Ingeniería en Informática

Escuela Politécnica Superior (Jaén)

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior (Jaén)

Proyecto Fin de Carrera

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN FRAMEWORK PARA EL DISEÑO DE MOTORES DE ANÁLISIS LINGÜÍSTICO BASADOS EN REGLAS

Alumno: Francisco Javier Estrella Liébana

Tutoras: Prof. Dra. Macarena Espinilla Estévez
Prof. Rosa Mª Rodríguez Domínguez

Dpto: Informática

Área de Conocimiento: Arquitectura y Tecnología de Computadores

Septiembre, 2011

UNIVERSIDAD DE JAÉN

DEPARTAMENTO DE INFORMÁTICA

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN FRAMEWORK PARA EL DISEÑO DE MOTORES DE ANÁLISIS LINGÜÍSTICO BASADOS EN REGLAS

Autor:
Francisco Javier Estrella Liébana

Tutoras:
Macarena Espinilla Estévez
Rosa Mª Rodríguez Domínguez

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Dra. Prof. Dª. Macarena Espinilla Estévez y Prof. Dª. Rosa Mª Rodríguez Domínguez, tutoras del Proyecto Fin de Carrera titulado: Análisis, diseño e implementación de un framework para el diseño de motores de análisis lingüístico basados en reglas, que presenta D. Francisco Javier Estrella Liébana, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, septiembre de 2011

El alumno: Las tutoras:

D. Francisco Javier Estrella Liébana

Dra. Prof. Dª. Macarena Espinilla Estévez
Prof. Dª. Rosa Mª Rodríguez Domínguez

Agradecimientos

Quisiera mostrar mi agradecimiento a todas las personas que, de un modo u otro, habéis contribuido en la realización de este proyecto.

Raquel, gracias por estar siempre ahí y ser quién eres.

Papá, mamá, gracias por el sacrificio que habéis hecho por mí, por apoyarme y por darme vuestra confianza.

Macarena y Rosa, gracias por la ayuda, el tiempo y la paciencia que me habéis brindado.

Luis, gracias por los consejos y por estar siempre dispuesto a prestarme tu ayuda.

Al resto, familia, compañeros y amigos, gracias por tantos buenos momentos.

Índice general

1. Introducción **1**

1.1. Introducción al proyecto 3

1.2. Propósito 4

1.3. Objetivos 4

1.4. Estructura de la memoria 5

2. Antecedentes **7**

2.1. Introducción 9

2.2. Iguáleme 11

2.3. Herramientas de análisis lingüístico existentes 12

3. Análisis lingüístico basado en reglas **19**

3.1. Introducción 21

3.2. Enfoque del análisis 23

3.2.1. Descripción del análisis 23

3.2.2. Tipo de análisis 26
3.2.3. Aplicabilidad del análisis 27
3.2.4. Campos de aplicación 28
3.2.4.1. Igualdad de género 31
3.2.5. Ventajas 34
3.2.6. Limitaciones 35
3.3. Funcionamiento del analizador 36
3.3.1. Procedimiento general 37
3.3.2. Tipos de reglas disponibles 39
3.3.2.1. Búsqueda contextual 39
3.3.2.2. Valores permitidos para los elementos 40
3.3.2.3. Tipo 1: Disyunción 41
3.3.2.4. Tipo 2: Palabra en conjunto de términos 42
3.3.2.5. Tipo 3: Existe palabra de conjunto de términos en un contexto 42
3.3.2.6. Tipo 4: Existe variante gramatical de palabra de un conjunto de términos en un contexto 43
3.3.2.7. Tipo 5: Palabra de una clase de palabras 43
3.3.2.8. Tipo 6: Existe palabra de una clase de palabras en un contexto 44
3.3.2.9. Tipo 7: Existe variante gramatical de palabra de una clase de palabras en un contexto 44
3.3.3. Algunos ejemplos 45

3.3.3.1. Comprobación de la adjetivación 45
3.3.3.2. Comprobar si una palabra está en masculino 47
3.3.4. Igualdad 48

4. Framework de análisis **51**

4.1. Arquitecturas basada en componentes 53
4.1.1. Introducción 53
4.1.2. Beneficios 55
4.1.3. Componentes 56
4.1.3.1. Características 56
4.1.4. Arquitecturas de componentes existentes 57
4.1.5. OSGi 59
4.1.5.1. Plataforma de servicios OSGi 59
4.1.5.2. Implementaciones de OSGi 63
4.1.5.3. Proyectos desarrollados mediante la arquitectura OSGi . . 64
4.2. Framework de análisis 68
4.2.1. Introducción 68
4.2.2. Características 69
4.2.2.1. Características generales 69
4.2.2.2. Eclipse Plug-Ins 70
4.2.2.3. Archivos JAR 71
4.2.3. Tipos de componentes 73

4.3. Tecnologías 74
4.3.1. Arquitectura del Framework 75
4.3.1.1. Alternativas existentes 75
4.3.1.2. Eclipse RCP 76
4.3.1.3. Aplicaciones desarrolladas con Eclipse RCP 77
4.3.2. Lenguajes de programación 81
4.3.3. Librerías, herramientas y programas 83
4.3.3.1. Librerías 83
4.3.3.2. Herramientas 86
4.3.3.3. Programas 87

5. Desarrollo del proyecto **89**

5.1. Descripción 91
5.2. Especificación de requerimientos 92
5.2.1. Requerimientos funcionales 93
5.2.2. Requerimientos no funcionales 94
5.2.2.1. Requerimientos software y hardware 94
5.3. Análisis del sistema 96
5.3.1. Casos de uso 96
5.3.1.1. Notación empleada 100
5.3.2. Escenarios 113
5.4. Diseño del sistema 119

5.4.1. Diseño de los componentes 119
5.4.1.1. Componentes 120
5.4.1.2. Tipo de los componentes 121
5.4.1.3. Grafos de dependencias 122
5.4.2. Diseño de la interfaz 127
5.4.2.1. Guía de estilo 127
5.4.2.2. Diccionario 133
5.4.2.3. Metáforas 138
5.4.2.4. Prototipos de la interfaz 142
5.4.2.5. Caminos de navegación 151
5.5. Implementación y pruebas 158
5.5.1. Desarrollo dirigido por pruebas 158
5.5.1.1. Formato de las pruebas 160

6. Conclusiones **163**

6.1. Conclusiones finales 165
6.2. Líneas de trabajo futuras 166

A. Manual de instalación **167**

A.1. Software requerido 169
A.1.1. Plataforma de ejecución Java 169
A.1.2. XULRunner 171

A.1.3. OpenOffice.org 172

B. Manual de usuario **173**

B.1. Acerca de este documento 175

B.2. Ejecutar Linguo 175

B.3. Interfaz de Linguo 177

B.3.1. Visor web 177

B.3.2. Visor OpenOffice 179

B.3.3. Visores informativos 180

B.3.4. Menús de la aplicación 183

B.3.5. Ayuda 185

B.3.6. Asistente para la generación de informes 185

B.3.7. Generación de reglas 186

B.4. Funcionalidad básica 188

B.4.1. Cambiar el visor activo 188

B.4.2. Modificar los visores informativos visibles 188

B.4.3. Ayuda 190

B.4.3.1. Acerca del programa 190

B.4.3.2. Ayuda del programa 193

B.4.4. Usar el navegador 194

B.4.4.1. Interfaz principal 194

B.4.4.2. Controles de navegación 195

B.4.5. Usar OpenOffice 196
B.4.5.1. Interfaz principal 196
B.4.5.2. Controles de OpenOffice 197
B.4.6. Usar el visor de resultados 198
B.4.7. Salir de Linguo 201
B.5. Funcionalidad avanzada 202
B.5.1. Generar un informe 202
B.5.1.1. Generar un informe de un documento 203
B.5.1.2. Generar un informe del visor activo 206
B.5.1.3. Generar un informe de un sitio web 207
B.5.2. Generador de reglas 210
B.5.2.1. Paquete de reglas 211
B.5.2.2. Conjunto de términos 213
B.5.2.3. Términos 214
B.5.2.4. Reglas 216
B.5.2.5. Importar y exportar paquetes de reglas 219
B.5.2.6. Visor de prueba 220
B.5.2.7. Almacenar los cambios realizados 221
B.6. Atajos de teclado 221
B.7. FAQ 222
B.7.1. Algunas incidencias detectadas no aparecen en el visor web 222

B.7.2. El contenido del navegador web se muestra mal una vez analizado . 222

B.7.3. El análisis falla con algunos documentos 222

B.7.4. El comportamiento del componente X no es correcto 223

Índice de cuadros

3.1. Valores permitidos 41

3.2. Elementos del tipo de regla 'Palabra en conjunto de términos' 42

3.3. Elementos del tipo de regla 'Existe palabra de conjunto de términos en un contexto' 43

3.4. Elementos del tipo de regla 'Existe variante gramatical de palabra de un conjunto de términos en un contexto' 43

3.5. Elementos del tipo de regla 'Palabra de una clase de palabras' 44

3.6. Elementos del tipo de regla 'Existe palabra de una clase de palabras en un contexto' 44

3.7. Elementos del tipo de regla 'Existe variante gramatical de palabra de una clase de palabras en un contexto' 45

5.2. Matriz de trazabilidad 99

5.6. Iconos de la interfaz 141

B.1. Controles de navegación 196

B.2. Controles de OpenOffice 197

B.3. Tipos de recursos 211

B.4. Atajos de teclado 221

Índice de figuras

2.1. Logotipo de Iguáleme 11

2.2. Interfaz de WordSmith Tools 13

2.3. Interfaz de Plagiarism Checker 14

2.4. Interfaz de Concordance 15

2.5. Interfaz de nombra.en.red 16

2.6. Interfaz de Themis 17

4.1. Curva de fallos del software 54

4.2. Logotipo de la OSGi Alliance 58

4.3. Logotipo de Eclipse 58

4.4. Dependencias en forma de grafo 61

4.5. Uso de múltiples versiones 62

4.6. Carga de componentes dinámica 63

4.7. Ejemplo de reporte generado con BIRT 64

4.8. Interfaz de Eclipse 65
4.9. Interfaz de GX WebManager 65
4.10. Interfaz de NetBeans 66
4.11. Interfaz de Nuxeo 66
4.12. Interfaz de WSO2 Carbon 67
4.13. Interfaz de Confluence 67
4.14. Aplicación constituida por Plug-ins 71
4.15. Conflicto entre versiones 73
4.16. Interfaz de Bioclipse 78
4.17. Interfaz de CCTVnet 78
4.18. Interfaz de Azureus 79
4.19. Interfaz de J3DWorkbench 79
4.20. Interfaz de Kalypso 80
4.21. Interfaz de BaZaa 80
4.22. Interfaz de Global Vision Systems 81
4.23. Índice Tiobe de lenguajes de programación [34] 82
4.24. Logotipo de HtmlParser 83
4.25. Logotipo de ODFDOM 83
4.26. Logotipo de Apache POI 84
4.27. Logotipo de SWTChart 84
4.28. Logotipo de NOA 85

4.29. Logotipo de HyperSQL 85
4.30. Logotipo de iText 85
4.31. Logotipo de jFeeChart 86
4.32. Logotipo de JUnit 86
4.33. Ejemplo de reporte generado por cobertura 86
4.34. Logotipo de Mockito 87
4.35. Logotipo de GIT 87
4.36. Logotipo de Java 88
4.37. Logotipo de OpenOffice.org 88

5.1. Diagrama frontera del sistema 98
5.2. Caso de uso 'Generar informe' 101
5.3. Caso de uso 'Gestión de paquetes de reglas' 110
5.4. Escenario 1: Apertura de un archivo en el editor 114
5.5. Escenario 2: Generación de un informe de un documento 116
5.6. Escenario 3: Guardar la dirección web actual 116
5.7. Escenario 4: Modificar un paquete de reglas 117
5.8. Escenario 5: Importar un paquete de reglas 118
5.9. Grafo de dependencias de linguo 123
5.10. Grafo de dependencias de linguo.reports 124
5.11. Grafo de dependencias de linguo.rulemaker 124
5.12. Grafo de dependencias de linguo.viewer.openoffice 125

5.13. Grafo de dependencias de linguo.viewer.results 125

5.14. Grafo de dependencias de linguo.viewer.webnavigator 126

5.15. Grafo de dependencias del resto de componentes 126

5.16. Logotipo Linguo 132

5.17. Isotipo Linguo 132

5.18. Logotipo e Isotipo Linguo 133

5.19. Ejemplos de metáforas comunes a cualquier interfaz de usuario 139

5.20. Pantalla principal navegador 142

5.21. Visor estadísticas 143

5.22. Pantalla principal OpenOffice 143

5.23. Menú archivo 144

5.24. Menú ver en vista navegador 144

5.25. Menú ver en vista OpenOffice 144

5.26. Menú herramientas 144

5.27. Menú OpenOffice 145

5.28. Menú navegador 145

5.29. Tipo de informe 146

5.30. Informe de documento 146

5.31. Informe de visor activo 147

5.32. Informe de sitio web 147

5.33. Generador de reglas 148

5.34. Nuevo paquete de reglas 148

5.35. Nueva regla 149

5.36. Nuevo conjunto de términos 149

5.37. Nuevo término 150

5.38. Menú contextual paquetes de reglas 150

5.39. Menú contextual paquete de reglas 150

5.40. Menú contextual conjunto de términos 151

5.41. Menú contextual regla 151

5.42. Creación de un nuevo paquete de reglas. 153

5.43. Generación de un informe de un sitio web. 154

5.44. Cambiar la vista del navegador 155

5.45. Exportar un paquete de reglas 156

5.46. Modificación de un término de un conjunto de términos 157

A.1. Activación de la plataforma Java proporcionada por Sun 170

A.2. Comprobar el estado de XULRunner 171

B.1. Error de ejecución de Linguo 175

B.2. Asignar permisos de ejecución 176

B.3. Visor web 177

B.4. Incidencia en visor web 178

B.5. Visor OpenOffice 179

B.6. Incidencia en visor OpenOffice 180
B.7. Visor de resultados 181
B.8. Visor de estadísticas 182
B.9. Menú en visor web 183
B.10. Menú en visor OpenOffice 184
B.11. Ayuda de Linguo 185
B.12. Asistente para la generación de informes 186
B.13. Generador de reglas 187
B.14. Cambiar el visor activo 188
B.15. Modificación de la visibilidad de un visor informativo 189
B.16. Todos los visores informativos ocultos 189
B.17. Abrir el diálogo Acerca de Linguo 190
B.18. Diálogo Acerca de Linguo 190
B.19. Detalles de la instalación de Linguo 191
B.20. Configuración de Linguo 192
B.21. Acceder a la ayuda 193
B.22. Búsqueda dentro de la ayuda 194
B.23. Incidencia por varios paquetes de reglas 195
B.24. Barra de navegación 195
B.25. Uso del visor de resultados 198
B.26. Panel de paquetes 199

B.27.Filtrado por paquetes 199

B.28.Panel de términos 200

B.29.Filtrado por términos 200

B.30.Salir de Linguo 201

B.31.Abrir el asistente de generación de informes 202

B.32.Página de informe 203

B.33.Informe de documento 204

B.34.Campos de un informe de documentos 205

B.35.Nombre del informe a generar 205

B.36.Campos de un informe del visor activo 206

B.37.Campos de un informe de un sitio web 207

B.38.Descarga de páginas 209

B.39.Análisis de páginas 209

B.40.Agregación de resultados 209

B.41.Acceso al generador de reglas 210

B.42.Paquetes de reglas existentes 211

B.43.Nuevo paquete de reglas 212

B.44.Nuevo conjunto de términos 213

B.45.Eliminar conjunto de términos en uso 214

B.46.Nuevo término 214

B.47.Introducción de un término 215

B.48.Término añadido al conjunto de términos 215

B.49.Nueva regla 216

B.50.Selección del tipo de regla 217

B.51.Elementos de regla válidos 218

B.52.Eliminar regla con reglas hijas 218

B.53.Importación de paquete duplicado 219

B.54.Marcado de incidencias en visor de prueba 220

B.55.Incidencia por múltiples paquetes 220

Capítulo 1

Introducción

1.1. Introducción al proyecto

De sobra es conocida la importancia que el lenguaje ha tenido para el desarrollo del ser humano. A través del lenguaje nos comunicamos, transmitimos ideas, manifestamos sentimientos, pero también podemos insultar, menospreciar, infundir odio, etc.

Esta capacidad de herir mediante el lenguaje puede ser buscada en determinadas ocasiones de manera intencionada, sin embargo, en muchas ocasiones surge por el simple hecho de utilizar aquellas palabras, frases y expresiones con las que estamos más comúnmente familiarizados.

Teniendo esto en cuenta, parece lógico pensar que si actuamos directamente sobre todos aquellos términos y expresiones que son hirientes o excluyentes para determinados colectivos podremos reducir de forma significativo el perjuicio que a estos se hace a través del lenguaje.

En la actualidad, conscientes de la conveniencia de llevar a cabo acciones de este tipo están surgiendo multitud de iniciativas que abordan esta problemática: políticas lingüísticas [1], guías de uso del lenguaje [2], manifiestos que defienden que se anteponga a la persona antes que su enfermedad, raza, discapacidad, genero o cualquier otro aspecto [3], etc.

Paralelamente a estas iniciativas están apareciendo en los últimos años varias herramientas software que intentan facilitar y servir de apoyo a todo el trabajo que se está realizando. Todas estas herramientas, aunque operativas en mayor o menor grado, presentan serias limitaciones: lentitud, elevado consumo de recursos, rigidez de funcionamiento, dificultad para analizar grandes volúmenes de información, etc.

En el núcleo de la mayoría de las limitaciones mencionadas anteriormente radica un mismo problema, la ausencia de un marco de trabajo común que permita erradicar cualquier esfuerzo que no este directamente enfocado a discernir cuales son los términos y expresiones inapropiados en un contexto determinado.

En este proyecto llevaremos a cabo el desarrollo de este marco de trabajo, abstrayendo toda la parte común a la realización de un análisis lingüístico sobre texto escrito (procesado de la información, búsqueda de incidencias, visualización de resultados de

forma agregada, ...). De esta forma, cualquier persona interesada en hacer uso de la solución implementada, tan solo deberá de establecer aquellas reglas que considere oportunas obviando todo lo demás.

Para mostrar el funcionamiento en un caso concreto realizaremos el desarrollo de un paquete de reglas orientado hacia la igualdad de género, el cual permitirá conocer dentro de un documento dado cualquier incidencia que tenga carácter ofensivo o excluyente en base a cuestiones de género.

1.2. Propósito

El propósito de este proyecto es analizar, diseñar e implementar un Framework modular y extensible para diseñar motores de análisis lingüísticos basados en paquetes de reglas.

1.3. Objetivos

1. Búsqueda y revisión bibliográfica.
2. Analizar las principales arquitecturas de componentes existentes en la actualidad y seleccionar la más adecuada a las características de la solución a implementar.
3. Desarrollar una solución que permita analizar el contenido de distintas fuentes notificando al usuario la adecuación o no del lenguaje utilizado a las reglas fijadas.
4. Realizar un diseño altamente modular que permita el uso en distintos ámbitos así como la reutilización de los componentes desarrollados.
5. Obtención de un corpus lingüístico para la creación de un diccionario gramatical que pueda ser empleado por las reglas.
6. Elaborar un paquete de reglas centrado en la igualdad de género.
7. Redacción de la memoria.

1.4. Estructura de la memoria

A continuación, se hará una breve introducción de los capítulos en los que se ha estructurado este proyecto.

Este primer capítulo pretende ser una introducción general mediante la cual conocer la problemática que nos atañe, el propósito del presente proyecto y los objetivos que perseguimos.

En el capítulo 2 se verán cuales son los antecedentes del proyecto. Mostraremos cual es el panorama actual en el campo del análisis lingüístico asistido por ordenador para después hablar de Iguáleme, herramienta de la cual se deriva el Framework desarrollado.

El capítulo 3 aborda el concepto de análisis lingüístico basado en reglas el que es el eje central de este proyecto. Realizaremos una breve introducción al análisis lingüístico para seguidamente explicar en que consiste el tipo de análisis que aquí planteamos. Pasaremos después a categorizar el tipo de análisis lingüístico a realizar así como la aplicabilidad del mismo dentro de un marco general. Después mostraremos algunos campos de aplicación de este tipo de análisis y detallaremos un caso concreto, la generación de un paquete de reglas para tratar con problemas de desigualdad de género. Por último veremos cual es el procedimiento general del analizador así como los tipos de reglas disponibles. Estas reglas se acompañarán de algunos ejemplos prácticos que permitirán una mayor comprensión sobre su funcionamiento.

El capítulo 4 presenta el Framework desarrollado a través de una explicación general del mismo, los tipos de componentes desarrollados y el ecosistema en el que estos componentes se ejecutan. Para ello realizaremos en primer lugar un breve análisis de las principales arquitecturas de componentes existentes en la actualidad y justificaremos cuales son los motivos que nos han llevado a emplear la arquitectura OSGi. A continuación explicaremos como llevaremos a cabo la realización del Framework haciendo uso de esta arquitectura e introduciremos cuales han sido las herramientas de desarrollo a emplear para su construcción.

El capítulo 5 describe el proceso de desarrollo completo del proyecto. Muestra cada una de las etapas de desarrollo del software para lleva a cabo este proyecto. Así, definirán los requerimientos funcionales y no funcionales, se abordará la etapa de análisis y diseño

del sistema para, finalmente, realizar la implementación. Hablaremos por último de Test Driven Development (TDD), exponiendo en que consiste esta filosofía de desarrollo y como nos hemos valido de ella para alcanzar unos requisitos mínimos de calidad.

El último capítulo recoge las conclusiones generales derivadas del desarrollo del proyecto.

La sección final de la memoria se compone de dos anexos. El primero es un manual de instalación de la aplicación y el segundo un manual de usuario.

Capítulo 2

Antecedentes

2.1. Introducción

A pesar de lo que pudiese parecer, la informática es una ciencia aún joven que lleva entre nosotros poco tiempo. Por este motivo a día de hoy no somos todavía plenamente conscientes de todas las posibilidades reales que nos brinda.

Paralelamente al desarrollo de la informática, más y más campos se están viendo beneficiados de ella, emergiendo continuamente nuevas utilidades. Esto es debido a que el fin mismo de la informática no es otro que el de servir como herramienta de apoyo a través del tratamiento automatizado de los datos.

Esta naturaleza no es siempre bien comprendida, existiendo visiones totalmente diferentes de la función misma de la informática. De esta forma nos encontramos con personas que opinan que la informática no sirve realmente para casi nada (o su versión más común, puede que para determinadas cosas sirva pero para lo que a mí me incumbe no tiene apenas valía) a otras que consideran la informática como una herramienta prácticamente mágica que permite hacer cualquier cosa, ambas visiones tienen parte de razón.

Ciertamente la informática por si sola carece realmente de valor, es el propio conocimiento humano el que le aporta utilidad, poniendo a nuestra disposición potentes herramientas de cálculo para solucionar una enorme variedad de problemáticas diversas. Por otro lado, la informática, a pesar de no disponer de poderes mágicos, permitirá hacer casi cualquier cosa siempre y cuando esta pueda ser abordada mediante una solución concreta.

A pesar de todo, este concepto de la informática como herramienta de ayuda va poco a poco imponiéndose en la visión general y cada vez más se recurre a ella para resolver problemas de una forma realista. Así, van quedándose progresivamente atrás las ideas de 'o todo o nada' y determinadas disciplinas que hasta hace recientemente poco consideraban la informática como inservible para sus propósitos empiezan a abordar sus problemas valiéndose de esta como herramienta de apoyo. Una de estas disciplinas y la que nos incumbe en el presente proyecto es la de la lingüística.

La lingüística, de forma genérica, versa sobre el estudio de las lenguas, abordando tanto el análisis de su estructura como la propia utilización de las mismas.

Bajo esta concepción tan amplia, nos encontramos con una disciplina con multitud de aplicaciones directas, útil para abordar problemas complejos y necesaria para entender aspectos intrínsecos en la propia naturaleza humana.

La lingüística presenta el problema general de que su estudio mediante herramientas informáticas es complejo. Existen cientos de connotaciones en cada caso concreto y es difícil abstraer un caso general. Esta dificultad para obtener una solución óptima lleva a que en muchos casos se deseche el uso de la informática bajo la idea de que "nada es equiparable a la pericia de un observador experto". Sin embargo, debemos considerar que si bien puede ser imposible a día de hoy obtener la mejor solución para un problema dado, si que podemos acotar de forma significativa el mismo y a la vez ofrecer una solución razonable al problema.

A medida que esta idea ha ido calando, han comenzado a surgir herramientas independientes que permiten tratar con problemas lingüísticos concretos. Si bien es cierto que la aparición de diferentes herramientas ya constituye de por si un significativo avance, la forma en la que se están desarrollando las diversas herramientas presenta dos problemas sustanciales:

- Muchas veces se abordan problemas similares de forma independiente. Esto redunda en que se vuelven a implementar una y otra vez las mismas funciones siendo este un trabajo totalmente innecesario.

- En otras ocasiones no se aborda un determinado problema por que no existen las herramientas necesarias, o bien no se dispone de los conocimientos necesarios para realizarlas.

El Framework que aquí se propone pretende ser una pequeña ayuda dentro del análisis lingüístico y disminuir un poco la problemática de los dos puntos anteriores, es decir, por un lado evitar que se vuelva a hacer una y otra vez lo que ya existe y por otro, proporcionar una herramienta de uso general que pueda ser utilizada en determinados casos para abordar problemas concretos.

Este Framework y su concepción no surgen de la nada si no que tienen su proceso de maduración previo, para entender mejor este proceso hablemos de Iguáleme.

2.2. Iguáleme

Iguáleme es fruto del proyecto de investigación "Mundo Virtual sin Desigualdades" (UNIVER09/2009/23/0005) concedido por el Instituto Andaluz de la Mujer de la Consejería para la Igualdad y Bienestar Social. Busca ser una herramienta que luche contra la discriminación basada en sexos y promueva la configuración de un "mundo virtual" en Internet sin desigualdades. Para ello, Iguáleme analiza el texto contenido en sitios Web, páginas Web y archivos de texto y proporciona informes sobre entidades dudosas que aparecen en el texto que no igualan a la mujer.

Figura 2.1: Logotipo de Iguáleme

La aplicación fue desarrollada por el autor de este proyecto y por el equipo investigador:

- Macarena Espinilla Estévez (como investigadora principal).
- Rosa Mª Rodríguez Domínguez.
- Rocío de Andrés de Calle.
- Patricia Gómez Costilla.
- Luis Martínez López.

Tras la finalización de la herramienta se observó que gran parte de la funcionalidad podría ser utilizada para resolver problemas similares. De este modo podríamos utilizar la aplicación para resolver otros problemas como la detección de expresiones racistas o la presencia de jerga profesional. Todo ello manteniendo una serie de funcionalidades ya disponibles (navegador web, generación de informes, ...).

De aquí surge la idea de abstraer la parte encargada de guiar el análisis y obtener una herramienta de carácter general que pudiese ser empleada en la resolución de problemas similares.

Pero al igual que se puede abstraer la parte encargada del análisis se puede también aislar el resto de partes funcionales que componen la aplicación (navegador web, visor de errores, analizador, ...) para ser utilizados en distintas aplicaciones de análisis.

En base a esta idea se decide llevar a cabo la elaboración de un Framework de análisis lingüístico que siga el modelo de análisis introducido en Iguáleme.

Este Framework es una ambiciosa remodelación de Iguáleme que lleva a cabo la descomposición del mismo en componentes, lo adapta para trabajar con información de forma genérica y le añade nuevas funcionalidades.

Sobre el análisis, diseño e implementación de este Framework versa este proyecto.

2.3. Herramientas de análisis lingüístico existentes

Para finalizar este capítulo mostraremos algunas de las herramientas de análisis lingüístico existentes a fin de hacernos una idea tanto de sus puntos fuertes como de sus limitaciones.

WordSmith Tools [7]

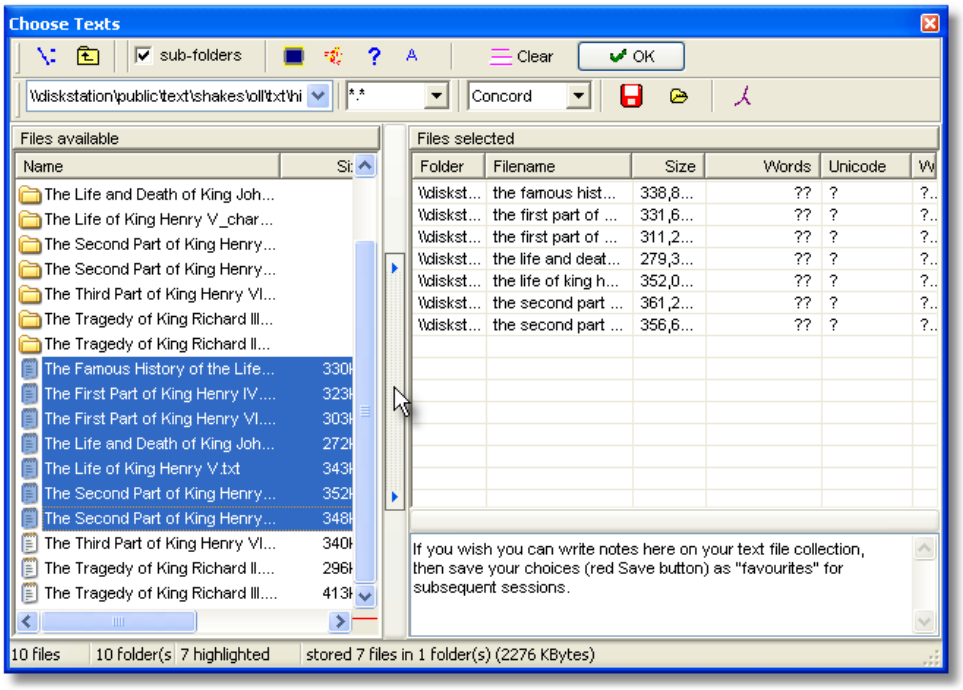


Figura 2.2: Interfaz de WordSmith Tools

WordSmith Tools (figura 2.2) es, posiblemente, el programa de análisis textual más popular. Está formado por tres aplicaciones:

- **WordList**: Un generador de listas de palabras en orden alfabético o de frecuencias.
- **Concord**: Un generador de listas de palabras en contexto.
- **KeyWords**: Un buscador de palabras clave.

Los distintos tipos de análisis permitidos la convierten en una de las herramientas más prácticas para la realización de análisis textuales, sin embargo, presenta serias limitaciones:

- Es software de código cerrado y su funcionalidad no puede ser utilizada para la construcción de nuevas herramientas.

- La versión gratuita solo permite obtener un número limitado de resultados.
- Sus aplicaciones no son capaces de trabajar de forma conjunta por lo que no es posible realizar un análisis conjugando toda la funcionalidad proporcionada.

Plagiarism Checker [8]

Figura 2.3: Interfaz de Plagiarism Checker

Plagiarism Checker (figura 2.3) es un detector en linea de plagios. Su funcionamiento es sumamente sencillo, solo se ha de indicar el fragmento de texto a comprobar y la herramienta buscará los posibles plagios existentes haciendo una consulta en los buscadores Google y Yahoo!.

Las limitaciones son muy parecidas a las detectadas para WordSmith Tools:

- Software cerrado.

- Versión gratuita limitada.
- Aplicabilidad muy limitada.

Concordance [9]

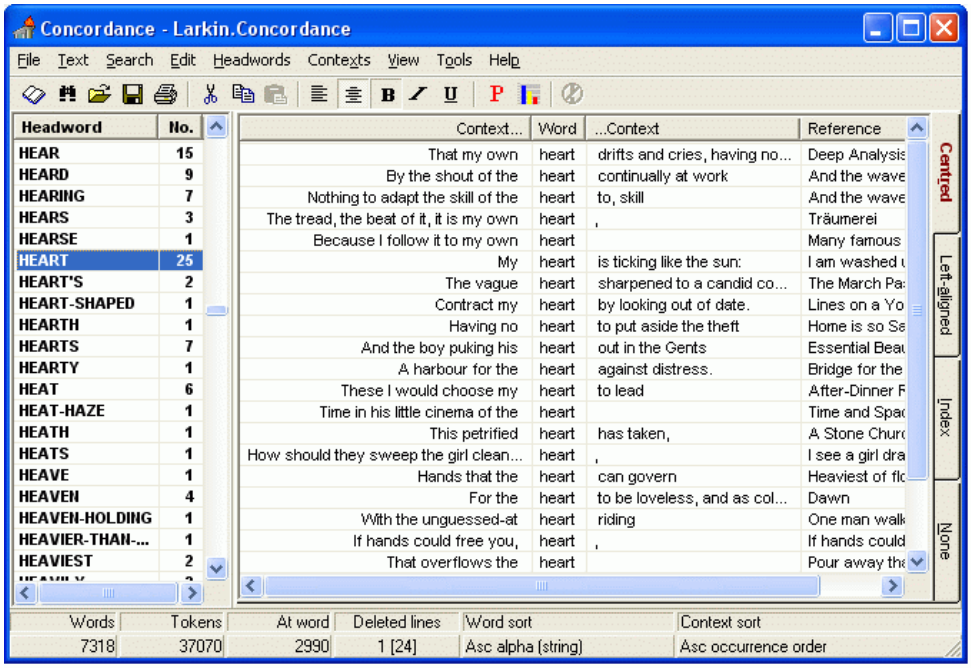


Figura 2.4: Interfaz de Concordance

Concordance (figura 2.4) es una herramienta de ayuda al análisis del lenguaje. Su funcionalidad es similar a la de WordsSmith Tools, pudiendo realizar tareas como:

- Crear listas de palabras.
- Contar frecuencias.
- Buscar los diferentes usos de una palabra.
- Buscar frases en un idioma dado.

Sin embargo, vuelve a presentar serias limitaciones:

- Software cerrado y de pago.
- Las funcionalidades no pueden ser empleadas de forma conjunta.

nombra.en.red [72]

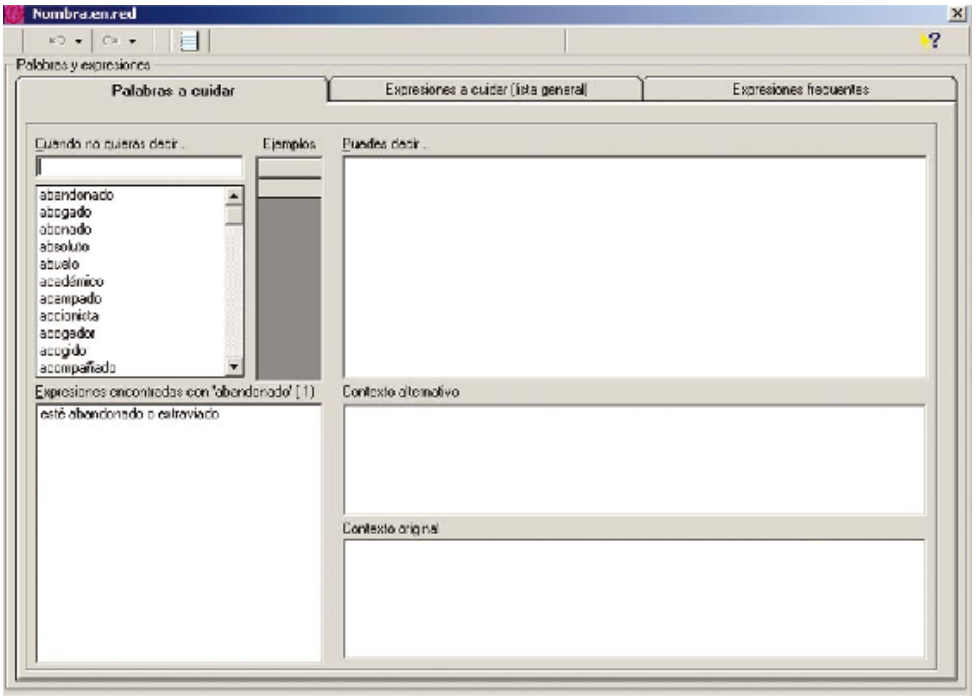


Figura 2.5: Interfaz de nombra.en.red

nombra.en.red (figura 2.5) es una herramienta de consulta sobre posibles expresiones sexistas. Más que como herramienta de análisis debe considerarse como herramienta de ayuda al análisis, ya que su única función es la de proporcionar propuestas de redacción alternativas que faciliten la utilización de un lenguaje inclusivo.

Las limitaciones más destacadas de nombra.en.red son las siguientes:

- Aplicabilidad muy limitada.
- No es posible realizar un análisis.

- Las expresiones sexistas y las propuestas alternativas no pueden ser modificadas por el usuario.

Themis [73]

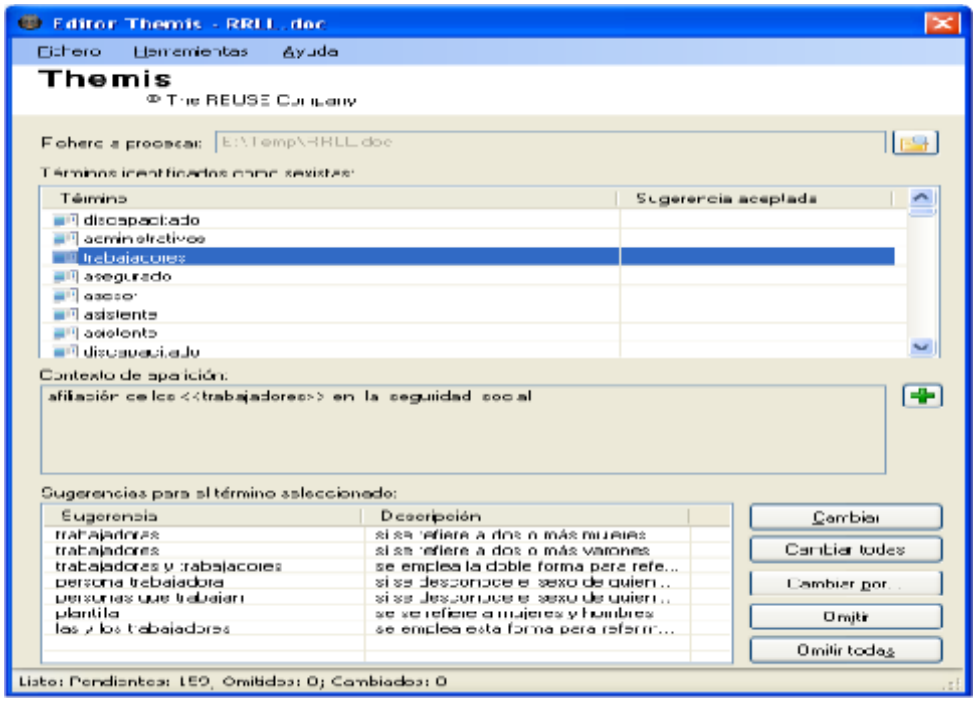


Figura 2.6: Interfaz de Themis

Themis (figura 2.6) es un corrector del lenguaje sexista. Su funcionamiento es simple, busca posibles términos sexistas en un documento y ofrece al usuario un editor mediante el cual poder sustituir estos términos por otros.

Presenta serias limitaciones:

- Software cerrado y de pago.
- Aplicabilidad muy limitada.
- El editor únicamente permite modificar los términos sexistas detectados en el documento (figura 2.6), no siendo posible editar el documento en sí.

- No es posible obtener información estadística del análisis.
- Las expresiones sexistas y las propuestas alternativas no pueden ser modificadas por el usuario.
- Los tipos de expresiones detectadas son fijos.

Capítulo 3

Análisis lingüístico basado en reglas

3.1. Introducción

Previamente a profundizar en el presente capítulo conviene aclarar a que nos estamos refiriendo al hablar de "análisis lingüístico basado en reglas".

La lingüística es un campo relativamente amplio que se ocupa del estudio científico de la estructura de las lenguas naturales así como del conocimiento que los hablantes poseen de ellas [4, 5, 6].

La lingüística puede dividirse en función de múltiples aspectos tales como el enfoque adoptado para su estudio, las estructuras lingüísticas a analizar, los temas de estudio u otros factores no lingüísticos. Veamos a continuación cuales serían las posibles divisiones en cada caso.

En función del *enfoque adoptado en su estudio*, podemos dividir el estudio de la lingüística en tres dicotomías [4]:

- **Lingüística teórica frente a aplicada**. La primera se refiere a la construcción de una teoría general de la estructura de la lengua y el sistema cognitivo que la hace posible. Busca describir la lengua en sí misma así como caracterizar el conocimiento implícito que los habitantes tienen de las mismas y como la adquieren. La lingüística aplicada o práctica se centra en estudiar los problemas sociales que tienen que ver con el lenguaje.
- **Lingüística sincrónica frente a diacrónica**. La lingüística sincrónica describe la lengua en un momento dado mientras que la diacrónica describe el desarrollo y cambios estructurales de la misma.
- **Microlingüística frente a macrolingüística**. Se diferencian en función de la amplitud del punto de vista tomado para su estudio. La microlingüística se centra en el estudio de la propia lengua obviando su función social, por contra, la macrolingüística abarca aspectos como la forma de adquisición, mecanismos psicológicos que subyacen en la producción y recepción del habla, función estética, etc.

Considerando los subcampos de la estructura analizados (tales como fonética, fonología, morfología, sintaxis, semántica, pragmática o análisis del discurso), el nivel de análisis

realizado (sonidos individuales, palabras, frases y discursos) y el método empleado, podemos realizar una nueva división centrada en el *estudio de las estructuras lingüísticas* [5].

Dependiendo de si se lleva a cabo un análisis de lengua hablada o escrita, diacrónico o sincrónico, descriptivo o prescriptivo, etc, podemos establecer una división que considera el *tema de estudio* empleado [4].

Por último, atendiendo a *otros factores no lingüísticos*, podemos llevar a cabo una división más de la lingüística. Algunos de los distintos tipos de lingüísticas en base a esta división serían [5]:

- **Biolingüística**: Realiza un estudio del lenguaje natural desde una perspectiva biológica y evolutiva.
- **Lingüística clínica**: Adopta un enfoque práctico para tratar con aspectos de patologías del lenguaje.
- **Lingüística computacional**: Lleva a cabo un estudio computacional de las implementaciones de las estructuras lingüísticas.
- **Desarrollo de la lingüística**: Analiza el desarrollo de las habilidades lingüísticas en los individuos.
- **Evolución del lenguaje**: Estudia la aparición y evolución del lenguaje en el ser humano siguiendo su linea evolutiva.
- **Geolingüística**: Encargada del análisis de hechos lingüísticos en función de su localización geográfica.
- **Neurolingüística**: Realiza un estudio de los mecanismos neuronales en el cerebro humano que controla la compresión, producción y adquisición del lenguaje.
- **Psicolingüística**: Trata con aspectos tales como la adquisición y utilización del lenguaje.
- **Sociolingüística**: Se interesa en como los aspectos sociales influyen en el uso de la lengua.

- **Estilística**: Comprende un estudio del uso artístico o estético del lenguaje.
- Otros campos tales como **tipología lingüística**, **lingüística histórica**, **lingüística aplicada**, **criptoanálisis**, **traducción**, **lingüística forense**, etc.

A la vista de estas divisiones se deduce que en función del enfoque tomado en los diversos aspectos a considerar, estaremos realizando un análisis lingüístico u otro, variando la aplicabilidad del mismo sustancialmente.

En el presente proyecto, al hablar de un "análisis lingüístico basado en reglas" nos referimos a un *estudio lingüístico guiado por reglas lingüísticas*. En el punto 3.2.2 estableceremos el tipo de análisis lingüístico realizado en función de la división establecida en este apartado.

3.2. Enfoque del análisis

3.2.1. Descripción del análisis

El análisis lingüístico dirigido por reglas planteado en el presente proyecto se realiza en función de una serie de **paquetes de reglas**. Cada paquete de reglas esta formado por una serie de reglas y conjuntos de términos que permiten analizar una problemática particular.

De este modo podríamos tener diferentes paquetes de reglas que nos permitirán realizar múltiples análisis sobre un mismo contenido. Por ejemplo, podemos desarrollar paquetes de reglas para realizar el análisis igualitario de un texto, el uso de términos racistas, la búsqueda de patologías del lenguaje, etc.

A su vez, cada paquete se puede enfocar al estudio de una única problemática o de varias. Siguiendo con el ejemplo anterior, sería posible construir un único paquete que podríamos denominar como "usos inapropiados del lenguaje" el cual analice tanto problemas de género como raciales.

Como hemos indicado, los paquetes de reglas están formados por conjuntos de términos y reglas. Los **conjuntos de términos** servirán para englobar todos aquellos términos

significativos para el análisis en cuestión, pudiendo existir varios en el mismo paquete. Así, podríamos tener un conjunto con términos potencialmente raciales o sexistas, un conjunto con términos que son frecuentemente usados de forma errónea por los hablantes no nativos de la propia lengua, etc.

Estos conjuntos nos permiten acotar de forma significativa el análisis, obteniendo una aproximación mucho más cercana a la problemática a analizar. De esta forma, para un análisis igualitario de un texto podemos contar con un conjunto de términos potencialmente exclusivos y así discernir cuando una expresión es potencialmente excluyente para la mujer.

Cada término dentro del conjunto de términos guardará información tanto del propio término en sí como sus géneros y números gramaticales. De este modo se podrá por un lado conocer una mayor información del término en cuestión (si es masculino, si carece de él, si es neutro, si está en plural, etc) y, por otro, agrupar bajo un mismo concepto las diversas variantes del mismo, permitiendo de este modo un estudio más realista.

Además de los conjuntos de términos se dispone de un completo **diccionario** gramatical que nos permitirá obtener la clase gramatical de una palabra dada o buscar palabras de una clase concreta. El diccionario es común a todos los paquetes de reglas y complementa la información de los conjuntos de términos. Haciendo uso del diccionario podemos buscar la ocurrencia de un término dado de un conjunto de términos y, posteriormente, comprobar si existe alguna palabra de una clase de palabras concreta próxima a él. También sería posible condicionar la aparición de un término a la pertenencia de este a una clase gramatical o incluso buscar términos de un conjunto de términos en las proximidades de una palabra de una clase gramatical dada. Igual que en los conjuntos, los términos del diccionario vienen acompañados de su género y número gramatical, opciones que pueden ser empleadas para una mayor precisión del análisis.

A pesar de todo lo dicho hasta ahora, los conjuntos de términos y el diccionario no servirían para mucho si no dispusiésemos de las reglas, verdadero motor del análisis. Las **reglas** son las que dotan de poder al análisis permitiéndonos realizar búsquedas lingüísticas complejas orientadas. Las reglas nos permitirán localizar los términos, analizar la clase gramatical de cada palabra, buscar ocurrencias próximas de términos dados en determinada forma gramatical, etc.

En síntesis, las reglas nos permiten buscar términos o palabras de un clase dada, conocer su variante gramatical y analizar la relación que guarda con el resto de palabras de su conjunto. Retomando el ejemplo de un análisis igualitario, las reglas nos permitirían localizar los posibles términos exclusivos y comprobar que se cumplen todos los requerimientos para considerar que, en efecto, tenemos una expresión potencialmente exclusiva.

Para un funcionamiento óptimo del análisis las reglas se pueden considerar independientes o dependientes de otras regla. Las reglas independientes solo dependerán de su resultado para ser consideradas afirmativas o negativas, en cambio una regla dependiente de otras reglas dependerá no solo de su propia evaluación, sino también del resultado de las reglas de las cuales depende. En el caso general podemos describir las reglas de las que depende una regla dada como condiciones de la misma, cumpliéndose la regla únicamente en caso de que todas las reglas de las que depende se cumplan.

