificación de Esquemas XML, Parte cero
<<http://www.w3.org/TR/xmlschema-0/#mixedContent>>

El contenido de la carta obviamente contiene elementos mezclados con texto, su contenido se podría describir de la siguiente manera:

```
<xsd:element name="letterBody">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="salutation">
        <xsd:complexType mixed="true">
          <xsd:element name="name"
type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="quantity"
type="xsd:positiveInteger"/>
      <xsd:element name="productName"
type="xsd:string"/>
      <xsd:element name="shipDate"
type="xsd:date" minOccurs="0"/>
      <!-- etc. -->
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

GRUPOS

Los grupos definen conjuntos de elementos. Los grupos pueden ser globales o locales, los grupos locales son los que se han estado mostrando en los ejemplos anteriores, un grupo global se declara independientemente (usando la etiqueta “xsd:group”) y se lo puede utilizar en otras partes del documento por medio de referencias.

Un grupo secuencial es aquel en que el orden de los elementos está predefinido, se usa la etiqueta “xsd:sequence” para especificar este tipo de grupo:

```

<xsd:group name="dataModule">
  <xsd:sequence>
    <xsd:element ref="version"/>
    <xsd:element ref="autor"/>
    <xsd:element ref="licencia"/>
    <xsd:element ref="contenidos"/>
  </xsd:sequence>
</xsd:group>

```

También se puede especificar un grupo opcional, de tal manera que sólo uno de sus elementos integrantes necesita aparecer:

```

<xsd:group name="tarjetaDeCredito">
  <xsd:choice>
    <xsd:element ref="masterCard"/>
    <xsd:element ref="visa"/>
    <xsd:element ref="diners"/>
  </xsd:choice>
</xsd:group>

```

ANOTACIONES

Las anotaciones permiten ingresar información no usada para validar los documentos XML en sí, pero pueden ser útil como medio de información para los humanos que lean el documento, así como puede proveer de información que las aplicaciones puedan usar.

Las anotaciones pueden ser ingresadas al principio de prácticamente cualquier entidad (cuando se refiere a esta), y en cualquier parte dentro del ámbito global del esquema. Las anotaciones tienen dos etiquetas:

Nombre	Descripción
documentation	Especifica información para los usuarios lectores del documento.
Appinfo	Permiten el paso de cualquier tipo de información estructura de manera XML a ser pasada a la aplicación.

Tabla G.5: Tipos de notación en los Esquemas XML

REPRESENTACIÓN GRÁFICA DE ELEMENTOS XSD

En esta sección se explica la forma en que se han representado gráficamente los documentos XSD que han sido descritos en este proyecto de tesis. Para generar estos gráficos, se ha empleado la herramienta Oracle JDeveloper 10G (versión 9.0.5.2). El editor XSD de este producto utiliza gráficos en dos dimensiones que representan tanto los elementos del documento como las relaciones existentes entre ellos.

A continuación se describen cada uno de estos elementos:



Figura G.1: Tipo Simple de Dato

El gráfico representa un elemento simple, los mismos que son representados como un rectángulo color celeste. En su interior va el nombre del tipo seguido por las restricciones establecidas para el mismo.

A continuación se muestra las representaciones para los distintos tipos de relaciones entre un dato compuesto y sus integrantes. La relación puede ser: electiva (el elemento compuesto debe contener solo uno de los elementos descritos), requerida de todos (todos los elementos especificados deben aparecer sin importar el orden), o secuencial (los elementos deben aparecer en el orden especificado):

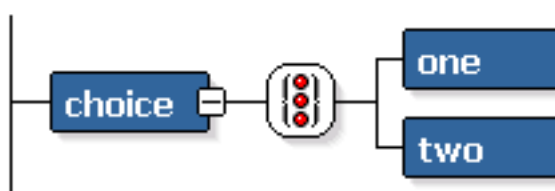


Figura G.2: Elementos de elección

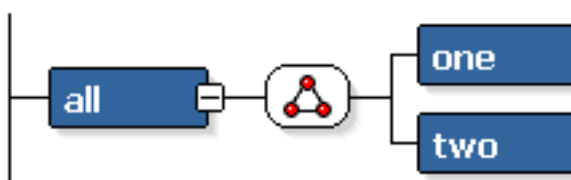


Figura G.3: Elementos Requeridos

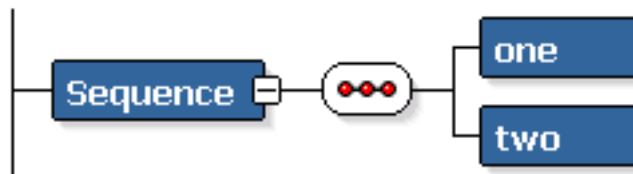


Figura G.4: Elementos Secuenciales

La cardinalidad de los elementos es mostrada de dos maneras: Cuando el mínimo es cero, el elemento es marcado como opcional y trazado con líneas punteadas. En los otros casos, en los cuales existe un rango, dicho rango es mostrado junto a la figura en la notación “mínimo:máximo”:

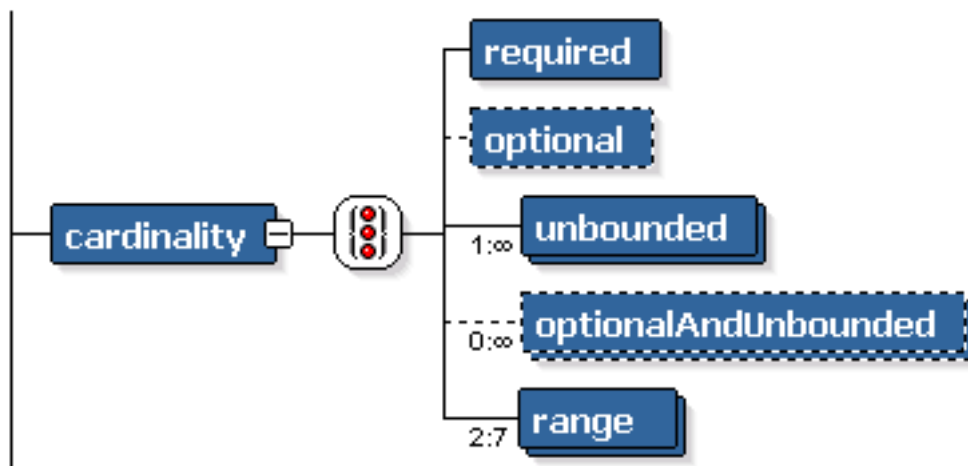


Figura G.5: Cardinalidad de Elementos

Los tipos compuestos son mostrados en naranja, en un cuadro expandible que contiene los detalles de la especificación del mismo. Las relaciones hacia otros elementos complejos también son mostradas, y son expandibles/contraíbles para optimizar el uso de espacio.