Puede darse el caso de que las reglas dependientes de una regla (o las condiciones de la misma) sean contrapuestas, es decir, una sea contraria a la otra. En general esto constituiría en todo caso una negación puesto que nunca podrían satisfacerse las dos simultáneamente (una palabra no puede ser un sustantivo y a la vez no serlo) y en tal caso el resultado de la regla sería negativo. Sin embargo, esta situación no es tan obvia como pueda parecer en un primer momento ya que, si de por sí dos reglas pueden constituir una contradicción, la evaluación condicional de cada una de ellas hace que la condición no sea únicamente su dependencia directa si no esta y, a su vez, todas sus condiciones propias. Pongámonos en el caso de que no pueda aparecer una palabra dada y, en el caso de que aparezca, solo se permita que venga precedida de otra palabra concreta. Como vemos tenemos dos reglas, una en caso de que aparezca y otra en caso de que no aparezca las cuales nunca se podrán cumplir simultáneamente. En este caso el resultado real sería una disyunción entre las dos, es decir, o bien la palabra no aparece o bien aparece pero precedida de una palabra concreta. Para estos casos es necesario incluir una regla propia que permita la disyunción de las condiciones.

Las reglas se evalúan de forma secuencial, guardando de este modo especial relación el orden de estas con el resultado obtenido. Por este motivo es aconsejable establecer en primer lugar aquellas reglas que nos permitan limitar de una forma mucho más significativa el problema.

Las reglas raíz o reglas de las que nadie depende dentro del paquete de reglas, aunque

siguen este modelo secuencial de análisis se evaluarán siempre independientemente del resultado de la regla que le precede. Esto nos permite incluir en el paquete una serie de reglas independientes las cuales agrupan los diversos problemas a considerar. Las reglas raíz pueden utilizarse también para eliminar las disyunciones en caso de que estas nos resulten confusas. Reconsiderando el problema anterior de que un término podría no aparecer o aparecer acompañado de otro, podríamos optar por una solución distinta al uso de la disyunción que implicaría crear una regla raíz para cada una de las condiciones. De este modo tendríamos una regla en la cual sería condición que no apareciese el término y otra en la que sería condición que apareciese acompañado de una palabra dada. Esto puede ser útil en determinados problemas y puede ayudar a simplificar la lógica del árbol de reglas, sin embargo, introduce una duplicidad de reglas que conlleva un análisis menos eficiente al evaluar las reglas previas múltiples veces. Por tanto se desaconseja este tipo de enfoque para los análisis medianamente complejos.

Partiendo de esta estructura de paquetes, conjuntos de términos, diccionario gramatical y reglas, se realiza un análisis secuencial de las distintas frases que conforman un texto. Teniendo el texto dividido en frases realizamos un análisis de cada frase palabra a palabra, comprobando en cada caso si una serie de reglas se verifican para cada palabra en su contexto actual. Una vez finalizado el análisis de todas las frases se devolverá un informe de aquellos términos que han verificado alguna regla raíz. Este informe estará agregado por paquetes y contendrá la información necesaria para poder localizar el punto exacto de la aparición de cada ocurrencia concreta.

3.2.2. Tipo de análisis

Pasamos a establecer aquí el tipo de análisis realizado en función de la división establecida en el apartado 3.1. No debemos caer en el error de confundir el tipo de análisis con la aplicabilidad del mismo. Establecemos el tipo en base a la forma en la que se realiza el análisis, surgiendo la aplicabilidad del análisis del tipo del mismo.

En primer lugar, atendiendo al *enfoque adoptado en el estudio*, podemos considerar el análisis como:

- **Teórico**, puesto que realiza un estudio de las palabras analizando aspectos como su

aparición, su categoría gramatical y su relación con el resto de las palabras.

- **Sincrónico**, ya que analiza la estructura del documento en función de la morfología y sintaxis actual.
- **Microlingüista**, al analizar la estructura del contenido sin entrar en aspectos sociales del mismo.

Considerando el *estudio de las estructuras lingüísticas* realizado se puede clasificar el análisis como un análisis **sintáctico de frases**.

En base al *tema de estudio* podríamos considerarlo como un análisis **descriptivo de lengua escrita**.

La restante división centrada en *otros factores no lingüísticos* carece de significado de cara a determinar el tipo aunque no así de cara a la aplicabilidad.

De esta forma, se puede establecer de modo general el tipo de análisis lingüístico realizado como *un análisis descriptivo de la lengua escrita a través del estudio de la sintaxis de las frases de carácter microlingüista, sincrónico y teórico.*

3.2.3. Aplicabilidad del análisis

Anteriormente hacíamos un inciso para destacar que había que distinguir claramente entre el tipo y la aplicabilidad del análisis realizado. Pasamos a ver en este subapartado la aplicabilidad en los distintos tipos de análisis lingüístico en función de las características del mismo.

En función del *enfoque adoptado para el estudio* tenemos que su aplicabilidad es principalmente:

- **Práctica y en menor grado teórica**, puesto que su principal función es analizar problemas que tienen relación con el lenguaje. No obstante también puede ser empleado para el estudio teórico de la lingüística (frecuencia de aparición conjunta de palabras, estructura de las frases, etc) aunque de forma muy limitada.

- **Sincrónica y en menor grado diacrónica**, ya que su principal uso es la búsqueda de la ocurrencia de determinadas circunstancias en un documento sin analizar otros aspectos. Sin embargo, también podría ser usado como herramienta de ayuda en un análisis diacrónico (frecuencia de términos en documentos de distintas épocas, caída de términos en desuso, aparición de nuevos términos, etc) aunque al igual que con el caso de la lingüística teórica, de forma tremendamente limitada.

- **Microlingüista y Macrolingüista**, al poder ser empleado tanto para analizar las características de un determinado contenido como los aspectos psicológicos que subyacen en la recepción del contenido (uso de términos inapropiados, minería de opiniones, ...).

Atendiendo al *estudio de las estructuras lingüísticas* a realizar con el análisis, su principal aplicabilidad se encuentra en el análisis del **discurso del texto y de las frases**. También puede ser de utilidad en un análisis sintáctico de palabras aunque no es la herramienta más adecuada para esta tarea.

Puede considerarse de utilidad para casi cualquier *tema de análisis* ya sea análisis descriptivo (características del texto) o prescriptivo (minería de opiniones, ingeniería social), de lengua escrita (análisis de documentos) o hablada (eliminación de términos inapropiados en discursos), etc.

Por último y centrándonos en el resto de *factores no lingüísticos* nos encontramos con que el análisis puede tener aplicabilidad en multitud de ellos tales como la lingüística clínica, la lingüística computacional, el desarrollo de la lingüística, la psicolingüística, la sociolingüística, ...

3.2.4. Campos de aplicación

En el subapartado anterior indicábamos la aplicabilidad del tipo de análisis realizado a multitud de problemáticas distintas. No queremos con ello ofrecer la solución propuesta en el presente proyecto como una solución definitiva para multitud de casuísticas distintas si no, más bien, poner de manifiesto que puede ser empleada como una herramienta de ayuda para el análisis lingüístico de modo general.

En este subapartado introducimos como ejemplo algunos de los campos en los cuales consideramos que puede ser de utilidad.

- **Uso de lenguaje inapropiado** [3, 17]. Puede ser una excelente solución para problemas de este tipo. En la mayoría de los casos a tratar podremos encontrar un paquete de reglas que se adapte al problema que nos concierne ya sea igualdad de género, uso de lenguaje racista, lenguaje ofensivo hacia enfermos, etc. Veremos en el punto 3.2.4.1una solución propuesta para detectar el uso de lenguaje poco apropiado por cuestiones de género.

- **Detección de jerga profesional**. Muchos textos están plagados de determinados términos que son muy comunes para determinados colectivos pero relativamente extraños para el resto. Ejemplos de esto los tenemos en el campo médico, informático, científico, etc. Si bien esto es en muchas ocasiones inevitable, en otras se podría hacer uso de cualquier otro término más común sin con ello modificar el concepto al cual nos estamos refiriendo. De este modo se podría elaborar un paquete de reglas el cual, equipado con un conjunto de términos que podrían ser reemplazados por otros términos más conocidos, podría ayudar a paliar esta circunstancia.

- **Categorización de documentos**. Al hablar de la categorización de un documento nos referimos a asignarle una categoría que nos de una idea de a que se refiere. Las categorizaciones a realizar pueden ser múltiples. Podríamos categorizar por temáticas, por dificultad, por idioma, etc. Se podría elaborar una paquete de reglas para realizar una categorización concreta el cual, midiendo la frecuencia de aparición de una serie de palabras clave en determinados conjuntos de términos, pudiese indicarnos cual es la categoría más probable para un documento.

- **Minería de opiniones**. La minería de opiniones busca conocer las opiniones de los individuos acerca de cualquier tipo de cosa (objetos, lugares, personas, ...). Para ello se realizan principalmente tres tareas, a saber: detectar la subjetividad de la opinión, determinar la polaridad de la misma y establecer al intensidad de esta opinión [18]. La minería de opiniones es una parcela de conocimiento relativamente nueva pero que, muy posiblemente, revolucionará el futuro de internet. Sería posible realizar todas estas tareas en mayor o menor grado mediante la solución propuesta, detectar la subjetividad de un discurso en base al tipo de palabras empleadas y a

la proximidad de nombres propios [19], determinar la polaridad (positiva o negativa) en función del numero de términos favorables o desfavorables que aparezcan y diferenciar los términos entre más o menos favorables incluyéndolos en distintos paquetes previamente baremados.

- **Ingeniería social**. La ingeniería social es la práctica de obtener información confidencial a través de la manipulación de usuarios legítimos. En la mayoría de los casos, la ingeniería social se vale de la tendencia natural de la gente a reaccionar de manera predecible en ciertas situaciones [20]. Este campo podría valerse del análisis desde dos vertientes diferentes, la del atacante y la del atacado. Desde el punto de vista del atacante se puede emplear para localizar datos personales de la gente, descubrir frecuencias de apariciones de términos inusuales, etc. Desde el otro punto de vista, para evitar un posible uso de la ingeniería social sería posible generar un paquete que descubriese posibles usos de ingeniería social (phising, solicitud de datos personales bajo solicitudes demasiado cordiales, ...)

- **Detección de spam**. El spam o mensaje basura es aquel mensaje que un usuario recibe sin haberlo solicitado, sin desearlo o bien, de un remitente desconocido. Típicamente al hablar de spam solemos pensar en el correo electrónico, sin embargo, existen múltiples tecnologías que son víctimas del spam tales como grupos de noticias, motores de búsqueda, redes sociales, wikis, foros, blogs, etc [21]. Como vemos, el spam se puede presentar en distintos sitios y bajo distintas formas. Un paquete de reglas capaz de detectar aquel contenido potencial de ser spam (presencia de determinados términos, detección de solicitudes de datos personales, ...) sería de ayuda para afrontar esta situación.

- **Búsqueda de síntomas de enfermedades**. La mayoría de enfermedades mentales así como muchos trastornos de desarrollo presentan algunos síntomas en el lenguaje significativos (problemas en la estructuración, lenguaje que se va volviendo progresivamente más pobre, frases carentes de sentido, lenguaje confuso, uso de palabras peculiares, ...)[22, 23]. El tipo de análisis realizado podría emplearse en este campo para la detección de gran parte de estos síntomas.

3.2.4.1. Igualdad de género

En el presente punto estableceremos un paquete de reglas centrado en la igualdad de género. El paquete será capaz de localizar potenciales problemas de desigualdad de género, ofreciendo información acerca de las incidencias localizadas.

Para la elaboración del paquete debemos tomar varias decisiones:

1. Que términos nos indican una posible desigualdad de género.
2. Bajo que condiciones los términos anteriormente seleccionados constituyen un potencial problema.
3. Que condiciones especiales o excepciones pueden darse al caso general.
4. Dependencias entre las reglas.

1) Términos que indican una posible desigualdad de género

Los términos que presentan una posible desigualdad de género serán aquellos que se empleen en masculino para referirse a una persona independientemente de su género.

Esta terminología aparece en multitud de ocasiones a lo largo de todo tipo de documentos independientemente del carácter del propio documento (documentos científicos, artículos de opinión, prensa, ...), siendo una situación bastante común y muchas veces obviada. Ciertamente en muchas ocasiones esta exclusión de la mujer no se realiza de forma intencionada si no por cuestiones de brevedad, falta de concienciación, descuido, estilismo del texto, etc. Sin embargo, debemos ser conscientes de que lo que para nosotros no tiene por que constituir un problema puede resultar ofensivo para otra persona, la cual se puede sentir herida cuando esta no era nuestra intención.

De este modo nos encontramos con que comúnmente empleamos expresiones como 'los investigadores', 'el estudiante', 'el alumno', 'los ministros', etc, cuando podríamos emplear en su lugar expresiones como 'los investigadores y las investigadoras', 'el estudiante o la estudiante', ...

Para nosotros estos serán los términos que constituirán un potencial problema de

desigualdad de género y estarán englobados dentro del conjunto de términos 'Términos exclusivos' del paquete de reglas.

No obviamos que existen otros términos que también pueden constituir un potencial problema de género (utilizar el femenino de un término como algo negativo y el masculino como algo positivo, emplear determinados términos que tratan de forma despectiva a la mujer, etc). Para poder tratar con estos problemas es indispensable obtener un conjunto que contenga todos los casos posibles, labor que escapa al propósito de este proyecto.

2) Bajo qué condiciones los términos constituyen un potencial problema

Al seleccionar los términos potencialmente exclusivos ya hemos adelantado parte de esta cuestión puesto que explicábamos por que para nosotros estos serían los términos a considerar.

Siendo más concretos, consideraremos que un término de los pertenecientes a los 'Términos exclusivos' constituye un potencial problema de género si se cumple que:

- Se emplea el término únicamente en masculino para referirse tanto a hombres como a mujeres.
- Siendo el término neutro, este está calificado con un determinante masculino.

3) Excepciones al caso general

Si bien el caso general nos puede parecer completo para tratar con la mayoría de los problemas de genero, debemos ser conscientes de que existen múltiples circunstancias en las cuales este caso general no es válido.

Las excepciones son en la mayoría de los casos lógicas y aparecen frecuentemente durante el propio análisis. Como veremos en el punto 3.2.5, una de las principales ventajas del sistema desarrollado es precisamente esta, la posibilidad de refinar las condiciones del análisis durante el mismo.

Las excepciones se añaden como una condición negada, es decir, se deben comportar como condiciones que no deberían cumplirse y lo más lógico es indicar que esa condición **no** debería cumplirse, en otras palabras, negando la condición.

Tomaremos como excepciones las siguientes:

- El término está actuando como un adjetivo. El término cumple la función de adjetivo y no se esta refiriendo a una persona sino que califica a un sustantivo.
- Existe un nombre propio en las proximidades del término. Si aparece el término en masculino para referirse a una persona y tenemos un nombre propio cerca de este término, muy probablemente el término designe a esta persona.

4) Dependencias entre las reglas

Llegados a este punto solo nos queda por decidir el orden de evaluación de las reglas. En el punto 3.3.4, una vez introducido el formato de las reglas existentes veremos las reglas concretas que crearíamos así como sus valores en cada caso.

El formato del paquete de reglas será el siguiente:

- Si aparece un término contenido en el conjunto de 'Términos exclusivos' en masculino o en forma neutra.
  - Si el término no está actuando como un adjetivo.
  - Si no existe el término localizado en género femenino en un contexto próximo.
  - Si no existe un nombre propio en un contexto próximo.
  - Si el término está calificado como masculino en caso de ser neutro.

Tenemos una única regla general o regla raíz que a su vez debe verificar que se cumplen las cuatro reglas de las que dependen. Si se cumplen las cinco reglas (regla raíz más las cuatro reglas de las que depende) tenemos que se verifica la regla general, se cumplen sus dos condiciones (regla necesarias para que se considere una la incidencia) y no se cumple ninguna de sus excepciones (reglas que descartarían una incidencia). Cada término con el que se verifiquen estas cinco reglas constituirá un potencial problema de igualdad de género.

3.2.5. Ventajas

Las ventajas del análisis lingüístico basado en reglas que aquí planteamos son múltiples. De entre ellas las más significativas son las siguientes:

- **Múltiples campos de aplicación**. Como se vio en 3.2.4, existen varios campos de aplicación en los cuales este análisis podría resultar de utilidad. Los que se muestran en el apartado 3.2.4 solo son algunos de los que se nos ocurren, pudiéndose adaptar el análisis a una variedad mucho mayor de problemas.

- **Solución de uso general**. El análisis no se limita a trabajar con un único caso concreto si no que puede ser utilizado para resolver varios problemas distintos. En el presente proyecto hacemos uso del análisis para obtener una solución completa (la búsqueda de desigualdad de género), siendo esta solución una parte poco significativa del resultado final del proyecto, pudiéndose haber obtenido otras soluciones distintas fácilmente.

- **Ampliable**. Otro punto a favor de este tipo de análisis es que no está limitado, pudiendo ser ampliado fácilmente. En el presente proyecto solo se introducen una serie de tipos de reglas para la realización del análisis, sin embargo, la adición de nuevos tipos de reglas sería un proceso relativamente sencillo. Cada nuevo tipo de regla añadido abre un nuevo abanico de posibilidades al análisis (estudio de la morfología de las propias palabras, derivaciones de verbos, etc).

- **Reutilizable**. Las reglas de un paquete de análisis pueden extrapolarse con facilidad a otros paquetes de reglas al no tomar los valores de análisis de forma absoluta. Nunca se emplea un término dado para realizar un análisis si no más bien podríamos decir que el análisis se realiza en base a las características de una serie de términos relacionados. Al trabajar sobre la forma y no sobre el contenido podemos cambiar con tranquilidad el contenido manteniendo el mismo mecanismo de análisis.

- **Refinable**. Esta es una de las ventajas más significativas de este tipo de análisis, la posibilidad de refinar el mismo en función de los datos obtenidos hasta obtener un paquete de reglas que se adapte a la problemática concreta. Las condiciones no son rígidos fragmentos programados cuya modificación puede tener efectos colaterales y prolongarse su codificación durante largos periodos de tiempo. Con este tipo de

análisis, la modificación es instantánea y, debido al nulo acoplamiento entre las reglas, ausente de efectos colaterales. Además, la facilidad con la que se modifican los valores del análisis nos incita a probar distintas combinaciones, favoreciendo de este modo la obtención de mejores resultados.

- **Estructura común para muchas aplicaciones**. El análisis basado en reglas puede ser implementado en cualquier herramienta que necesite hacer uso de él pudiendo ser empleado en procesadores de texto, navegadores web, servidores de aplicaciones, etc.

3.2.6. Limitaciones

A pesar de las ventajas expuestas, también existen algunas limitaciones que se deberán considerar para, en su caso, optar por otra solución. Las principales limitaciones son:

- **Dificultad para trabajar con lenguaje mal escrito**. El análisis puede errar si el lenguaje a analizar presenta faltas de ortografía o los términos no se introducen correctamente. Esta no es una limitación particular del tipo de análisis aquí propuesto si no, en general, de cualquier análisis infoasistido. Esta deficiencia podría solventarse en parte si previamente al análisis se realiza una corrección ortográfica del texto a tratar y si se introduce con cuidado el conjunto de términos.
- **Dificultad para trabajar con lenguaje mal estructurado**. Estrechamente relacionado con lo anterior nos encontramos con que un lenguaje mal estructurado puede disminuir la precisión del análisis. Al igual que antes, esto no es una deficiencia del tipo de análisis, sino del análisis de textos asistido por ordenador en general, sin embargo, en nuestro caso esta problemática es más significativa puesto que muchas veces orientamos las reglas en función de la estructura esperada.
- **Las soluciones alcanzadas no son las más eficientes**. Normalmente, a medida que se alcanza una solución más genérica esta se vuelve menos eficiente que una solución específica para el mismo problema. Una solución construida a medida para tratar con un problema concreto será, en la mayoría de los casos, más eficiente que la que se alcanzaría con el análisis dirigido por reglas puesto que se podrían considerar

otros tipos de análisis y situaciones específicas (análisis por árbol, uso de expresiones regulares, tratamiento previo del texto para eliminar palabras vacías, ...).

- **Deficiencias en los términos del diccionario**. La precisión del análisis depende en gran parte de la calidad de los recursos con los que trabaja, entendiéndose estos como el diccionario gramatical y el conjunto de términos. Cualquier error presente en alguno de estos recursos mermará la precisión del análisis. Considerando que no existe en la actualidad ningún diccionario gramatical completo del español[1] hemos construido este haciendo uso de varios corpus gramaticales libres disponibles. La ausencia de cualquier término o la mal clasificación de alguno de ellos redundará en una probable mal detección de una incidencia.

- **Tipos de reglas limitados**. Los tipos de reglas (los cuales se verán en el punto 3.3.2) aunque buscan ser generales, se han incluido para tratar con problemas similares al de la igualdad de género puesto que se busca demostrar la utilidad del análisis dirigido por reglas en general y para un problema con unas características concretas en particular. Para problemas diferentes al tratado podría ser necesario añadir nuevos tipos de reglas que consideren un abanico mayor de posibilidades.

3.3. Funcionamiento del analizador

En este punto veremos cual será el funcionamiento del análisis dirigido por reglas que en este proyecto planteamos.

Veremos en primer lugar cual es el procedimiento general que guía el análisis a través del algoritmo del mismo (acompañado de su pertinente explicación).

Después veremos los tipos de reglas que estarán disponibles para el análisis y algunos ejemplos que mostrarán su uso.

Por último, se verán las reglas necesarias para el paquete definido en el punto 3.2.4.1.

[1] A pesar de una ardua búsqueda no nos ha sido posible encontrar ninguno.

3.3.1. Procedimiento general

El procedimiento general del análisis será el que se muestra en el algoritmo 3.1.

Algorithm 3.1 Procedimiento general del análisis

```
 1:Inicio Analizador
 2: Frases := Extraer_frases_del_texto(texto)
 3: Para cada paquete de reglas
 4:   Para cada frase en Frases
 5:     Palabras := Extraer_palabras_de_frase(frase)
 6:     Para cada palabra de Palabras
 7:       Para cada regla raíz en el paquete
 8:         Evaluación := Evaluar(Palabra, Frase, Paquete, Regla)
 9:         Si (Evaluación == Verdadero) entonces
10:           Resultado := Añadir(Palabra, Frase, Paquete)
11:         Fin Si
12:       Fin Para
13:     Fin Para
14:   Fin Para
15: Fin Para
16: Devolver Resultado
17:Fin Analizador
```

El analizador recibe el texto en formato de texto plano en el cual cada frase se separa haciendo uso de un punto.

Ya dentro del analizador tomamos el texto y separamos cada una de las frases del mismo haciendo uso de los puntos de final de frase (2). Una vez el texto está separado por frases, realizamos el análisis de las palabras (5, 6) de cada frase (4) en cada uno de los paquetes (3).

Con cada una de las palabras evaluaremos cada una de las reglas raíz[2] presentes en el paquete (7, 8) y, si el resultado de la evaluación es afirmativo (9), añadiremos la incidencia al resultado (10) del análisis.

Esta sería la estructura general del análisis que, como vemos, analiza todas las frases de un texto palabra a palabra con todas las reglas raíz de un paquete. El paso más

[2]Como ya hemos indicado anteriormente en este capítulo, denominamos como reglas raíz aquellas reglas de las que ninguna otra regla depende.

importante del pseudocódigo anterior sería el 8, en el cual se lleva acabo la evaluación de una regla dada para cada palabra de la frase. Veamos con mayor detalle como se realiza este proceso siguiendo el algoritmo 3.2.

Algorithm 3.2 Evaluación de una regla

```
 1:Inicio Evaluar(Palabra, Frase, Paquete, Regla)
 2: Condiciones := Obtener_dependencias(Regla, Paquete)
 3: Si (Palabra verifica Regla en Frase) entonces
 4:   Si (Regla tiene Condiciones) entonces
 5:     Si (Regla es Disyunción) entonces
 6:       Para cada condición en Condiciones
 7:         Evaluación := Evaluar(Palabra, Frase, Paquete, Condición)
 8:         Si (Evaluación == Verdadero) entonces
 9:           Devolver Verdadero
10:         Fin Si
11:       Fin Para
12:       Devolver Falso
13:     En otro caso
14:       Para cada condición en Condiciones
15:         Evaluación := Evaluar(Palabra, Frase, Paquete, Condición)
16:         Si (Evaluación == Falso) entonces
17:           Devolver Falso
18:         Fin Si
19:       Fin Para
20:       Devolver Verdadero
21:     Fin Si
22:   Fin Si
23:   Devolver Verdadero
24: Fin Si
25: Devolver Falso
26:Fin Evaluar_regla
```

El evaluador de reglas recibe la palabra actual (deberá verificar la regla), la frase en la que está contenida (contexto en el que se deberá cumplir la regla), la regla a analizar (condición a cumplir) y el paquete de reglas (guarda las dependencias entre las reglas y los conjuntos de términos).

El primer paso será obtener todas las condiciones de la regla (2), entendidas estas como las reglas de las cuales depende. Conociendo la regla y sus condiciones pasamos a realizar la evaluación como tal que pasa por comprobar que la regla se cumple para la

palabra indicada en el contexto actual (3).

Si la evaluación es negativa devolvemos el resultado (25) y, si es afirmativa, debemos considerar dos casos, que la regla tenga condiciones o no (4). Si no tiene condiciones la evaluación es afirmativa (23) y, si las tiene, dependerá tanto de la evaluación de las mismas como del tipo de la regla actual. Si la regla actual tiene condiciones y se trata de una disyunción (5) la evaluación será afirmativa si se cumple al menos una de sus condiciones (6-11), si no se cumple ninguna la evaluación será negativa (12). Si no se trata de una disyunción (13) la evaluación será afirmativa (20) siempre y cuando lo sean todas sus condiciones (14-19).

En resumen, la evaluación se encarga de comprobar que la regla actual se cumple para la palabra indicada en el contexto actual y que también lo hacen las condiciones de esta en caso de existir.

3.3.2. Tipos de reglas disponibles

Los tipos de regla establecen condiciones de búsqueda y comprobación gramatical. Cada uno de los tipos de regla cumple una función específica de búsqueda o comprobación cuya evaluación vendrá dada por el valor de sus elementos.

Se escriben en un formato comprensible (aunque rígido), pensado para poder ser entendido fácilmente.

Previamente a ver los tipos de reglas existentes introduciremos la búsqueda contextual y los valores permitidos para los distintos elementos.

3.3.2.1. Búsqueda contextual

La búsqueda contextual es empleada por la mayoría de las reglas para conocer el entorno de un término dado. Es un mecanismo simple pero que dota de gran potencial al análisis, pudiendo afirmar que, en caso de prescindir de él, la aplicabilidad del tipo de análisis sería prácticamente nula.

La búsqueda contextual consiste básicamente en realizar búsquedas no solo en la

posición actual (que también podría estar incluida), sino en las inmediaciones de esta. El contexto se configura en base a tres parámetros:

- **Tipo de contexto**: Si se debe buscar en el contexto anterior, posterior o anterior y posterior.
- **Tamaño del contexto**: Número de palabras a considerar en el contexto (pudiendo ser ilimitado).
- **Inclusión del centro**: Si la palabra central debe ser considerada o no.

En el punto siguiente veremos los valores permitidos para los parámetros en los elementos de las reglas.

3.3.2.2. Valores permitidos para los elementos

Existe un tipo de elemento de carácter especial y sumamente importante, el 'conjunto de términos'. Este elemento es de tipo cadena de caracteres y permite cualquier valor siempre y cuando se corresponda con el nombre de un conjunto de términos existente en el paquete de reglas.

Además de este tipo de elemento especial existen una serie de elementos comunes a la mayoría de las reglas recogidos todos ellos en la tabla 3.1. La interpretación de las columnas de la tabla es la siguiente:

- **Nombre**: Nombre del elemento en cuestión.
- **Tipo**: Tipo del elemento que puede ser lógico, entero o cadena de caracteres.
- **Valores permitidos**: Conjunto de valores permisibles para el elemento.
- **Formato**: Formato del valor del elemento en la regla.
- **Interpretación**: Como se debe interpretar el valor del elemento.

Nombre	Tipo	Valores permitidos	Formato	Interpretación
Negación	Lógico	Verdadero	'no '	Negado
		Falso	''	No negado
Variante gramatical	Cadena	Masculino	'masculino'	Masculino
		Femenino	'femenino'	Femenino
		Neutro	'neutro'	Neutro
		Singular	'singular'	Singular
		Plural	'plural'	Plural
		Masculino singular	'masculino singular'	Masculino singular
		Femenino singular	'femenino singular'	Femenino singular
		Neutro singular	'neutro singular'	Neutro singular
		Masculino plural	'masculino plural'	Masculino plural
		Femenino plural	'femenino plural'	Femenino plural
		Neutro plural	'neutro plural'	Neutro plural
		Cualquiera	'de cualquier tipo'	Cualquiera
Clase de palabras	Cadena	Sustantivo	'un sustantivo'	Sustantivo
		Adjetivo	'un adjetivo'	Adjetivo
		Preposición	'una preposición'	Preposición
		Conjunción	'una conjunción'	Conjunción
		Nombre propio	'un nombre propio'	Nombre propio
		Determinante	'un determinante'	Determinante
		Palabra vacía	'una palabra vacía'	Palabra vacía
Tipo de contexto	Entero	0	''	Completo
		1	'anterior '	Anterior
		2	'posterior '	Posterior
Tamaño del contexto	Entero	-1	'infinitas'	Ilimitado
		0	'0'	Cero palabras
		.	.	.
		.	.	.
		20	'20'	Veinte palabras
Inclusión del centro	Lógico	Verdadero	''	Inclusión
		Falso	'no '	Exclusión

Cuadro 3.1: Valores permitidos

3.3.2.3. Tipo 1: Disyunción

- **Función**: La disyunción es un tipo especial de regla cuyo resultado depende únicamente del resultado de sus dependencias, cumpliéndose cuando lo hace al menos uno de ellas. En caso de que la disyunción no tenga ninguna dependencia se considerará

que se cumple.

- **Formato**: *si se cumple al menos una subregla.*
- **Elementos**: Es el único tipo de regla que no tiene elementos al depender única y exclusivamente del resultado de sus dependencias.

3.3.2.4. Tipo 2: Palabra en conjunto de términos

- **Función**: Comprueba si una palabra está contenida (o no) dentro de un conjunto de términos del paquete. Se permite especificar la variante gramatical del término.
- **Formato**: *si el término #negación#está contenido en '#conjunto de términos#' siendo #variante gramatical#.*
- **Elementos**: Cuenta con tres elementos que son:

Nombre	Descripción
negación	Elemento en el conjunto de términos o no
conjunto de términos	Nombre del conjunto de términos en el cual se buscará el término
variante gramatical	Variante gramatical del término

Cuadro 3.2: Elementos del tipo de regla 'Palabra en conjunto de términos'

3.3.2.5. Tipo 3: Existe palabra de conjunto de términos en un contexto

- **Función**: Comprueba si en un contexto dado existe una palabra que está contenida (o no) dentro de un conjunto de términos del paquete. Se permite especificar la variante gramatical del término.
- **Formato**: *si #negación#existe un término contenido en '#conjunto de términos#' siendo #variante gramatical# en el contexto #tipo de contexto#de #tamaño del contexto# palabras #inclusión del centro#considerando el centro.*
- **Elementos**: Cuenta con seis elementos que son:

Nombre	Descripción
negación	Elemento en el conjunto de términos o no
conjunto de términos	Nombre del conjunto de términos en el cual se buscará el término
variante gramatical	Variante gramatical del término
tipo de contexto	Tipo de contexto en el que buscar
tamaño del contexto	Tamaño del contexto
Inclusión del centro	Considerar la posición central en el contexto o no

Cuadro 3.3: Elementos del tipo de regla 'Existe palabra de conjunto de términos en un contexto'

3.3.2.6. Tipo 4: Existe variante gramatical de palabra de un conjunto de términos en un contexto

- **Función**: Comprueba si en un contexto dado existe (o no) una palabra perteneciente a un conjunto de términos del paquete en una determinada variante gramatical. Se permite especificar la variante gramatical del término.
- **Formato**: *si #negación#existe el término actual en #variante gramatical# contenido en '#conjunto de términos#' en el contexto #tipo de contexto#de #tamaño del contexto# palabras #inclusión del centro#considerando el centro.*
- **Elementos**: Cuenta con seis elementos que son:

Nombre	Descripción
negación	Elemento en el conjunto de términos o no
conjunto de términos	Nombre del conjunto de términos en el cual se buscará el término
variante gramatical	Variante gramatical del término
tipo de contexto	Tipo de contexto en el que buscar
tamaño del contexto	Tamaño del contexto
Inclusión del centro	Considerar la posición central en el contexto o no

Cuadro 3.4: Elementos del tipo de regla 'Existe variante gramatical de palabra de un conjunto de términos en un contexto'

3.3.2.7. Tipo 5: Palabra de una clase de palabras

- **Función**: Comprueba si una palabra es (o no) de una clase gramatical concreta. Se permite especificar la variante gramatical del término.

- **Formato**: *si el término #negación#es #clase de palabras# #variante gramatical#.*
- **Elementos**: Cuenta con tres elementos que son:

Nombre	Descripción
negación	Elemento del tipo indicado o no
clase de palabras	Clase de palabras deseada
variante gramatical	Variante gramatical del término

Cuadro 3.5: Elementos del tipo de regla 'Palabra de una clase de palabras'

3.3.2.8. Tipo 6: Existe palabra de una clase de palabras en un contexto

- **Función**: Comprueba si en un contexto dado existe (o no) una palabra de una clase gramatical concreta. Se permite especificar la variante gramatical del término.
- **Formato**: *si #negación#existe #clase de palabras# #variante gramatical# en el contexto #tipo de contexto#de #tamaño del contexto# palabras #inclusión del centro#considerando el centro.*
- **Elementos**: Cuenta con seis elementos que son:

Nombre	Descripción
negación	Existencia o no de un término de la clase indicada
clase de palabras	Clase de palabras deseada
variante gramatical	Variante gramatical del término
tipo de contexto	Tipo de contexto en el que buscar
tamaño del contexto	Tamaño del contexto
Inclusión del centro	Considerar la posición central en el contexto o no

Cuadro 3.6: Elementos del tipo de regla 'Existe palabra de una clase de palabras en un contexto'

3.3.2.9. Tipo 7: Existe variante gramatical de palabra de una clase de palabras en un contexto

- **Función**: Comprueba si en un contexto dado existe (o no) la variante gramatical de una palabra de una clase gramatical concreta. Se permite especificar la variante gramatical del término.

- **Formato**: *si #negación#existe el término actual siendo #clase de palabras# #variante gramatical# en el contexto #tipo de contexto#de #tamaño del contexto# palabras #inclusión del centro#considerando el centro.*

- **Elementos**: Cuenta con seis elementos que son:

Nombre	Descripción
negación	Existencia o no de un término de la clase indicada
clase de palabras	Clase de palabras deseada
variante gramatical	Variante gramatical del término
tipo de contexto	Tipo de contexto en el que buscar
tamaño del contexto	Tamaño del contexto
Inclusión del centro	Considerar la posición central en el contexto o no

Cuadro 3.7: Elementos del tipo de regla 'Existe variante gramatical de palabra de una clase de palabras en un contexto'

3.3.3. Algunos ejemplos

3.3.3.1. Comprobación de la adjetivación

Existen sustantivos que pueden actuar en determinados casos como adjetivos (bonito, estudiante, ...). En estos casos si preguntamos por la categoría gramatical de una palabra de este tipo de forma aislada tenemos que podría ser tanto un adjetivo como un sustantivo. Una posible forma de detectar algunos de estos casos podría ser comprobar si la palabra es un posible adjetivo y, en tal caso, comprobar si existe un sustantivo próximo a ella. Si el resultado de ambas reglas es afirmativo es muy probable que la palabra esté actuando como un adjetivo.

Las reglas que introduciríamos serían:

- Si el término es un adjetivo de cualquier tipo (Tipo 5)

    - Si existe un sustantivo de cualquier tipo en el contexto de 2 palabras no considerando el centro (Tipo 6)

La situación se complica si lo que queremos es descartar la adjetivación, en este caso tendríamos dos condiciones, que la palabra no sea un adjetivo o que siéndolo, no este actuando como tal, es decir, no tenga un sustantivo próximo a ella. En un primer momento podría pensarse en introducir una serie de reglas como las siguientes:

- Si el término no es un adjetivo de cualquier tipo (Tipo 5)
- Si el término es un adjetivo de cualquier tipo (Tipo 5)
    - Si no existe un sustantivo de cualquier tipo en el contexto de 2 palabras no considerando el centro (Tipo 6)

Lo anterior parece obvio pero llevado a la práctica no funcionaría salvo en el caso de que se tratasen de reglas raíz. Para ver el motivo por el cual esto no es correcto analicemos los posibles casos considerando que las reglas se evalúan de forma secuencial.

1. El término no es un adjetivo y no existe un sustantivo cerca. La primera regla se verificaría pero la segunda no.
2. El término no es un adjetivo y existe un sustantivo cerca. Igual que en el caso anterior.
3. El término es un adjetivo y no existe un sustantivo cerca. La primera regla no se verifica y por tanto la segunda no llega a evaluarse.
4. El término es un adjetivo y existe un sustantivo cerca. Igual que en el caso anterior.

Como vemos, no existe ninguna combinación que permita que se verifiquen las dos reglas de más bajo nivel simultáneamente y por tanto no habría forma de descartar la adjetivación de este modo.

La solución más sencilla pasaría por emplear la disyunción para indicar que, de las dos formas de descartar la adjetivación que hemos planteado, con que se cumpla una sería suficiente.

Incluyendo la disyunción las reglas quedarían de la siguiente forma:

- Si se cumple al menos una subregla (Tipo 1)
  - Si el término no es un adjetivo de cualquier tipo (Tipo 5)
  - Si el término es un adjetivo de cualquier tipo (Tipo 5)
    - Si no existe un sustantivo de cualquier tipo en el contexto de 2 palabras no considerando el centro (Tipo 6)

3.3.3.2. Comprobar si una palabra está en masculino

La comprobación de si una palabra dada de un conjunto de términos es masculina es más compleja de lo que podría parecer en un primer momento. En la mayoría de los casos nos bastará con una sola regla:

- Si el término está contenido en 'Nombre de un paquete' siendo masculino (Tipo 2)

Si bien es cierto que en un gran número de casos lo anterior funcionará correctamente podría suceder que un término dado fuese neutro.

Póngase como ejemplo el término 'estudiante' el cual consideraremos está contenido en el conjunto de términos que buscamos. ¿Qué debería el analizador devolver al preguntar si el término está en masculino? ¿Y si preguntamos si el término está en femenino? En ambos casos la respuesta debería ser afirmativa puesto que el término puede ser de ambos tipos y no tenemos ninguna información más.

Comprobar si el término 'estudiante' está en masculino pasaría por comprobar si existe algún determinante que lo califique como masculino ('el estudiante', 'un estudiante', ...). Así, asegurarnos de que muy posiblemente un término está en masculino pasaría por incluir una serie de reglas como las siguientes:

- Si el término está contenido en 'Nombre de un paquete' siendo masculino (Tipo 2)
  - Si se cumple al menos una subregla (Tipo 1)
    - Si el término no está contenido en 'Nombre de un paquete' siendo neutro (Tipo 2)

- Si el término está contenido en 'Nombre de un paquete' siendo neutro (Tipo 2)
  - Si existe un determinante masculino en el contexto anterior de 2 palabras no considerando el centro (Tipo 6)

3.3.4. Igualdad

Llegados a este punto estamos en condiciones de especificar cual será el paquete de reglas para realizar un análisis lingüístico basado en reglas para detectar la desigualdad de género.

En el punto 3.2.4.1 introdujimos cuales serían los términos a tratar, las condiciones y las excepciones, siendo las pautas del análisis para detectar potenciales casos de desigualdad de género las siguientes:

- Si aparece un término contenido en el conjunto de 'Términos exclusivos' en masculino o en forma neutra.
  - Si el término no está actuando como un adjetivo.
  - Si no existe el término localizado en género femenino en un contexto próximo.
  - Si no existe un nombre propio en un contexto próximo.
  - Si el término está calificado como masculino en caso de ser neutro.

En base a los tipos de reglas disponibles, una posible solución que realice este análisis sería la que sigue:

- Si el término está contenido en 'Términos exclusivos' siendo masculino (Tipo 2)
  - Si se cumple al menos una subregla (Tipo 1)
    - Si el término no es un adjetivo de cualquier tipo (Tipo 5)
    - Si el término es un adjetivo de cualquier tipo (Tipo 5)
      - Si no existe un sustantivo de cualquier tipo en el contexto de 2 palabras no considerando el centro (Tipo 6)

- Si no existe el término actual en femenino contenido en 'Términos exclusivos' en el contexto de 4 palabras no considerando el centro (Tipo 4)
- Si no existe un nombre propio de cualquier tipo en el contexto de 3 palabras no considerando el centro (Tipo 6)
- Si se cumple al menos una subregla (Tipo 1)
  - Si el término no está contenido en 'Términos exclusivos' siendo neutro (Tipo 2)
  - Si el término está contenido en 'Términos exclusivos' siendo neutro (Tipo 2)
    - Si existe un determinante masculino en el contexto anterior de 2 palabras no considerando el centro (Tipo 6)

Como vemos, el formato dado a las reglas permite que el modelado del análisis sea natural, pudiendo entender fácilmente el objetivo de un paquete con solo leer las reglas que lo componen. Los paquetes de reglas se convierten de este modo en una especie de enunciados de análisis, con un lenguaje y estructura muy próximos a los que emplearíamos para formular el problema.

El paquete de reglas 'Igualdad', definido en este punto, se incorporará a la herramienta desarrollada en este proyecto a fin de probar la validez de la misma.

Capítulo 4

Framework de análisis

4.1. Arquitecturas basada en componentes

4.1.1. Introducción

Tradicionalmente, el desarrollo de cualquier solución software suele presentar una serie de problemas que se repiten una y otra vez independientemente de la naturaleza del problema a tratar y que se acentúan a medida que avanza el desarrollo. Algunas de las problemáticas recurrentes son las siguientes:

- **Dificultad de reutilización**. Idealmente los problemas solo se deberían resolver una vez. Si hemos implementado una funcionalidad concreta para una solución dada deberíamos poder añadir esta funcionalidad a cualquier solución similar con relativa facilidad. En la práctica nos encontramos con que la reutilización es, en la mayoría de los casos, sumamente costosa, llegando al punto de que en muchas ocasiones es más práctico reimplementar que reutilizar.

- **Complejidad creciente**. El funcionamiento del código se va volviendo más complejo y su comportamiento más extraño a medida que se añaden nuevas funcionalidades. Su diseño se vuelve confuso, se empiezan a multiplicar los fallos, se acaban tomando muchas soluciones de compromiso y, en definitiva, se va produciendo un progresivo deterioro del producto que, en muchos casos, obliga a desechar todo lo realizado hasta el momento y a replantear por completo el proyecto para poder continuar.

- **Aumento progresivo de los fallos**. El software no se estropea. Por su propia naturaleza es imposible que su funcionamiento se resienta con el paso del tiempo puesto que la lógica del mismo no varía. De forma idílica podría pensarse en base a lo anterior que un producto software deberá seguir siendo siempre igual de estable tras sucesivas modificaciones si estas solo añaden nuevas funcionalidades y se realizan bajo unos parámetros mínimos de calidad. En la práctica esta situación no se cumple y nos encontramos que cada cambio realizado suele incrementar considerablemente el numero de fallos en el producto. La tasa de fallos con el transcurrir del tiempo sería similar a la mostrada en la figura 4.1.

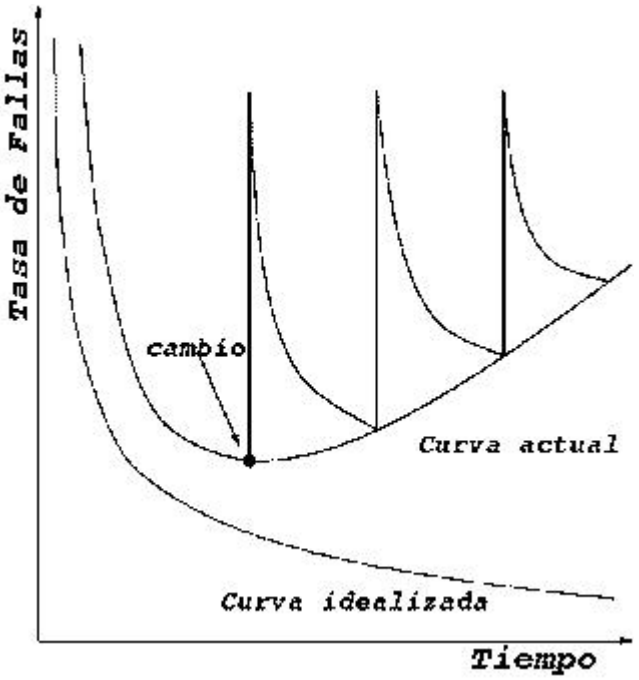


Figura 4.1: Curva de fallos del software

Si nos paramos a analizar los problemas expuestos veremos que en la base de todos radica un denominador común, el alto acoplamiento y la falta de cohesión funcional. A medida que introducimos cambios en la solución inicial debemos adaptarla para tratar con circunstancias no previstas inicialmente. Progresivamente, las fronteras entre las diversas funcionalidades se van difuminando, el flujo del código se vuelve cada vez más difícil de seguir, comienzan a aparecer objetos gigantes que solucionan múltiples cosas, los efectos colaterales se multiplican y, en definitiva, disminuye la cohesión y aumenta de manera exponencial el acoplamiento.

Sin embargo, si dirigimos la atención hacia otras áreas de ingeniería nos encontraremos con multitud de problemas de naturaleza sumamente compleja que son completados dentro del plazo previsto y sin grandes contratiempos, ambiciosos proyectos formados por cientos de elementos desarrollados a su vez por miles de personas que desconocen el trabajo del resto de las parte e incluso, en muchos casos, el resultado final esperado y que finalizan de forma satisfactoria. En el extremo opuesto nos encontramos con un gran porcentaje de proyectos software que sufren continuos retrasos, que ralentizan la salida al mercado de productos y que, en resumen, resultan progresivamente más difíciles de

estimar a medida que aumenta la complejidad.

Llegados a este punto, cabe plantearse la pregunta de por que los problemas complejos son relativamente fáciles de estimar en un caso y no en otro, si en ambos casos se abordan bajo un enfoque sistemático y disciplinado. La respuesta, en la mayoría de las ocasiones, pasa por la división funcional realizada, aspecto fundamental a considerar en el primer caso y no abordado con la suficiente relevancia en el segundo.

La idea general es que es más probable que una pieza cumpla correctamente con su función, si establecemos y delimitamos de forma clara cual será esta función. Si estructuramos el diseño del sistema de modo que los componentes funcionales y lógicos queden agrupados en una única unidad, estos serán fácilmente reutilizables e inmunes a los fallos ante la introducción de una nueva funcionalidad. Si además adaptamos los componentes para acceder y ser accedidos por otros a través de las interfaces públicas de los mismos permitiremos clarificar el flujo del programa y minimizar el acoplamiento entre las distintas partes.

En esto radica una arquitectura basada en componentes la cual es, en esencia, un enfoque de construir el software de la forma anteriormente descrita.

Una arquitectura basada en componentes va más allá del paradigma de la programación orientada a objetos, acercándose más a asuntos como qué funcionalidad proporciona cada componente, cómo se comunican estos componentes entre ellos, cómo comparten la información, cómo se integran entre si y cómo funcionan todos de forma cooperativa en un mismo ecosistema.