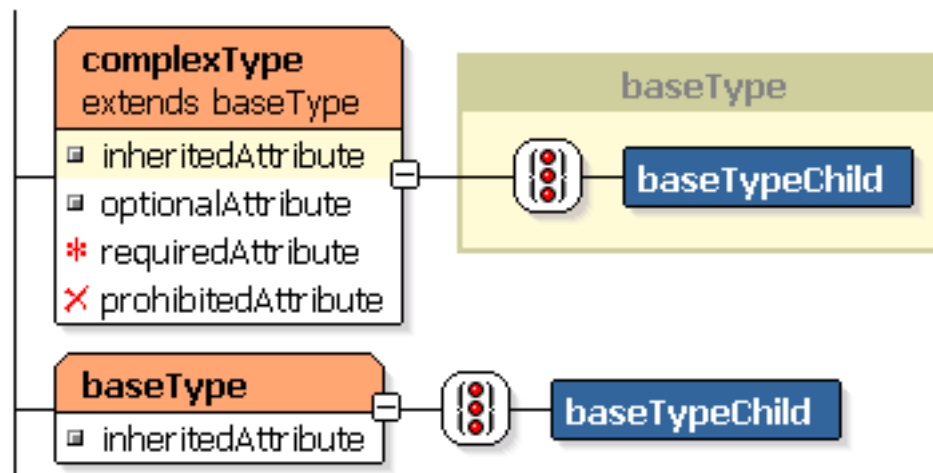


Figura G.6: Definición de Tipos Compuestos

Finalmente, las uniones entre elementos se denotan con un rectángulo amarillo estableciendo la unión; y las listas son mostradas en elementos púrpura, numerando los contenidos permisibles de la misma:

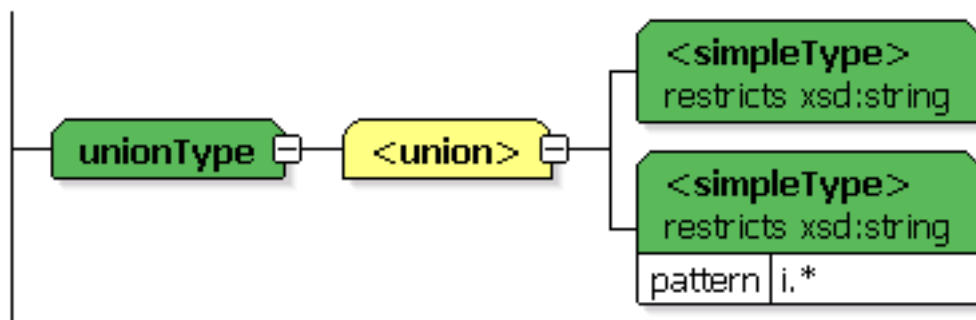


Figura G.7: Definición de uniones

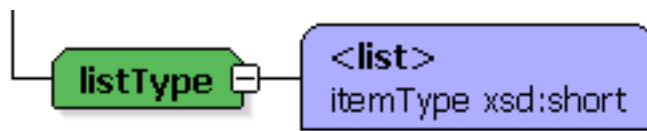


Imagen G.8: Definición de Listas

BIBLIOGRAFÍA

- [I] Jussi Penttilä. "Introduction to Mediation" 20 de noviembre del 2001.
<http://www.netlab.hut.fi/opetus/s38151/Luentomatsku/MDS_SAS.ppt>
- [II] ACE*COMM Corporation "Convergent Mediation" . Tomado en enero 2 del 2004.
<http://www.acecomm.com/pdfs/acecomm_cmtechtools_spanish.pdf>
- [III] ACE*COMM Corporation "Using Mediation for Intercarrier Validation and Billing" . Tomado en enero 2 del 2004.
<http://www.acecomm.com/pdfs/acecomm_billing_verification.pdf>
- [IV] INTEC Telecom Systems "Interconnection – an introduction" . Tomado en enero 2 del 2004.
<<http://www.intec-telecom-systems.com/its/pressroom/whitepapers>>

- [V] Chorleywood Consulting. "IP Mediation". Tomado en enero 2 del 2004.
<<http://www.chorleywood.com/publications/brochures/M68brochure.pdf>>
- [VI] Anderson Harvy. "The One-Stop, Dynamic, Event Detail Cache for Telcos".
Tomado en enero 2 del 2004.
<http://www.verizonit.com/pdf/ah_wp_ums_ods.pdf>
- [VII] Universal Wireless Communications Consortium "Fraud Tutorial" . Tomado en
enero 2 del 2004.
<http://www.3gamericas.org/english/technology_center/presentations/fraud/fraud_tutorial.PPT>
- [VIII] NTels Co., Ltd. "OSS Solutions for Next Generation". Tomado en enero 2 del
2004.
<<http://www.hp.co.kr/event/telecom/pdf/A4.pdf>>
- [IX] NTels Co., Ltd. "nTels Mediation Solution". 23 de julio del 2003.
<http://www.ntels.com/image/PD_NMD.pdf>
- [X] Kirsi Valtari. "Telecom Architectures" . Tomado en enero 2 del 2004.