4.1.2. Beneficios

Una arquitectura basada en componentes nos proporciona grandes beneficios de entre los que destacamos:

- **Efectiva división del trabajo**. Distintos individuos pueden ocuparse de la realización de distintos componentes sin necesidad de conocer poco más que la información relativa al componente a desarrollar.
- **Alto nivel de abstracción**. Podemos centrarnos en el resultado esperado por la

solución final sin detenernos a pensar cual es el funcionamiento concreto de un componente específico. Un caso claro es cuando empleamos un Toolkit gráfico para desarrollar una interfaz, nosotros no nos preocupamos por el funcionamiento interno de un botón o de una etiqueta, simplemente los utilizamos.

- **Reutilización inmediata de los componentes**. Cada componente cumple una función dada independientemente de que se este empleando en un programa o en otro. La reutilización de los componentes es inmediata.

- **Facilidad de mantenimiento**. Al disponer de cada funcionalidad agrupada en un único componente, tareas como la corrección de errores son mucho más sencillas. Podemos delimitar con rapidez cual es el código que falla y generar cambios sin afectar a los demás componentes.

- **Alta cohesión y bajo acoplamiento**. Los componentes son, por su naturaleza, altamente cohesivos y mínimamente acoplados. Una arquitectura que se construye con elementos de este tipo heredará las mismas propiedades.

- **Facilidad de instalación**. Un componente cumple una funcionalidad y solo él sabe como la hace. Gracias a esto, instalar un nuevo componente en una aplicación se convierte en un proceso trivial.

- **Facilidad de desarrollo**. Al desarrollar un componente únicamente debemos considerar la función que va a cumplir por lo que podemos centrarnos completamente en su desarrollo obviando el funcionamiento interno del resto de componentes con los que interactuará.

4.1.3. Componentes

4.1.3.1. Características

Los componentes o módulos tienen una serie de características que los hacen diferentes a otras unidades software. En general, podemos considerar tres propiedades básicas que un componente debería de cumplir [25]:

- **Autocontenidos**. Un componente es una unidad lógica y debería poder ser movido, instalado o desinstalado como una unidad independiente.
- **Altamente cohesivos**. Un componente debería cumplir idealmente con una única función.
- **Bajamente acoplados**. La implementación de un componente no debería depender en ningún caso de la implementación de otros componentes.

Es vital para soportar estas tres propiedades que cada componente tenga una interfaz bien definida mediante la cual otros componentes puedan hacer uso de la funcionalidad que este proporciona.

A parte de estas propiedades podemos citar otras características comunes a los componentes tales como:

- Son capaces de indicar cuales son sus dependencias así como la versión concreta de cada una de ellas. De esta forma podemos olvidarnos del tedioso problema de tratar con dependencias y solo preocuparnos de cual es la funcionalidad o funcionalidades que queremos incluir.
- Definen su propio espacio de trabajo por lo que evitan cualquier tipo de conflicto con otros componentes.
- Pueden coexistir dos versiones de un mismo componente en un mismo software.

4.1.4. Arquitecturas de componentes existentes

De entre las principales arquitecturas de componentes existentes en la actualidad podemos destacar cinco, Maven, Ivy, OSGi, Eclipse Plug-Ins y JSR 277. Veamos a continuación las características principales de cada una de ellas.

OSGi [26]

OSGi es la principal arquitectura de componentes existente en la actualidad. Define una forma de crear componentes que cumplan con las tres propiedades básicas expuestas

en 4.1.3.1. En una arquitectura OSGi, cada componente es un paquete que incorpora un conjunto de metadatos que añaden información a este.

Destaca especialmente por la gestión que hace de los componentes en tiempo de ejecución, aplicando un enfoque en el cual cada componentes dispone de su propio entorno de ejecución. Al trabajar de este modo se soluciona de forma instantánea cualquier conflicto que pudiese existir entre los componentes.

Figura 4.2: Logotipo de la OSGi Alliance

Eclipse Plug-Ins System [27]

Un Plug-In es similar a un componente OSGi, de hecho el propio sistema de plug-in está construido haciendo uso de la arquitectura OSGi. La diferencia entre ambos es más histórica que real y radica esencialmente en los metadatos contenidos en el componente, los cuales difieren levemente. En las sucesivas versiones del sistema de plug-ins de Eclipse estas diferencias se han solventado e incluso existe la posibilidad de construir un plug-in declarándolo directamente como un componente OSGi.

Figura 4.3: Logotipo de Eclipse

Maven e Ivy

Ambas herramientas tienen características similares a las de una arquitectura de componentes pero están más enfocadas a la gestión de la construcción de los componentes que

a la ejecución de los mismos. Ambas se basan en la adición de metadatos a los paquetes a utilizar, pudiendo de este modo gestionar de forma eficiente las dependencias.

La principal deficiencia de ambas es que no realizan ninguna gestión de los componentes en tiempo de ejecución por lo que no pueden manejar la aparición de conflictos entre los componentes.

JSR 277

JSR 277 define la especificación para la construcción de una arquitectura de componentes en lenguaje Java de características similares a la arquitectura OSGi. Su principal limitación es que es relativamente reciente por lo que aún no existen tantos componentes disponibles como para una arquitectura OSGi.

4.1.5. OSGi

4.1.5.1. Plataforma de servicios OSGi

La plataforma de servicios OSGi es una especificación de la OSGi Alliance para diseñar una arquitectura orientada a componentes que verifique las tres propiedades básicas expuestas en 4.1.3.1. Aunque a priori pudiese parecer que una simple especificación de diseño no puede cambiar gran cosa en el panorama del desarrollo del software, la realidad demuestra que, si estructuramos un sistema siguiendo esta especificación, conseguiremos grandes beneficios.

Pasamos a continuación a ver las características más relevantes de la plataforma OSGi y que le confieren un mayor valor añadido que al resto de arquitecturas de componentes existentes.

- **Uso de metadatos en los paquetes**. Realmente, un componente OSGi no es más que un paquete JAR[1] al que se adicionan una serie de metadatos en su manifiesto (MANIFEST.MF). Los metadatos consisten en:

[1] Contenedor de código estandarizado para el lenguaje de programación Java.

- *El nombre del componente*. Usado para determinar de forma única un componente concreto.
- *La versión del componente*. Empleado para poder trabajar con distintas versiones de un mismo componente en una misma aplicación.
- *Listado de importaciones y exportaciones*. Necesario para conocer lo que el componente emplea y lo que proporciona.
- *Otros metadatos opcionales*.
  - Versión mínima de la máquina virtual Java.
  - Información del creador del paquete.
  - Licencia.
  - Dirección de contacto con el desarrollador.
  - Descripción del paquete.
  - ...

Un ejemplo de metadatos de un componente sería el siguiente:

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Linguo Rulebase
Bundle-SymbolicName: linguo.rulebase
Bundle-Version: 1.0.0
Bundle-Activator: linguo.rulebase.Activator
Bundle-Vendor: linguo
Require-Bundle: org.eclipse.core.runtime,
 linguo.libs.hypersql;bundle-version="1.0.0",
 linguo.utils;bundle-version="1.0.0"
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
Bundle-ActivationPolicy: lazy
Export-Package: linguo.rulebase,
 linguo.rulebase.db,
 linguo.rulebase.model,
 linguo.rulebase.utils
```

- **Arquitectura de componentes en forma de grafo**. Una de sus principales virtudes es que las dependencias no tiene por que estructurarse en formato de árbol si no que pueden constituir un grafo tan complejo como necesitemos. De esta forma un componente puede indicar como dependencia cualquier componente que desee y a su vez puede ser usado como dependencia en cualquier otro componente. La única excepción es que no se permite el uso de dependencias circulares, es decir, dependencias reciprocas entre paquetes (A depende de B y a su vez B depende de A).

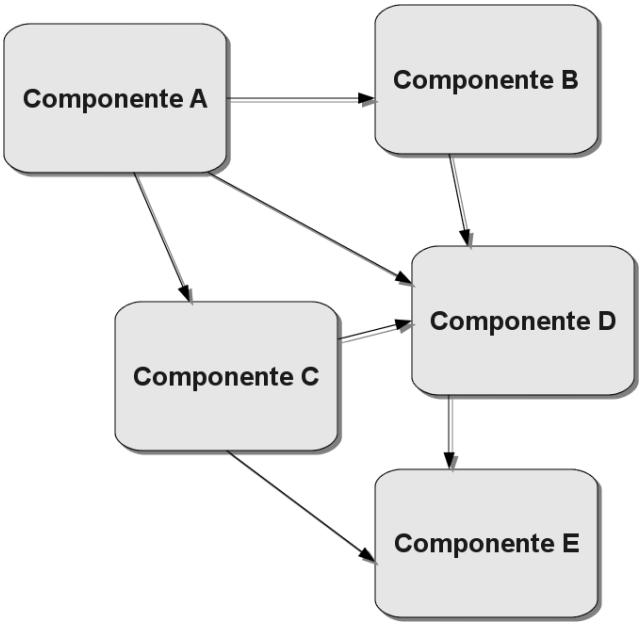


Figura 4.4: Dependencias en forma de grafo

- **Exportación selectiva**. Cada componente puede definir de forma explícita que parte del mismo estará visible desde el exterior. De esta forma podemos ocultar aquellas partes del mismo relativas a su funcionamiento externo y dejar únicamente visible lo necesario para hacer uso de él, cumpliendo así de forma efectiva con el principio de ocultación de información.

- **Uso de múltiples versiones de forma simultanea**. Al disponer de un sistema de dependencias en forma de grafo podemos solucionar problemáticas comunes en cualquier esquema de dependencias en forma de árbol como la imposibilidad de trabajar en una misma aplicación con diferentes versiones de una misma librería.

Así, si tenemos un componente que necesita emplear la versión 3.4 de una librería y a su vez, otro componente necesita la versión 3.2, esto no constituye ningún problema puesto que cada uno puede emplear su propia versión de la librería que será cargada en su propio espacio de trabajo y no entrará en conflicto con las dependencias del resto de componentes con los que interactúa. Esta problemática se verá con mayor detalle en el punto 4.2.2.3.

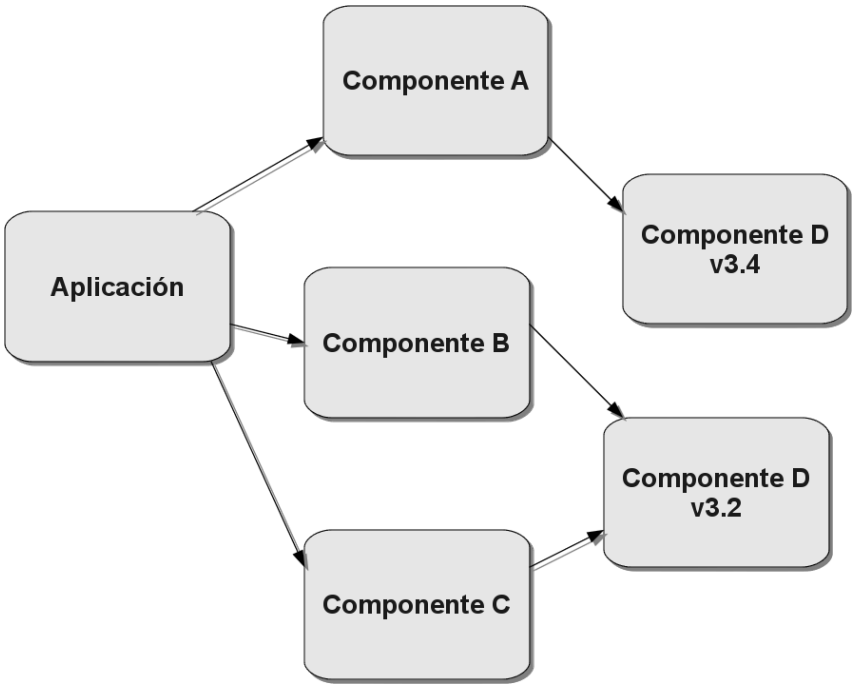


Figura 4.5: Uso de múltiples versiones

- **Carga de componentes dinámica**. Un componente no necesita ser cargado en memoria hasta que no es necesario utilizarlo. Esto proporciona ventajas significativas:
  - Podemos limitar el consumo de recursos de la aplicación puesto que una funcionalidad no se activa hasta que no se necesita.
  - Podemos modificar en tiempo de ejecución la versión de un determinado componente y, en consecuencia, de una determinada funcionalidad.
  - Podemos liberar los componentes una vez han dejado de ser útiles por lo que podemos optimizar aún más el consumo de recursos de la aplicación.

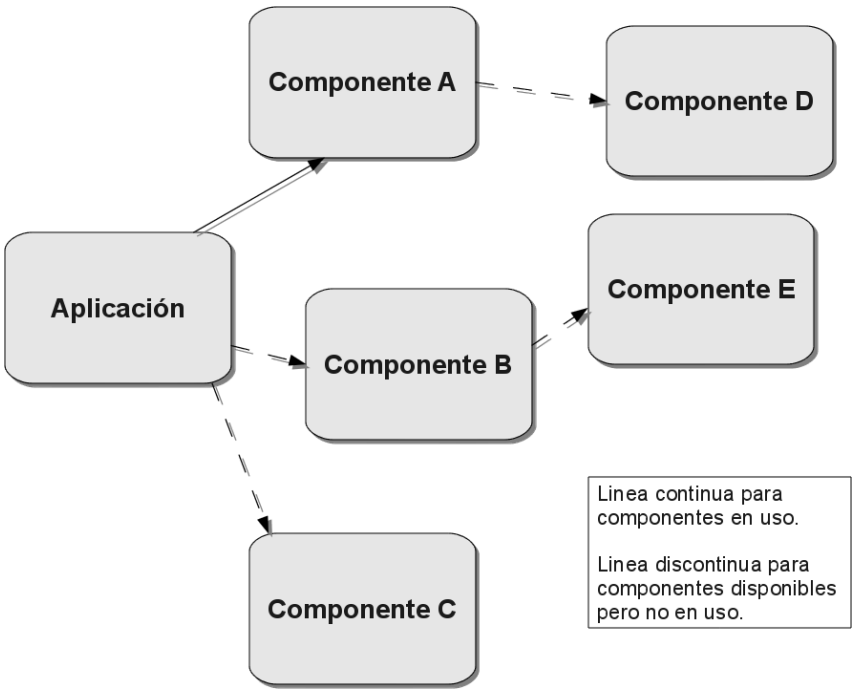


Figura 4.6: Carga de componentes dinámica

4.1.5.2. Implementaciones de OSGi

Existen varias implementaciones de la plataforma OSGi. Algunas de las más relevantes en el panorama actual son:

- **Equinox**: Implementa la versión 4.X de las especificaciones OSGi y es desarrollada por Eclipse Foundation. Es la alternativa más empleada hoy en día por constituir el núcleo de Eclipse, habiendo sido adaptada en multitud de proyectos.
- **Knopflerfish**: Implementa la versión 4.2 de las especificaciones OSGi y es desarrollado por Makeware. Surge como una versión derivada de una implementación anterior (Gatespace) siendo en la actualidad una de las alternativas más estables y populares.
- **Felix**: Implementa la versión 4.2 de las especificaciones OSGi y es desarrollado por la fundación Apache. Igual que Knoplerfish surge como versión derivada de un proyecto anterior (Proyecto Óscar). Su vinculación con la fundación Apache la convierte en

una alternativa muy interesante, factor que ha hecho que se haya utilizado para proyectos como EasyBeans, GlassFish, NetBeans y otros.

- **Concierge**: Implementa la versión 3.0 de las especificaciones OSGi y es desarrollado por el Institute for Pervasive Computing de Zurich bajo licencia BSD. Su principal baza es que está muy optimizado de cara al consumo de recursos lo que posibilita que pueda ser embebido en multitud de dispositivos.
- **Apache Karaf**: Es un contenedor ligero de componentes y aplicaciones. Sobre él se han desarrollado proyectos como Apache Geronimo o Fuse ESB.

4.1.5.3. Proyectos desarrollados mediante la arquitectura OSGi

Con objeto de mostrar la potencia de la arquitectura OSGi presentamos aquí algunos de los proyectos que hacen uso de esta.

- **Apache Aries** [58]: Un contenedor de implementaciones blueprints y extensiones de aplicaciones.
- **DataNucleus** [59]: Una herramienta para la persistencia de datos.
- **BIRT** [60]: Una herramienta de generación de reportes (figura 4.7).

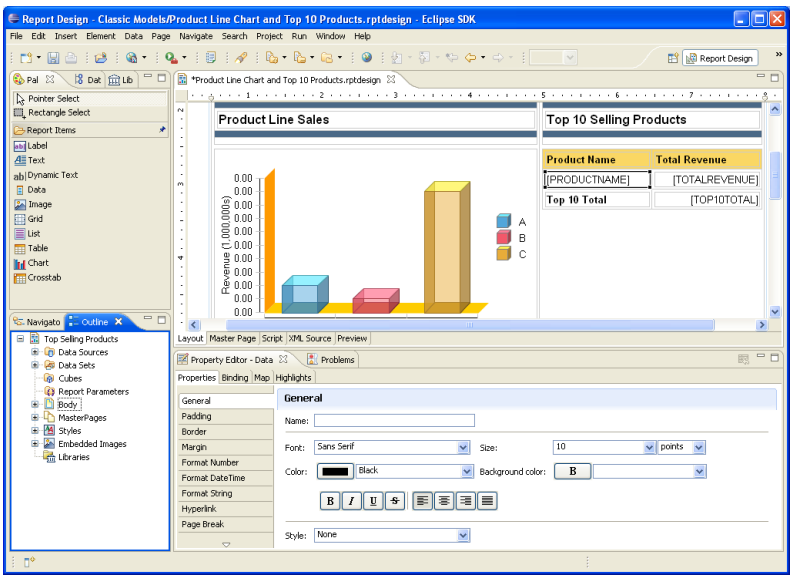


Figura 4.7: Ejemplo de reporte generado con BIRT

- **Eclipse Virgo** [64]: Un servidor de aplicaciones con soporte para aplicaciones OSGi.
- **Eclipse** [32]: Un entorno de desarrollo integrado para el desarrollo de aplicaciones de cliente enriquecido (figura 4.8).

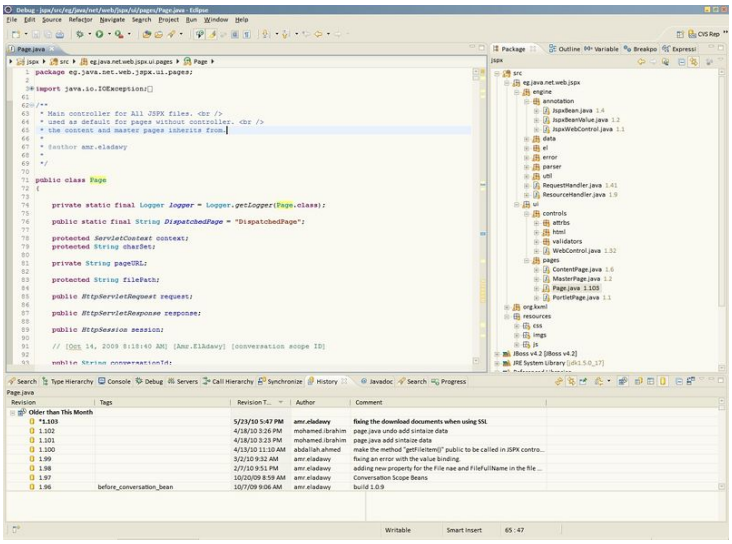


Figura 4.8: Interfaz de Eclipse

- **GX WebManager** [61]: Un sistema de gestión del contenido web de la empresa (figura 4.9).

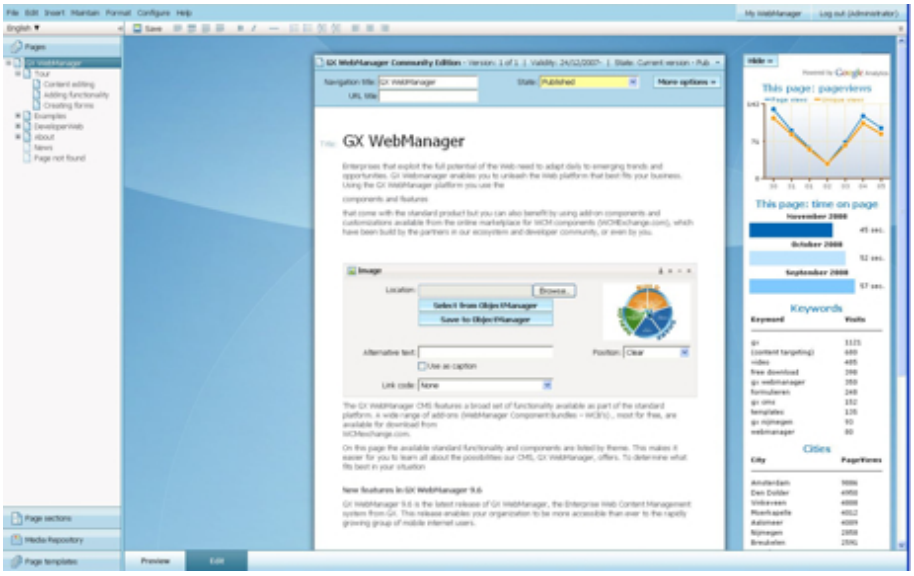


Figura 4.9: Interfaz de GX WebManager

- **EasyBeans** [65]: Un contenedor EJB3.
- **Netbeans** [30]: Un entorno de desarrollo integrado para el desarrollo de aplicaciones de cliente enriquecido (figura 4.10).

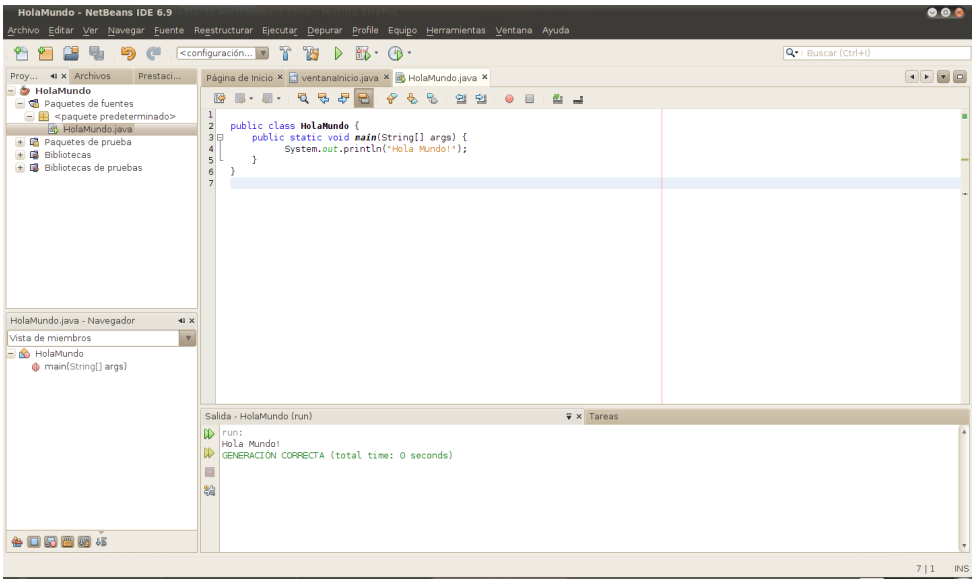


Figura 4.10: Interfaz de NetBeans

- **Nuxeo** [63]: Una plataforma de administración de contenidos (figura 4.11).

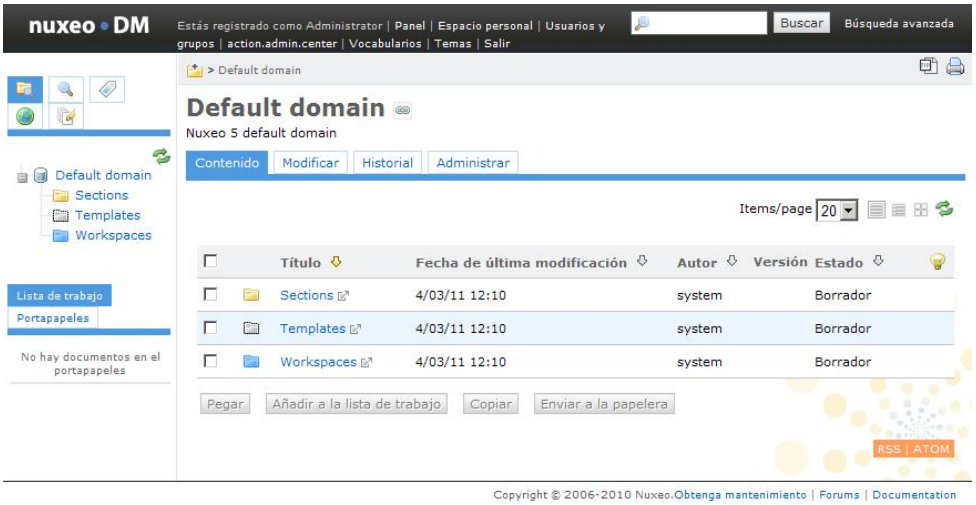


Figura 4.11: Interfaz de Nuxeo

- **WSO2 Carbon** [71]: Una plataforma de desarrollo de middleware[2] de carácter empresarial (figura 4.12).

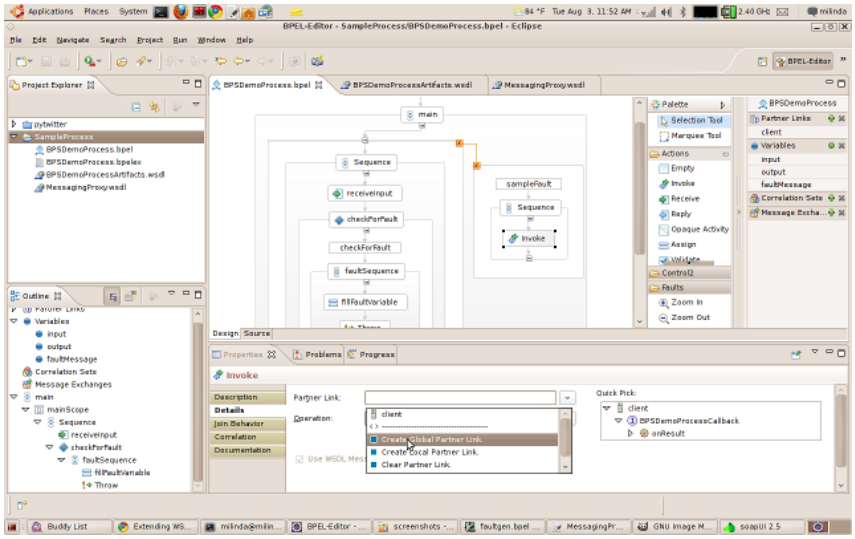


Figura 4.12: Interfaz de WSO2 Carbon

- **Confluence** [62]: Una plataforma para la gestión de contenidos (figura 4.13).

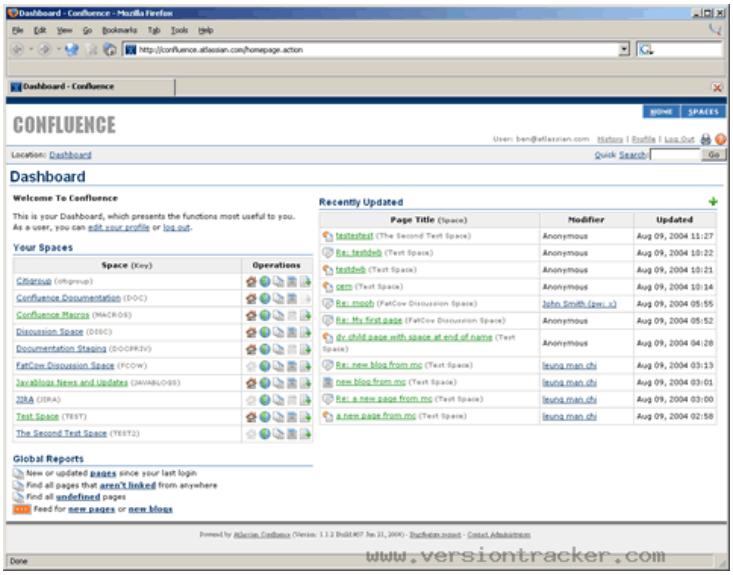


Figura 4.13: Interfaz de Confluence

[2]Software de asistencia a la interacción entre aplicaciones.

- **Weblogic** [63]: Un servidor de aplicaciones J2EE de Oracle.
- **SAP BusinessObjects Event Insight** [66]: Un analizador de eventos complejos SDN.
- **JBoss** [67]: Un servidor de aplicaciones J2EE.
- **GlassFish** [68]: Un servidor de aplicaciones J2EE.
- **JOnAS 5** [69]: Un servidor de aplicaciones J2EE 5.
- **WebSphere Application Server** [70]: Un servidor de aplicaciones J2EE de IBM.

4.2. Framework de análisis

4.2.1. Introducción

En la actualidad estamos viendo como aparecen Framework para casi todo, Framework para aplicaciones web, Framework multimedia, Framework para sistemas de apoyo a la toma de decisiones, etc. Sin embargo, al usar este término que parece estar tan de moda en la actualidad, muy pocos saben a lo que se están refiriendo realmente.

Entendemos un Framework de forma general como *un conjunto de mecanismos bajo los cuales resolver un tipo concreto de problema.* Dentro del mundo del software estos mecanismos pueden verse como componentes o módulos concretos relativos al tipo de problema a tratar, los cuales, correctamente estructurados y organizados, nos permitirán obtener una solución concreta para un problema dado.

De este modo, al hablar de un Framework para aplicaciones web nos referimos a un conjunto de componentes que pueden ser empleados para construir una página web completa simplemente tomando una serie de funcionalidades deseadas (comentarios, wikis, blogs, correo interno, ...). Al hablar de un Framework multimedia de una serie de componentes que, debidamente enlazados nos proporcionarán una aplicación multimedia a medida (reproductor de audio, vídeo, codificación, visor de imágenes, etc).

En el presente proyecto planteamos la creación de un Framework para el diseño de motores de análisis lingüístico basados en reglas. Entendiendo la naturaleza de este tipo de análisis de la forma que se vio en el capítulo 3 y las características propias de un Framework tal y como lo acabamos de presentar el Framework que aquí proponemos puede verse como, *un conjunto de componentes software, específicamente diseñados para la realización de un análisis lingüístico basado en reglas, mediante los cuales poder construir aplicaciones específicas para tratar con problemáticas relacionadas con el análisis lingüístico.*

Así, debemos concebir el Framework de análisis como un entorno para la realización de aplicaciones específicas de análisis lingüístico, cada una de las cuales tomará para sus fines los componentes que sean necesarios.

4.2.2. Características

4.2.2.1. Características generales

Si bien es cierto que no existe una lista de propiedades estricta que nos diga las características que debería cumplir un Framework para ser considerado como tal, si que podemos fijar una serie de características deseables en él para que pueda ser considerado de utilidad con independencia de su funcionalidad concreta.

En lineas generales podemos decir que un Framework debería:

- **Permitir tratar con el tipo de problemas soportados de forma genérica**. Un Framework no puede estar diseñado para solucionar un caso concreto, su diseño se debe de realizar pensando en la problemática general a la que dará soporte.
- **Facilitar la construcción de soluciones a medida**. Los distintos componentes deben poder adaptarse y reutilizarse. De este modo debe ser posible emplear un subconjunto de ellos para la construcción de una solución a medida.
- **Ser estable**. Aquella funcionalidad ofrecida por cada componente debería ser llevada a cabo de la forma descrita por su interfaz y en todos los casos posibles.
- **Ser ampliable**. Un Framework debe estar preparado para soportar cualquier nuevo componente sin necesidad de modificar ninguno de los anteriores.

Este listado de características deseables es el que guía el desarrollo del Framework que aquí planteamos. Así, el Framework de análisis será:

- **Genérico**. Permitirá tratar de forma general con cualquier problema de análisis lingüístico siempre y cuando pueda ser enfocado mediante el análisis dirigido por reglas propuesto.
- **Adaptable**. Los distintos componentes podrán reutilizarse para la construcción de distintas soluciones diferentes a la aquí propuesta.
- **Estable**. Los componentes se desarrollaran bajo unos criterios mínimos de calidad que garanticen la estabilidad de la aplicación.
- **Ampliable**. Estará preparado para trabajar con cualquier componente no considerado previamente.

En la búsqueda de un Framework cuyos componentes aseguren que cumpla con las características aquí descritas, se ha optado por elegir realizar estos componentes como Plug-ins de Eclipse. Las características y peculiaridades de un componente de este tipo se detallan en el siguiente punto.

4.2.2.2. Eclipse Plug-Ins

Un plug-in de Eclipse es un componente OSGi con ligeras modificaciones sobre la especificación original. El núcleo mismo de Eclipse esta construido haciendo uso de plug-ins, proporcionando un entorno en el que los plug-ins pueden a su vez desarrollarse y ser ejecutados [27].

Cada plug-in contribuye al conjunto de forma estructurada, pudiendo a su vez depender de otros servicios proporcionados por otro plug-in y este a su vez proporcionar servicios de los que dependan otros plug-ins.

Este diseño permite que sea posible reutilizar funciones independientes y diferenciadas para construir múltiples aplicaciones diferentes con una misma base común.

Los plug-ins son archivos JAR que definen sus dependencias y servicios mediante los archivos MANIFEST.MF y plugin.xml [27]. Haciendo uso de estos archivos Eclipse es

capaz de cargar un plug-in así como sus dependencias en el momento que sea necesario y no antes, pudiendo de este modo reducir tanto el tiempo de carga como el consumo de recursos.

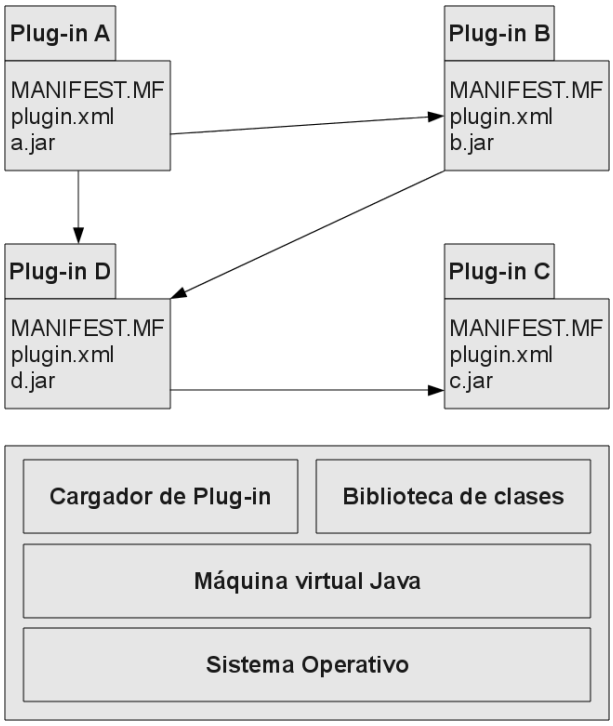


Figura 4.14: Aplicación constituida por Plug-ins

Los plug-ins, por su naturaleza como componentes OSGi, heredan todas las funcionalidades descritas para un componente en esta plataforma, esto es: dependencias en forma de grafo, exportación selectiva, posibilidad de usar diferentes versiones en una misma ejecución, carga dinámica, etc.

4.2.2.3. Archivos JAR

Hasta ahora puede parecer que un Plug-in es poco más que un archivo JAR. Para comprender las ventajas que nos aporta un Plug-in sobre un archivo JAR y así entender mejor que diferencias existen entre ambos haremos un breve inciso para explicar las deficiencias de un archivo JAR con respecto a un Plug-in.

Un archivo JAR parece un componente pero realmente no lo es porque no cumple

con las propiedades descritas en 4.1.3.1 para los componentes. Los JAR son archivos comprimidos que contienen una serie de código destinado a cumplir con una o varias funciones. Son usados principalmente para proporcionar librerías y, en caso de pequeños programas, para contener toda la aplicación final. Incorporan un archivo de manifiesto en el que definen algunos parámetros, siendo este manifiesto muy limitado en comparación con el de un componente.

Sus principales deficiencias con respecto a un plug-in pueden ser resumidas en las siguientes [27]:

- **No tienen concepto de tiempo de ejecución**: Los archivos JAR no diferencian entre el entorno en tiempo de compilación y de ejecución. Por este motivo todos los archivos JAR de una misma aplicación compartirán un mismo espacio de trabajo. Esto hace que un sistema constituido mediante archivos JAR sea muy poco escalable (conflictos de nombres, imposibilidad de trabajar con distintas versiones, ...).

- **Ausencia de dependencias explícitas**: Un archivo JAR no es capaz de indicar por el mismo sus dependencias, solo pudiendo descubrir que falta alguna por los errores producidos al intentar ejecutarlo.

- **No especifican la versión necesaria de una librería**: No podemos saber que versión concreta necesitan de una librería dada, debiendo recurrir a la documentación del mismo (en caso de existir). Si intentamos ejecutar un archivo JAR con una dependencia en una versión diferente a la necesitada los resultados pueden ser totalmente imprevisibles.

- **Imposibilidad de trabajar con múltiples versiones de forma simultanea**: En aplicaciones medianamente grandes habrá partes que necesitarán ejecutar distintas versiones de una misma librería. Un ejemplo típico puede ser el de una aplicación que emplea una librería para acceder a una base de datos en local y otra versión distinta de la misma librería para su acceso en red. En estos casos, nos encontramos con la limitación de que al ejecutarse todos los archivos en un mismo entorno de trabajo, no podemos tener dos versiones distintas de la misma librería funcionando a la vez. En un plug-in esto no es problema al definir cada uno su propio entorno de trabajo.

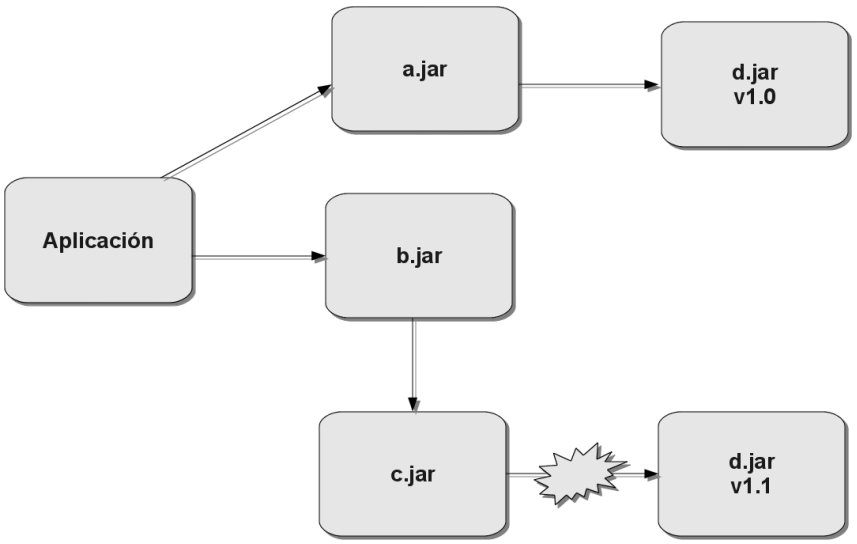


Figura 4.15: Conflicto entre versiones

- **Mecanismo de ocultación de información deficiente**: Un archivo JAR solo puede establecer los niveles de ocultación de información según los modificadores de acceso propios de Java (local, privado, público o protegido). El problema es que estos modificadores de acceso son iguales tanto dentro como fuera del paquete, es decir, lo que sea público dentro del paquete lo será también fuera de él. Un componente va más allá al definir explícitamente qué es lo que desde fuera de él estará accesible.

4.2.3. Tipos de componentes

Cada uno de los componentes desarrollados cumplirá con un función específica y bien delimitada, pudiendo ver cada componente como una unidad funcional única y diferente al resto. A pesar de esta estricta división en el funcionamiento podemos abstraer de ellos una serie de características comunes que nos permitirán clasificarlos a groso modo, como de un tipo u otro.

En base a esta división podremos, posteriormente, intuir cuales son las características concretas esperadas para cada componente particular desarrollado. En general cada componente podrá considerarse de uno de los siguientes tipos:

- **De soporte**: Serán los componentes centrales que constituirán el núcleo de la apli-

cación, sirviendo de soporte para la realización de las distintas tareas necesarias. Bajo un mayor nivel de abstracción estos componentes de soporte podrían verse como una unidad única e independiente, indispensable para el funcionamiento del resto de las partes.

- **De análisis sintáctico de documentos**: Permitirán extraer el contenido de un documento con independencia de su tipo concreto para poder trabajar de forma uniforme sobre este contenido. De este modo podrá haber un componente para tratar con documentos en formato PDF, HTML, TXT, etc.
- **De visualización de resultados**: Ofrecerán distintas visiones de los resultados del análisis.
- **De creación y modificación de recursos**: Posibilitarán la introducción de contenido para que pueda ser analizado siguiendo el esquema de reglas fijado. Si un mismo componente sirve a su vez para la introducción de los datos y la observación de los resultados se podrá considerar como de creación y visualización.
- **De configuración**: Permitirán al usuario modificar y adaptar el comportamiento de determinados componentes.
- **De integración**: Servirán como lazo de unión entre componentes de distinto tipo o del mismo. Así, un componente de integración puede almacenar los resultados concretos de un análisis y servirlos a cualquier componente que desee hacer uso de ellos.

4.3. Tecnologías

Antes de llevar a cabo el desarrollo del software y en vista de las características que buscamos para el mismo, es conveniente realizar un estudio de las tecnologías más convenientes para nuestro caso concreto.

En este último apartado del capítulo introduciremos cuales serán las tecnologías a emplear, concretamente veremos cuál es la arquitectura sobre la que se construirá el Framework, cuáles serán los lenguajes de programación a emplear en su desarrollo y que librerías, herramientas y programas de terceros se emplearán.

4.3.1. Arquitectura del Framework

Independientemente de los componentes que conforman el Framework deberá haber una estructura base sobre la que los componentes puedan ser ejecutados.

Esta estructura base es la que permitirá que los componentes puedan ser iniciados, que interactúen entre ellos sin necesidad de conocer los componentes concretos disponibles, que un nuevo componente pueda ser añadido en cualquier momento sin necesidad de realizar modificaciones en el resto y, en resumen, que la arquitectura del propio Framework pueda ser considerada como de componentes.

Si bien esta base puede ser creada por nosotros siguiendo alguna de las especificaciones de las arquitecturas de componentes descritas en el punto 4.1.4, es una labor totalmente innecesaria puesto que ya existen muchas soluciones implementadas de nos podemos utilizar.

Una muy buena opción es emplear la arquitectura central de un Framework Rich Client Platform (RCP) para conseguir los componentes mínimos necesarios para nuestros objetivos. Un Framework RCP, es una herramienta enfocada en la creación de aplicaciones ricas de escritorio [28]. Para ello pone a disposición del usuario de un entorno de ejecución mínimo en el cual poder ejecutar los componentes desarrollados.

Un Framework RCP se convierte por tanto en la mejor opción para obtener la base de ejecución mínima sobre la que los componentes de nuestro Framework de análisis puedan ser ejecutados. En los siguientes puntos veremos cuales son las principales alternativas existentes actualmente así como las características propias del Framework RCP seleccionado.

4.3.1.1. Alternativas existentes

Las principales alternativas existentes en la actualidad para desarrollar una aplicación RCP son las siguientes:

- **Eclipse RCP** [29]: Eclipse es una herramienta útil para una gran variedad de tareas en base a su naturaleza como un conjunto de componentes OSGi. Dentro

de este conjunto de componentes existe un pequeño grupo que puede ser empleado para desarrollar nuestras propias aplicaciones siguiendo una arquitectura similar a la de Eclipse. Este núcleo es lo que se denomina comúnmente como Eclipse RCP. A día de hoy es la alternativa más popular y no es fruto del azar. Factores como el número de componentes disponibles, la cantidad de documentación o su comunidad de desarrolladores la avalan. Será la alternativa seleccionada para llevar a cabo el desarrollo de nuestro Framework.

- **NetBeans Platform** [30]: De forma parecida a Eclipse RCP surge NetBeans Platform. NetBeans es uno de los Integrated Development Environment (IDE) más extendidos para Java en la actualidad y, al igual que Eclipse, se sustenta en una arquitectura de componentes OSGi. Si abstraemos del conjunto total de componentes únicamente aquellos que se encargan de proporcionar el entorno de ejecución básico podremos disponer de una aplicación general con la que construir nuestras propias aplicaciones, esta es la base de NetBeans Platform. El principal motivo que nos lleva a desechar NetBeans Platform frente a Eclipse RCP es que, en el primero, el número de componentes mínimo necesario para construir la aplicación es mayor, lo que se traduce en unos programas de mayor tamaño y en un consumo de recursos mucho más elevado.

- **Spring Framework** [31]: Es la alternativa más libre de las tres, entendiendo por libre que es la que mayor grado de libertad proporciona al desarrollador. No obliga a usar un modelo concreto de programación, razón que ha propiciado su popularización. Puede ser empleado para desarrollar cualquier aplicación Java desde aplicaciones de escritorio a aplicaciones web.

4.3.1.2. Eclipse RCP

La propia definición que la fundación Eclipse da para su herramienta es "una especie de herramienta universal, un IDE abierto y extensible para todo y nada en particular"[32], una definición tan difusa como la naturaleza misma de Eclipse.

Eclipse es una herramienta genérica, útil para multitud de cosas aunque sin un objetivo concreto. Su potencia por tanto no radica en la propia herramienta en si, si no en lo que en esta permite. Cualquier componente de Eclipse puede ser reutilizado para

ser usado en cualquier otra aplicación, cualquier módulo desarrollado puede ser ejecutado fácilmente dentro de un entorno Eclipse, en resumidas cuentas, cualquier parte de Eclipse está a la entera disposición del desarrollador. Pero esto no se queda ahí.

Dentro de la variedad de componentes que conforman Eclipse podemos encontrar algunos de naturaleza común ya sea compiladores, editores, vistas, herramientas de refactorización, herramientas de marcado, etc, y de entre todos estos podemos destacar unos de especial relevancia en nuestro caso, los que proporcionan el entorno de ejecución básico.

Existe un reducido grupo de plug-ins encargados de construir la infraestructura de la aplicación a través del soporte para vistas, editores, perspectivas, etc. Este conjunto de plug-ins constituye un entorno de aplicación genérico en el cual, cualquier componente desarrollado desde cero puede ser ejecutado. Esto es lo que se conoce como el núcleo RCP[27, 29].

El núcleo RCP sustenta el entorno de aplicación Eclipse y puede ser empleado a su vez para construir nuestras propias aplicaciones. De esta forma podemos desarrollar cualquier aplicación valiéndonos de este núcleo RCP a la cual podremos incorporar si deseamos alguno de los muchos componentes ya existentes para Eclipse de forma sencilla.

Las aplicaciones desarrolladas con el núcleo RCP reciben de forma genérica el nombre de aplicaciones 'Eclipse RCP' en referencia a la arquitectura sobre la que están construidas.

El desarrollo del Framework se llevará a cabo haciendo uso de Eclipse RCP.

4.3.1.3. Aplicaciones desarrolladas con Eclipse RCP

Para ilustrar la versatilidad y potencia de Eclipse RCP veremos aquí algunas de las muchas aplicaciones desarrolladas mediante esta herramienta.

- **Bioclipse** [10]: Una aplicación de investigación científica centrada en la diseño, la edición y la interacción con estructuras biológicas. Permite trabajar con elementos de interés en este campo tales como proteínas y moléculas (figura 4.16).

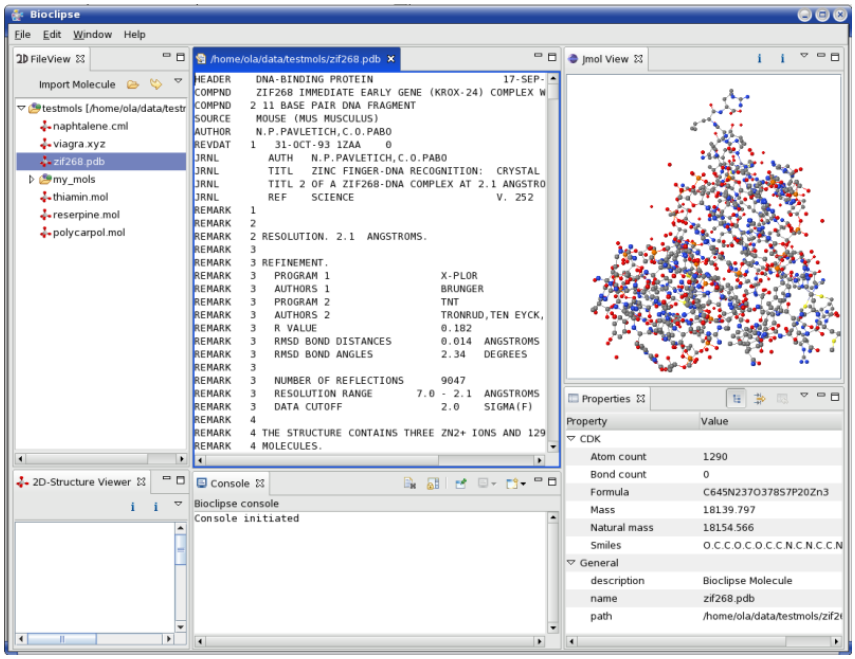


Figura 4.16: Interfaz de Bioclipse

- **CCTVnet** [13]: Un software para la vídeo monitorización en red (figura 4.17).

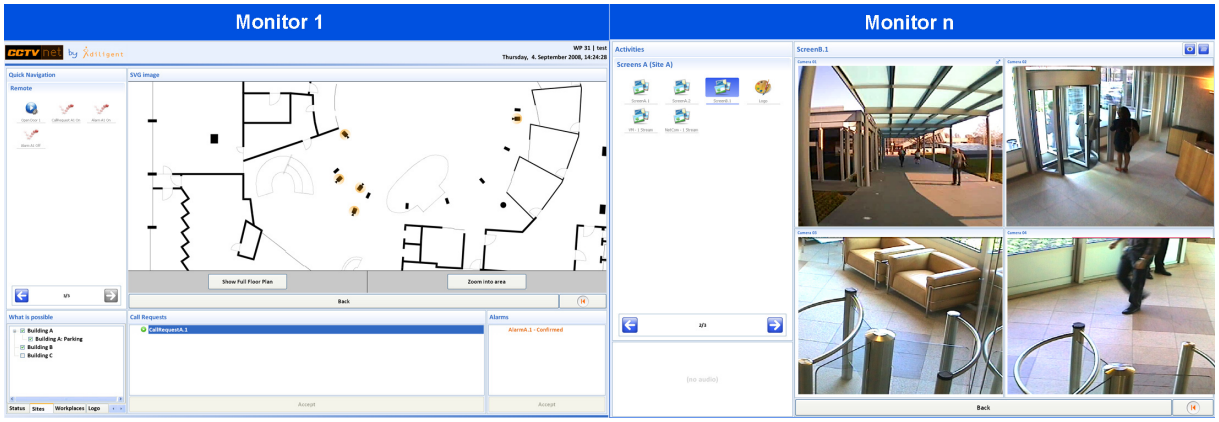


Figura 4.17: Interfaz de CCTVnet

- **Azureus** [11]: Recientemente renombrado como Vuze, Azureus es un programa de intercambio de archivos P2P a través del protocolo BitTorrent (figura 4.18).

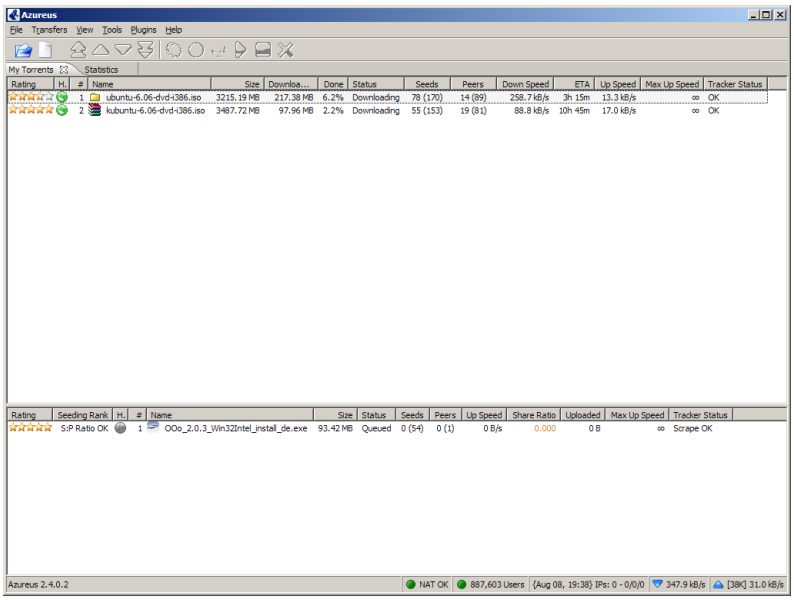


Figura 4.18: Interfaz de Azureus

- **J3DWorkbench** [12]: Una aplicación de modelado y visualización 3D (figura 4.19).

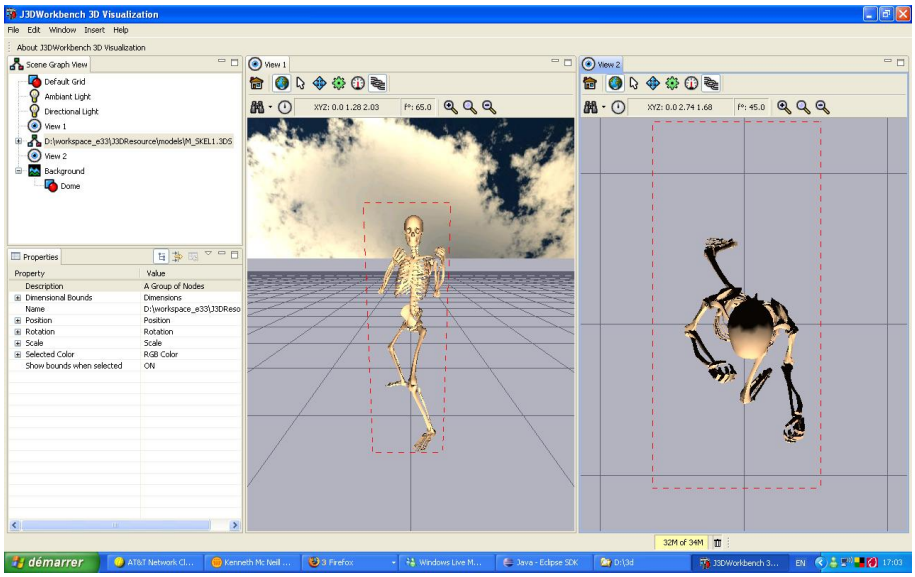


Figura 4.19: Interfaz de J3DWorkbench

- **Kalypso** [14]: Una aplicación de modelado y visualización geoespacial (figura 4.20).

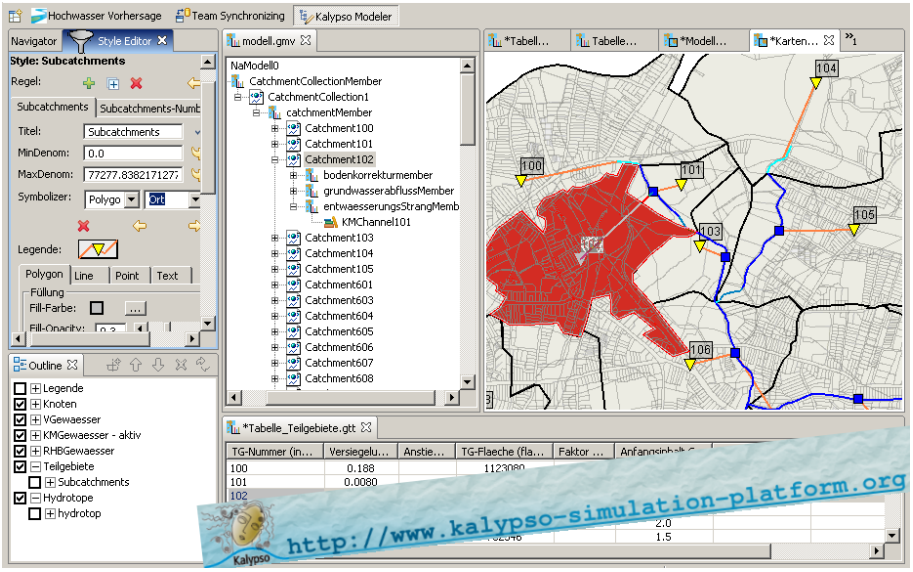


Figura 4.20: Interfaz de Kalypso

- **BaZaa** [15]: Un buscador P2P en tiempo real (figura 4.21).

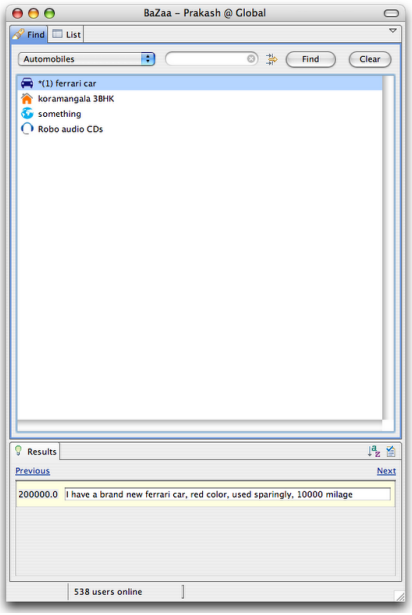


Figura 4.21: Interfaz de BaZaa

- **Global Vision Systems** [16]: Una plataforma de supervisión y análisis en 3D y en tiempo real (figura 4.22).

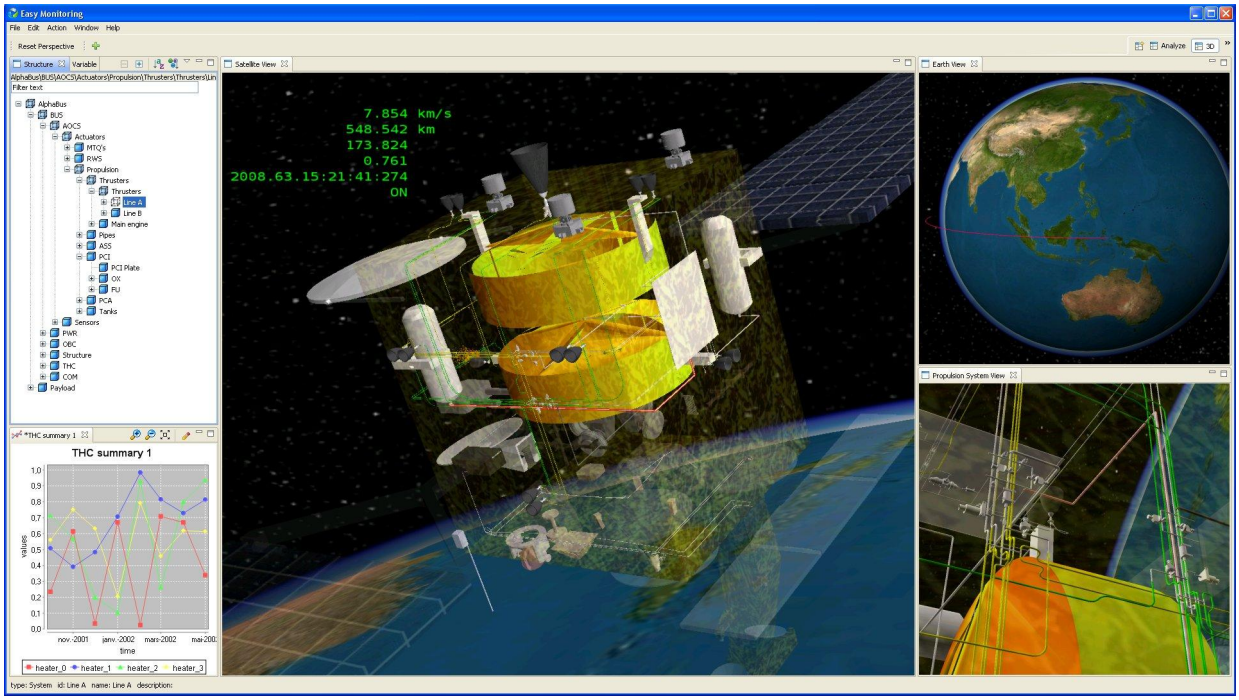


Figura 4.22: Interfaz de Global Vision Systems

4.3.2. Lenguajes de programación

En nuestro caso el lenguaje de programación pasa más por ser una imposición que una opción. Así, en base a la utilización de Eclipse RCP como Framework de desarrollo nos vemos condicionados a emplear Java como lenguaje de desarrollo principal.

Si bien es cierto que también podríamos optar por alguno de los lenguajes de programación implementados sobre la plataforma Java como Groovy, Jython, JRuby o Scala, la utilización de Java pasa por ser la mejor opción en nuestro caso, principalmente por cuestiones de eficiencia[3].

[3]Los lenguajes construidos sobre Java son, por definición, más lentos que Java puesto que previamente a su ejecución necesitan convertir el código a lenguaje Java que pueda ser ejecutado por la máquina virtual de Java.

A pesar de lo dicho inicialmente de que la elección de Java ha venido determinada por el Framework a usar, esto no constituye realmente ningún problema puesto que en cualquier otro caso la elección habría sido la misma. Y no decimos que Java sea la mejor opción sin motivo, de entre las múltiples ventajas que nos aporta podemos destacar:

- **Multiplataforma**. La filosofía de Java es 'Escribe una vez, ejecuta en todas partes' [33]. Si bien en la práctica esto raramente se cumple, si que podemos decir que Java es uno de los lenguajes que más nos facilitarán la labor en este aspecto.
- **Idóneo para el desarrollo de componentes**. Que los principales Framework RCP existentes estén desarrollados pensando en Java no es fruto del azar. Las características propias de Java tales como su posibilidad de ser compilado o interpretado, su orientación a objetos, su extensa librería de clases o su carácter multiplataforma, lo convierten en la mejor alternativa en este campo.
- **Extendido**. Java es el lenguaje de programación más extendido en la actualidad (figura 4.23). Si desarrollamos un Framework bajo la premisa de que este debe ser ampliable, debemos optar necesariamente por aquel lenguaje de programación que sea conocido por un mayor número de usuarios.

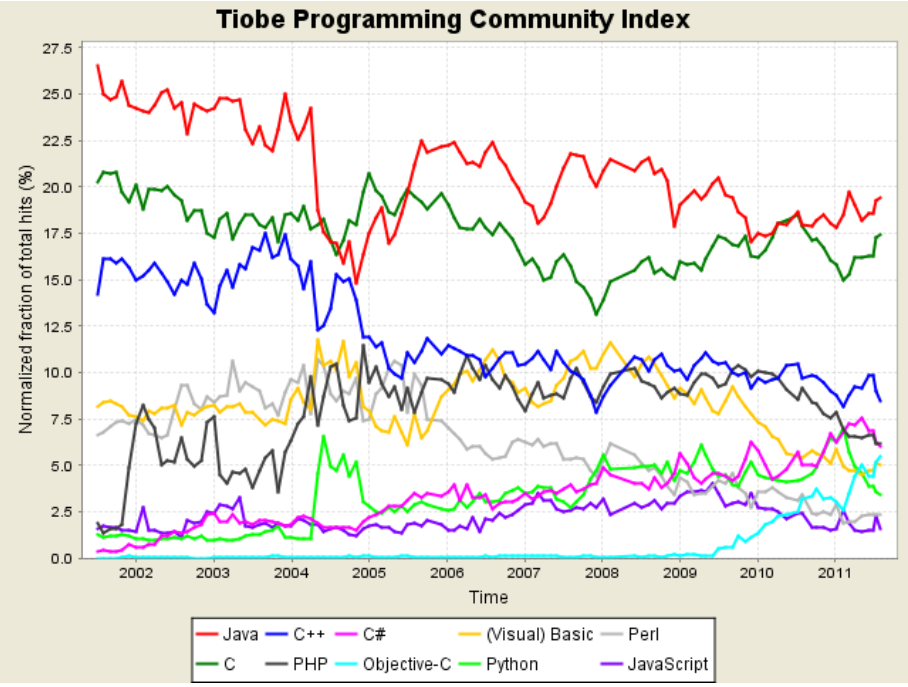


Figura 4.23: Índice Tiobe de lenguajes de programación [34]

4.3.3. Librerías, herramientas y programas

Para el desarrollo del Framework de análisis, así como de la propia memoria del proyecto se hará uso de una serie de librerías, herramientas y programas que presentamos a continuación.

4.3.3.1. Librerías

HtmlParser [35]

HtmlParser es una librería Java para el tratamiento sintáctico de documentos en formato Html. Mediante HtmlParser podemos dividir un documento Html en los distintos elementos que lo componen, trabajar sobre ellos y modificarlos. Emplearemos HtmlParser para dar soporte al uso de documentos Html dentro del Framework.

Figura 4.24: Logotipo de HtmlParser

ODFDOM [46]

ODFDOM es una librería para el acceso a documentos en formato OpenDocument (ODF). Con ODFDOM podremos acceder a documentos ODT para analizar su contenido.

Figura 4.25: Logotipo de ODFDOM

Apache POI [57]

Apache POI es una librería desarrollada por la fundación Apache que permite extraer

el contenido de documentos en formato Office Open XML. En nuestro Framework emplearemos Apache POI para el análisis de documentos DOC y DOCX.

Figura 4.26: Logotipo de Apache POI

SWTChart [36]

SWTChart es un componente Eclipse construido sobre el Toolkit gráfico SWT que permite la creación de gráficas de varios tipos. Utilizaremos SWTChart para mostrar información agregada del análisis al usuario.

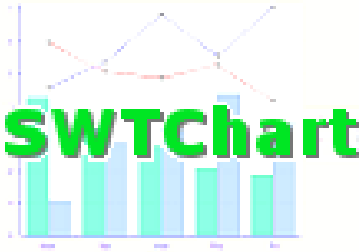


Figura 4.27: Logotipo de SWTChart

NOA [37]

NOA viene de las siglas (Nice Office Access) que podríamos traducir como acceso agradable a Office. Es una librería que nos permite desarrollar de forma sencilla aplicaciones basadas en OpenOffice.org. Emplearemos NOA para dar soporte a OpenOffice.org dentro del Framework.

Figura 4.28: Logotipo de NOA

HyperSQL [38]

HyperSQL es una pequeña base de datos desarrollada en Java que puede ser embebida dentro de una aplicación. HyperSQL nos servirá para almacenar los paquetes de reglas así como el diccionario gramatical.

Figura 4.29: Logotipo de HyperSQL

iText [39]

iText es una librería Java que permite tanto acceder a documentos PDF, RTF y HTML como generarlos. Haremos uso de iText para la generación de informes en PDF y para el análisis de documentos de este tipo.

Figura 4.30: Logotipo de iText

jFreeChart [40]

jFreeChart es una librería Java para la generación de gráficas que pueden ser fácil-

mente embebidas dentro de la aplicación o de los documentos generados. Emplearemos jFreeChart para la generación de las gráficas en los informes PDF.

Figura 4.31: Logotipo de jFeeChart

4.3.3.2. Herramientas

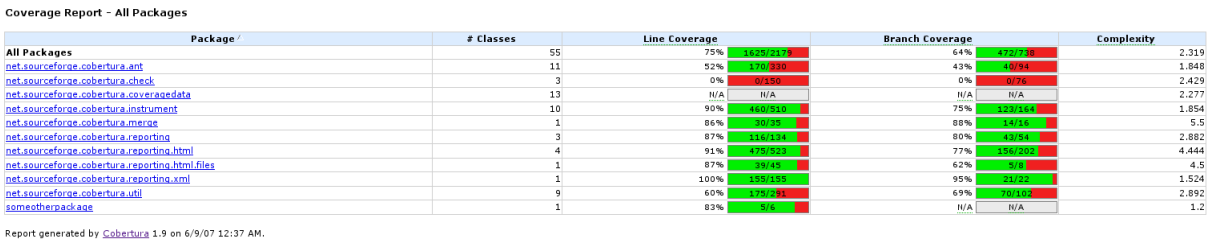
JUnit [41]

JUnit es un Framework Java para la realización de pruebas unitarias que utilizaremos para testear el código desarrollado.

Figura 4.32: Logotipo de JUnit

Cobertura [42]

Cobertura es una herramienta Java que conjuntamente con JUnit nos servirá para calcular el porcentaje de código fuente probado mediante los test unitarios.



Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	55	75%	1625/2179	64%	472/738	2.319
net.sourceforge.cobertura.ant	11	52%	170/330	43%	40/94	1.848
net.sourceforge.cobertura.check	3	0%	0/150	0%	0/76	2.429
net.sourceforge.cobertura.coveragedata	13	N/A	N/A	N/A	N/A	2.277
net.sourceforge.cobertura.instrument	10	90%	460/510	75%	123/164	1.854
net.sourceforge.cobertura.merge	1	86%	30/35	88%	14/16	5.5
net.sourceforge.cobertura.reporting	3	87%	116/134	80%	43/54	2.882
net.sourceforge.cobertura.reporting.html	4	91%	475/523	77%	156/202	4.444
net.sourceforge.cobertura.reporting.html.files	1	87%	39/45	62%	5/8	4.5
net.sourceforge.cobertura.reporting.xml	1	100%	155/155	95%	21/22	1.524
net.sourceforge.cobertura.util	9	60%	175/291	69%	70/102	2.892
someotherpackage	1	83%	5/6	N/A	N/A	1.2

Report generated by Cobertura 1.9 on 6/9/07 12:37 AM.

Figura 4.33: Ejemplo de reporte generado por cobertura

Mockito [43]

Mockito es una librería Java para la creación de objetos Mock[4] que usaremos para realizar las pruebas unitarias en aislamiento.

Figura 4.34: Logotipo de Mockito

SWTBot [44]

SWTBot es un plug-in Eclipse que permite probar los elementos de la interfaz gráfica. Lo emplearemos para probar los botones de la interfaz de usuario de la aplicación.

Git [45]

Git es un sistema de control de versiones distribuido que nos valdrá para gestionar el código fuente desarrollado.

Figura 4.35: Logotipo de GIT

4.3.3.3. Programas

Máquina virtual Java [47]

Será necesario disponer de la máquina virtual Java tanto para el desarrollo como para la posterior ejecución del Framework. La versión que emplearemos será la 6.0.

[4] Objetos simulados que imitan el comportamiento de objetos reales.

Figura 4.36: Logotipo de Java

Plataforma de desarrollo Eclipse RCP [32]

Como hemos indicado anteriormente en 4.3.1.2, se hará uso de la plataforma de desarrollo Eclipse RCP para la construcción del Framework y dar soporte de ejecución básico a los distintos componentes. La versión de Eclipse utilizada será la 3.6 también conocida como Helios.

XULRunner [48]

XULRunner es un entorno de ejecución web elaborado por Mozilla que emplearemos para proveer al Framework de un completo navegador web. La versión será la 1.9.

OpenOffice.org [49]

OpenOffice.org es una popular suite ofimática que usaremos para proporcionar un editor de texto enriquecido al Framework. La versión necesaria será la 3.2 o superior.

Figura 4.37: Logotipo de OpenOffice.org

Capítulo 5

Desarrollo del proyecto

5.1. Descripción

Una vez vistas las bases teóricas en las que se fundamenta este proyecto es el momento de abordar su desarrollo.

Como hemos indicado en los capítulos anteriores, el proyecto consiste en desarrollar un Framework para la realización de análisis lingüísticos dirigidos por reglas constituido sobre una arquitectura de componentes OSGi.

El Framework se acompaña de Linguo, una aplicación desarrollada sobre Eclipse RCP que incorpora todos los componentes creados, y que proporciona todas las funcionalidades del Framework. El desarrollo de Linguo se realizará de forma simultanea al del Framework por lo que, por simplicidad, en la mayoría de las ocasiones emplearemos Linguo para explicar las distintas actividades llevadas a cabo durante el desarrollo. Para mostrar el correcto funcionamiento de la aplicación, ésta va acompañada de un paquete de reglas para la detección de problemas de desigualdad de género. Este paquete se corresponde con el visto en el punto 3.3.4.

El desarrollo se hará siguiendo los principios de desarrollo establecidos por la Ingeniería del software la cual, aún careciendo de una definición formal establecida, podemos entender como:

La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software [51].

El proceso de desarrollo bajo la ingeniería del software variará en función del modelo de desarrollo escogido. Considerando que el desarrollo va a ser llevado a cabo por una única persona, lo más adecuado es optar por seguir el ciclo de vida clásico del software, el cual pasa por realizar las siguientes actividades:

En primer lugar se realizará la **especificación de requerimientos**, estableciendo cual va a ser el propósito del sistema, sus restricciones y propiedades.

Después pasaremos a **analizar el sistema**, tras lo cual obtendremos un modelo del sistema correcto, completo, consistente, claro y verificable.

A continuación nos ocuparemos de **diseñar el sistema**, obteniendo como resultado

una solución lógica al modelo planteado en la etapa anterior.

Al finalizar el diseño llevaremos a cabo la **implementación**. En esta fase partiremos de la solución lógica obtenida en el diseño y la implementaremos.

Para finalizar, pasaremos a realizar las **pruebas**, con las que verificaremos y validaremos el sistema.

En los siguientes puntos profundizaremos en cada una de estas actividades.

5.2. Especificación de requerimientos

Los requerimientos software son un conjunto de condiciones o capacidades que la solución final deberá poseer. Mediante la especificación de los mismos establecemos lo que el sistema deberá hacer, pero no cómo hacerlo, de ello se ocupan las etapas posteriores.

Haciendo un símil, podríamos decir que los requerimientos son como cláusulas de un contrato al que el desarrollo queda ligado. Por tanto, es transcendental que los requerimientos estén bien formulados, siendo esencial que sean realistas, completos y verificables. Además, su formulación debe ser en todo caso una ayuda y nunca una fuente de controversias, por lo que deberán ser redactados con sumo cuidado. Así, se debe intentar que sean concisos, consistentes y carentes de ambigüedad, de modo que siempre quede claro cuál es el requerimiento.

En ingeniería del software, los requerimientos se suelen dividir dependiendo de si tratan sobre el funcionamiento expreso del sistema o no[1]. Se denominan respectivamente *requerimientos funcionales* y *requerimientos no funcionales.*

[1]En los casos en que existan regulaciones, leyes, factores ambientales, etc, determinantes para el desarrollo, puede ser necesario incluir un nuevo apartado para la especificación de los requerimientos de este tipo.

5.2.1. Requerimientos funcionales

Como indicamos en el punto anterior, los requerimientos funcionales son aquellos que definen el funcionamiento del sistema de forma expresa.

En base al sistema que deseamos desarrollar los requerimientos funcionales serán los siguientes:

1. Realizar análisis a través del navegador web.
2. Realizar análisis a través del editor de textos.
3. Realizar análisis de sitios web.
4. Realizar análisis de documentos de texto.
5. Abrir documentos de texto en el editor de textos.
6. Guardar los documentos de texto creados con el editor.
7. Guardar las páginas web mostradas en el navegador web.
8. Generar informes de los análisis realizados.
9. Visualizar en el mismo navegador web los resultados del análisis.
10. Visualizar en el mismo editor de textos los resultados del análisis.
11. Visualizar resultados del análisis de forma agregada.
12. Visualizar resultados del análisis de forma separada.
13. Crear paquetes de reglas para realizar análisis a medida.
14. Modificar los paquetes de reglas que guían el análisis.
15. Importar paquetes de reglas externos.
16. Exportar los paquetes de reglas.

5.2.2. Requerimientos no funcionales

Los requerimientos no funcionales complementan a los funcionales y recogen aspectos relacionados con el desempeño del sistema, su mantenimiento, plazos de entrega, etc.

En el desarrollo de un Framework, los requerimientos no funcionales son críticos para garantizar aspectos como su ampliabilidad o su separación funcional. En caso contrario, el Framework podría ser prácticamente inservible para el desarrollo de nuevas aplicaciones.

Los requerimientos no funcionales serán los siguientes:

- Cada funcionalidad independiente debe estar agrupada en un único componente funcional.
- La interfaz de cada componente debe estar perfectamente especificada.
- La funcionalidad de cada componente debe estar documentada.
- Cada componente podrá ser reemplazado sin afectar a los anteriores.
- El sistema en su conjunto debe ser ampliable de forma sencilla.
- Cada componente podrá ser probado de forma independiente.
- El desarrollo se debe realizar haciendo uso de buenas prácticas de programación.
- La interfaz de la aplicación debe ser usable.
- El sistema en su conjunto debe ser estable.

Los requerimientos software y hardware por ser más extensos se incluyen en el siguiente punto.

5.2.2.1. Requerimientos software y hardware

El equipo en el que se realizará el proyecto tendrá unas especificaciones hardware reducidas, pues tan solo se deberá utilizar para desarrollar y testear el proyecto. A nivel software, haremos uso de diversas herramientas de libre acceso, tanto por su carácter

gratuito como por la necesidad de no encadenarnos a ninguna herramienta en particular. En conjunto, se dispondrá de un equipo en el que se deben contar al menos con las siguientes especificaciones:

- **A nivel hardware**
  - Procesador Pentium IV 3.2 Ghz o equivalente.
  - 1 GB de memoria RAM.
  - Espacio de almacenamiento de 10 GB.
- **A nivel software**
  - Sistema operativo GNU/Linux.
  - Entorno de desarrollo Java.
  - Motor de bases de datos HyperSQL 1.9.0
  - Plataforma de desarrollo RCP Eclipse 3 RCP.5.2.
  - XULRunner 1.9.
  - OpenOffice.org 3.2.

Para hacer uso de la herramienta el equipo debe tener al menos las siguientes especificaciones:

- **A nivel hardware**
  - Procesador Pentium IV 2 Ghz o equivalente.
  - 512 MB de memoria RAM.
  - Espacio de almacenamiento de 10 GB.
- **A nivel software**
  - Sistema operativo GNU/Linux.
  - XULRunner 1.9.
  - Máquina virtual Java 6.0.
  - OpenOffice.org 3.2.

5.3. Análisis del sistema

Al realizar el análisis del sistema llevamos a cabo la creación de un modelo concreto que capture el comportamiento deseado, el cual se ha fijado al especificar los requerimientos. Esta fase está estrechamente ligada con la anterior, realizándose en muchas ocasiones de manera conjunta.

Para la obtención del modelo del análisis realizaremos dos tareas, definiremos los casos de uso y los principales escenarios de los mismos.

5.3.1. Casos de uso

Podemos definir un caso de uso como:

Una especificación de secuencias de acciones, incluidas secuencias variantes y secuencias de error que un sistema, subsistema o clase puede realizar al interactuar con actores externos [52].

Un caso de uso es algo que el actor quiere que el sistema haga. Es un 'caso de uso' del sistema por un actor específico [53]:

- Los casos de uso se inician siempre por un actor.
- Los casos de uso se escriben siempre desde el punto de vista de los actores.

Normalmente pensamos en los casos de uso en el nivel del sistema, pero como indica la definición, también podemos aplicar casos de uso para describir 'casos de uso' de un subsistema o incluso una clase individual.

Aún no existiendo un estándar UML para la especificación de casos de uso, la siguiente plantilla es prácticamente un estándar de facto para su definición [53]:

- **Nombre del caso de uso**: Identificador unívoco del caso de uso.
- **Breve descripción**: Un párrafo que captura el objetivo del caso de uso.

- **Actores**: Actores implicados en el caso de uso.
- **Precondiciones**: Condiciones que deben ser verdaderas antes de que el caso de uso pueda ejecutarse. Son restricciones en el estado del sistema.
- **Flujo principal**: Los pasos en el caso de uso.
- **Postcondiciones**: Condiciones que deben ser ciertas al final del caso de uso.
- **Flujos alternativos**: Una lista de alternativas al flujo principal.

Por tanto, es necesario que definamos cuales son los actores que van a partir en los casos de uso. En nuestro sistema vamos a contar con un único actor que definiremos como Usuario y será el que interactuará con la aplicación.

Definidos los actores podemos pasar a elaborar los diferentes casos de uso. Debemos procurar que cada uno de los requerimientos funcionales planteados esté cubiertos por al menos uno de los casos de uso. Esto convierte la creación de los casos de uso en una tarea complicada ya que se dará una relación muchos a muchos entre ambos. Así, cada caso de uso podrá tratar con muchos requerimientos diferentes y, a su vez, un requerimiento puede estar presente en varios casos de uso.

El primer paso para la realización de los casos de uso será la creación del diagrama frontera de nuestro sistema el cual describirá de forma completa la funcionalidad del mismo (figura 5.1).

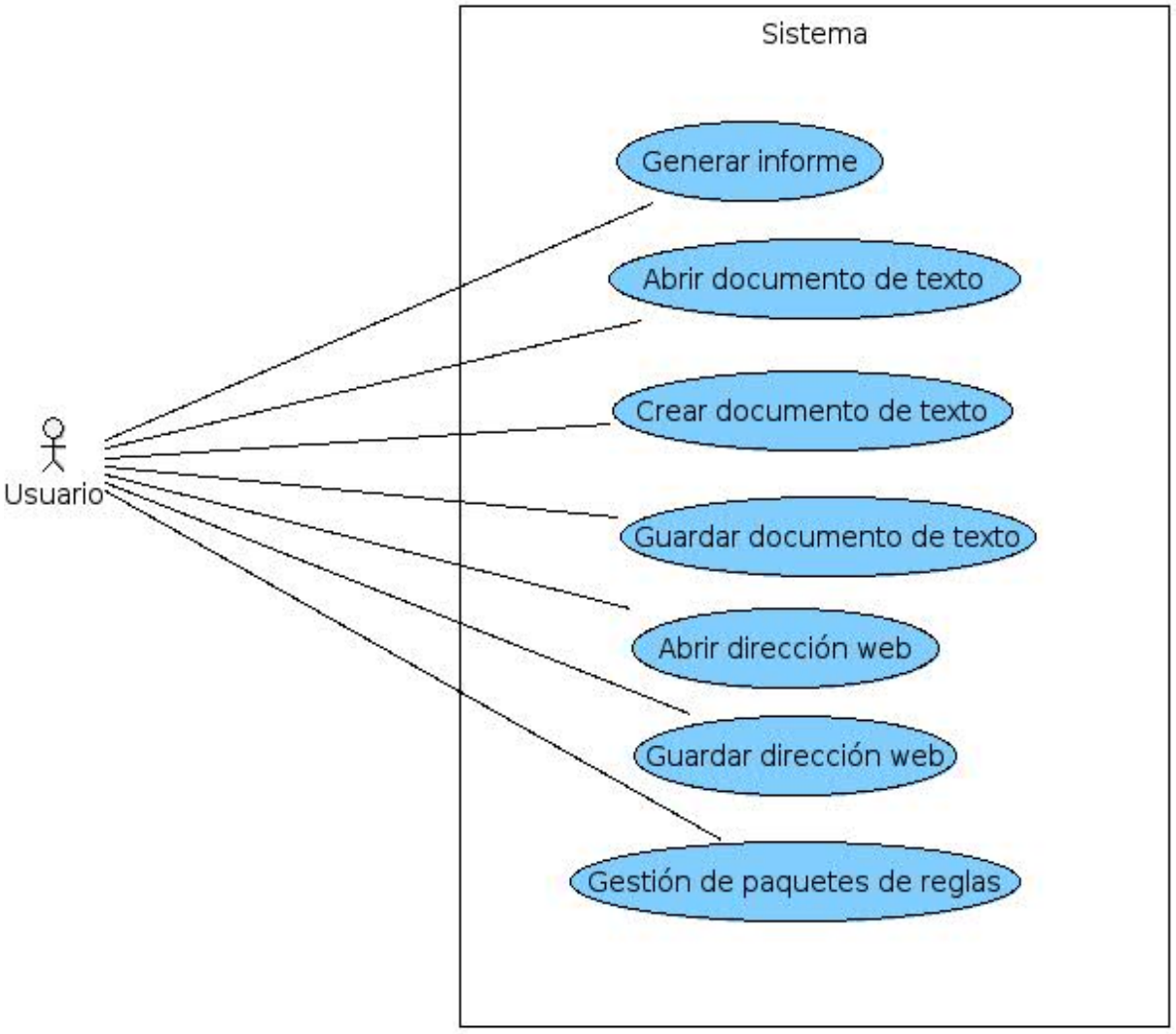


Figura 5.1: Diagrama frontera del sistema

Para comprobar si cada uno de los requerimientos está recogido en al menos un caso de uso emplearemos una matriz de trazabilidad. Si en dicha matriz existiese algún requerimiento que no está presente en ningún caso de uso sería un claro indicativo de que nos falta por incluir alguno.

Los valores de las columnas se corresponden con los casos de uso mostrados en la figura 5.1 de forma descendente ('Generar informe' CU1, 'Abrir documento de texto' CU2, ...).

Requerimiento\Caso de uso	**CU1**	**CU2**	**CU3**	**CU4**	**CU5**	**CU6**	**CU7**
R1	x				x		
R2	x	x	x				
R3	x						
R4	x						
R5		x					
R6				x			
R7						x	
R8	x						
R9					x		
R10		x	x	x			
R11		x	x	x	x		
R12		x	x	x	x		
R13							x
R14							x
R15							x
R16							x

Cuadro 5.2: Matriz de trazabilidad

La matriz de trazabilidad 5.2 muestra que el conjunto de casos de uso trata con la totalidad de los requerimientos por lo que podemos proceder a detallar cada uno de ellos.

Los casos de uso obtenidos pueden ser suficientemente precisos o necesitar ser explicados en mayor detalle. Para detallar un caso de uso podemos emplear dos tipos de relaciones:

- <<**include**>>: Una relación entre casos de uso que permite que un caso de uso incluya comportamiento de otro.

- <<**extend**>>: Una relación entre casos de uso que permite que un caso de uso extienda su comportamiento con uno o más fragmentos de comportamiento de otros.

A continuación describiremos cada uno de los casos de uso mostrados en 5.1 exponiendo los diagramas de aquellos que necesiten un mayor nivel de detalle.

5.3.1.1. Notación empleada

La ausencia de un estándar para la especificación de los casos de uso hace que sea necesario indicar, previamente a la descripción de los mismos, cual va a ser la notación que emplearemos. Esto evitará posibles confusiones debidas a una mayor familiaridad con un estilo de definición de casos de uso diferente al aquí empleado.

- Los casos de uso comienzan siempre con una acción del actor principal y terminan con una respuesta del sistema.
- No se especificará en ningún caso cómo se realizará un paso sino qué paso se realiza. De establecer la forma en la que se realiza un paso concreto nos ocuparemos en la etapa de diseño.
- Los caminos alternativos se notan de la forma '[número de acción][letra]'. Así, si la acción 4 del flujo de eventos admite dos alternativas se notaran:

  4a. Acción 4 del flujo principal.
  5. Acción siguiente a 4a.

  Caminos alternativos:
  4b. Acción alternativa a 4a.
  5. Acción siguiente a 4b.

- Los caminos optativos se notan de la forma '[número de acción].[número]'. De esta forma, si a partir de la acción 4 existe un camino optativo compuesto de dos acciones la notación sería:

  4. Acción 4 del flujo principal.
     4.1. Acción 1 del camino optativo.
     4.2. Acción 2 del camino optativo.
  5. Acción 5 del flujo principal.

- Si los caminos alternativos son medianamente complejos se englobarán en un subflujo de eventos que se notará de la forma '[número de subflujo].[número]'.

Caso de uso 1: Generar informe

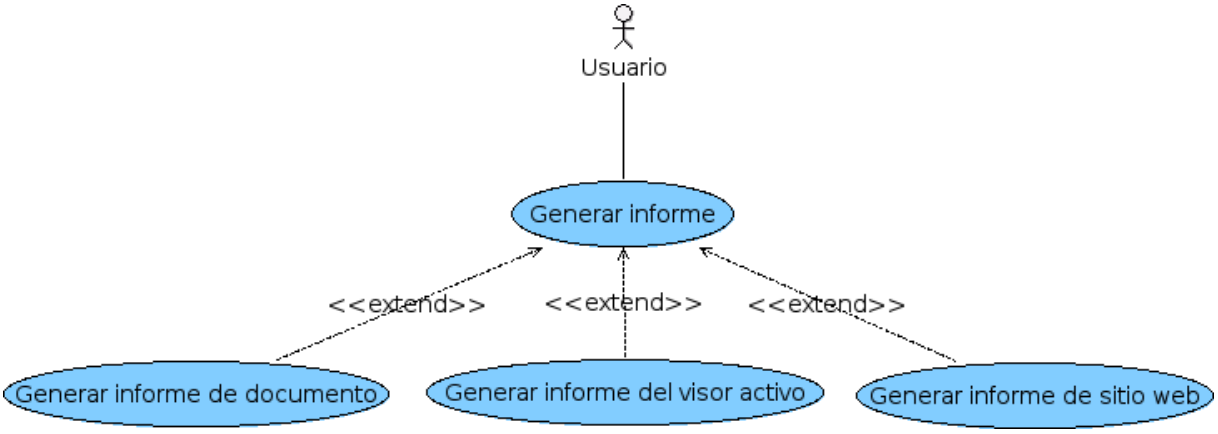


Figura 5.2: Caso de uso 'Generar informe'

Breve descripción:

Generar un informe de un contenido.

Actores:

Usuario.

Precondiciones:

No debe existir ningún análisis en curso.

Flujo principal:

1. El usuario elige generar un informe.

2. El sistema solicita el tipo de informe.

S1. El usuario elige informe de documento.

S2. El usuario elige informe del visor activo.

S3. El usuario elige informe de sitio web.

Postcondiciones:

El informe es generado.

Subflujos de eventos:

S1. Informe de documento.

1.1 El sistema solicita la ruta del archivo a analizar.

1.2. El usuario introduce la ruta del archivo a analizar.

1.3a. El sistema solicita la ruta del archivo de salida.

1.4. El usuario introduce la ruta del archivo de salida.

1.5a. El sistema analiza el archivo.

1.6. El sistema genera el archivo de salida.

S2. Informe del visor activo.

2.1. El sistema solicita la ruta del archivo de salida.

2.2. El usuario introduce la ruta del archivo de salida.

2.3a. El sistema analiza el visor activo.

2.4. El sistema genera el archivo de salida.

S3. Informe de sitio web.

3.1 El sistema solicita la dirección del sitio web.

3.2. El usuario introduce la dirección del sitio web.

3.3. El sistema solicita la ruta del archivo de salida.

3.4. El usuario introduce la ruta del archivo de salida.

3.5a. El sistema analiza el sitio web.

3.6. El sistema genera el archivo de salida.

Flujos alternativos en S1:

Flujo alternativo a 1.3a:

1.3b. El sistema indica que el archivo no es válido y retorna a 1.1.

Flujo alternativo a 1.5a (b):

1.5b. El sistema informa que no es posible crear el archivo en la dirección especificada y retorna a 1.3a.

Flujo alternativo a 1.5a (c):

1.5c. El sistema solicita confirmación para sobreescribir el archivo.

1.6a. El usuario confirma la solicitud.

1.7. El sistema analiza el archivo.

1.8. El sistema genera el archivo de salida.

Flujo alternativo a 1.6a:

1.6b. El usuario deniega la solicitud.

1.7. El sistema retorna a 1.3a.

Flujos alternativos en S2:

Flujo alternativo a 2.3a (b):

2.3b. El sistema informa que no es posible crear el archivo en la dirección especificada y retorna a 2.1.

Flujo alternativo a 2.3a (c):

2.3c. El sistema solicita confirmación para sobreescribir el archivo.

2.4a. El usuario confirma la solicitud.

2.5. El sistema analiza el visor activo.

2.6. El sistema genera el archivo de salida.