<<http://www.tml.hut.fi/Opinnot/T-110.300/2002/Luennot/l1128.pdf>>

- [XI] Conor Ryan. "Mediation, Rating and Billing in an IP services environment - architecture and approach". Tomado en enero 2 del 2004.
<<http://www.ist-albatross.org/AccountingWhitePaper.pdf>>
- [XII] Mickael Badar. "Delivery of Integrated Services in AlbatrOSS". Tomado en enero 2 del 2004.
<<http://www.ist-albatross.org/DISWhitePaper.pdf>>
- [XIII] "Bill Format Conversion.pdf", Documentación de la central Huawei, Linkotel S.A.
- [XIV] "Ticket Format of OVS003B.pdf" ", Documentación de la central Huawei, Linkotel S.A.
- [XV] Michael Morrison, "XML al descubierto" Ed. Prentice Hall, 2000.
- [XVI] MorganDoyle Limited. "Mediation for Convergent Billing Systems". Septiembre del 2001.
<http://www.morgandoyle.co.uk/white_papers/Mediation.pdf>
- [XVII] Lucent Technologies. "LUCENT'S KENAN SYSTEMS RECEIVES FIRST CONTRACT FOR NEW ENERGY BILLING SOFTWARE FROM WESSEX WATER SERVICES, LTD". 22 de febrero del 2000.
<<http://www.prnewswire.co.uk/cgi/news/release?id=11422>>

- [XVIII] Kabira Technologies, Inc. "UK'S BIGGEST ELECTRICITY SUPPLIER, npower IMPLEMENTS KABIRA'S POWERFUL MEDIATION SOLUTION TO SUPPORT FAST EXPANSION INTO UK TELECOMMUNICATIONS MARKETPLACE". 4 de febrero del 2003.
<http://www.real-time-enterprise-software.com/news/pr2_4_03.html>
- [XIX] MIND C.T.I. Ltd. All. "Mediation and AAA". Tomado en enero 2 del 2004.
<<http://www.mindcti.com/mediationAAA.html>>
- [XX] Peter Lambert. "Value-based IP Billing Prompts Standards Debate". Abril del 2000.
<<http://www.phoneplusmag.com/articles/041opcen.html?wts=20040708081347&hc=121&req=Peter+and+Lambert>>
- [XXI] Intec Telecom Systems PLC. Tomado en agosto 2 del 2004.
<<http://www.intec-telecom-systems.com>>
- [XXII] ACE*COMM Corporation. Tomado en agosto 2 del 2004.
<<http://www.acecomm.com>>
- [XXIII] Eric Armstrong, Stephanie Bodoff, Debbie Carson, Ian Evans. "The J2EE 1.4 Tutorial". Mayo 30 del 2003.
<<http://java.ittoolbox.com/documents/document.asp?i=2544>>

- [XXIV] Eric Armstrong, Stephanie Bodoff y otros. "The Java Web Services Tutorial".
17 de diciembre del 2003.
<<http://java.sun.com/webservices/docs/1.3/tutorial/doc>>
- [XXV] W3C, "Extensible Markup Language", Tomado en Julio del 2004.
<<http://www.w3.org/XML/>>
- [XXVI] Charles F. Goldfarb, "The ROOTS of SGML – A Personal Recollection", 1996.
<<http://www.sgmlsource.com/history/roots.htm>>
- [XXVII] SGML Users' Group. "A Brief History of the Development of SGML", Junio 11
de 1990.
<<http://www.sgmlsource.com/history/sgmlhist.htm>>
- [XXVIII] Didier Courtaud, "From GenCode to XML: An history of Markup Languages",
Noviembre del 2002.
<www.renater.fr/Video/2003ATHENS/DC-XMLHistory_eng/all.pdf>
- [XXIX] W3C, "Extensible Markup Language (XML) 1.0 (Third Edition)".
Recomendación del 4 de Febrero del 2004.
<<http://www.w3.org/TR/2004/REC-xml-20040204/>>

- [XXX] W3C. “Extensible Markup Language (XML) 1.1”, Recomendación del 4 de Febrero del 2004, editada el 15 de Abril del 2004.
<<http://www.w3.org/TR/2004/REC-xml11-20040204/>>
- [XXXI] W3C. “XML Schema Part 0: Primer”. Recomendación del 2 de Mayo del 2001.
<<http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/>>
- [XXXII] W3C, “XML Schema Part 1: Structures”, Recomendación del 2 de Mayo del 2001.
<<http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>>
- [XXXIII] W3C, “XML Schema Part 2: Datatypes”, Recomendación del 2 de Mayo del 2001.
<<http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>>
- [XXXIV] Pankaj Kamthan, “A Gift of "Life" : The Document Object Model”, 15 de Enero de 1999.
<<http://tech.irt.org/articles/js143/>>

- [XXXV] W3C, “Document Object Model (DOM) Technical Reports”, según las versiones mas recientes disponibles en Julio del 2004.
<<http://www.w3.org/DOM/DOMTR>>
- [XXXVI] SAX. Tomado en Julio del 2004.
<<http://sax.sourceforge.net/>>
- [XXXVII] The Apache Software Foundation, “Xerces2 Java Parser Readme”, The Apache XML Project. Tomado en Julio del 2004.
<<http://xml.apache.org/xerces2-j/index.html>>
- [XXXVIII] W3C, “XML Path Language (XPath)”. Recomendación del 16 de Noviembre de 1999.
<<http://www.w3.org/TR/1999/REC-xpath-19991116>>
- [XXXIX] Jonathan Robie, Texcel Research, “¿Qué es el modelo de Objetos del Documento?”, tomado el 26 de octubre de 2004.
<<http://html.conclase.net/w3c/dom1-es/introduction.html>>
- [XL] SAX, Javadoc, Class InputSource, tomado el 26 de octubre de 2004.
<<http://www.saxproject.org/apidoc/org/xml/sax/InputSource.html>>

- [XLII] Intec Telecom Systems – Customer list. Tomado en agosto 2 del 2004.
<<http://www.intec-telecom-systems.com/its/aboutus/customers/list/>>
- [XLIII] Intec Telecom Systems – Convergent IP and Billing Mediation. Tomado en agosto 2 del 2004.
<<http://www.intec-telecom-systems.com/its/productsservices/products/intermediate/>>
- [XLIV] Intec Telecom Systems – Intec launches Version 5 of Inter-mediate. Tomado en agosto 2 del 2004.
<<http://www.intec-telecom-systems.com/its/pressroom/pressreleases/pr2004/2004-03-03/>>
- [XLV] ACE*COMM – Service Provider Solution. Tomado en agosto 2 del 2004.
<http://www.acecomm.us/serviceprovider/convergent_mediation.htm>
- [XLVI] ACE*COMM – Capture. Tomado en agosto 2 del 2004.
<<http://www.acecomm.us/serviceprovider/capture.htm>>
- [XLVII] ACE*COMM – Collection. Tomado en agosto 2 del 2004.
<<http://www.acecomm.us/serviceprovider/collection.htm>>
- [XLVIII] ACE*COMM – Manage. Tomado en agosto 2 del 2004.
<<http://www.acecomm.us/serviceprovider/manage.htm>>

[XLVIII] ACE*COMM – Price/Cost. Tomado en agosto 2 del 2004.

<<http://www.acecomm.us/serviceprovider/price.htm>>

[XLIX] ACE*COMM – Present. Tomado en agosto 2 del 2004.

<<http://www.acecomm.us/serviceprovider/present.htm>>

[L] Documentación de centrales Ericsson. Pacifictel S.A.

[LI] Documentación de centrales Alcatel. Pacifictel S.A.

[LII] Documentación de centrales Siemens. Pacifictel S.A.

[LIII] Documentación de centrales Honet. Pacifictel S.A.

Diagrama de Clases del Sistema *