Flujo alternativo a 2.4a:

2.4b. El usuario deniega la solicitud.

2.5. El sistema retorna a 2.1.

Flujos alternativos en S3:

Flujo alternativo a 3.5a (b):

3.5b. El sistema informa que no es posible crear el archivo en la dirección especificada y retorna a 3.3.

Flujo alternativo a 3.5a (c):

3.5c. El sistema solicita confirmación para sobreescribir el archivo.

3.6a. El usuario confirma la solicitud.

3.7. El sistema analiza el sitio web.

1.8. El sistema genera el archivo de salida.

Flujo alternativo a 3.6a:

3.6b. El usuario deniega la solicitud.

3.7. El sistema retorna a 3.a.

Caso de uso 2: Abrir documento de texto

Breve descripción:

Abrir un documento de texto en el editor.

Actores:

Usuario.

Precondiciones:

El documento debe existir.

El editor debe estar abierto.

Flujo principal:

1. El usuario elige abrir documento.

2. El sistema solicita la ruta del archivo.

3. El usuario introduce la ruta del archivo.

4a. El sistema muestra el archivo en el editor.

5. El sistema analiza el contenido del documento.

6. El sistema muestra los resultados del análisis.

Postcondiciones:

El documento y las incidencias del mismo son mostradas.

Flujos alternativos:

Flujo alternativo a 4a:

4b. El sistema informa que el archivo no es válido y retorna a 2.

Caso de uso 3: Crear documento de texto

Breve descripción:

Crear un nuevo documento de texto en el editor.

Actores:

Usuario.

Precondiciones:

El editor debe estar abierto.

Flujo principal:

1. El usuario elige crear un nuevo documento.
2. El sistema crea un nuevo documento en el editor.
3. El sistema analiza el contenido del documento.
4. El sistema muestra los resultados del análisis.

Postcondiciones:

El documento creado y las incidencias del mismo son mostradas.

Flujos alternativos:

Ninguno.

Caso de uso 4: Guardar documento de texto

Breve descripción:

Guardar un documento de texto.

Actores:

Usuario.

Precondiciones:

El editor debe estar abierto.

Flujo principal:

1. El usuario elige guardar el documento.
2. El sistema solicita la ruta del archivo.

3. El usuario introduce la ruta del archivo.

4a. El sistema informa que se ha guardado el documento.

Postcondiciones:

El documento es almacenado.

Flujos alternativos:

Flujos alternativo a 4a (b):

4b. El sistema informa que no es posible crear el archivo en la dirección especificada y retorna a 2.

Flujos alternativo a 4a (c):

4c. El sistema solicita confirmación para sobreescribir el archivo.

5a. El usuario confirma la solicitud.

6. El sistema informa que se ha guardado el archivo.

Flujo alternativo a 5a:

5b. El usuario deniega la solicitud.

6. El sistema retorna a 2.

Caso de uso 5: Abrir dirección web

Breve descripción:

Abrir una dirección web.

Actores:

Usuario.

Precondiciones:

El navegador debe estar abierto.

Flujo principal:

1. El usuario introduce una nueva dirección.

2a. El sistema carga la dirección introducida.

3. El sistema analiza el contenido de la página.

4. El sistema muestra los resultados del análisis.

Postcondiciones:

La página y las incidencias de la misma son mostradas.

Flujos alternativos:

Flujos alternativo a 2a:

2b. El sistema informa que no es posible cargar la dirección especificada y abandona el caso de uso.

Caso de uso 6: Guardar dirección web

Breve descripción:

Guardar una dirección web.

Actores:

Usuario.

Precondiciones:

El navegador debe estar abierto.

Flujo principal:

1. El usuario elige guardar la dirección.

2. El sistema solicita la ruta del archivo.

3. El usuario introduce la ruta del archivo.

4a. El sistema informa que se ha guardado la dirección.

Postcondiciones:

La dirección web es almacenada.

Flujos alternativos:

Flujos alternativo a 4a (b):

4b. El sistema informa que no es posible crear el archivo en la dirección especificada y retorna a 2.

Flujos alternativo a 4a (c):

4c. El sistema solicita confirmación para sobreescribir el archivo.

5a. El usuario confirma la solicitud.

6. El sistema informa que se ha guardado la dirección.

Flujo alternativo a 5a:

5b. El usuario deniega la solicitud.

6. El sistema retorna a 2.

Caso de uso 7: Gestión de paquetes de reglas

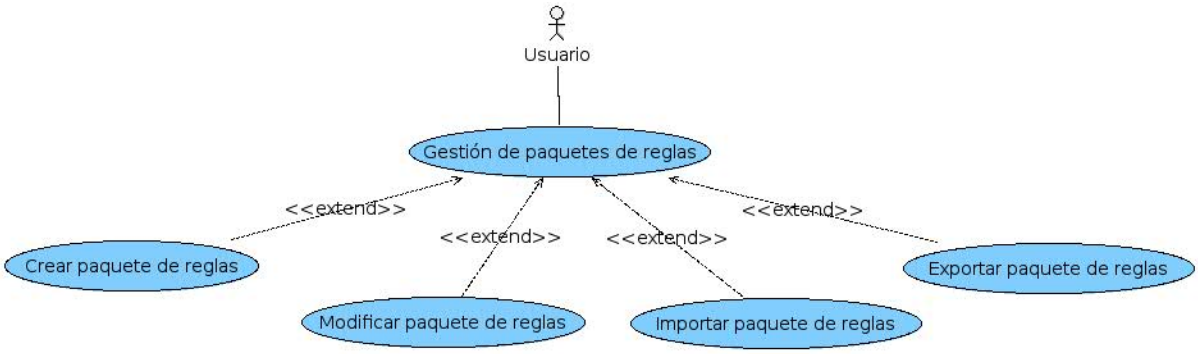


Figura 5.3: Caso de uso 'Gestión de paquetes de reglas'

Breve descripción:

Gestionar los paquetes de reglas de la aplicación.

Actores:

Usuario.

Precondiciones:

No debe existir ningún análisis en curso.

Para la modificación y exportación el paquete deberá existir.

Flujo principal:

1. El usuario elige gestionar los paquetes de reglas.

2. El sistema muestra la herramienta de gestión de paquetes de reglas.

S1. El usuario elige crear un nuevo paquete de reglas.

S2. El usuario elige modificar un paquete de reglas.

S3. El usuario elige importar un paquete de reglas.

S4. El usuario elige exportar un paquete de reglas.

Postcondiciones:

Las modificaciones en los paquetes de reglas son almacenadas.

Subflujos de eventos:

S1. Creación de un nuevo paquete de reglas.[2]

1.1 El sistema solicita los valores para el paquete.

1.2. El usuario introduce los valores para el paquete.

1.3a. El sistema almacena el paquete.

S2. Modificación de un paquete de reglas.[3]

2.1 El sistema muestra los valores del paquete.

2.2. El usuario modifica los valores del paquete.

2.3a. El sistema almacena los cambios en el paquete.

S3. Importar un paquete de reglas.

3.1. El sistema solicita la ruta del archivo.

3.2. El usuario introduce la ruta del archivo.

3.3a. El sistema crea un nuevo paquete de reglas a partir del archivo.

S4. Exportar un paquete de reglas.

4.1. El sistema solicita la ruta del archivo de salida.

4.2. El usuario introduce la ruta del archivo de salida.

4.3a. El sistema informa que se ha exportado el paquete.

[2]El subflujo estaría compuesto por más acciones a nivel inferior para la inserción, modificación y borrado de conjuntos de términos, términos y reglas, pero no entraremos en mayor detalle a fin de simplificar el mismo.

[3]Igual que en S1.

Flujos alternativos en S1:

Flujo alternativo a 1.3a:

1.3b. El sistema indica que los datos son inválidos y retorna a 1.1.

Flujos alternativos en S2:

Flujo alternativo a 2.3a:

2.3b. El sistema indica que los datos son inválidos y retorna a 1.1.

Flujos alternativos en S3:

Flujo alternativo a 3.3a (b):

3.3b. El sistema informa que el archivo no es válido y retorna a 3.1.

Flujo alternativo a 3.3a (c):

3.3c. El sistema renombra el paquete importado para no sobreescribir un paquete existente.

3.4. El sistema crea un nuevo paquete de reglas a partir del archivo importado.

Flujos alternativos en S4:

Flujo alternativo a 4.3a (b):

4.3b. El sistema informa que no es posible crear el archivo en la dirección especificada y retorna a 4.1.

Flujo alternativo a 4.3a (c):

4.3c. El sistema solicita confirmación para sobreescribir el archivo.

5a. El usuario confirma la solicitud.

6. El sistema informa que se ha exportado el paquete.

Flujo alternativo a 5a:

5b. El usuario deniega la solicitud.

6. El sistema retorna a 4.1.

5.3.2. Escenarios

Un escenario toma un caso de uso como funcionalidad abstracta y sigue su flujo de eventos mediante el uso de valores ficticios. Así, un escenario mostrará una representación concreta de un caso de uso a fin de descubrir posibles problemas y así poder anticiparse a ellos.

Un escenario quedará formado al establecer:

- El nombre del mismo.
- Su descripción.
- Los actores que participarán.
- El flujo de eventos que se seguirá.

De lo anterior se deduce que para cada caso de uso existirán múltiples escenarios y, por tanto, será necesario discernir cuales son los más interesantes a realizar. En nuestro caso definiremos un escenario para cada caso de uso que entrañe una mayor complejidad. Concretamente hemos elaborado cinco escenarios, los cuales se muestran a continuación.

Nombre:

Apertura de un archivo en el editor.

Descripción:

El usuario Roberto inicia la aplicación y decide abrir un documento para analizar su contenido.

Actores participantes:

Roberto.

Flujo de eventos:

1.Roberto inicia la aplicación y abre el editor de textos.

2. El sistema le muestra el editor de textos con un documento nuevo creado que no contiene ninguna incidencia.

3.Roberto pulsa sobre la opción de 'Abrir archivo'.

4. El sistema muestra una ventana con un navegador de archivos para que introduzca el archivo que desea abrir.

5.Roberto entra a su carpeta personal y selecciona el archivo 'estudio de viabilidad.odt'.

6.El sistema abre el archivo en el editor y, automáticamente, muestra junto al mismo todas las incidencias que ha detectado.

Figura 5.4: Escenario 1: Apertura de un archivo en el editor

Nombre:

Generación de un informe de un documento de texto.

Descripción:

El usuario Roberto inicia la aplicación para obtener un informe de la memoria de su tesis.

Actores participantes:

Roberto.

Flujo de eventos:

1.Roberto inicia la aplicación y pulsa en 'Generar un informe'.

2. El sistema muestra un asistente para que el usuario elija el tipo de informe que desea.

3.Roberto pulsa sobre la opción 'Informe de documentos'.

4. El sistema muestra una ventana con un navegador de archivos para que introduzca el archivo que desea abrir.

5.Roberto abre la carpeta 'Memoria de la Tesis' y selecciona el archivo 'Versión final.pdf'.

6. El sistema muestra una ventana con un navegador de archivos para que introduzca el nombre del archivo de salida.

7.Roberto pulsa sin querer sobre el archivo 'Versión final.pdf' y comienza a temer lo peor.

8. El sistema pide al usuario que confirme si desea sobreescribir el archivo.

9.Roberto indica aliviado que no quiere sobreescribir este archivo.

9. El sistema vuelve a solicitar el archivo de salida.

10.Roberto indica que el informe se genere en el archivo 'Análisis Tesis.pdf'.

11. El sistema genera el informe y notifica al usuario de ello.

Figura 5.5: Escenario 2: Generación de un informe de un documento

Nombre:

Guardar la dirección web mostrada en el navegador web.

Descripción:

El usuario Roberto mientras emplea el navegador web descubre una página que desea conservar para analizar con mayor detalle.

Actores participantes:

Roberto.

Flujo de eventos:

1.Roberto encuentra una página web que desea conservar y pulsa en 'Guardar página'.

2. El sistema muestra una ventana con un navegador de archivos para que introduzca el nombre del archivo de salida.

3.Roberto introduce como archivo de salida 'web relevante.html'.

4. El sistema guarda la página y notifica al usuario de ello.

Figura 5.6: Escenario 3: Guardar la dirección web actual

Nombre:

Modificación de los elementos de un paquete de reglas.

Descripción:

El usuario Roberto quiere modificar un paquete de reglas para que trate con un caso que había pasado por alto.

Actores participantes:

Roberto.

Flujo de eventos:

1.Roberto inicia la aplicación y pulsa en 'Gestionar los paquetes de reglas'.

2. El sistema muestra la herramienta de gestión de paquetes de reglas.

3.Roberto selecciona el paquete a modificar y pulsa en 'Modificar paquete'.

4. El sistema abre el paquete seleccionado en la herramienta de modificación de paquetes.

5.Roberto añade una regla, modifica dos anteriores y pulsa en 'Guardar'.

6. El sistema almacena los cambios y notifica de ello al usuario.

Figura 5.7: Escenario 4: Modificar un paquete de reglas

Nombre:

Importar un paquete de reglas externo a la aplicación.

Descripción:

El usuario Roberto quiere importar a la aplicación un paquete de reglas que ha creado su compañero.

Actores participantes:

Roberto.

Flujo de eventos:

1.Roberto inicia la aplicación y pulsa en 'Gestionar los paquetes de reglas'.

2. El sistema muestra la herramienta de gestión de paquetes de reglas.

3.Roberto pulsa en 'Importar paquete'.

4. El sistema muestra una ventana con un navegador de archivos para que introduzca el archivo que desea importar.

5.Roberto selecciona el archivo 'racismo-backup.linguo'.

6. El sistema detecta dentro del archivo el paquete de reglas 'Racismo', nombre igual a otro paquete existente. Por tanto renombre el paquete como 'Racismo-1' y, como no existe ningún archivo con este nombre, lo almacena, notificando de ello al usuario.

Figura 5.8: Escenario 5: Importar un paquete de reglas

5.4. Diseño del sistema

En esta etapa del desarrollo nos centramos en establecer la arquitectura de la aplicación a través de la definición de sus datos, componentes e interfaz.

Es la actividad más compleja de todo el proceso de desarrollo, dando respuesta a cómo se van a realizar los requerimientos planteados al inicio del desarrollo.

Aunque aquí se presenten las distintas tareas realizadas de forma consecutiva, realmente su realización se ha llevado a cabo de forma paralela, trabajando en todas ellas de manera simultanea. La razón es simple, no podemos hacerlo de otra forma si queremos realizar un diseño correcto del sistema. Es fácil deducir en base a lo anterior que, el diseño, pasa por ser la actividad más laboriosa de todo el proceso de desarrollo.

Nuestra aplicación, al estar basada en una arquitectura de componentes, hace que sea innecesario realizar el diseño de los datos.

Al ser los componentes unidades funcionales mínimas, los tipos de datos manejados en cada componente son, en el peor de los casos, dos o tres distintos. Esto no quiere decir en absoluto que en una arquitectura de componentes sea posible prescindir del diseño de los datos, ya que esto solo se cumplirá en determinadas ocasiones.

En nuestro caso, debido a las características y arquitectura de la aplicación podemos omitir esta tarea, que viene a ser sustituida por un diseño detallado de los componentes que conforman la aplicación.

5.4.1. Diseño de los componentes

Nuestra aplicación se basa en una arquitectura de componentes OSGi. En el punto 4.2.3 comentamos que haríamos uso de distintos tipos de ellos, a continuación mostramos cuales serían los componentes, indicando tanto su tipo como la relación existente entre ellos.

5.4.1.1. Componentes

- **linguo**: Este componente es diferente al resto. Su función es la de integrar el resto de componentes de cara a proveer la aplicación final. En cierto modo podríamos decir que su labor es la de estructurar la interfaz, organizando los elementos visuales proporcionados por el resto de componentes y habilitando el acceso a los demás a través de los menús.
- **linguo.parser.core**: Núcleo central del análisis. Recibe un texto etiquetado, lo analiza y devuelve el resultado.
- **linguo.parser.dictionary**: Provee a la aplicación de un diccionario gramatical.
- **linguo.parser.html**: Extrae y marca el contenido de documentos html.
- **linguo.parser.plaintext**: Extrae y marca el contenido de documentos en texto plano.
- **linguo.parser.searcher**: Busca la existencia de determinados elementos en el texto.
- **linguo.parser.website**: Analiza sitios web.
- **linguo.reports**: Genera informes del análisis.
- **linguo.resultsprovider**: Suministra la información del análisis a los visores.
- **linguo.rulebase**: Almacena los paquetes de reglas de la aplicación.
- **linguo.rulemaker**: Gestiona la creación y modificación de reglas.
- **linguo.utils**: Utilidades de uso general.
- **linguo.viewer.openoffice**: Embebe la aplicación OpenOffice, permitiendo analizar el contenido de su editor.
- **linguo.viewer.results**: Ofrece visores informativos del análisis.
- **linguo.viewer.webnavigator**: Embebe el navegador XULRunner, permitiendo analizar el contenido de las páginas mostradas.

Además de estos componentes existen un conjunto de librerías que han sido convertidas en plug-ins para una mayor homogeneidad. Los 'plug-ins librerías' serían:

- **linguo.libs.htmlparser**
- **linguo.libs.hypersql**
- **linguo.libs.itext**
- **linguo.libs.jfreechart**
- **linguo.libs.mockito**
- **linguo.libs.noa**
- **linguo.libs.odfdom**
- **linguo.libs.poi**
- **linguo.libs.swtchart**

5.4.1.2. Tipo de los componentes

Componente	**Tipo**
linguo	Integración, Configuración
linguo.parser.core	Soporte
linguo.parser.dictionary	Soporte
linguo.parser.html	Análisis sintáctico
linguo.parser.plaintext	Análisis sintáctico
linguo.parser.searcher	Soporte
linguo.parser.website	Análisis sintáctico
linguo.reports	Visualización de resultados
linguo.resultsprovider	Integración
linguo.rulebase	Soporte
linguo.rulemaker	Creación y modificación, Configuración
linguo.utils	Soporte

Componente	Tipo
linguo.viewer.openoffice	Creación y modificación, Visualización de resultados
linguo.viewer.results	Visualización de resultados
linguo.viewer.webnavigator	Creación y modificación, Visualización de resultados
linguo.libs.htmlparser	Soporte
linguo.libs.hypersql	Soporte
linguo.libs.itext	Soporte
linguo.libs.jfreechart	Soporte
linguo.libs.mockito	Soporte
linguo.libs.noa	Soporte
linguo.libs.odfdom	Soporte
linguo.libs.poi	Soporte
linguo.libs.swtchart	Soporte

5.4.1.3. Grafos de dependencias

A continuación se incluyen los grafos de dependencias de los componentes.

Se ha separado el grafo en varias figuras para facilitar su comprensión. También se ha omitido de forma deliberada las librerías (componentes 'linguo.libs.*') y el componente 'linguo.utils' a fin de simplificar la representación.

Grafo de dependencias de linguo

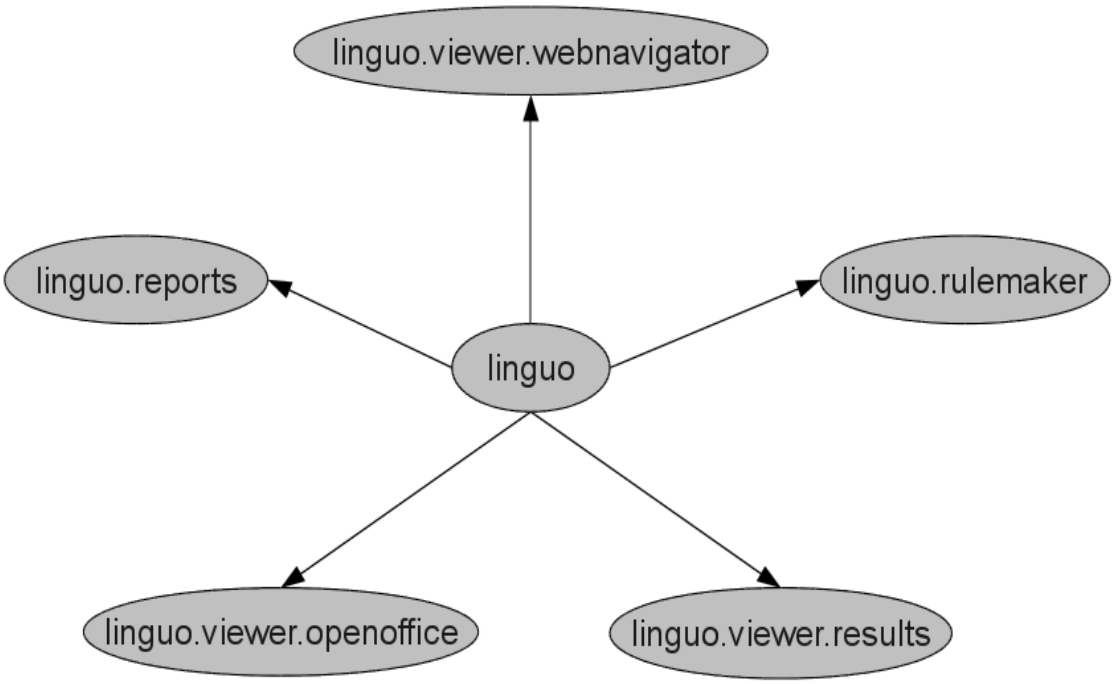


Figura 5.9: Grafo de dependencias de linguo

Grafo de dependencias de linguo.reports

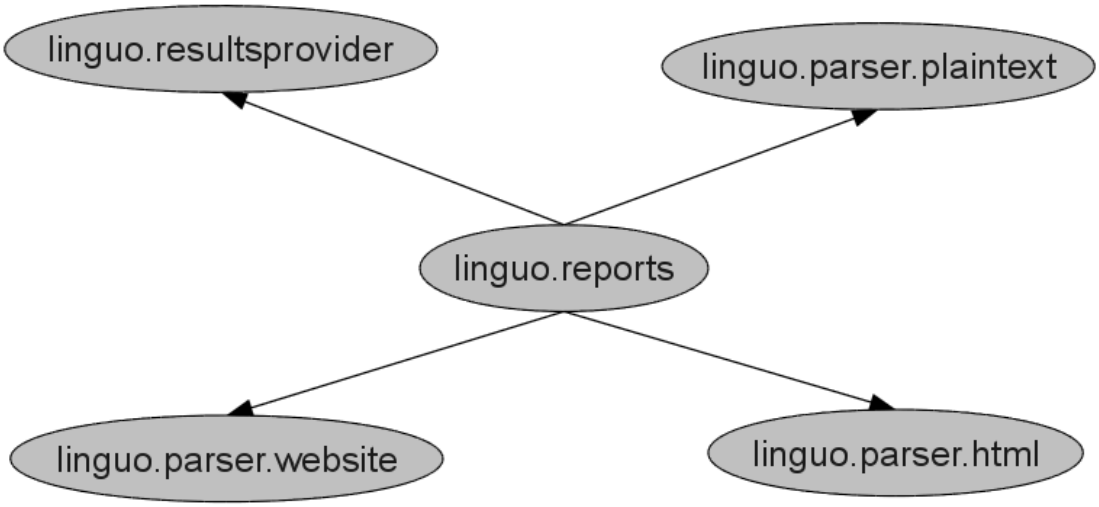


Figura 5.10: Grafo de dependencias de linguo.reports

Grafo de dependencias de linguo.rulemaker

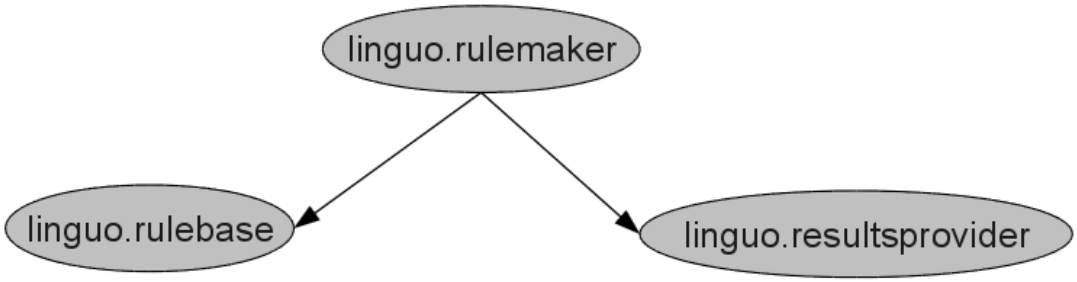


Figura 5.11: Grafo de dependencias de linguo.rulemaker

Grafo de dependencias de linguo.viewer.openoffice

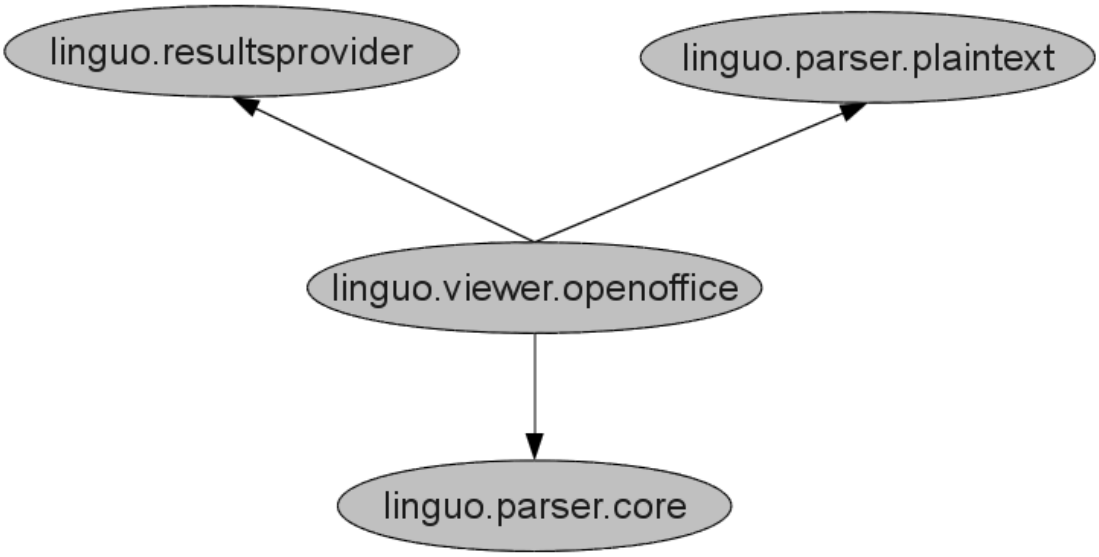


Figura 5.12: Grafo de dependencias de linguo.viewer.openoffice

Grafo de dependencias de linguo.viewer.results

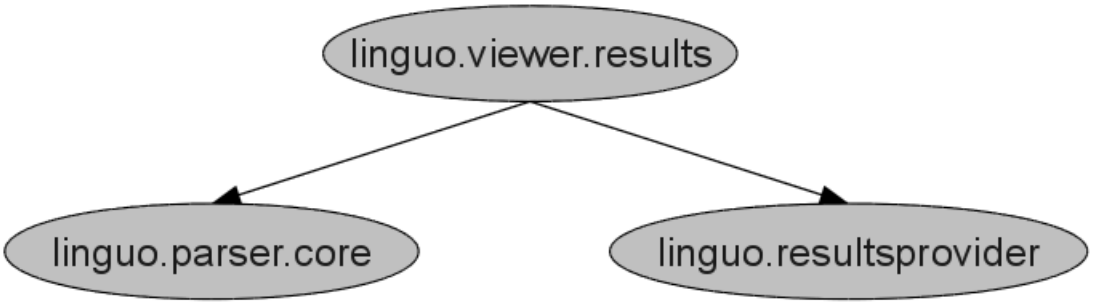


Figura 5.13: Grafo de dependencias de linguo.viewer.results

Grafo de dependencias de linguo.viewer.webnavigator

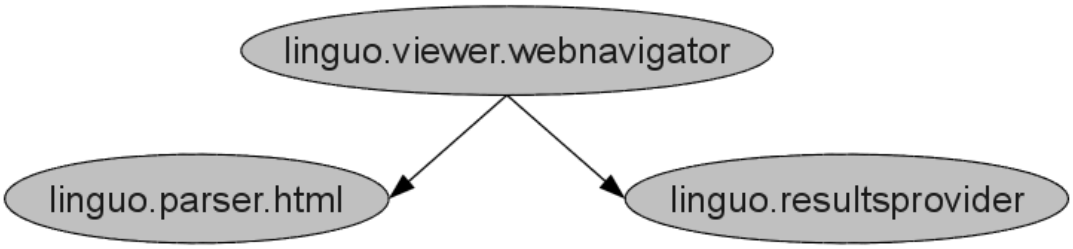


Figura 5.14: Grafo de dependencias de linguo.viewer.webnavigator

Grafo de dependencias del resto de componentes

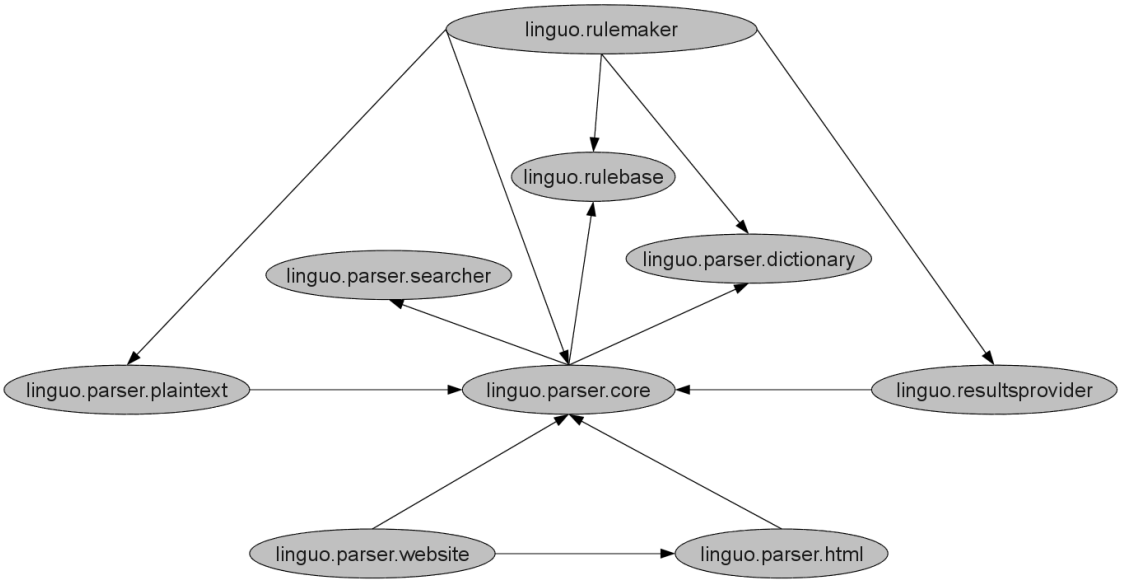


Figura 5.15: Grafo de dependencias del resto de componentes

5.4.2. Diseño de la interfaz

En esta fase del diseño del sistema definiremos cuál va a ser la forma visual de la interfaz entre el usuario y la aplicación, así como su comportamiento. Constituye una fase crítica que puede condicionar el resultado final obtenido. Así, será de vital importancia que la interfaz cumpla con unos principios mínimos de usabilidad a fin de facilitar la interacción con la aplicación. Si esto no se cumple, será muy difícil que los usuarios se sientan cómodos usando la aplicación, o lo que es peor, que estos desechen la aplicación tras recibir una primera impresión nefasta.

En el diseño de la interfaz abordamos aspectos como qué apariencia tendrá, cómo se distribuyen los distintos elementos, cómo puede interactuar el usuario con la misma, cuáles serán los términos a usar o cuál será su aspecto estético.

Los siguientes puntos de este apartado muestran las tareas que se han llevado a cabo en esta fase del diseño.

5.4.2.1. Guía de estilo

Mediante la guía de estilo especificaremos cuál es el estilo deseado para la interfaz, buscando que este sea atractivo y que esté en consonancia con el de la plataforma de ejecución de la aplicación. De esta forma, deberemos definir los colores a usar, la separación entre los distintos elementos, el aspecto de los menús y demás elementos gráficos.

La aplicación hará uso del Toolkit gráfico SWT lo que nos facilitará en parte esta tarea. SWT es un Toolkit Java para el diseño de interfaces de usuario desarrollado por IBM [50]. Los componentes SWT emplean los componentes nativos de la plataforma en la que el programa se esté ejecutando, adaptándose a la misma y consiguiendo una mayor integración con el resto de aplicaciones [54].

Al usar SWT, podemos simplificar la guía de estilo, ya que será el propio Toolkit el que se encargará de construir una interfaz en consonancia con el resto de aplicaciones de la plataforma de ejecución. Así, aspectos como la separación entre los distintos elementos de la interfaz, formatos de las fuentes, forma de los cuadros de diálogo o apariencia de los mensajes de error, estarán resueltos de forma óptima al hacerse exactamente igual que en

el resto de aplicaciones del sistema (siempre y cuando hayan sido desarrolladas siguiendo la guía de estilo para el gestor gráfico de ventanas en uso).

En nuestra caso, la definición de una guía de estilo para los distintos elementos de la interfaz, pasa por ser una tarea totalmente innecesaria, pudiendo afirmar que su confección sería en todo caso contraproducente. Decimos esto en base a que:

- La creación de una guía de estilo que cumpla de forma estricta con las recomendaciones para un gestor de ventanas dado, conseguiría en el mejor de los casos, que la aplicación se viese igual que si dejásemos a SWT hacer su trabajo.
- Si diseñamos una guía de estilo en base a las características propias de una plataforma, el aspecto de la aplicación solo estará en consonancia con las aplicaciones de este sistema, rompiéndose la integración visual en el resto de plataformas.

Como vemos, el disponer de un Toolkit como SWT, capaz de adaptar la apariencia de la interfaz a la del entorno de ejecución, hace que esta tarea sea desaconsejada.

De esta forma, para saber cuál es el estilo final de los elementos de la aplicación, no nos queda más que remitir a las guía de estilo de la plataforma de ejecución.

Los únicos apartados que incluiremos en esta guía de estilo serán los informes y logos, al ser su apariencia común independientemente del entorno de ejecución. Pasemos a ver el estilo de ambos.

Informes

Los informes ofrecen una visión de distintos aspectos del análisis. Se generarán en formato PDF, por lo que se verán igual independientemente de la plataforma en la que se visualicen.

Para una correcta visualización de los mismos, deberemos considerar aspectos tales como las fuentes a emplear (tamaño, color, efectos de fuente, etc), separación entre bloques de información diferentes y estilos de la página. A continuación detallamos las especificaciones de estilo que guiarán su diseño.

- **Información del documento**

- Título: Informe de Linguo.
- Creador: Linguo.
- Autor: Linguo.

- **Fuentes**
  - *Estándar* (Comprende texto informativo, valores de celdas de tablas, numeración de páginas y etiquetas de gráficas)
    - Tipo: Helvética (Winansi si no está disponible).
    - Tamaño: 12pt.
    - Color: Negro.
    - Efectos de fuente: Ninguno.
  - *Etiquetas informativas*
    - Tipo: Helvética (Winansi si no está disponible).
    - Tamaño: 12pt.
    - Color: Negro.
    - Efectos de fuente: Negrita.
  - *Secciones*
    - Tipo: Helvética (Winansi si no está disponible).
    - Tamaño: 20pt.
    - Color: Negro.
    - Efectos de fuente: Negrita.
  - *Título*
    - Tipo: Helvética (Winansi si no está disponible).
    - Tamaño: 40pt.
    - Color: Negro.
    - Efectos de fuente: Negrita.

- **Especificaciones de página**
  - Tamaño: A4 (210 x 297 mm).

- Fondo: Blanco.
- Márgenes
  - Superior: 13,5 mm.
  - Inferior: 13,5 mm.
  - Izquierdo: 13 mm.
  - Derecho: 10 mm.

- **Especificaciones del documento**
  - Separación entre secciones: 18mm.
  - Interlineado: Simple.
  - Alineación de texto: Izquierda.
  - Fondo de página: Blanco.

- **Tablas**
  - *Cabecera*
    - Alineación: Centro.
    - Fondo: Gris 80 % de opacidad.
  - *Celdas*
    - Alineación: Izquierda.
    - Fondo: Blanco.

- **Gráficas**
  - Tamaño: 70 x 70 mm.
  - Fondo ejes: Transparente.
  - Fondo valores: Gris.
  - Color valores: Rojo (#ff6262).

- **Marcas de agua**
  - Logotipo aplicación sobre fondo azul claro a 15 % de opacidad.

- Situación
  - Impresión a una cara por folio
    - 1/3 del margen izquierdo ocupando todo el alto.
  - Impresión a doble cara
    - Páginas impares: 1/3 del margen izquierdo ocupando todo el alto.
    - Páginas pares: 1/3 del margen derecho ocupando todo el alto.

- **Numeración de páginas**

  - Formato: 'Página X de Y' siendo X el número de página e Y el total de páginas del documento.
  - Situación
    - Impresión a una cara por folio
      - Alineado a la derecha en el pie de página.
    - Impresión a doble cara
      - Páginas impares: Alineado a la derecha en el pie de página.
      - Páginas pares: Alineado a la izquierda en el pie de página.

Logo

El logo de la aplicación la identificará y diferenciará, asignándole una imagen que el usuario podrá identificar con la misma.

Se incluirán dos logos, un logotipo[4] y un isotipo[5]. Ambos se mostrarán en distintos lugares de la aplicación (informes, ventana de carga, icono, etc).

El logotipo será el nombre del programa con la letra 'g' terminando en forma de clavija. Esta clavija hace referencia a la naturaleza de la aplicación como conjunto de Plug-ins[6].

[4]Representación tipográfica de la aplicación.
[5]Icono visual que remite a la aplicación.
[6]Del inglés enchufable.

Figura 5.16: Logotipo Linguo

El isotipo será una pieza de puzzle con la letra 'g' característica del logotipo. La forma del isotipo como pieza de puzzle hace referencia a la arquitectura de componentes[7] de la aplicación.

Figura 5.17: Isotipo Linguo

En algunas ocasiones se hará uso de ambos de forma conjunta (marcas de agua, navegador, pantalla de carga, etc). En estas situaciones se dispondrá siempre del isotipo a la izquierda del logotipo, separados por una distancia igual a la mitad del tamaño del isotipo.

[7]Los cuales podrían ser vistos como las distintas piezas que conforman la aplicación.

Figura 5.18: Logotipo e Isotipo Linguo

5.4.2.2. Diccionario

El diccionario de la aplicación contiene los términos más significativos empleados a lo largo de la interfaz, junto con el concepto al cual nos referimos al emplearlos. Su uso nos aporta varias ventajas:

- **Uso de una terminología común**: Evitaremos denominar un mismo concepto de formas diferentes en distintos lugares.
- **Eliminación de la ambigüedad**: Disponer de un documento en el cual estén contenidos todos los términos usados así como su significado concreto nos permite descubrir aquellas denominaciones que puedan resultar confusas.
- **Simplificación del desarrollo**: El uso del diccionario clarifica el desarrollo, tanto de la interfaz como del resto de la aplicación, ya que al usar una determinada palabra, sabemos en todo momento a lo que nos estamos refiriendo.

Como vemos, el diccionario es un documento sencillo de confeccionar y que nos aporta una serie de ventajas significativas. Si es cierto que su uso es a veces innecesario, especialmente en pequeños problemas, sin embargo, a medida que aumenta el tamaño de la aplicación, resulta más útil.

En una aplicación constituida en base a componentes, su elaboración pasa a ser prácticamente obligatoria puesto que, de otro modo, ¿cómo podemos garantizar qué componentes desarrollados por distintos grupos de usuarios utilizan la misma terminología

para referirse a un mismo concepto?. La respuesta a esta cuestión pasa necesariamente por la generación de un diccionario.

A continuación mostramos el diccionario de nuestra aplicación.

Término	**Definición**
Abrir archivo	Abrir un nuevo archivo dentro de OpenOffice.
Abrir el informe generado	Apertura posterior del informe una vez generado.
Adjetivo	Categoría gramatical para los adjetivos.
Anterior (Navegador)	Ir a la dirección anterior de navegación.
Anterior (Tipo de contexto)	Considerar las palabras anteriores a la actual.
Añadir	Crear un nuevo elemento del tipo especificado.
Archivo	Menú archivo de la aplicación.
Archivo a analizar	Ruta en el sistema del archivo a analizar.
Aumentar subnivel	Considerar la regla como una regla hija de la anterior.
Aumentar zoom	Aumentar el tamaño de la página que se muestra en el navegador.
Bajar regla	Evaluar la regla después de la siguiente.
Borrar (Conjuntos de términos)	Eliminar un término del conjunto de términos.
Borrar (Generador de reglas)	Borrar un elemento del tipo especificado.
Buscar	Buscar un archivo para su análisis.
Cancelar	Deshacer cualquier modificación realizada.
Características del informe	Características de impresión del informe a generar.
Color	Color en el que se mostrarán las incidencias de un paquete de reglas.
Clase de palabras	Clase gramatical de palabras.
Completo	Tipo de contexto que considera las palabras anteriores y posteriores a la actual.
Conjunción	Categoría gramatical para las conjunciones.
Conjunto de términos	Conjunto de términos relacionados.
Contraer	Ocultar todos los subelementos de un elemento.
Cualquiera	Omitir la búsqueda por variante gramatical.
Descripción	Texto descriptivo de un determinado elemento.
Determinante	Categoría gramatical para los determinantes.

Término	**Definición**
Dirección	Dirección web a analizar en un informe de sitios.
Disminuir subnivel	Considerar que la regla no es hija de su padre actual.
Disminuir zoom	Disminuir el tamaño de la página que se muestra en el navegador.
Dispersión	Distribución de incidencias en el documento.
Doble cara	Tipo de informe preparado para ser impreso a doble cara.
Documento	Documento a analizar para la generación de un informe.
Editar	Editar el valor de un término de un conjunto de términos.
Elegir	Seleccionar un color concreto.
Elemento	Parámetro de una regla que tomará un valor concreto.
Estadísticas	Información estadística de un análisis.
Expandir	Mostrar todos los subelementos de un elemento.
Exportar paquete de reglas	Generar un archivo de un paquete de reglas.
Femenino	Variante gramatical femenina de un término.
Femenino plural	Variante gramatical femenina plural de un término.
Femenino singular	Variante gramatical femenina singular de un término.
Forma compilada	Apariencia de una regla una vez sustituidos los elementos por valores concretos.
Formato	Formato de un tipo de regla mostrando los distintos elementos del mismo.
Fuente de análisis	Contenido a analizar.
Generador de reglas	Herramienta para trabajar sobre las reglas de la aplicación.
Gráfico	Visor de información gráfica de un análisis.
Guardar	Almacenar cualquier cambio realizado.
Guardar archivo	Almacenar el archivo abierto en OpenOffice.
Guardar página	Almacenar la página web mostrada actualmente.

Término	Definición
Herramientas	Menú de herramientas de la aplicación.
Ilimitado	Tipo de contexto que no considera tamaño.
Importar paquete de reglas	Añadir un paquete de reglas desde un archivo externo.
Incidencia	Ocurrencia que concuerda con las reglas fijadas.
Inclusión del centro	Considerar la posición central de un contexto.
Información a añadir	Información que se debe añadir a un informe.
Información agregada	Información generada como suma de distintos análisis sobre un mismo contenido.
Información separada	Información con cada paquete de reglas.
Informe	Documento que muestra información del análisis.
Inicio	Dirección inicial del navegador.
Ir	Abrir la dirección introducida en la barra del navegador.
Linguo	Nombre de la aplicación.
Listado de páginas	Incluir en un informe el listado de páginas analizadas en un sitio web.
Masculino	Variante gramatical masculina de un término.
Masculino plural	Variante gramatical masculina plural de un término.
Masculino singular	Variante gramatical masculina singular de un término.
Modificar	Modificar el elemento seleccionado.
Navegador	Visor web de la aplicación.
Negación	Considerar que la regla no se debe verificar.
Neutro	Variante gramatical neutra de un término.
Neutro plural	Variante gramatical neutra plural de un término.
Neutro singular	Variante gramatical neutra singular de un término.
Nivel de profundidad	Nivel de profundidad en el análisis de un sitio web.
No (Inclusión del centro)	No incluir el elemento central en el contexto.
No (Negación)	No negar el resultado.
Nombre propio	Conjunto de nombres propios.
Nuevo	Crear un nuevo elemento del tipo especificado.

Término	**Definición**
Nuevo archivo	Crear un nuevo documento en OpenOffice.
OpenOffice	Visor OpenOffice de la aplicación.
Palabra	Término en una variante gramatical concreta.
Palabra vacía	Palabras comunes que aportan poco significado.
Paquete de reglas	Agrupación de conjuntos de términos y reglas.
Paquetes de reglas	Agrupación de paquetes de reglas.
Parar	Detener la carga de la página web actual.
Posición	Situación en la que apareció una incidencia dada.
Posterior	Considerar las palabras posteriores a la actual.
Preposición	Categoría gramatical para las preposiciones.
Ratios	Relación entre número de términos y de palabras.
Refrescar	Volver a cargar la dirección web actual.
Regla	Tipo de regla con valores establecidos para sus elementos.
Reiniciar zoom	Establecer el tamaño de la página web que se muestra en el navegador al original.
Resultados	Visor de resultados de análisis.
Salir	Abandonar la aplicación.
Seleccionar visor	Especificar el visor de documentos a mostrar.
Si (Inclusión del centro)	Incluir el elemento central en el contexto.
Si (Negación)	Negar el resultado.
Siguiente	Ir a la dirección siguiente en el navegador web.
Sitio web	Conjunto de páginas contenidas en una misma dirección web.
Subir regla	Evaluar la regla antes que la anterior.
Sustantivo	Categoría gramatical para los sustantivos.
Tamaño del contexto	Número de elementos a considerar en el contexto.
Término	Palabra con independencia de su variante gramatical.
Tipo de contexto	Forma del contexto en el que buscar.
Tipo de regla	Variante de regla para realizar una búsqueda concreta.
Valor	Valor dado a un elemento de una regla.

Término	Definición
Variante gramatical	Variación gramatical de un término.
Ver	Menú para mostrar u ocultar los visores de resultados.
Visor	Visor de documentos de la aplicación.

5.4.2.3. Metáforas

Entendemos por metáfora el uso de algún objeto o proceso real, familiar para el usuario, con el fin de representar alguna tarea abstracta realizada por el ordenador. Las metáforas son un recurso común en cualquier programa informático, y muchas de ellas forman ya parte del conocimiento general de cualquier usuario (papelera para tirar documentos, un disquete de 3.5 para guardar archivos, una flecha para representar la posición apuntada por el ratón, etc). Así, cualquier proceso abstracto de un programa, puede ser representado mediante una metáfora, la cual nos permitirá identificar este proceso y realizarlo.

El diseño de las metáforas es un aspecto crucial del diseño de la interfaz al ser el nexo entre usuario y tareas, si esta unión falla, el usuario no sabrá como realizar las tareas. El diseño de metáforas pasa por tanto por establecer metáforas efectivas, entendiendo como tales, aquellas que cumplen con su cometido de forma eficaz. Para que una metáfora sea eficaz, es esencial que represente algo conocido para el usuario, y además, que se adapte al tipo concreto de usuario al que la aplicación está destinada.

En el diseño de una interfaz de usuario las metáforas pueden considerarse de dos tipos, de proceso o visuales. Metáforas de proceso típicas serían el 'copiar y pegar' arrastrando el icono de un archivo de una carpeta a otra o la selección del elemento concreto representado por una tabla al pulsar sobre la fila en la que se muestra su descripción. Como ejemplos de metáforas visuales tenemos los iconos, botones de pulsación, cursores, etc.

Las metáforas de proceso no son siempre bien entendidas por los desarrolladores y muchas veces se cae en el error de obviarlas en la etapa de diseño. Esto no quiere decir que finalmente no se incluyan, si no que no se realiza un diseño detallado de las mismas en búsqueda de la metáfora más efectiva.

El diseño de las metáforas no debería ser en ningún caso un proceso exhaustivo, existiendo varias metáforas que, por su extendido uso, suelen ser comunes a cualquier interfaz de usuario (figura 5.19). Las metáforas que nos deben preocupar al realizar el diseño son aquellas que son exclusivas a nuestra interfaz, sobreentendiéndose que, el resto de ellas, son las comunes en una interfaz de usuario.

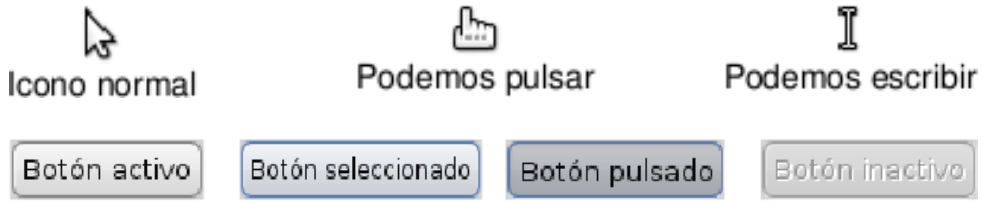


Figura 5.19: Ejemplos de metáforas comunes a cualquier interfaz de usuario

A continuación mostramos las metáforas más relevantes de nuestra interfaz. Para diferenciar entre las metáforas de proceso y las visuales emplearemos las letras 'P' y 'V'.

1) Búsqueda en el visor OpenOffice de cualquier incidencia al pulsar sobre ella en el visor de resultados (P)

Las incidencias encontradas al analizar el visor OpenOffice se muestran en el visor de resultados. Si el usuario realiza la pulsación sobre cualquiera de estas incidencias automáticamente se buscarán dentro de OpenOffice, pudiendo comprobar el usuario instantáneamente donde se encuentra dicha incidencia en el documento.

2) Ordenación de los términos de un conjunto de términos al realizar cualquier modificación sobre ellos (P)

Cada vez que el usuario inserte o modifique un término en un conjunto de términos se llevará a cabo la ordenación de la tabla que mostrará estos términos.

3) Marcado en color de las incidencias detectadas por el visor web (V)

Todas las incidencias detectadas dentro del navegador web se mostrarán al usuario haciendo uso de un cambio de color del fondo de las mismas. Este color será igual al

seleccionado por el usuario en las opciones del paquete de reglas que haya marcado la palabra como incidencia.

4) Subrayado en color de las incidencias detectadas por el visor de pruebas (V)

Las incidencias detectadas dentro del visor de pruebas se mostrarán al usuario haciendo uso de un subrayado en color de las mismas. Al igual que en el caso del navegador, este color será igual al establecido en las preferencias del paquete de reglas que indique la incidencia.

5) Iconos (V)

Para que el usuario pueda identificar con facilidad la acción a realizar añadiremos iconos para aquellas más comunes. Los iconos usados han sido tomados del proyecto Tango! [55] por dos motivos:

- Los iconos liberados por el proyecto Tango! son de dominio público.
- Existen multitud de aplicaciones que hacen uso de ellos por lo que es más fácil que resulten familiares al usuario y que, por tanto, la metáfora sea lo más efectiva posible.

En la tabla 5.6 mostramos los iconos empleados en nuestra aplicación así como su función concreta dentro de la misma.

Icono	Función
	Cargar dirección siguiente
	Nuevo archivo*
	Abrir archivo
	Cargar dirección anterior
	Detener la carga actual
	Refrescar la página
	Ir a la página inicial
	Ir a la página introducida
	Disminuir zoom
	Aumentar zoom
	Guardar la página actual
	Acerca de Linguo
	Cambiar visor
	Salir de la aplicación
	Nuevo informe
	Generador de reglas
	Paquetes de reglas
	Paquete de reglas
	Conjunto de términos
	Regla
	Visor de resultados*
	Visor de estadísticas

Cuadro 5.6: Iconos de la interfaz

* Nunca son mostrados de forma conjunta, apareciendo en el visor OpenOffice oculto el icono de este visor.

5.4.2.4. Prototipos de la interfaz

Los prototipos de la interfaz nos servirán para esbozar la interfaz de usuario de la aplicación. Debemos considerar que estos prototipos no reflejan exactamente el diseño final de la interfaz, más bien dan una idea de como será, pudiendo ser modificados posteriormente si se decide optar por una disposición distinta.

Para realizar los prototipos emplearemos diseños de pantallas que básicamente son dibujos a lápiz y papel. Los diseños de pantallas son un forma rápida y barata de analizar la disposición de la interfaz, pudiendo considerar varios estilos distintos de forma sumamente sencilla. Una vez decidido cuál será el diseño de la interfaz, estos diseños se desecharán, por lo que no debemos preocuparnos excesivamente por la calidad de los mismos.

Además de las pantallas que diseñaremos, existirán otras pantallas adicionales que nos vendrán proporcionadas por el sistema operativo tales como la búsqueda de archivos, la selección de color o las ventanas de progreso. Estas ventanas pueden ser omitidas del diseño de pantallas.

Pantalla principal navegador

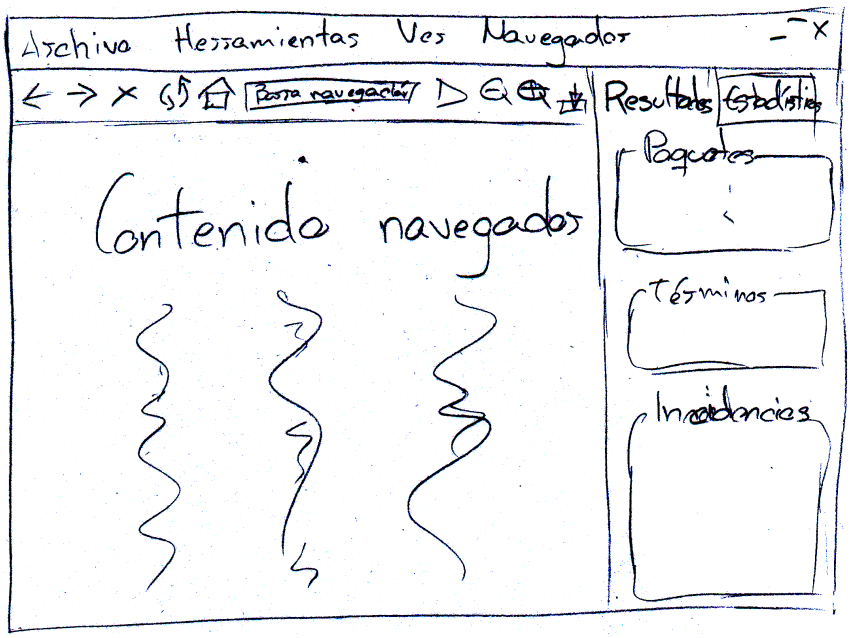


Figura 5.20: Pantalla principal navegador

Figura 5.21: Visor estadísticas

Pantalla principal OpenOffice

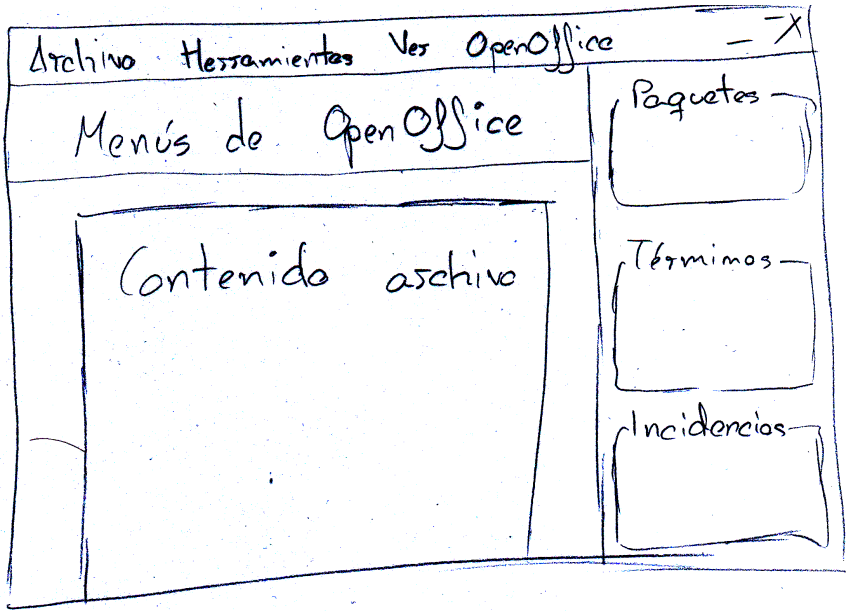


Figura 5.22: Pantalla principal OpenOffice

Menús

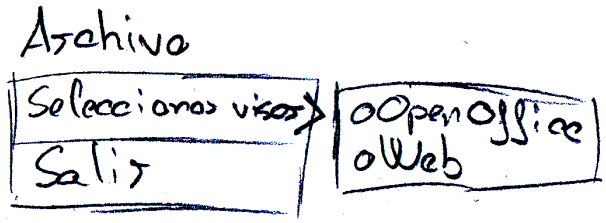


Figura 5.23: Menú archivo

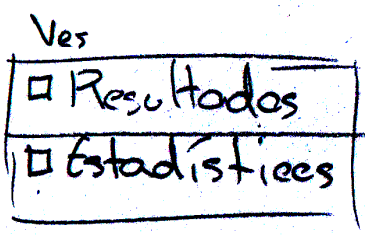


Figura 5.24: Menú ver en vista navegador

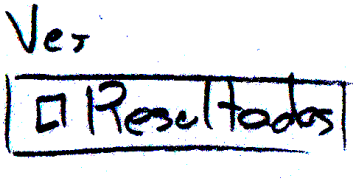


Figura 5.25: Menú ver en vista OpenOffice

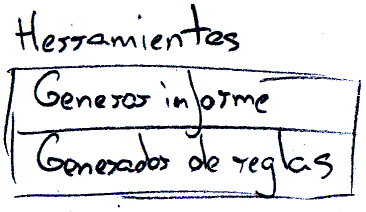


Figura 5.26: Menú herramientas

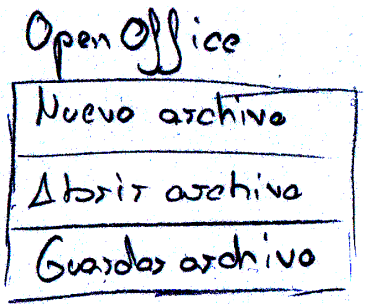


Figura 5.27: Menú OpenOffice

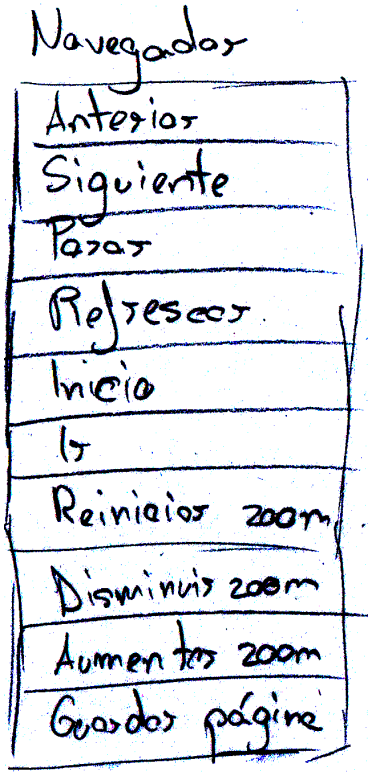


Figura 5.28: Menú navegador

Generador de informes

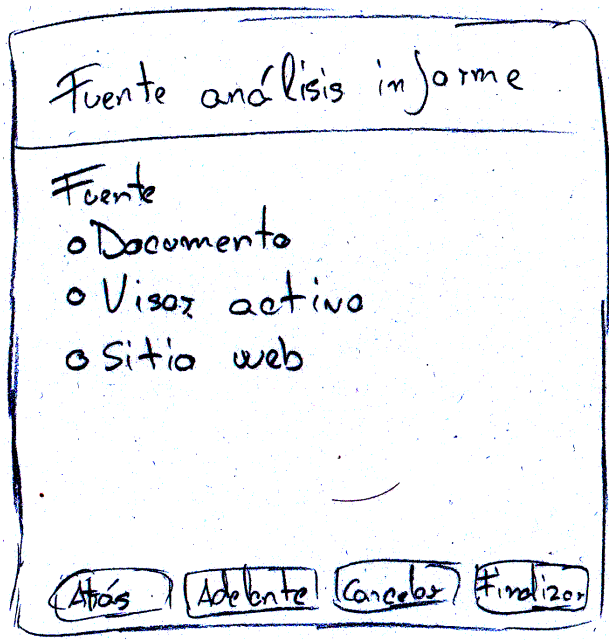


Figura 5.29: Tipo de informe

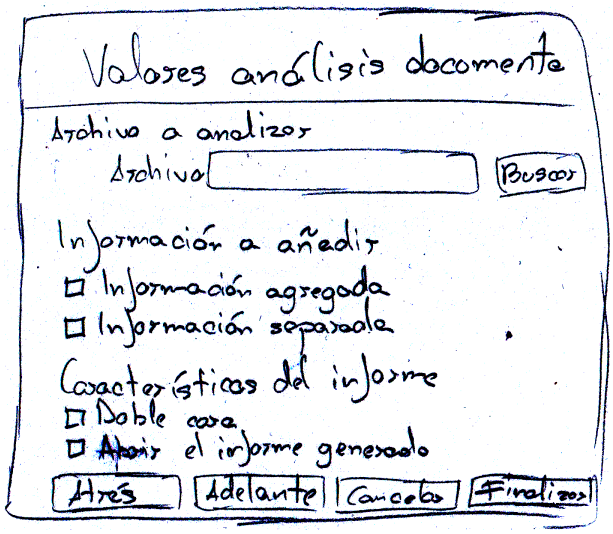


Figura 5.30: Informe de documento

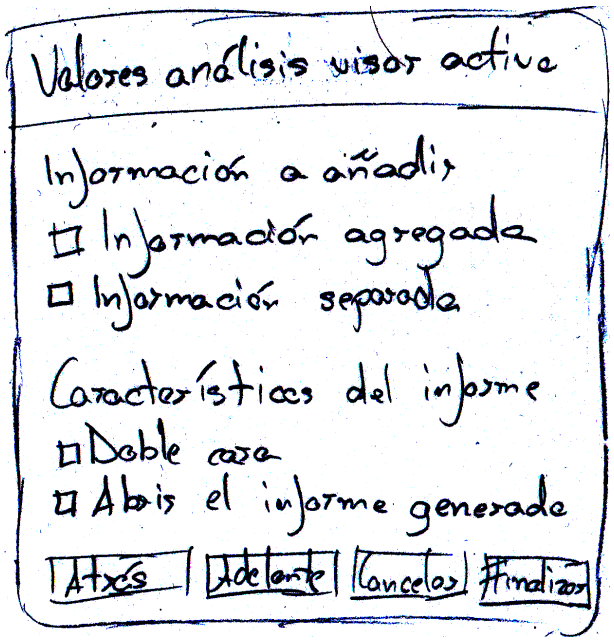


Figura 5.31: Informe de visor activo

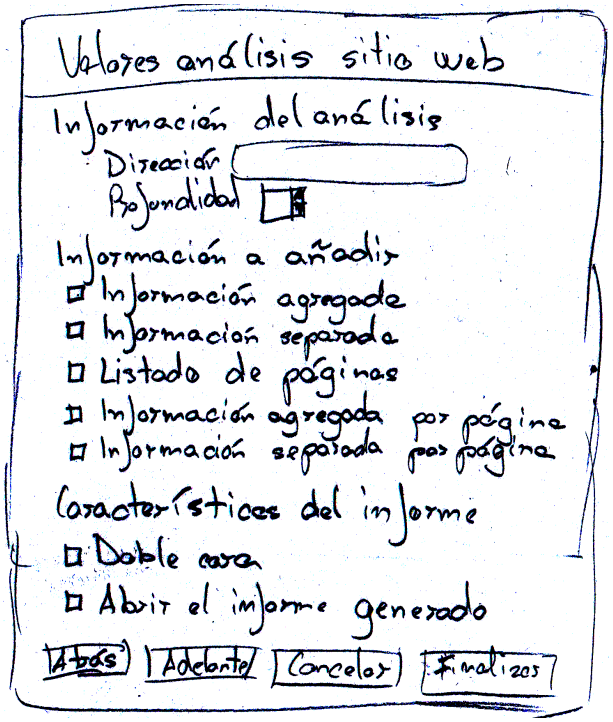


Figura 5.32: Informe de sitio web

Generador de reglas

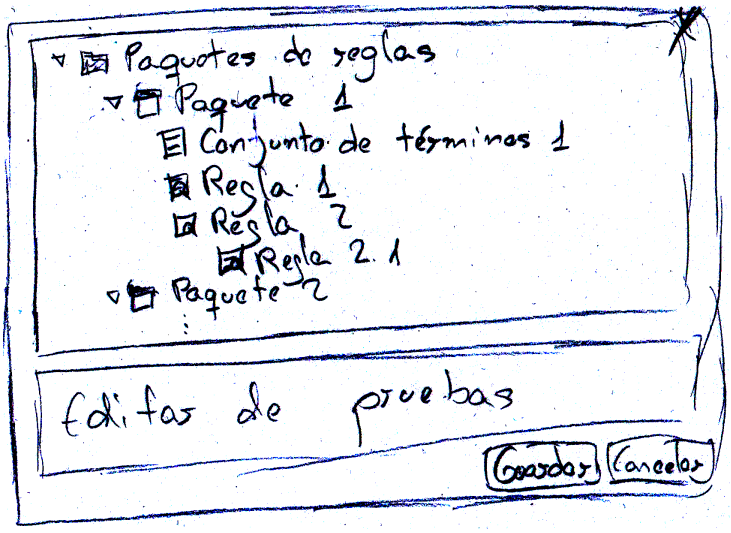


Figura 5.33: Generador de reglas

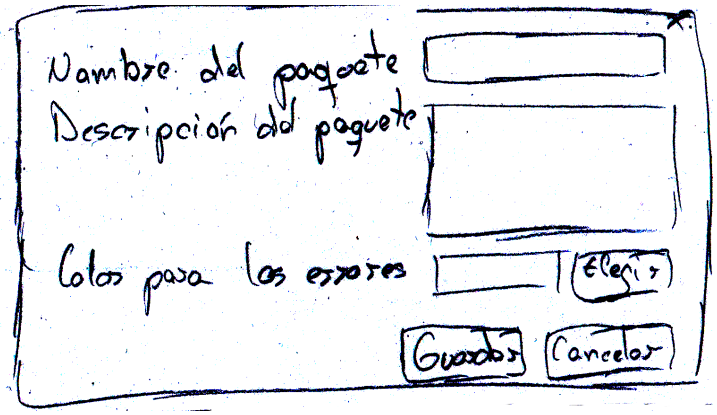


Figura 5.34: Nuevo paquete de reglas

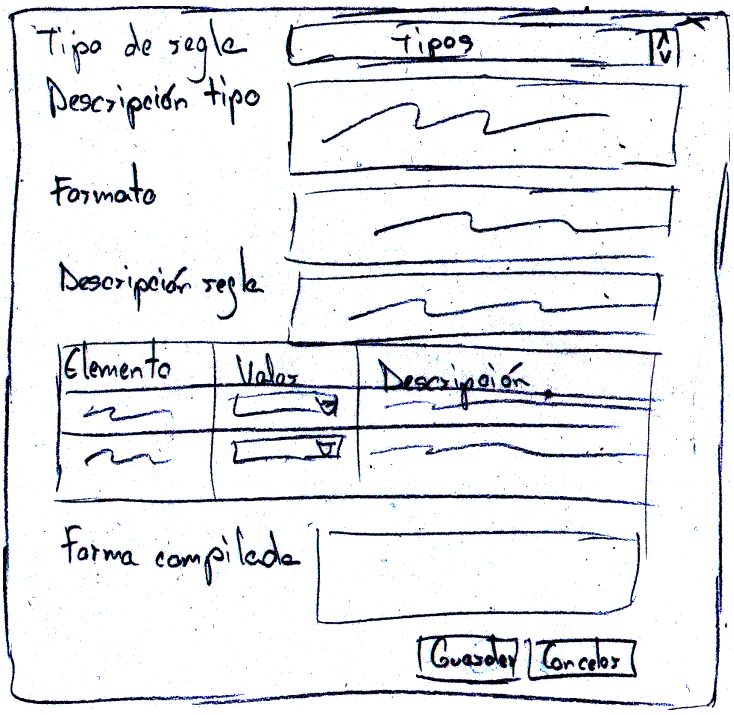


Figura 5.35: Nueva regla

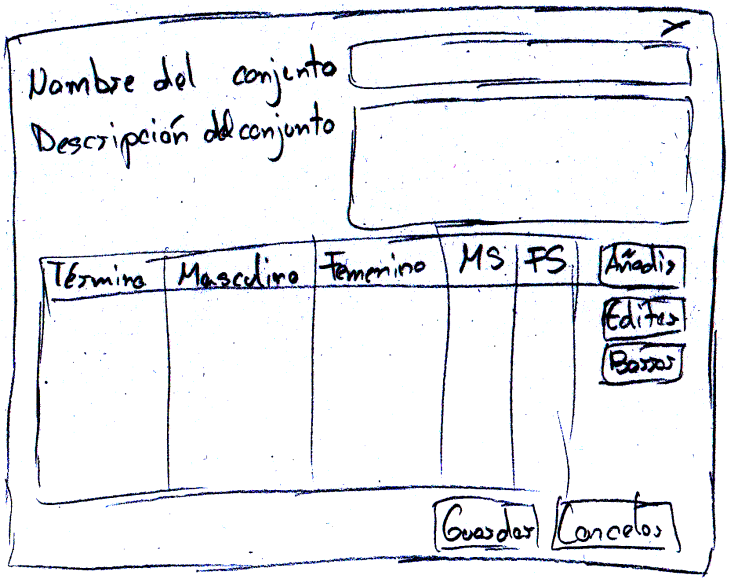


Figura 5.36: Nuevo conjunto de términos

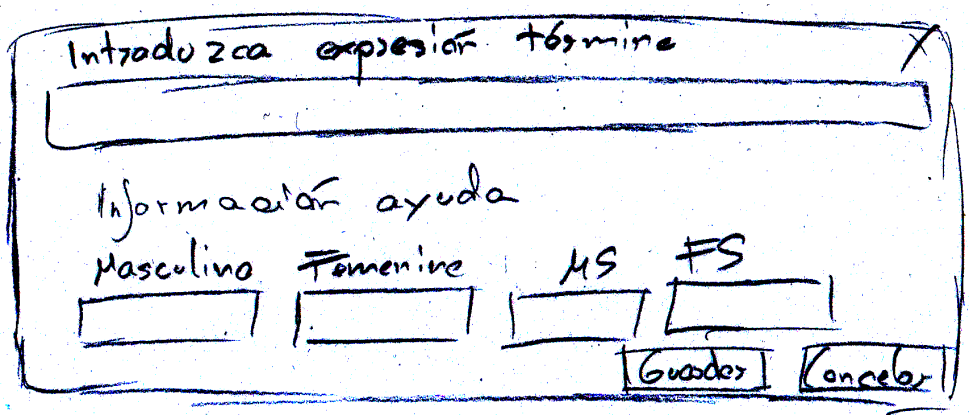


Figura 5.37: Nuevo término

Menús contextuales generador de reglas

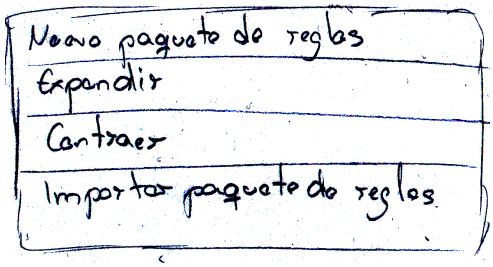


Figura 5.38: Menú contextual paquetes de reglas

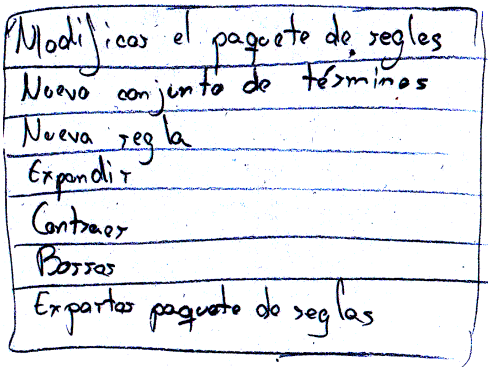


Figura 5.39: Menú contextual paquete de reglas

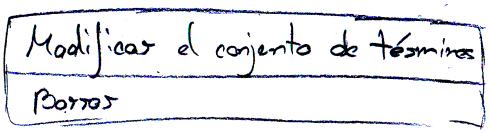


Figura 5.40: Menú contextual conjunto de términos

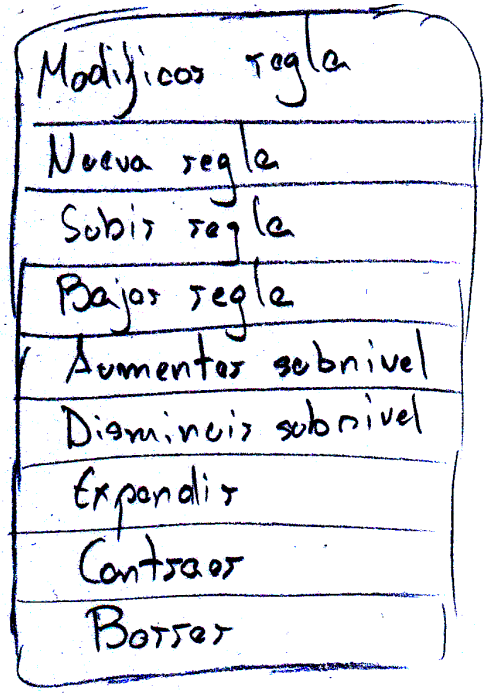


Figura 5.41: Menú contextual regla

5.4.2.5. Caminos de navegación

Los diseños de pantalla realizados en el punto anterior dan una idea de la interfaz de la aplicación de forma estática, sin embargo, no nos ofrecen ningún tipo de información de cómo será la interacción en sí. Para ello, debemos recurrir a los caminos de navegación, los cuales nos ofrecen una visión de como de fluida y comprensible resulta la acción para el usuario.

Existen distintos modos de abordar los caminos de navegación, principalmente mediante snapshots (similares a las pantallas pero con valores concretos asignados), escenarios y storyboards. Nosotros emplearemos estos últimos para estudiar los caminos de

navegación.

Los storyboards, también conocidos como guiones gráficos, son un conjunto de pantallas unidas con el fin de analizar una posible interacción. De esta forma, para confeccionar un storyboard concreto, utilizaremos los diseños de pantallas e iremos uniéndolos con flechas para conocer el conjunto de pasos necesarios para completar una acción. La secuencia de flechas nos dará una idea de lo fluida que resulta la interacción con la aplicación. Si se da el caso de que el número de pasos en un storyboard concreto resulta excesivo o carente de lógica, estaremos ante un claro indicador de que la interfaz no es tan usable como debería.

No es necesario incluir todos los storyboards posibles, ya que en tal caso, más que una ayuda serían un lastre. Debemos diseñar únicamente storyboards para aquellas acciones que consideremos más complejas, cerciorándonos así de si su realización está correctamente resuelta en la interfaz. Incluiremos storyboards para las siguientes acciones:

- Creación de un nuevo paquete de reglas. (5.42)
- Generación de un informe de un sitio web. (5.42)
- Cambiar la vista del navegador. (5.44)
- Exportar un paquete de reglas. (5.45)
- Modificación de un término de un conjunto de términos. (5.46)

Creación de un nuevo paquete de reglas

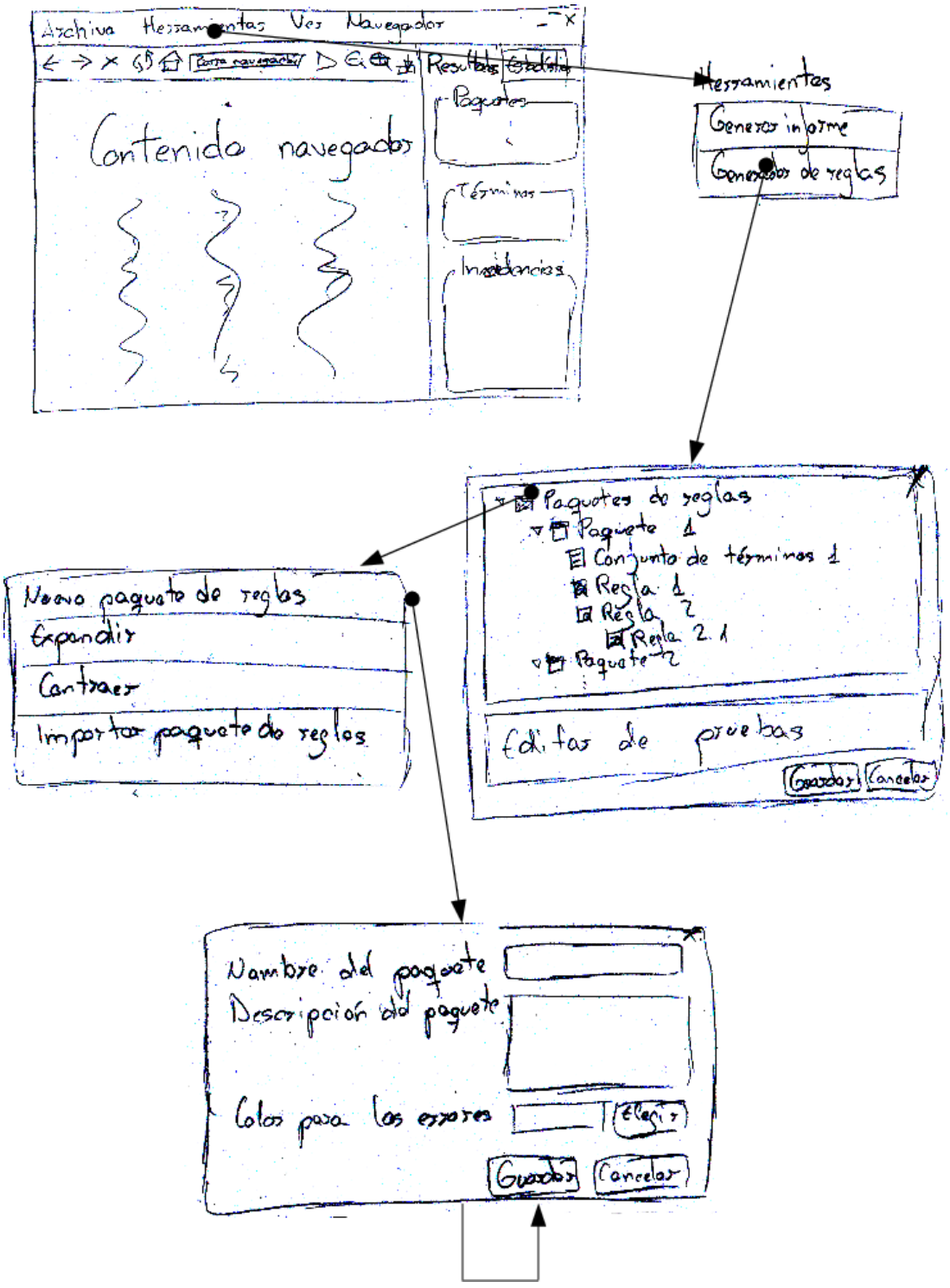


Figura 5.42: Creación de un nuevo paquete de reglas.

Generación de un informe de un sitio web

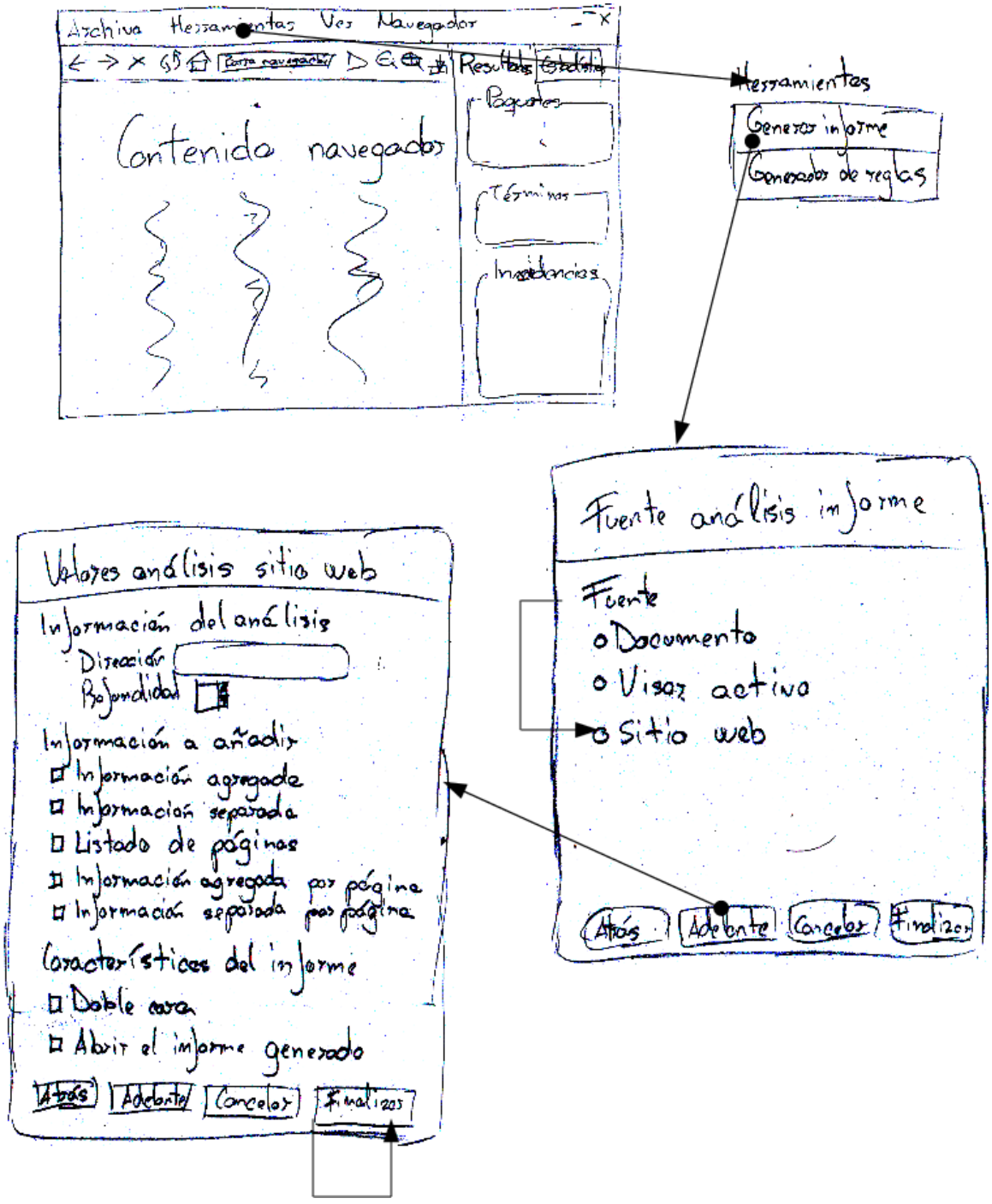


Figura 5.43: Generación de un informe de un sitio web.

Cambiar la vista del navegador

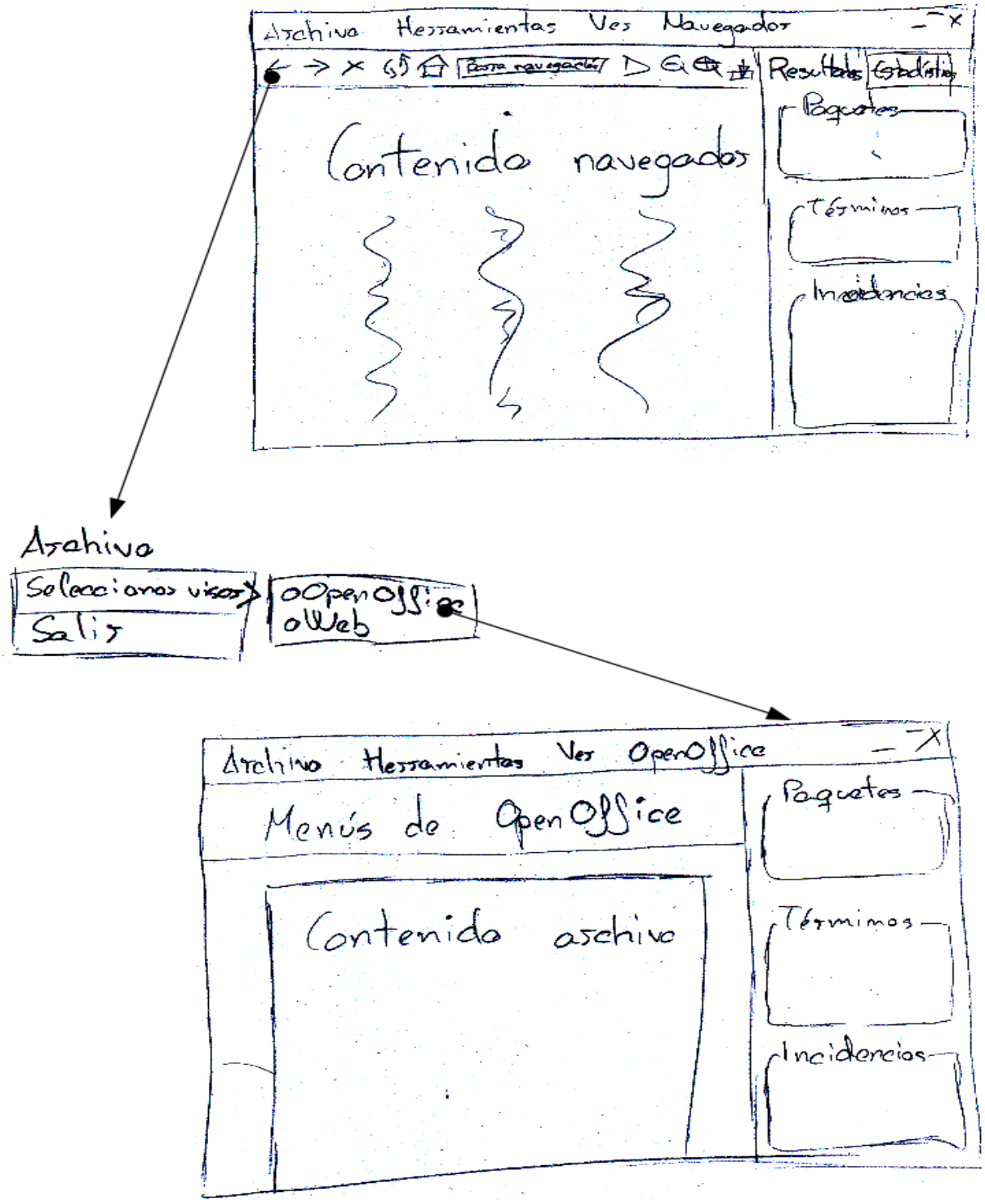


Figura 5.44: Cambiar la vista del navegador

Exportar un paquete de reglas

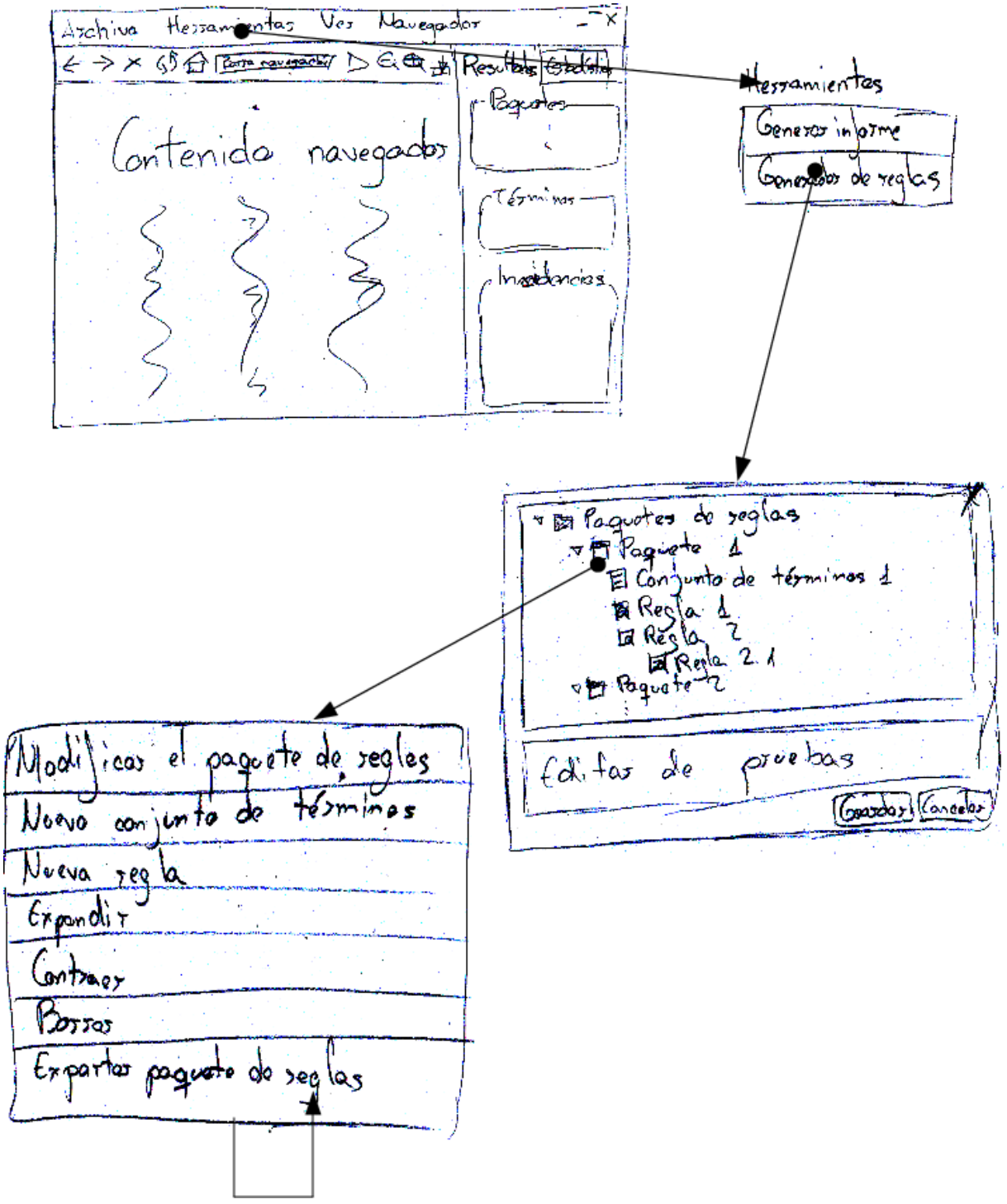


Figura 5.45: Exportar un paquete de reglas

Modificación de un término de un conjunto de términos

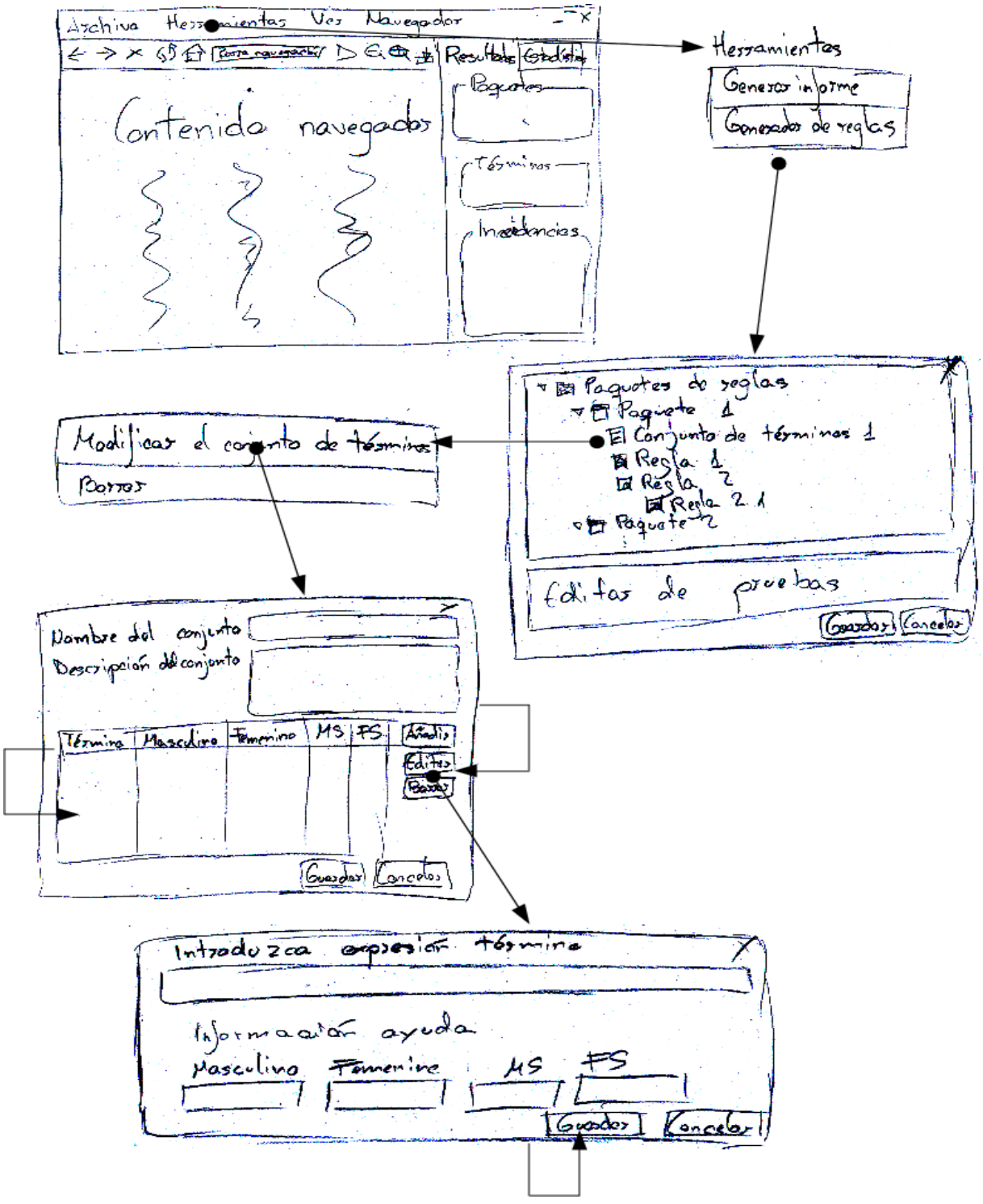


Figura 5.46: Modificación de un término de un conjunto de términos

5.5. Implementación y pruebas

En esta última etapa del desarrollo llevamos a cabo la transformación del modelo obtenido en las fases anteriores en código fuente. Se deben realizar tareas tan importantes para la creación de cualquier aplicación software como seleccionar la arquitectura, el lenguaje de programación o las herramientas a usar.

En el capítulo anterior de la memoria ya introducimos qué arquitectura, lenguaje de programación y herramientas íbamos a emplear, por lo que no es necesario volver a indicarlo aquí. Nos centraremos por tanto, en ver cómo hemos realizado la implementación de nuestra aplicación, remitiendo al anterior capítulo para cualquier duda relacionada.

5.5.1. Desarrollo dirigido por pruebas

En nuestro caso la implementación se ha realizado siguiendo un enfoque híbrido. Así, primero se ha dividido la aplicación en sus unidades mínimas funcionales, obteniendo los distintos componentes que le dan forma. La implementación de cada uno de estos componentes se ha hecho siguiendo un enfoque ascendente guiado por Test Driven Development (TDD).

TDD es una técnica de desarrollo e implementación que se fundamenta en tres pilares fundamentales [56]:

- Implementar las funciones justas.
- Minimizar el número de defectos.
- Producir software modular.

Para implementar según TDD solo debemos seguir un algoritmo iterativo de tres pasos [56]:

- Escribimos un test para cada requerimiento funcional.
- Se implementa el código necesario para que dicho test se cumpla.

- Refactorizamos para eliminar duplicidad y mejorar el código.

En mi opinión, este algoritmo debería incluir un cuarto paso:

- Escribir nuevos test para los casos no considerados.

Siguiendo el algoritmo, el primer paso sería diseñar un test para cada uno de los requisitos de nuestro programa. Asegurar que se cumplen los requisitos pasaría entonces por comprobar que los test se cumplen.

Una vez sabemos que es lo que nuestro programa debería hacer, lo único que nos debe preocupar es implementar el fragmento de código mínimo que permita que los test se verifiquen y, por tanto, los requisitos.

Cuando hemos conseguido pasar los test, es esencial refactorizar el código, buscando que este sea lo más eficiente posible. Como tenemos diversos test que nos comprueban en todo momento si el programa continua haciendo lo que debería, este proceso se realiza de forma controlada.

Por último, deberemos añadir nuevos test para todos aquellos requerimientos que consideremos que no quedan recogidos inicialmente.

Como se deduce del proceso anterior, la idea esencial de TDD es simple, convertir los requerimientos en pruebas software a verificar y desarrollar el código necesario para que estas pruebas se cumplan. Cuando desarrollamos con TDD y plasmamos los requerimientos iniciales, muchas veces descubrimos de forma instintiva muchas condiciones que nuestro software también debería cumplir.

Póngase como ejemplo que estamos desarrollando un programa que debería calcular raíces cuadradas. Nuestro requerimiento podría ser algo similar a 'El programa debe ser capaz de calcular la raíz cuadrada de un número', y, en consecuencia, podríamos escribir inicialmente un test como:

- La raíz cuadrada de 4 debe ser 2.

Es fácil descubrir que este test no cubrirá todos los casos posibles de las raíces cuadradas. La metodología TDD nos dice que todos aquellos casos que consideremos conflictivos,

deben quedar recogidos, pasando a englobar una batería de test. De este modo, mientras desarrollamos el código para que se cumpla el test anterior, nos daremos cuenta de que no hemos considerado el funcionamiento con otros valores distintos de 4, ni con números negativos, ni con valores nulos, etc. Estos aspectos deberán registrarse para la realización de nuevos test.

El problema es que, muchos de estos casos, podrían pasarse por alto si únicamente confiamos en que 'nos demos cuenta' durante el desarrollo. De ahí la razón de incluir un paso adicional al algoritmo. Si una vez probado el test inicial nos detenemos a analizar qué casos no hemos considerado, será más difícil que se nos pasen por alto.

5.5.1.1. Formato de las pruebas

Se han desarrollado múltiples pruebas unitarias para validar la implementación. Estas pruebas tienen, en la mayoría de los casos, el siguiente formato:

```
@Test${exception}
public final void ${methodName}_${test}_${result} () {
    ${code}
}
```

Donde el significado de los parámetros es el siguiente:

- **exception**: Excepción esperada. Se emplea en aquellos casos en los que el resultado esperado es que alguna excepción sea lanzada.
- **methodName**: Nombre del método a probar.
- **test**: Prueba concreta realizada.
- **result**: Resultado esperado.
- **code**: Código a probar.

El formato dado a los test es por tanto autodescriptivo, siendo el propio nombre suficiente para saber qué es lo que se prueba, bajo qué circunstancias y cuál es el resultado esperado.

Constituyen por tanto una excelente documentación complementaria de la interfaz de los componentes.

Para la implementación de los test se han utilizado varias librerías:

- **JUnit**: Es la implementación Java del Framework de pruebas unitarias xUnit. Permite probar fragmentos de código de forma aislada con solo añadir delante de cada función de prueba la anotación Java '@Test'. Cada prueba unitaria es un conjunto de código de cuyo resultado afirmamos algo con los métodos 'assert'. Si el resultado de la prueba es el esperado el test pasará sin más, en caso contrario, se nos indicará que la evaluación ha fallado.
- **Cobertura**: Es una librería que nos indica el porcentaje de código que se ha ejecutado de una aplicación. Es un complemento ideal a las pruebas unitarias ya que nos permite conocer qué porcentaje del código estamos probando e, incluso, aquellos fragmentos de 'código muerto'[8]existentes.
- **Mockito**: Lo ideal a la hora de realizar las pruebas es que estas se realicen de forma independiente. Esto lo podemos conseguir mediante Mockito. Con Mockito podemos crear objetos simulados (también llamados pseudoobjetos) que permitan probar un objeto independientemente de sus dependencias. Cada una de estas dependencias se convierte en el entorno de pruebas en un objeto Mock que simula ser el original. Así, podemos simular el comportamiento de las dependencias y centrarnos en el comportamiento del fragmento que se prueba. Si la prueba falla, será responsabilidad del objeto.
- **SWTBot**: Aunque las pruebas de la interfaz de usuario no suelen ser muy útiles debido a su alta complejidad y su escaso beneficio, en determinados casos si que pueden resultarnos útiles (comprobar si un botón funciona, si está habilitado un menú, etc). Para la realización de las pruebas de la interfaz emplearemos SWTBot, una librería que nos permite automatizar determinados comportamientos de la interfaz de usuario y comprobar el resultado obtenido.

Los test se incluyen en fragmentos de plug-ins. Un fragmento es un tipo de plug-in especial que solo existe en tiempo de ejecución. Su funcionamiento es simple, toma un plug-in y lo

[8]Código que no llega nunca a ejecutarse.

amplía. En el momento de ejecutar el fragmento se creará uniendo su código al del plug-in original lo que le permite acceder a todas las clases del original.

De esta forma garantizamos que:

- El código productivo y el código de test están separados.
- Se pueden probar los objetos y métodos privados del componente sin recurrir a complejos mecanismos de reflectividad.

En el CD del programa están disponibles todos los fragmentos de test desarrollados.

Capítulo 6

Conclusiones

6.1. Conclusiones finales

Este proyecto nace de la idea de establecer un marco de trabajo común para la realización de análisis lingüísticos. Partimos de Iguáleme, una buena idea pero tremendamente limitada. A lo largo de los capítulos hemos observado como esta idea tomaba forma hasta convertirse en una realidad.

Empezamos el proyecto planteando un modelo general de análisis lingüístico basado en la forma de trabajar de Iguáleme. Definimos este tipo de análisis como: un análisis descriptivo de la lengua escrita a través del estudio de la sintaxis de las frases de carácter microlingüista, sincrónico y teórico. Este análisis se adapta a multitud de problemáticas distintas, ya disponemos de un analizador.

El siguiente punto a resolver es la arquitectura a emplear. Considerando que buscamos una herramienta que pueda ser usada por terceros, cuestiones como la estabilidad o la ampliabilidad son fundamentales. De las distintas arquitecturas existentes, OSGi es la más adecuada para nuestro Framework de análisis. Ya solo queda realizar su implementación.

El desarrollo de una herramienta de las características planteadas implica utilizar multitud de librerías, herramientas externas, parsers, aplicaciones, etc. Además, su carácter genérico obliga a tener en todo momento una visión general, intentando no caer en el error de trabajar pensando en una problemática concreta. Es necesario por tanto, conjugar múltiples técnicas y herramientas tradicionales de la ingeniería del software tales como, los casos de uso o los diseños de pantallas con otras más novedosas como TDD.

Lo anterior, lejos de resultar un impedimento, es uno de los principales alicientes que me han motivado durante la realización del proyecto, habiendo podido poner en práctica muchos de los conocimientos acumulados durante mis años de estudio y, además, descubriendo que todos estos conocimientos me ayudan a aprender nuevas técnicas de desarrollo de forma sumamente sencilla.

Para finalizar, decir que la realización de este proyecto ha sido para mí una gran satisfacción, tanto por haber cumplido con los objetivos y requerimientos propuestos inicialmente, como por todo el conocimiento adquirido durante la realización del mismo.

6.2. Líneas de trabajo futuras

El proyecto realizado es ampliable por naturaleza, ya que este es uno de los pilares centrales que guían su desarrollo.

Si nos detuviésemos a dar una lista de todas las ampliaciones posibles podríamos rellenar varias páginas. No obstante, si que podemos destacar una serie de mejoras que, en caso de ser realizadas, mejorarían significativamente la funcionalidad del Framework.

Algunas de estas posibles mejoras serían:

- Desarrollar un componente para la modificación del diccionario gramatical de forma análoga al componente que permite la gestión de los paquetes de reglas.
- Implementar visores de documentos para todos los tipos de documentos analizados.
- Permitir la importación y exportación de diccionarios.
- Incorporar nuevos tipos de reglas de análisis.
- Habilitar un centro de actualizaciones para los componentes.
- Añadir nuevos tipos de análisis complementarios al lineal.
- Adaptar el analizador para permitir búsquedas empleando expresiones regulares.
- Crear nuevos visores de resultados.
- Construir aplicaciones específicas a fin de abarcar un espectro de usuarios más amplio.
- Realizar la conversión a una aplicación Rich Ajax Platform (RAP)[1].

[1] Aplicación web construida con el mismo código de una aplicación Eclipse RCP

Apéndice A

Manual de instalación

A.1. Software requerido

Linguo viene listo para ser ejecutado por lo que no es necesario realizar su instalación. Sin embargo, la aplicación hace uso de diversos programas para ofrecer toda su funcionalidad, si estos no están disponibles el resultado será claro, la ejecución fallará.

Por favor, verifique que todo el software que se muestra en este punto está correctamente instalado para una correcta ejecución del programa.

A.1.1. Plataforma de ejecución Java

Linguo está desarrollado en Java y por tanto es fundamental disponer de un entorno de ejecución Java versión 6 para utilizar el programa.

Recomendamos hacer uso de la máquina virtual Java proporcionada por Sun en lugar de la implementación libre OpenJDK. Los motivos son claros:

- Ha sido la plataforma en la que se ha desarrollado Linguo y en la única en la que podemos asegurar su funcionamiento.
- La implementación libre de la plataforma Java es, a día de hoy, más lenta que la versión oficial.

En base a esto no podemos más que recomendar que haga uso de la versión proporcionada por Sun.

Instalar la máquina virtual en un entorno GNU\Linux es un proceso sumamente sencillo. Únicamente deberá utilizar el gestor de paquetes de su distribución.

Como ejemplo, en un sistema con el gestor de paquetes 'aptitude'[1] el proceso sería tan simple como introducir desde linea de comandos una única instrucción:

```
aptitude install sun-java6-jre
```

[1] Gestor de paquetes empleado en algunas de las distribuciones más extendidas como Ubuntu o Debian.

Finalizado el proceso tendrá la máquina virtual Java instalada.

En el caso de que exista otra versión de Java en el sistema deberá establecer la versión instalada como la versión a usar para ejecutar las aplicaciones Java.

Para ello introduzca la siguiente instrucción:

```
update-alternatives --config java
```

Obtendrá algo similar a la figura A.1, la cual muestra un listado de las alternativas Java existente en el equipo.

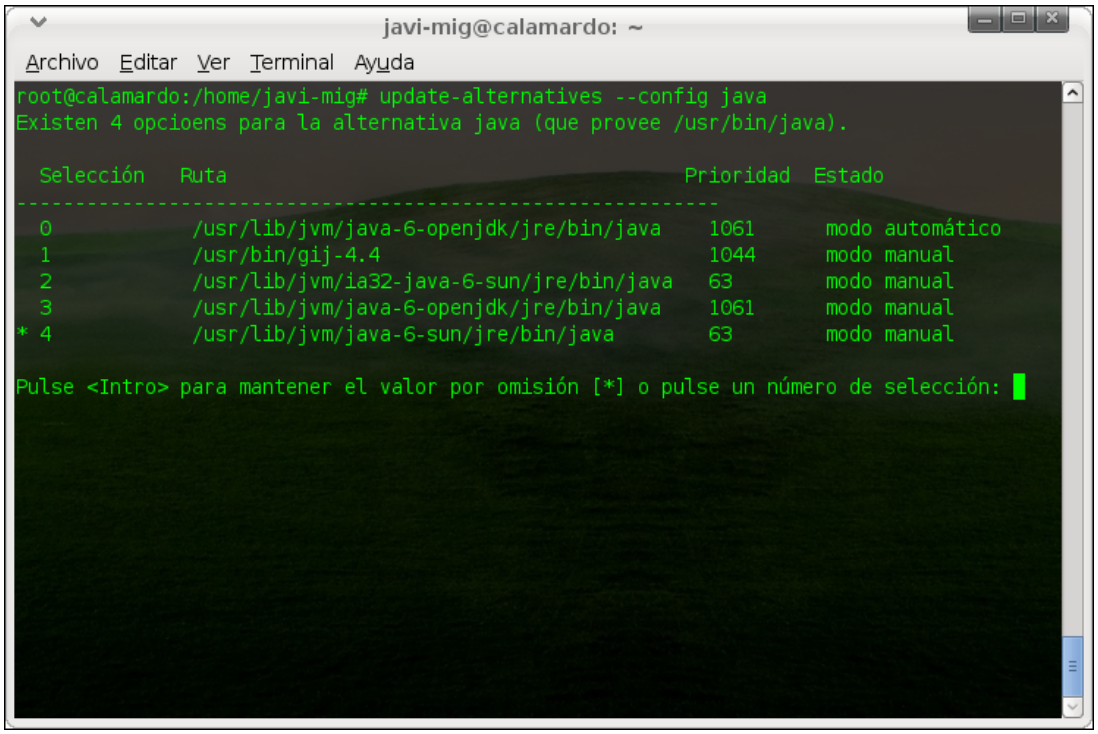


Figura A.1: Activación de la plataforma Java proporcionada por Sun

Introduzca el número de la alternativa seleccionada (en la figura A.1 sería '4') y pulse 'enter'. Tras esto la nueva versión será la que se emplee para ejecutar las aplicaciones Java.

A.1.2. XULRunner

Linguo hace uso de XULRunner para proveerle de un completo navegador web. Por este motivo es necesario que XULRunner esté instalado para poder utilizar la aplicación.

En la mayoría de distribuciones GNU\Linux XULRunner ya viene instalado por lo que no necesitará hacer nada. Para comprobar si XULRunner está instalado compruebe su estado mediante el gestor de paquetes de su distribución.

Comprobar el estado de un programa determinado mediante un gestor de paquetes es un proceso tan sencillo como introducir algo parecido a 'buscar [nombre del paquete]'.

En un sistema con el gestor de paquetes 'aptitude' el comando a introducir sería el siguiente:

```
aptitude search xulrunner
```

Si ejecuta el comando anterior el resultado será similar al mostrado en la figura A.2.

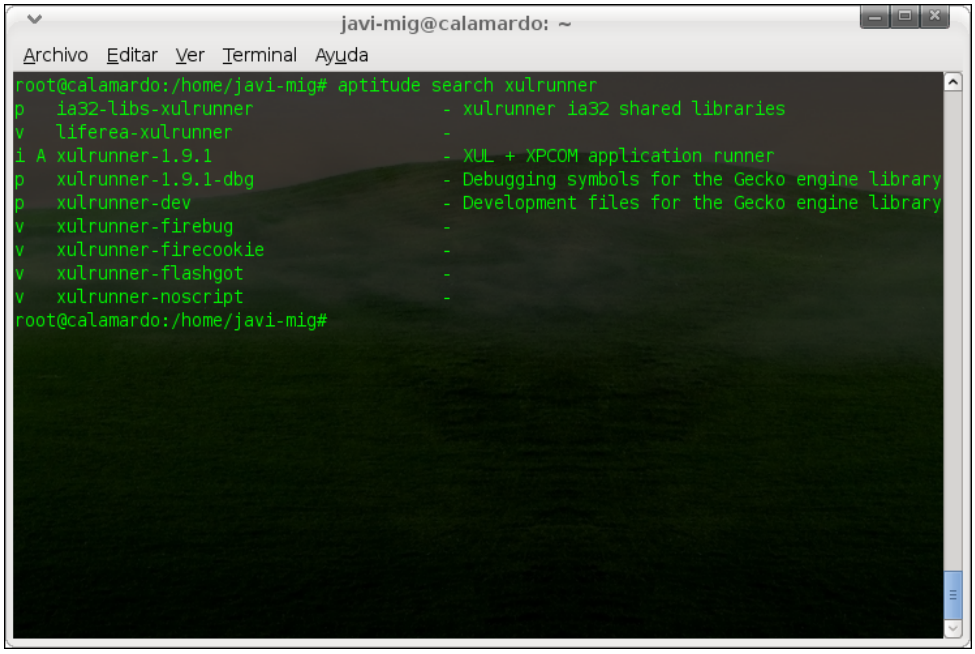


Figura A.2: Comprobar el estado de XULRunner

En la tercera linea de salida aparece 'i A xulrunner-1.9.1', la letra 'i' nos indica que XULRunner se encuentra instalado en su versión 1.9.1.

Si en su caso XULRunner no está instalado tan solo ejecute[2]:

```
aptitude install xulrunner-1.9.1
```

En unos segundos XULRunner estará instalado y listo para ser usado.

A.1.3. OpenOffice.org

Para proveer a la aplicación de un completo editor de textos, Linguo hace uso de la suite ofimática OpenOffice.org.

Es importante que la versión instalada de OpenOffice.org sea la 3.2 o superior, en caso contrario no sería posible acceder a la aplicación.

Al igual que XULRunner, OpenOffice.org suele venir instalado en la mayoría de las distribuciones GNU\Linux, habiéndose convertido casi en un estándar.

El proceso de instalación sería exactamente el mismo que en los casos anterior, tan solo busque si está instalado:

```
aptitude search openoffice.org
```

Si el resultado es negativo, instálelo:

```
aptitude install openoffice.org
```

Transcurridos unos minutos OpenOffice.org estará plenamente operativo.

[2]Si su gestor de paquetes es aptitude.

Apéndice B

Manual de usuario

B.1. Acerca de este documento

La aplicación Linguo le permite realizar análisis lingüísticos basados en reglas a medida.

En este manual exponemos todas sus funcionalidades y modos de uso. También se incluye una lista FAQ[1] con algunas preguntas comunes.

B.2. Ejecutar Linguo

Para ejecutar la aplicación el proceso es muy sencillo, solo debe copiar el archivo comprimido de la aplicación a su equipo, descomprimirlo y ejecutar el archivo 'linguo' que encontrará en la carpeta descomprimida. Tras este simple proceso se desplegará la aplicación y podrá comenzar a usarla.

Puede que al intentar ejecutar Linguo obtenga una ventana similar a la de la figura B.1.

Figura B.1: Error de ejecución de Linguo

No tema, se debe a que la aplicación no tiene permiso para ejecutarse. Para asignarle este permiso pulse sobre el archivo ejecutable con el botón derecho del ratón y seleccione la opción de 'Propiedades'. Dentro de la ventana que aparecerá diríjase a la pestaña 'Permisos' y marque la opción 'Permitir ejecutar el archivo como un programa' tal y

[1]Del inglés Frequently Asked Questions

como se muestra en la figura B.2. Seleccione 'Cerrar' e intente ejecutar el programa de nuevo.

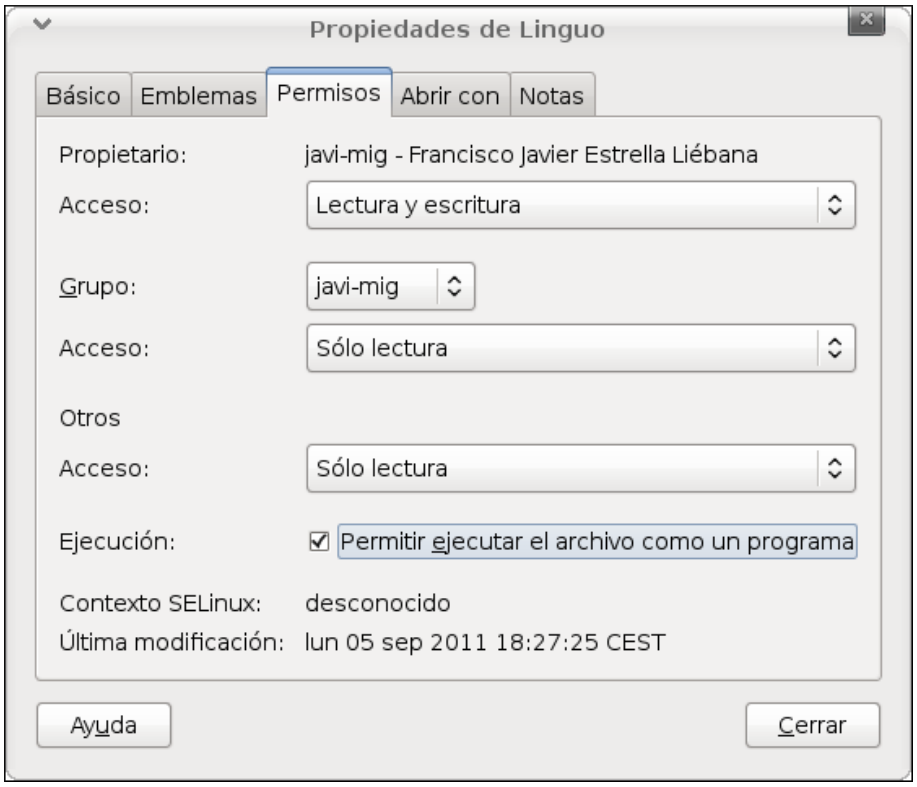


Figura B.2: Asignar permisos de ejecución

Si continua teniendo problemas para ejecutar Linguo compruebe si está instalado todo el software necesario para dar soporte a la aplicación. La lista de este software está disponible en el Manual de Instalación.

B.3. Interfaz de Linguo

B.3.1. Visor web

El visor web le permite navegar por distintas páginas web a la vez que se realiza un análisis de las mismas en base a los paquetes de reglas disponibles. Su apariencia es la que se muestra en la figura B.3.

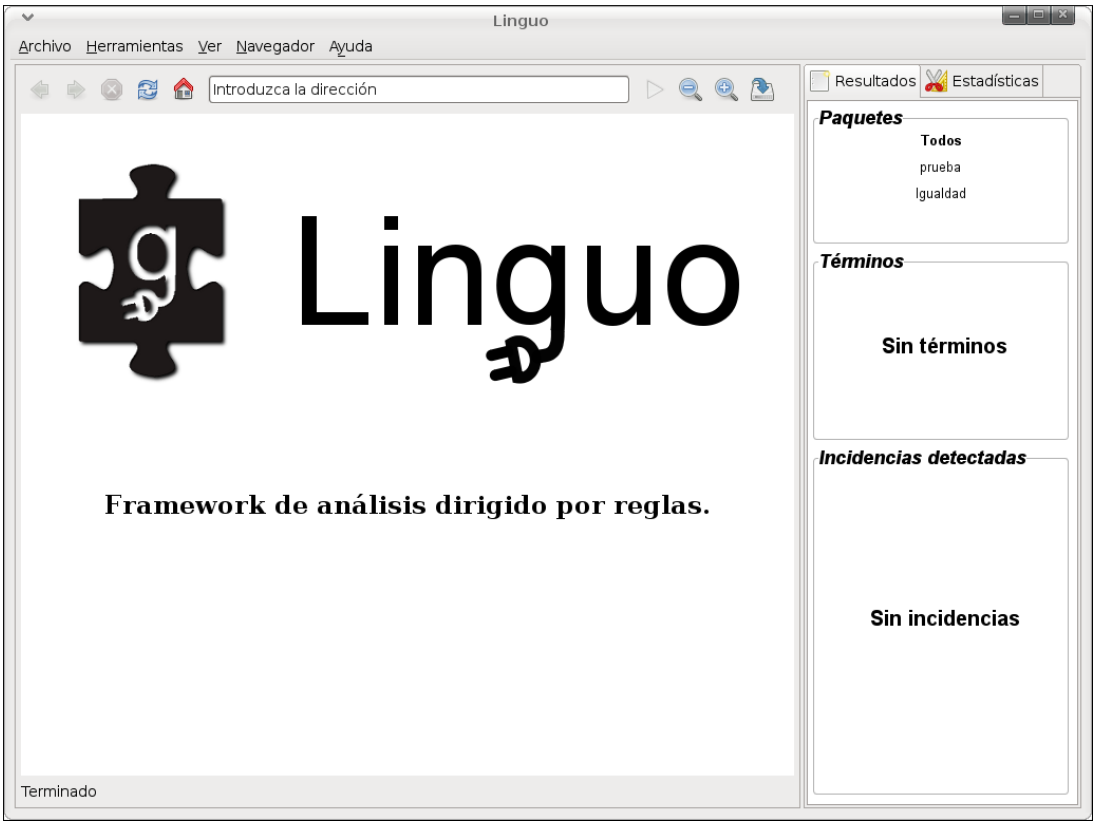


Figura B.3: Visor web

Es el visor predeterminado de Linguo y será el que aparezca cada vez que se inicie la aplicación. El análisis se realizará de forma automática una vez finalizada la carga de la página, pudiendo conocer en el momento cuales han sido las incidencias detectadas.

Las incidencias se resaltan dentro del visor según el color establecido en las preferencias de cada paquete. Al pasar el cursor por cualquier incidencia destacada aparecerá un

diálogo emergente en el que se indicará el paquete que ha marcado la palabra como una incidencia. Un ejemplo de esto se observa en la figura B.4.

Figura B.4: Incidencia en visor web

B.3.2. Visor OpenOffice

El visor OpenOffice integra el editor de texto de la suite ofimática OpenOffice.org dentro de la aplicación. Con el visor podrá confeccionar sus documentos tal y como lo haría normalmente, disponiendo además de la posibilidad de analizar el contenido.

La apariencia del visor OpenOffice.org es la de la figura B.5, aunque variará dependiendo de la versión instalada en su equipo.

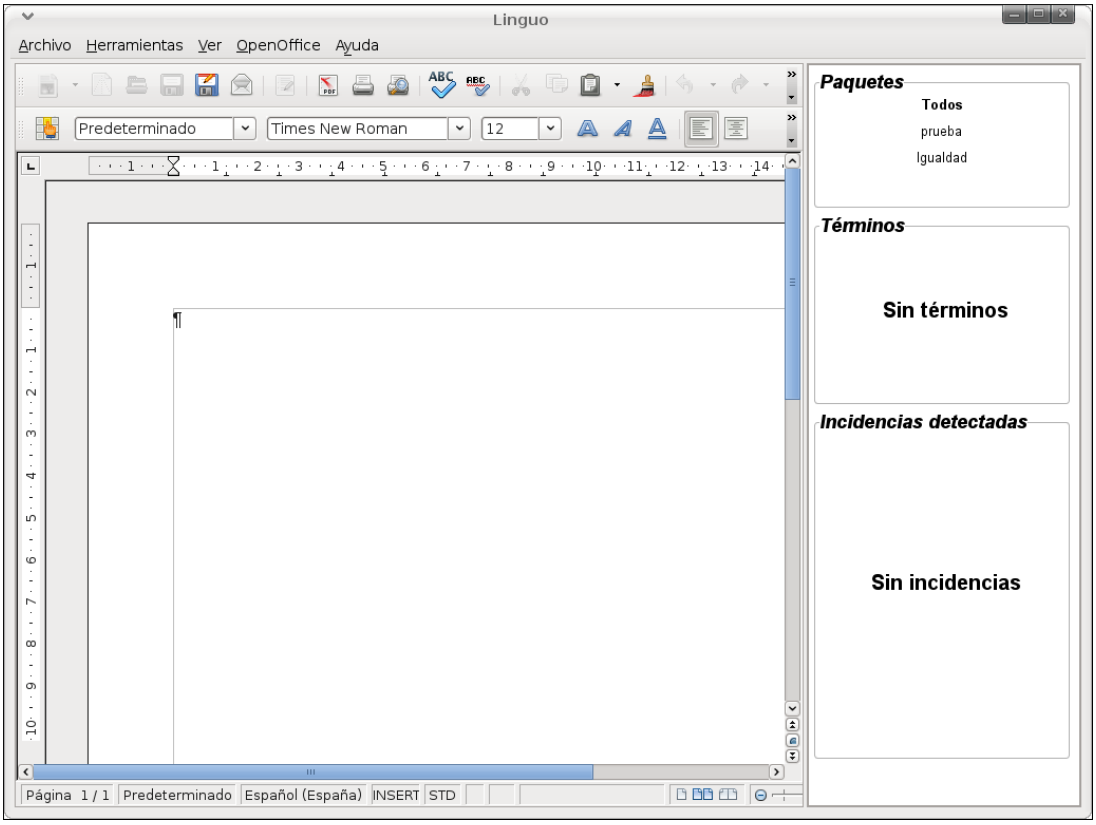


Figura B.5: Visor OpenOffice

El análisis del contenido se realiza periódicamente cada vez que transcurre un determinado tiempo sin que se modifique el contenido del documento.

Los resultados del análisis se muestran en el visor de resultados que se dispone a la derecha de la ventana principal de OpenOffice. Puede localizar el lugar exacto de aparición de cualquier incidencia detectada pulsando sobre ésta en dicho visor (figura B.6).

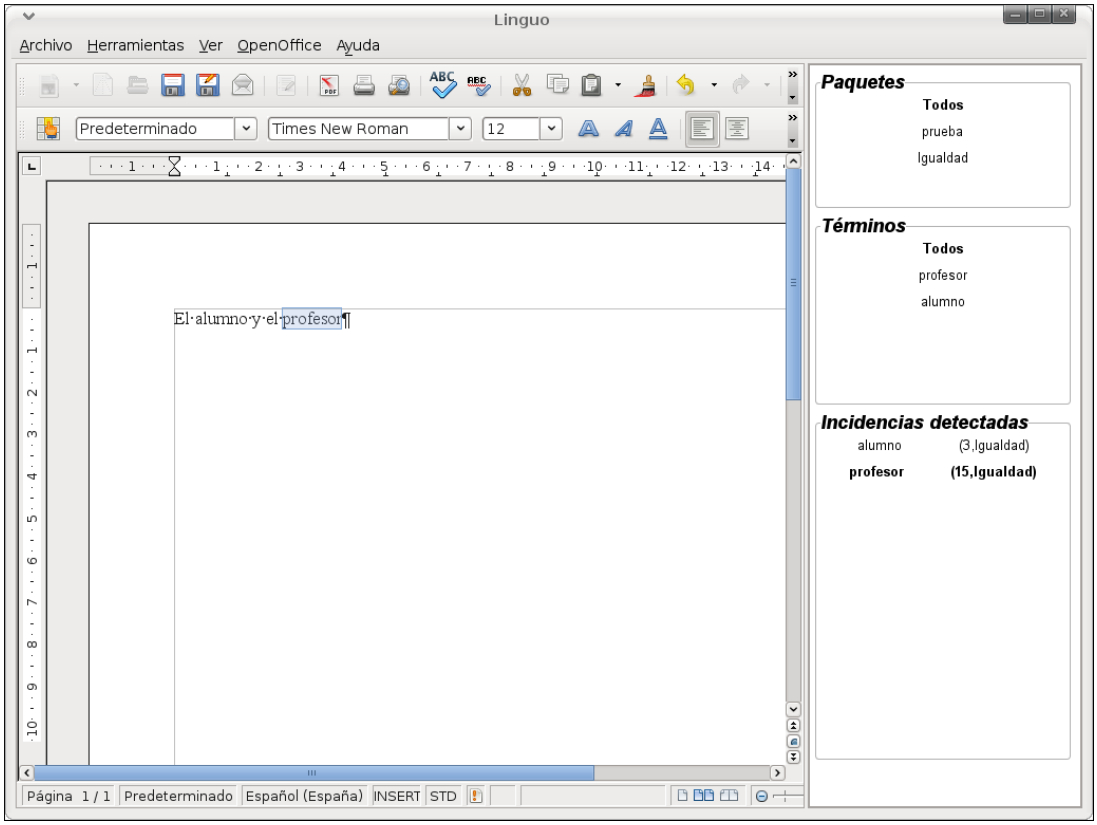


Figura B.6: Incidencia en visor OpenOffice

B.3.3. Visores informativos

Los visores informativos muestran los resultados del análisis de forma agregada y separada (por paquete de reglas). Se disponen a la derecha del visor principal, variando en función del que se encuentre activo en cada momento.

Los dos visores disponibles son el visor de resultados y el visor de estadísticas.

El *visor de resultados* (figura B.7) contiene información de las distintas incidencias detectadas y se divide en tres paneles:

- **Paquetes**: Paquetes de reglas en los cuales se han detectado incidencias.
- **Términos**: Términos para los que ha aparecido alguna incidencia.

- **Incidencias detectadas**: Listado de incidencias detectadas.

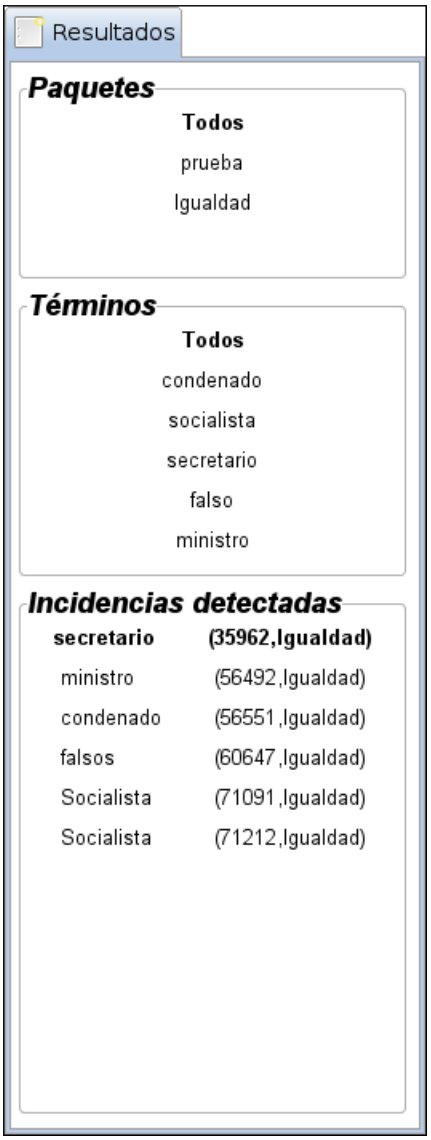


Figura B.7: Visor de resultados

El *visor de estadísticas* (figura B.8) muestra información estadística del análisis y se divide en tres paneles:

- **Info**: Información general del análisis. Engloba el número de términos y palabras en el documento, la cantidad de incidencias para ambos valores y el ratio de incidencias.

- **Gráfico de ratios**: Muestra la relación existente entre la cantidad de términos o palabras y el número de incidencias. Permite detectar aquellos casos en los que las incidencias se deben a un conjunto reducido de términos (cuando el ratio de términos con incidencias sea menor al de palabras).

- **Gráfico de dispersión**: Refleja como las incidencias se reparten a lo largo del documento. Da una idea visual de si se distribuyen de forma homogénea a lo largo del documento, o están concentradas en determinados fragmentos del mismo.

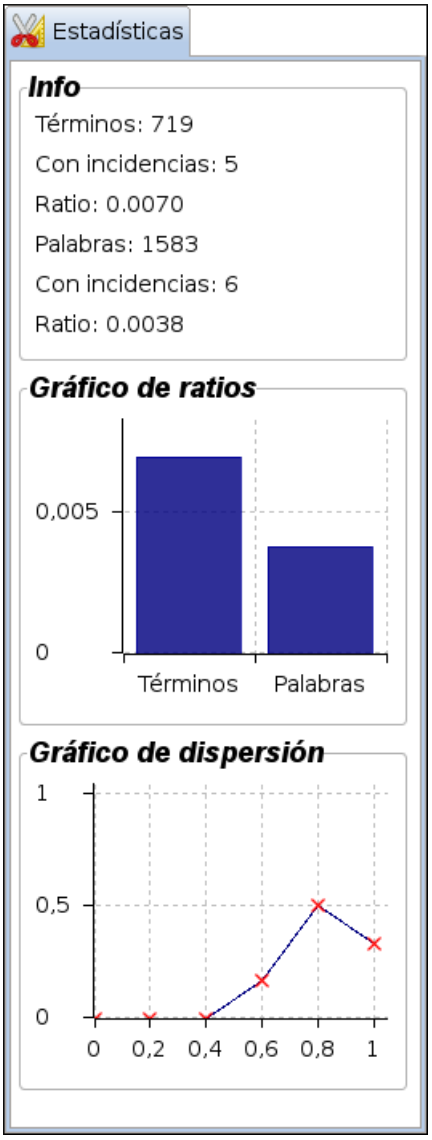


Figura B.8: Visor de estadísticas

B.3.4. Menús de la aplicación

A través de los menús de Linguo podrá acceder a toda la funcionalidad de la aplicación. Los menús se disponen en la parte superior al visor principal, variando su contenido en función de cual sea el activo.

En el caso de que el visor web sea el que se encuentra activo la apariencia del menú será la de la figura B.9.

Figura B.9: Menú en visor web

En este menú podrá encontrar las siguientes funciones:

- **Archivo**: Funciones generales de la aplicación.
  - **Seleccionar visor**: Modificar el visor activo en la aplicación.
    - **Visor OpenOffice**: Abrir el visor OpenOffice.
    - **Visor Web**: Abrir el visor web.
  - **Salir**: Abandonar la aplicación.
- **Herramientas**: Herramientas de utilidad de Linguo.
  - **Generar informe**: Asistente para la generación de informes.
  - **Generador de reglas**: Ventana de modificación de reglas.
- **Ver**: Controles para habilitar o deshabilitar los visores de información.
  - **Resultados**: Modificar la visibilidad del visor de resultados.
  - **Estadísticas**: Modificar la visibilidad del visor de estadísticas.
- **Navegador**: Controles específicos del navegador.
  - **Anterior**: Ir a la dirección anterior del navegador.

- **Siguiente**: Ir a la dirección posterior del navegador.
- **Parar**: Detener la carga de la página actual.
- **Refrescar**: Volver a cargar la página actual.
- **Inicio**: Ir a la página inicial del navegador.
- **Ir**: Ir a la dirección introducida en la barra de navegación del visor web.
- **Reiniciar zoom**: Reiniciar el tamaño del contenido mostrado en el navegador.
- **Disminuir zoom**: Disminuir el tamaño del contenido mostrado en el navegador.
- **Aumentar zoom**: Aumentar el tamaño del contenido mostrado en el navegador.
- **Guardar página**: Almacenar una copia física de la página web visible.

- **Ayuda**: Herramientas de ayuda al uso de la aplicación.
  - **Acerca de Linguo**: Información descriptiva sobre el propio programa.
  - **Contenido de la ayuda**: Información detallada de las diversas opciones del programa.

Si el visor activo es OpenOffice los menús variarán, desapareciendo el menú 'Navegador' y sustituyéndose por el menú 'OpenOffice'. A su vez, el contenido del menú 'Ver' solo contendrá un elemento que será 'Resultados'. La apariencia de este menú será tal y como se refleja en la figura B.10.

Figura B.10: Menú en visor OpenOffice

El contenido del menú 'OpenOffice' será el siguiente:

- **OpenOffice**: Controles específicos de OpenOffice.
  - **Nuevo archivo**: Crear un nuevo archivo en OpenOffice.
  - **Abrir archivo**: Abrir un archivo existente en OpenOffice.

B.3.5. Ayuda

Linguo contiene una completa ayuda en la que podrá encontrar respuesta a la mayoría de sus dudas respecto al uso de la aplicación. La ayuda no es una simple secuencia de páginas ya que permite realizar búsquedas e incluso añadir marcadores.

La apariencia del visor de ayuda será similar a la de la figura B.11.

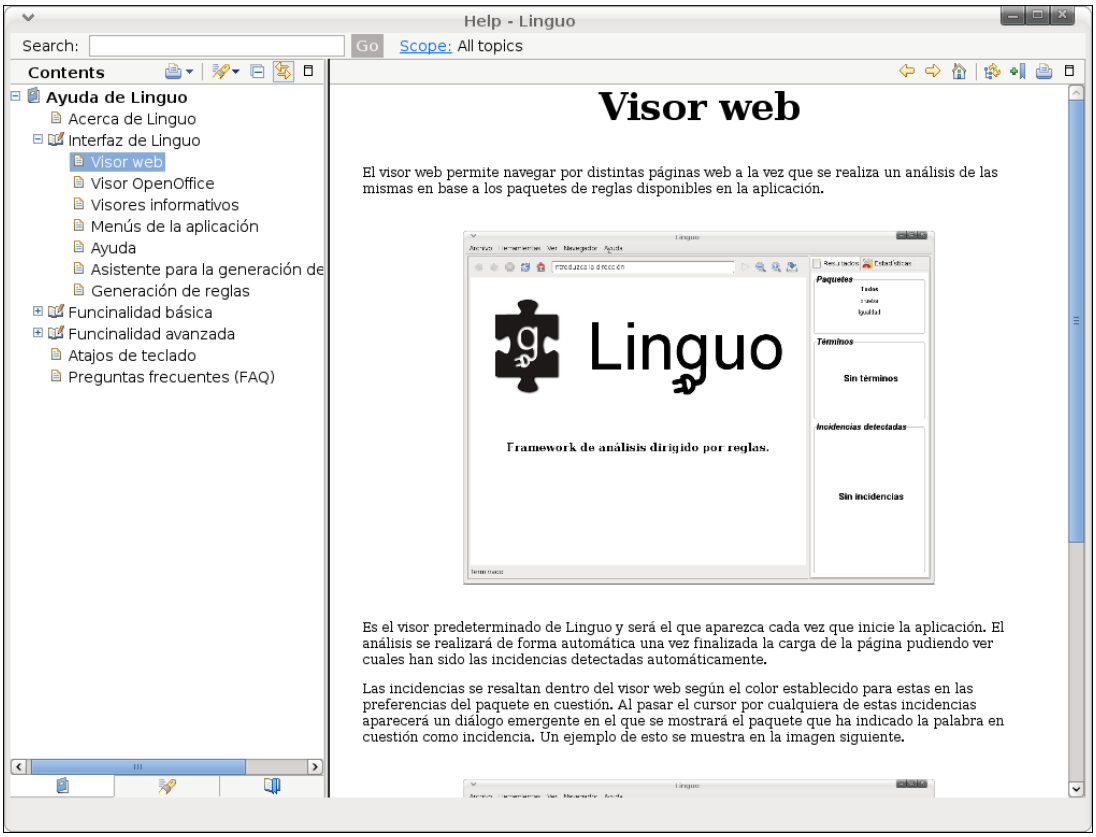


Figura B.11: Ayuda de Linguo

B.3.6. Asistente para la generación de informes

Mediante el asistente para la generación de informes podrá obtener un informe detallado de las incidencias detectadas en un contenido.

Las fuentes de contenido que es posible analizar son las siguientes:

- **Documento**: Archivo físico en alguno de los formatos soportados por la aplicación.
- **Visor activo**: Visor que actualmente se encuentra activado en la aplicación.
- **Sitio web**: Análisis de una página web así como de todos los enlaces que aparecen dentro de ella.

La pantalla inicial del asistente es la de la figura B.12.

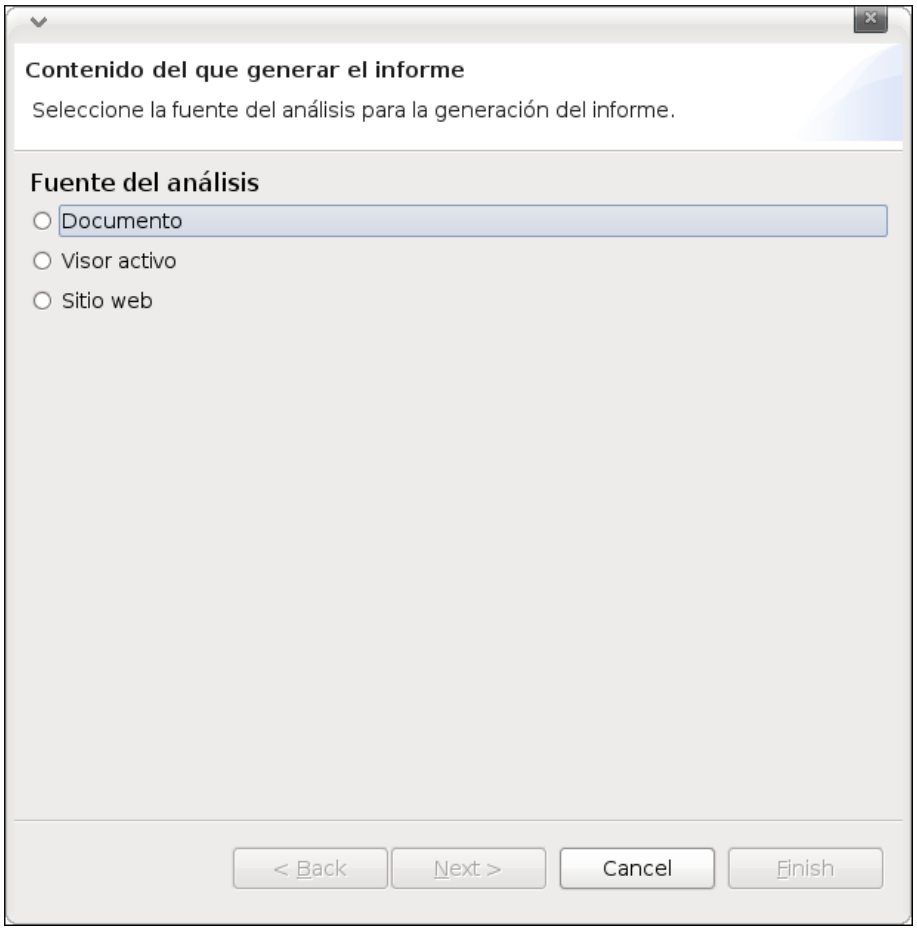


Figura B.12: Asistente para la generación de informes

B.3.7. Generación de reglas

El generador de reglas de Linguo le permite realizar análisis lingüísticos a medida a través de la creación y modificación de paquetes de reglas.

La ventana principal del generador de reglas se divide en dos regiones:

- **Recursos disponibles**: Muestra los recursos actualmente disponibles dentro de la aplicación tales como paquetes de reglas, conjuntos de términos y reglas. Puede interactuar sobre cualquiera de ellos pulsando con el botón derecho del ratón.
- **Visor de prueba**: Le permite insertar texto y comprobar cual sería el resultado del análisis. Las incidencias detectadas se indicarán en función del color establecido para el marcado en cada uno de los paquetes de reglas.

La ventana principal del generador de reglas será similar a la de la figura B.13.

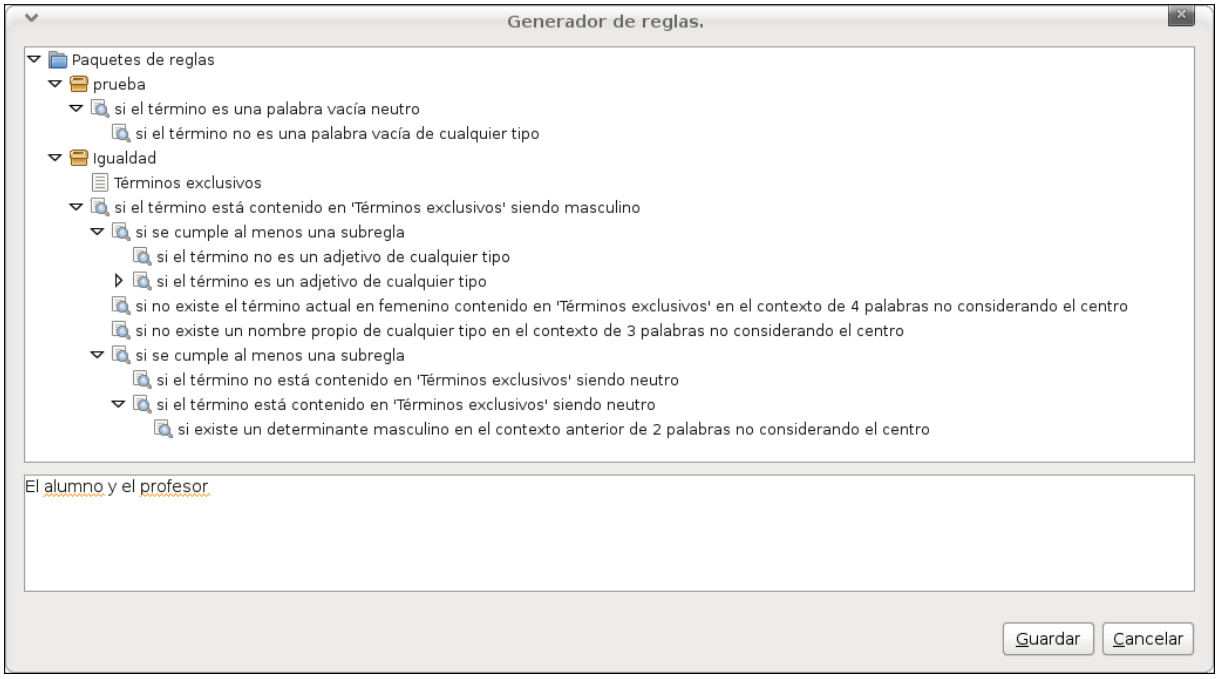


Figura B.13: Generador de reglas

B.4. Funcionalidad básica

B.4.1. Cambiar el visor activo

Para cambiar el visor activo diríjase al menú 'Archivo' y pulse 'Seleccionar visor', a continuación marque el visor deseado (figura B.14). En el caso de que el visor seleccionado sea OpenOffice, tenga en cuenta que debe ser iniciado por el sistema y que su carga puede demorar unos segundos.

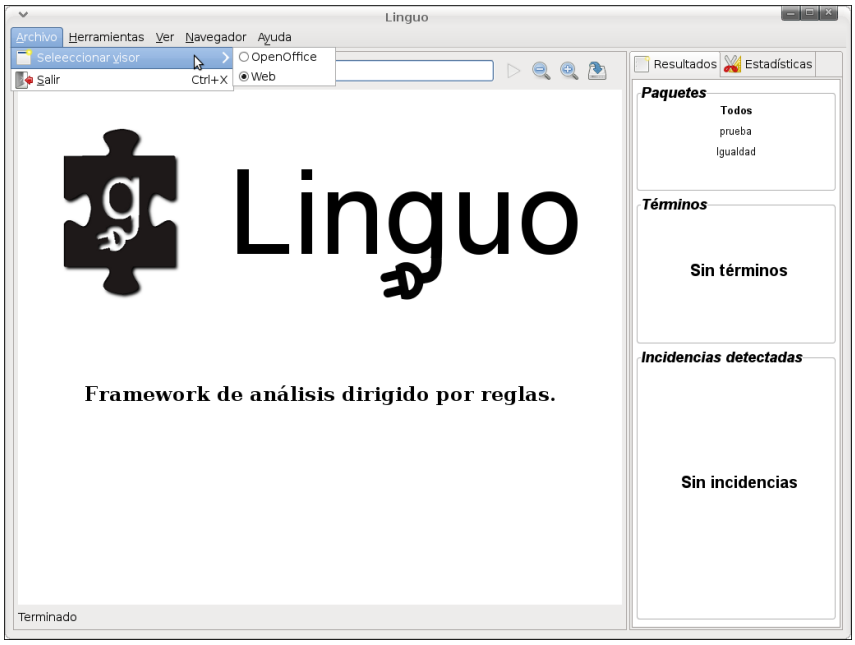


Figura B.14: Cambiar el visor activo

B.4.2. Modificar los visores informativos visibles

Para modificar la visibilidad de los visores informativos pulse en el menú 'Ver' y a continuación modifique el estado de cualquier visor pulsando sobre él (figura B.15).

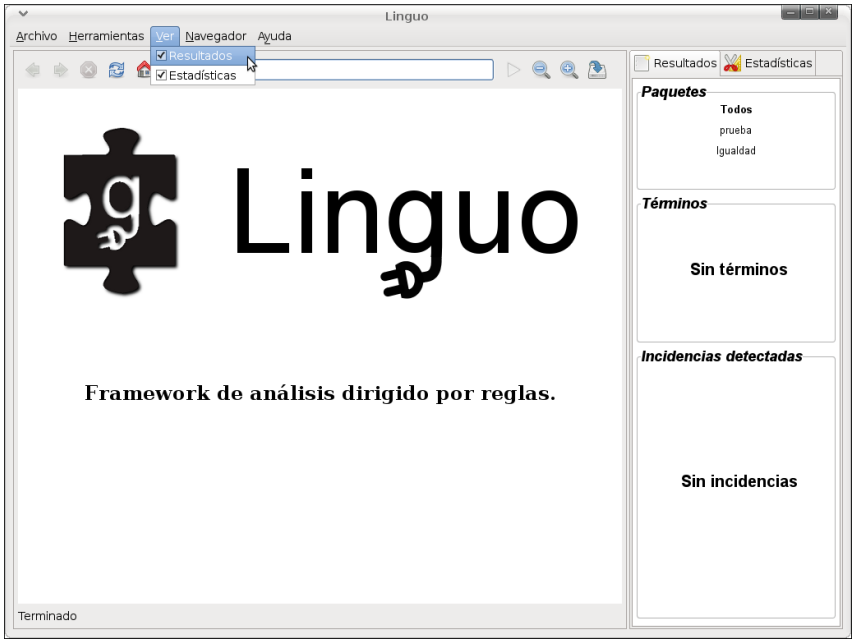


Figura B.15: Modificación de la visibilidad de un visor informativo

Si oculta todos los visores informativos el área completa de la aplicación se dedicará al visor principal (figura B.16).

Figura B.16: Todos los visores informativos ocultos

B.4.3. Ayuda

B.4.3.1. Acerca del programa

En el diálogo 'Acerca de Linguo' podrá encontrar la información relativa a la versión del Linguo así como una breve presentación del mismo. Para abrir el diálogo pulse en el menú 'Ayuda' y a continuación seleccione 'Acerca de Linguo' (figura B.17).

Figura B.17: Abrir el diálogo Acerca de Linguo

Al pulsar en 'Acerca de Linguo' se abrirá el diálogo de la figura B.18.

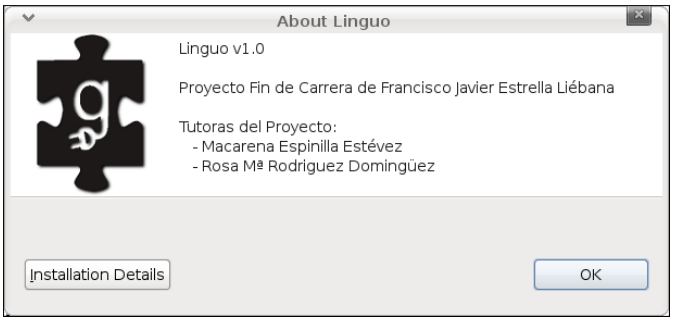


Figura B.18: Diálogo Acerca de Linguo

Dentro de este diálogo dispone de la posibilidad de comprobar la versión concreta de todos los plug-ins que componen la aplicación, para ello solo pulse en 'Installation Details' y obtendrá el diálogo de la figura B.19.

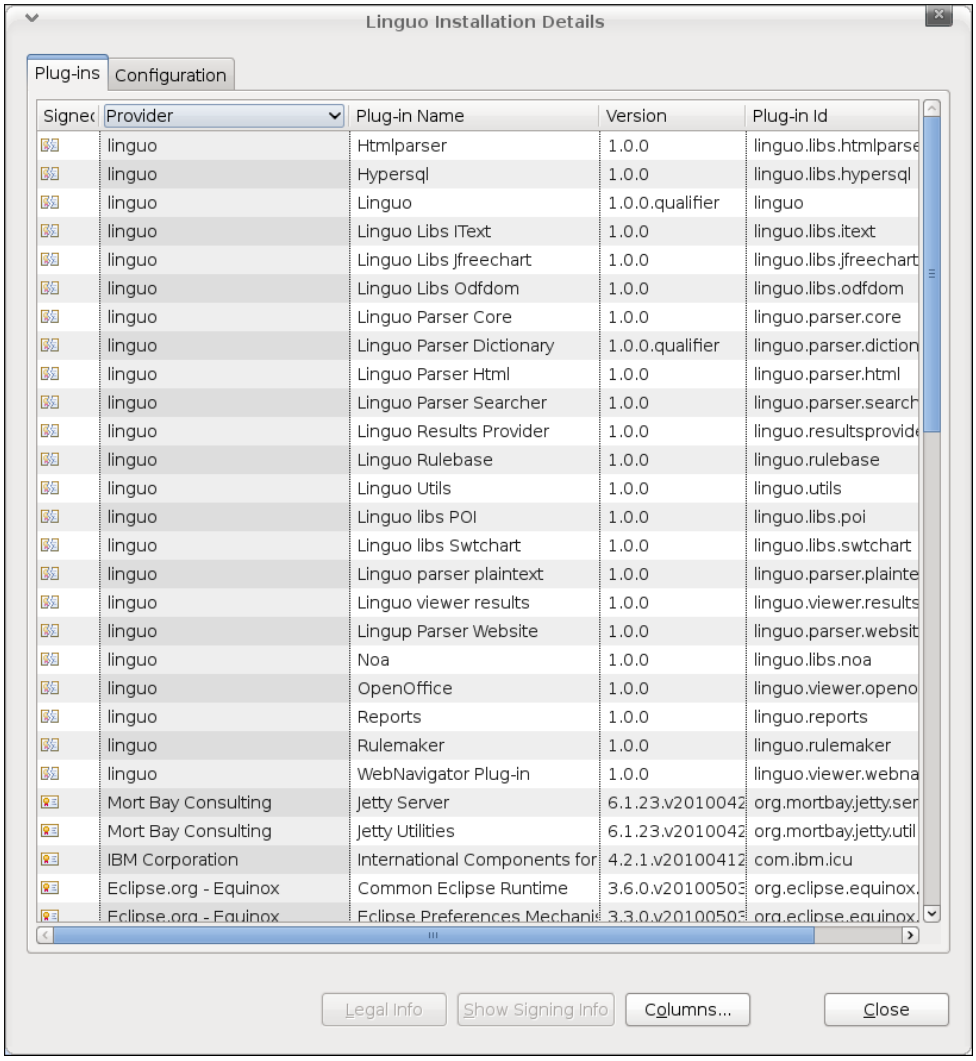


Figura B.19: Detalles de la instalación de Linguo

También es posible consultar en este diálogo la información relativa a la configuración del programa. Para ello seleccione la pestaña 'Configuration' (figura B.20).

Si pulsa en el botón 'View Error Log' de la pestaña 'Configuration' podrá obtener un registro de cualquier fallo que se haya producido desde el inicio del programa.

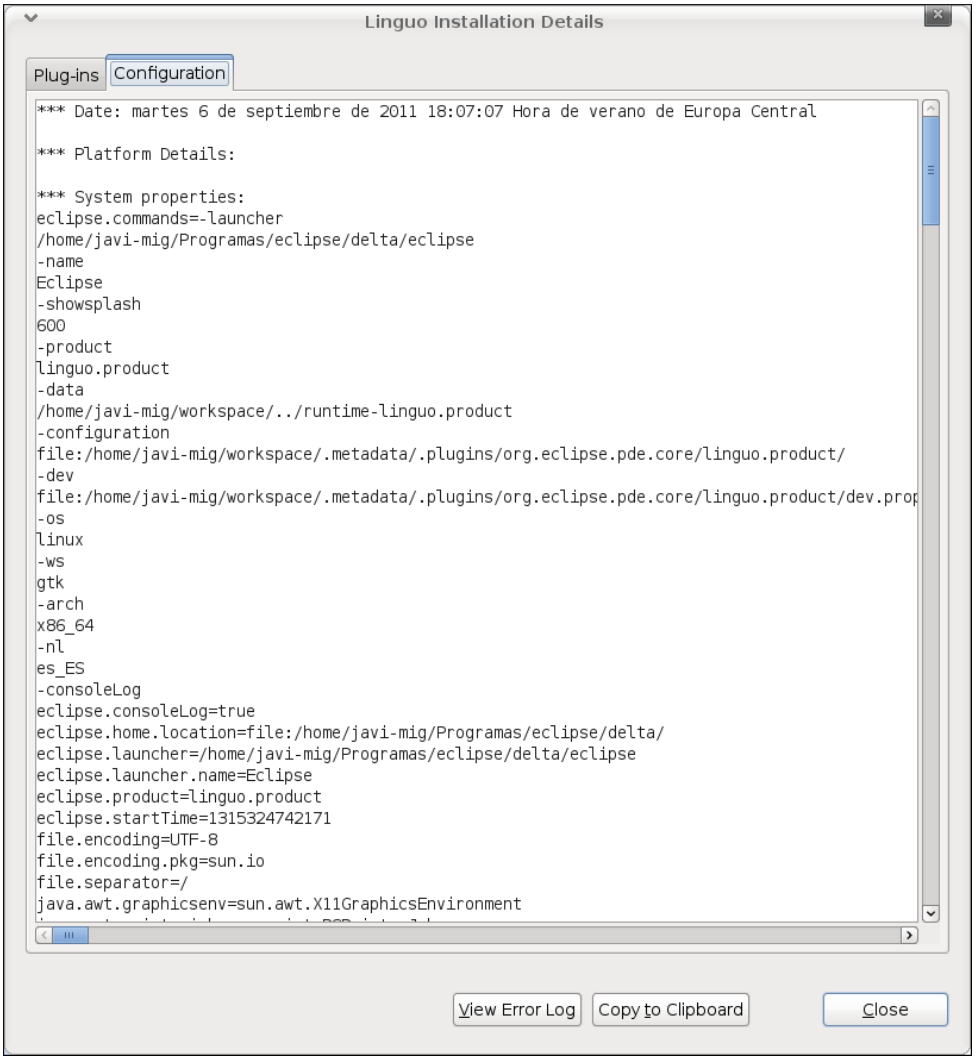


Figura B.20: Configuración de Linguo

B.4.3.2. Ayuda del programa

Para acceder a la ayuda de Linguo diríjase al menú 'Ayuda' y pulse en 'Contenido de la ayuda' (figura B.21) o bien introduzca la combinación de teclas 'Ctrl+H'.

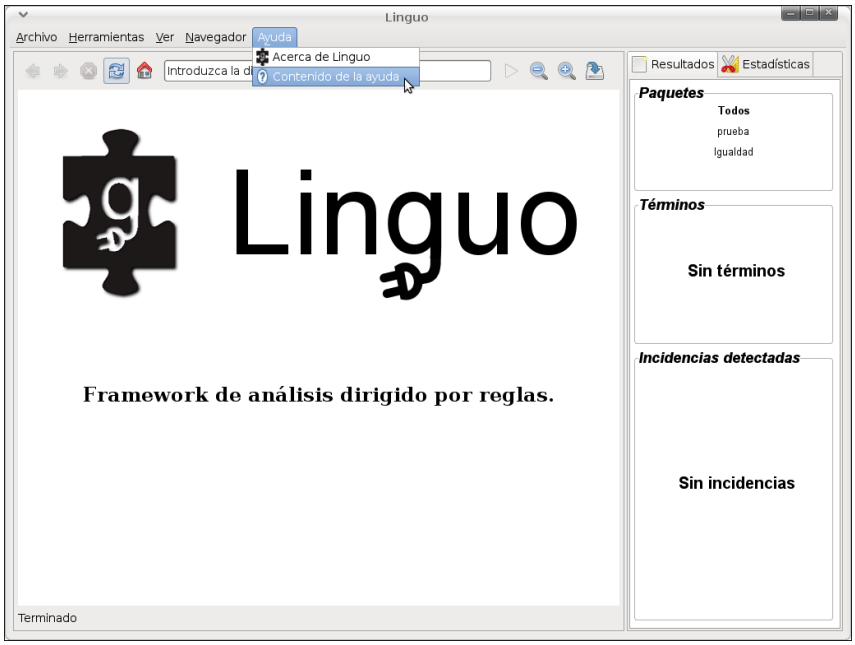


Figura B.21: Acceder a la ayuda

La ayuda puede ser consultada de dos modos distintos:

1. **Árbol de navegación**: Situado en la parte izquierda de la ayuda dispone de un árbol de navegación mediante el cual podrá consultar cualquier página que desee.

2. **Búsqueda**: Para realizar una búsqueda dentro de la ayuda introduzca la expresión a buscar en el cuadro superior de la ventana etiquetado bajo 'Search' y pulse en 'Go'. Transcurridos unos instantes aparecerán todos los contenidos relacionados con su consulta (figura B.22).

Es posible asignar un marcador a cualquier contenido de la ayuda para un rápido acceso posterior.

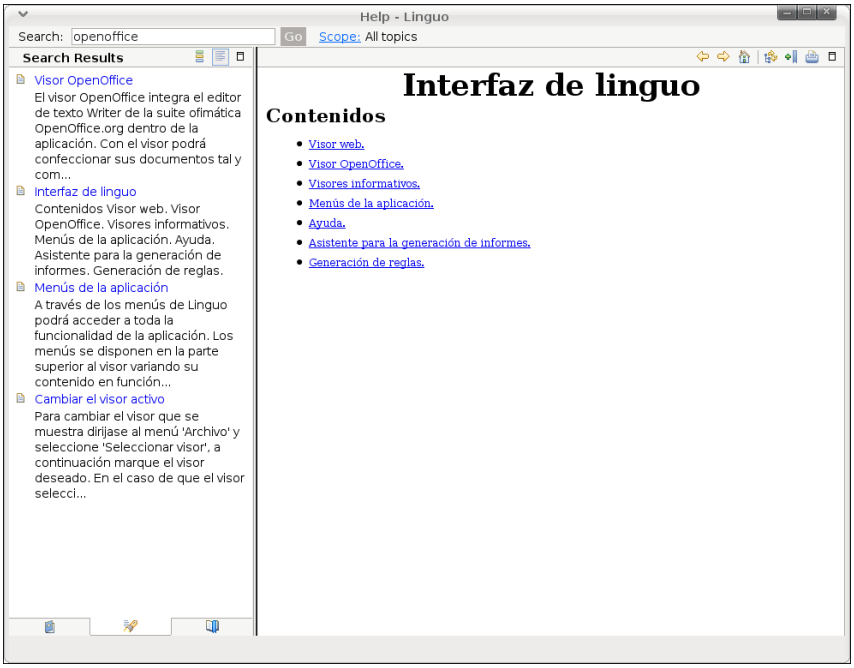


Figura B.22: Búsqueda dentro de la ayuda

B.4.4. Usar el navegador

B.4.4.1. Interfaz principal

Para usar el navegador lo primero que debe hacer es activar el visor web como el activo pulsando en 'Archivo → Seleccionar visor →Web'. Al iniciar el programa el visor web es el activo por lo que no es necesario que realice esta acción si no ha realizado ningún cambio.

Una vez el visor web está establecido como el visor principal podrá hacer uso del navegador para abrir cualquier dirección web. Para ello introduzca la página a abrir en la barra de navegación y pulse 'Enter', transcurridos unos instantes aparecerá el contenido de la dirección introducida en el visor web y las incidencias detectadas marcadas según el color establecido en las preferencias del paquete de análisis (puede ver un ejemplo de esto en el punto B.3).

Si se diese el caso de que una misma palabra es incidencia en más de un paquete

aparecerá destacada en color amarillo (figura B.23).

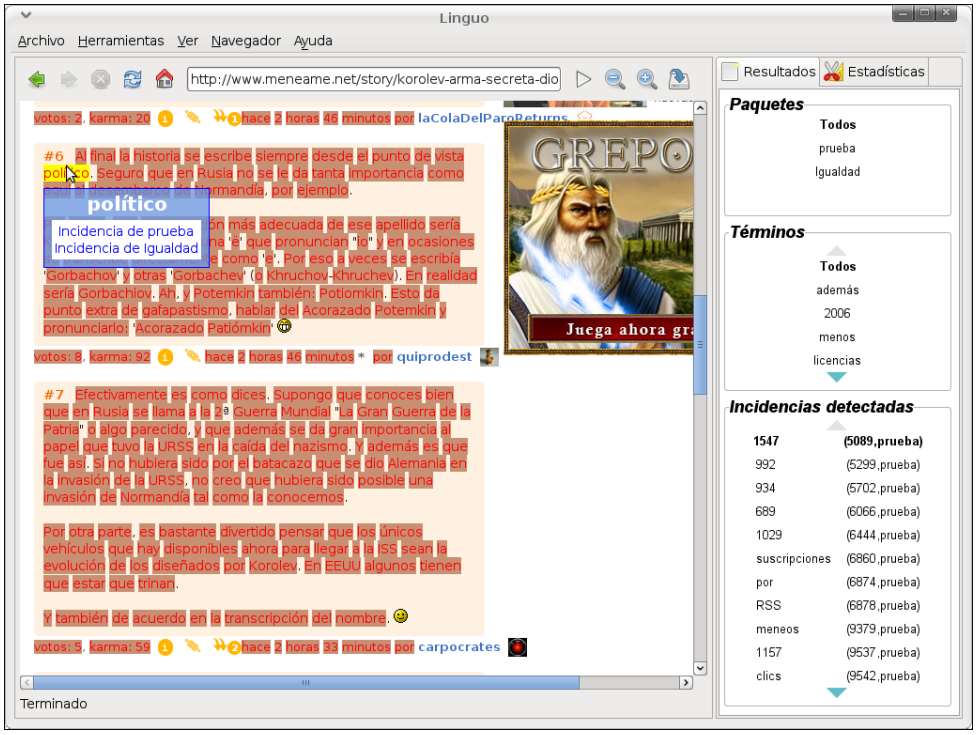


Figura B.23: Incidencia por varios paquetes de reglas

Para obtener información detallada de los resultados del análisis dispone de los paneles informativos. Una descripción de estos paneles se encuentra en el punto B.3.3.

B.4.4.2. Controles de navegación

Existen diferentes comandos para controlar la navegación en el visor web. Podrá acceder a ellos de tres modos distintos, a través del menú 'Navegador', mediante la barra de navegación (figura B.24) o pulsando una combinación de teclas (tabla B.1).

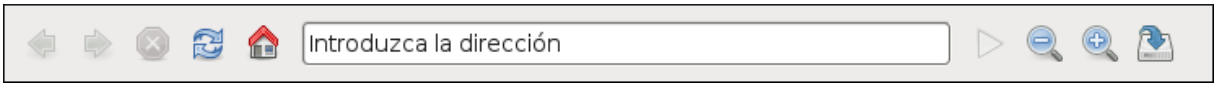


Figura B.24: Barra de navegación

Comando	Icono	Acceso	Atajo de teclado	Función
Anterior		B/M/A	Shift+Alt+Left	Cargar dirección anterior
Siguiente		B/M/A	Shift+Alt+Right	Cargar dirección siguiente
Parar		B/M/A	Esc	Detener la carga actual
Refrescar		B/M/A	Ctrl+F5	Refrescar la página
Inicio		B/M/A	Home	Ir a la página inicial
Ir		B/M/A	Shift+Enter	Ir a la página introducida
Reiniciar zoom	-	M/A	Ctrl+0	Restablecer zoom
Disminuir zoom		B/M/A	Ctrl+-	Disminuir zoom
Aumentar zoom		B/M/A	Ctrl++	Aumentar zoom
Guardar página		B/M/A	Ctrl+Alt+S	Guardar la página actual
Cambiar URL	-	A	Ctrl+L	Introducir una nueva URL

Cuadro B.1: Controles de navegación

B Barra de navegación.

M Menú.

A Atajo de teclado.

B.4.5. Usar OpenOffice

B.4.5.1. Interfaz principal

Igual que para usar el navegador, lo primero que debe hacer para usar OpenOffice es establecerlo como el visor activo. Para ello pulse en 'Archivo → Seleccionar visor →OpenOffice'.

Cuando finalice la carga de OpenOffice podrá trabajar con él al igual que lo haría normalmente, salvo por dos excepciones:

1. **No puede abrir archivos externos mediante el menú tradicional de OpenOffice**: Para utilizar un archivo externo debe de hacer uso del comando 'Abrir archivo' habilitado en el menú 'OpenOffice' de Linguo.

2. **No puede crear un nuevo archivo mediante el menú de tradicional de OpenOffice**: Al igual que en el punto anterior, para crear un nuevo archivo diríjase al menú 'OpenOffice' de Linguo y pulse en 'Nuevo archivo'.

El motivo de las dos salvedades al uso normal de OpenOffice es simple, deseamos que tanto los nuevos archivos creados como los archivos abiertos se muestren dentro de la ventana de Linguo para poder analizar desde la aplicación los archivos. Si recurriésemos al modo tradicional en el que OpenOffice realiza estas tareas, se llevarían a cabo en una nueva ventana de OpenOffice, por lo que sería imposible analizar su contenido desde la aplicación.

Ambas opciones de OpenOffice se encuentran deshabilitadas dentro de Linguo por lo que no tema equivocarse accidentalmente, solo hay un modo de realizar estas tareas dentro de Linguo y es a través del menú proporcionado.

Para más información sobre el uso de OpenOffice dentro de Linguo le remitimos al punto B.3.2 del manual.

B.4.5.2. Controles de OpenOffice

Los comandos de OpenOffice dependerán de la versión instalada del programa, Linguo solo los pone a su disposición. Por este motivo para una detallada documentación de como usar OpenOffice consulte la propia ayuda del programa (podrá encontrarla en la barra principal del mismo).

Como se indicó en el punto anterior, existen dos excepciones al uso normal de OpenOffice, la creación y apertura de archivos, debiendo recurrir en estos casos a los comandos proporcionados por Linguo.

Los comandos propios de Linguo para el uso de OpenOffice se presentan en la tabla B.2.

Comando	Icono	Acceso	Atajo de teclado	Función
Nuevo archivo		M/A	Ctrl+Alt+N	Crear un nuevo archivo
Abrir archivo		M/A	Ctrl+O	Abrir un archivo

Cuadro B.2: Controles de OpenOffice

M Menú.

A Atajo de teclado.

B.4.6. Usar el visor de resultados

El visor de resultados muestra las incidencias detectadas al analizar una página web o el contenido del editor de textos OpenOffice. Está dividido en tres paneles que permiten ajustar la visualización de los resultados. Así, la selección en cada uno de los paneles condicionará el contenido que se mostrará en los que se disponen por debajo de él. Veámoslo en un caso concreto.

El visor de la figura B.25 muestra los resultados de un análisis realizado sobre una página web.

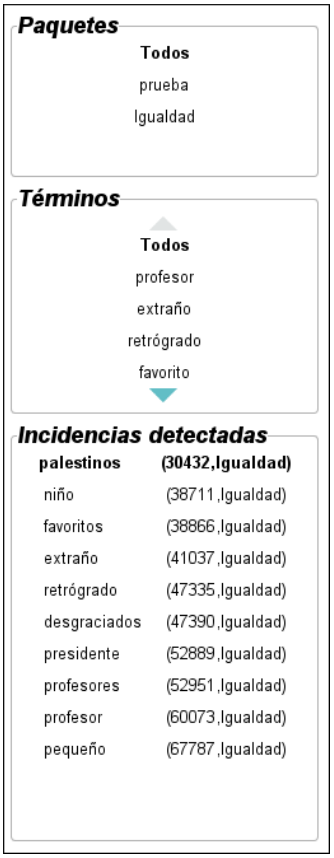


Figura B.25: Uso del visor de resultados

El panel superior etiquetado como 'Paquetes' nos indica los paquetes con los que se ha analizado el contenido, en nuestro caso 'prueba' e 'Igualdad' (figura B.26).

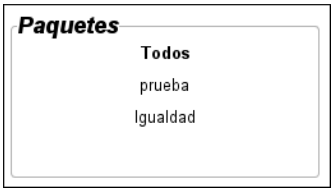


Figura B.26: Panel de paquetes

Al seleccionar cualquiera de ellos el contenido de los paneles inferiores se restringirá a aquellas incidencias encontradas en el paquete seleccionado (figura B.27). Si la selección es 'Todos' no se realiza ningún filtrado.

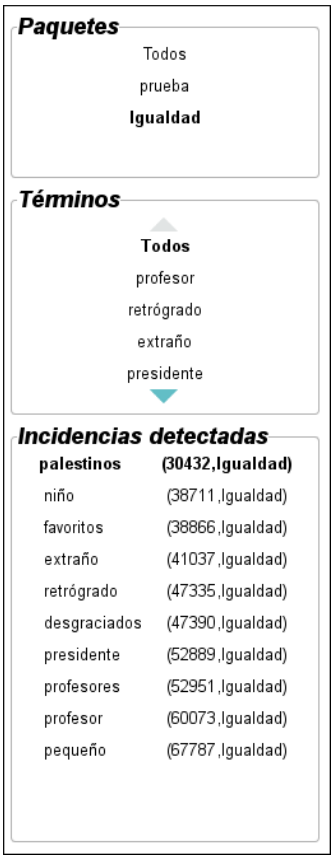


Figura B.27: Filtrado por paquetes

El panel central denominado 'Términos' muestra los términos concretos para los

cuales se han detectado incidencias (figura B.28).

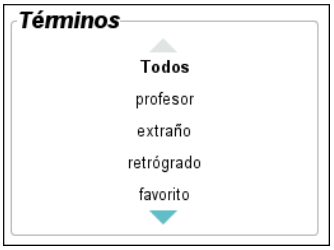


Figura B.28: Panel de términos

Al igual que ocurre con el panel de 'Paquetes', si seleccionamos alguno de los términos mostrados en este panel se limitará el contenido del panel inferior a aquellas incidencias que se corresponden con el término fijado (figura B.29). Si la selección es 'Todos' el filtrado no se llevará a cabo.

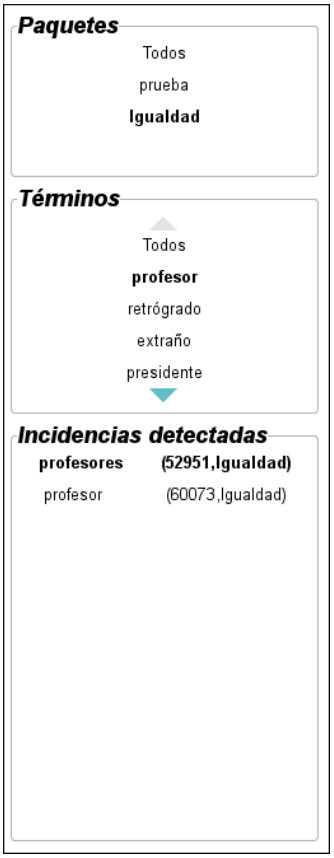


Figura B.29: Filtrado por términos

El panel inferior de 'Incidencias detectadas' refleja todas las incidencias detectadas, indicando tanto la incidencia concreta como su posición y el paquete por el que se ha considerado una incidencia.

Como ya se indicó en B.3.2, si seleccionamos cualquier incidencia del panel de 'Incidencias detectadas' se resaltará automáticamente en este visor el lugar exacto de aparición de la misma.

B.4.7. Salir de Linguo

Para salir de Linguo diríjase al menú 'Archivo' y seleccione 'Salir' (figura B.30) o bien pulse la combinación de teclas 'Ctrl+X'.

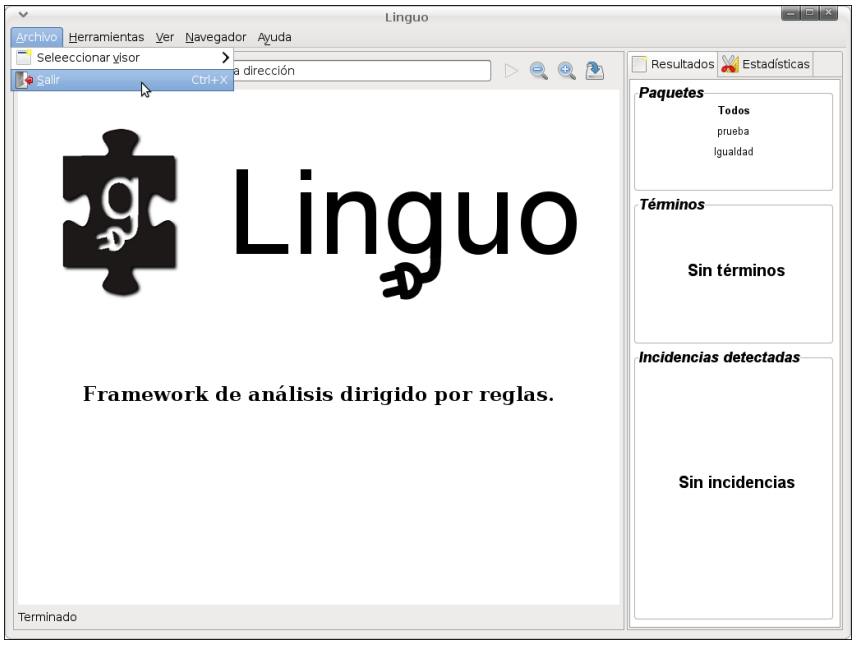


Figura B.30: Salir de Linguo

B.5. Funcionalidad avanzada

B.5.1. Generar un informe

Linguo permite la generación de informes tanto del contenido de los visores del programa como de recursos externos, ya sean estos documentos de texto o sitios web.

Para la generación de un informe seleccione dentro del menú 'Herramientas' la opción 'Generar Informe' (figura B.31) o bien pulse 'Ctrl+I'.

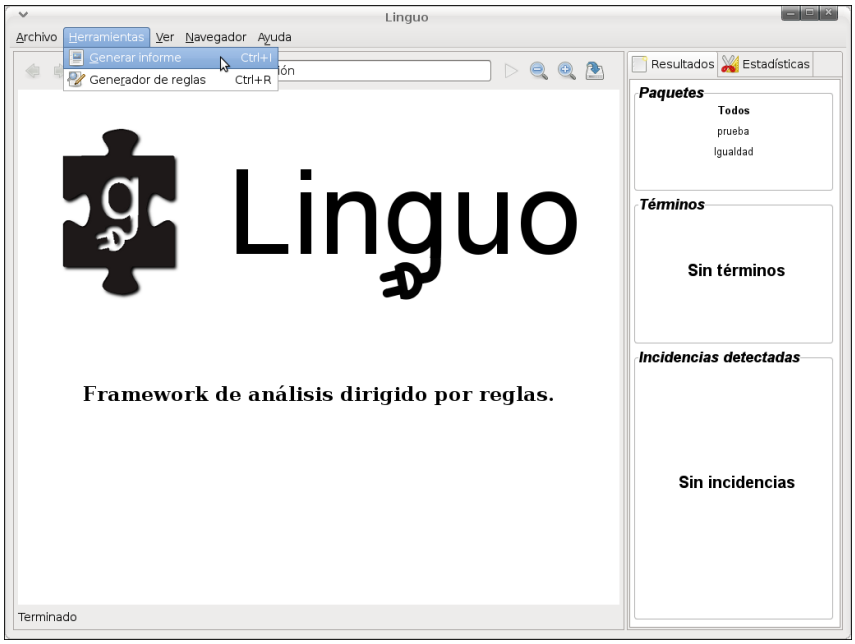


Figura B.31: Abrir el asistente de generación de informes

Durante el análisis del contenido de los visores el generador de informes estará deshabilitado temporalmente. Si encuentra la generación de informes desactivada vuelva a probar transcurridos unos segundos.

El generador de informes es un asistente que nos ayudará a generar un informe a medida de nuestras necesidades, variando las opciones disponibles en función del tipo de contenido a analizar.

Los informes son documentos PDF que contienen información gráfica y textual. Un

ejemplo de una página de un informe generado con Linguo se muestra en la figura B.32.

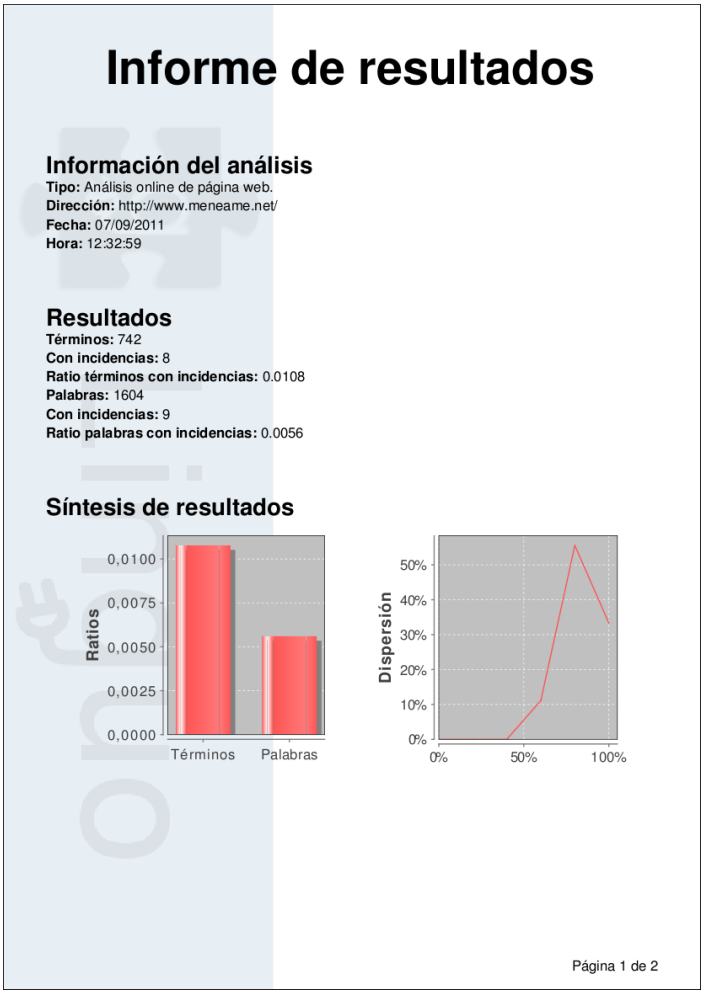


Figura B.32: Página de informe

En los siguientes puntos se muestran las diversas opciones habilitadas en cada caso.

B.5.1.1. Generar un informe de un documento

Linguo puede analizar documentos TXT, PDF, HTML, DOC, DOCX, ODT y RTF. Para realizar un informe de cualquiera de ellos inicie el asistente y seleccione como fuente de análisis 'Documento' (figura B.33).

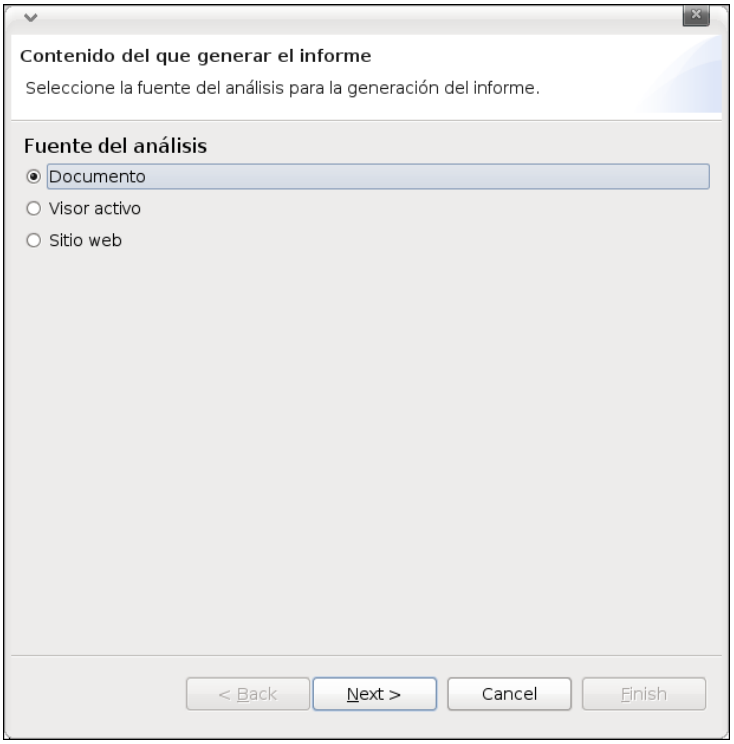


Figura B.33: Informe de documento

Al pulsar en 'Next >' le aparecerá una ventana como la de la figura B.34 en la que podrá establecer las preferencias sobre el informe.

El significado de los campos es el siguiente (con * los campos obligatorios):

- **Archivo a analizar***: Documento que se analizará para la realización del informe.
- **Incluir información agregada**: Incluye en el informe una sección con los resultados del análisis considerando todos los paquetes de reglas como uno solo.
- **Incluir información separada**: Incluye una sección para cada paquete de reglas con los que se realizará el análisis.
- **Informe a doble cara**: Si se marca esta opción tanto las marcas de agua como los números de página del informe generado variarán en función de si la página es par o impar.
- **Abrir el informe generado**: Abre el informe una vez creado.

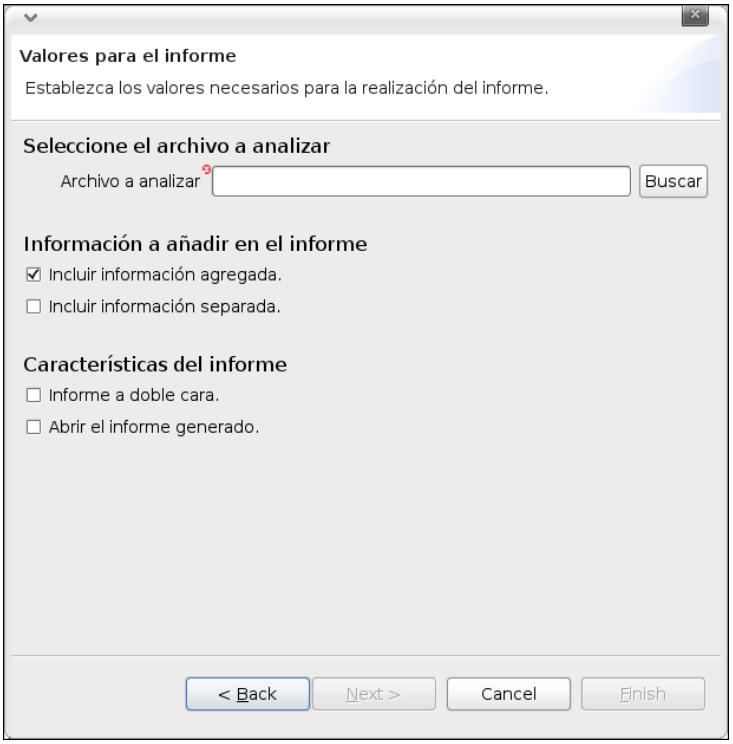


Figura B.34: Campos de un informe de documentos

Tras establecer todos los campos pulse en 'Finish' y se le pedirá que indique el nombre del archivo de salida (figura B.35).

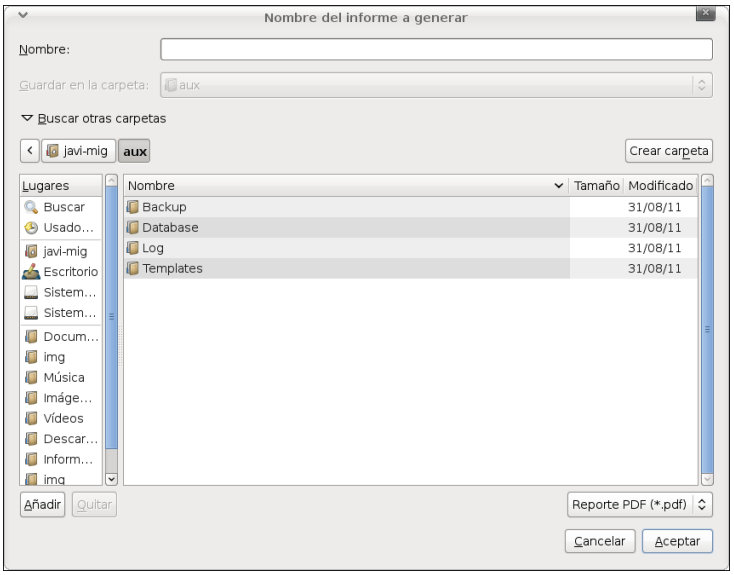


Figura B.35: Nombre del informe a generar

En el caso de que este archivo no pueda ser creado en la dirección especificada o de que ya exista un archivo con idéntico nombre al indicado se le notificará en el momento.

B.5.1.2. Generar un informe del visor activo

Puede obtener un informe del visor activo en cualquier momento, para ello abra el generador de informes, seleccione como fuente de análisis 'Visor activo' y pulse en 'Next >'. Tras esto le aparecerá la ventana de la figura B.36.

Los campos son iguales a los mostrados en el punto exterior. La única salvedad es el campo 'Archivo a analizar', que desaparece.

Cuando termine de configurar el análisis pulse en 'Finish' y se le pedirá que indique el nombre para el archivo generado. Si el nombre es válido se generará el informe.

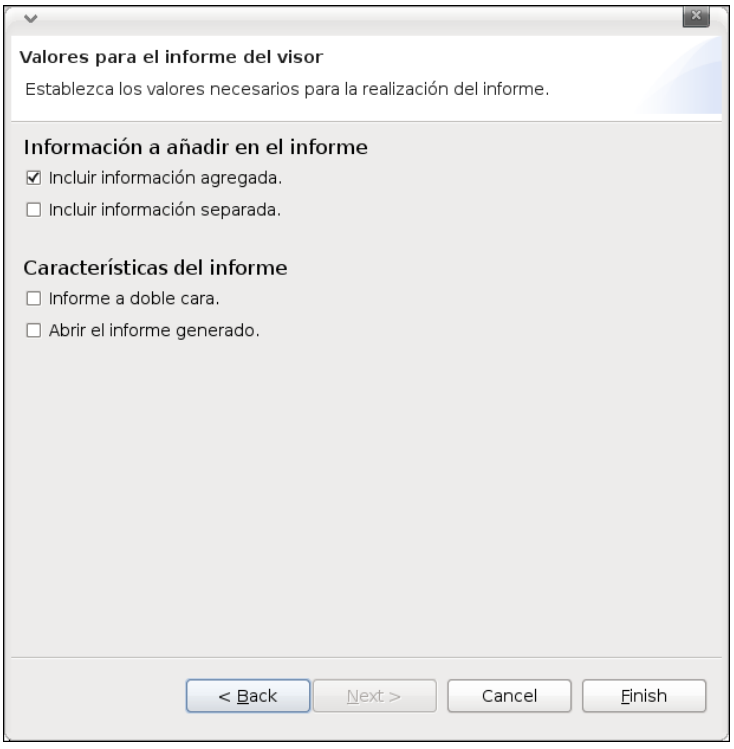


Figura B.36: Campos de un informe del visor activo

B.5.1.3. Generar un informe de un sitio web

Con los informes de sitios web puede analizar dominios completos hasta el nivel de profundidad que establezca.

Para realizar un informe de un sitio web inicie el generador de informes y seleccione 'Sitio web' como fuente del análisis. A continuación pulse en 'Next >'. Aparecerá una ventana como la de la figura B.37 en la que podrá indicar los valores concretos del análisis.

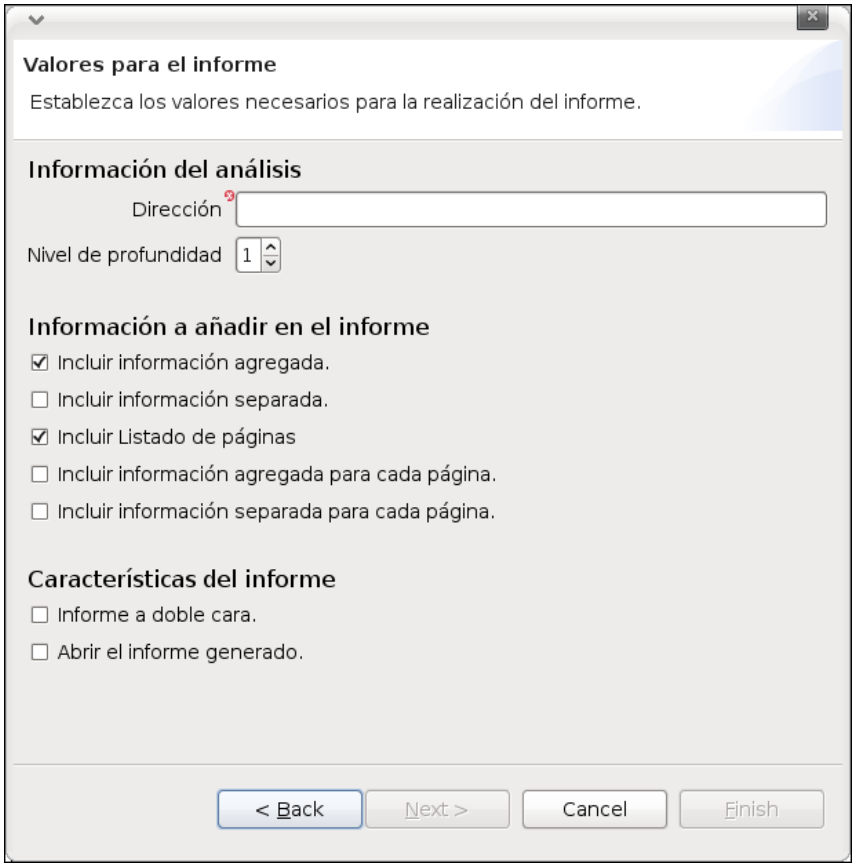


Figura B.37: Campos de un informe de un sitio web

El número de campos disponibles en este caso es superior a los anteriores. Mostramos a continuación el significado de cada campo nuevo introducido (con * los campos obligatorios):

- **Dirección***: Dirección web inicial a analizar.

- **Nivel de profundidad**: Numero de veces que se repite el proceso de captación de enlaces. Si se fija el nivel de profundidad a cero solo se analizará la página introducida, si es uno se analizarán también todas aquellas páginas enlazadas en la dirección introducida, si el nivel es de dos se analizarán también las páginas que aparezcan en cada una de las páginas enlazadas por la inicial, etc. La captación de enlaces se realiza considerando el host de la página inicial por lo que el análisis queda restringido al sitio web que contenga la dirección especificada.

- **Incluir información agregada**: Incluye en el informe una sección con los resultados del análisis considerando todos los paquetes de reglas como uno solo.

- **Incluir información separada**: Incluye una sección para cada paquete de reglas con los que se realizará el análisis.

- **Incluir Listado de páginas**: Mostrar un listado de las páginas analizadas.

- **Incluir información agregada para cada página**: Incluye para cada página analizada los resultados agregados de la misma.

- **Incluir información separada para cada página**: Incluye para cada página analizada los resultados obtenidos con cada paquete de reglas.

Cuando establezca las opciones del análisis pulse en 'Finish' e introduzca el nombre del archivo de salida en el diálogo que se le mostrará. Tras esto comenzará el análisis del sitio.

El análisis de un sitio web es un proceso costoso que se lleva a cabo en distintas etapas.

El primer paso es la descarga de las páginas, las cuales son 'descubiertas' al descargar cada nueva página, tomando los enlaces presentes en ella. Mientras dure este proceso se le mostrará un diálogo de proceso como el de la figura B.38.

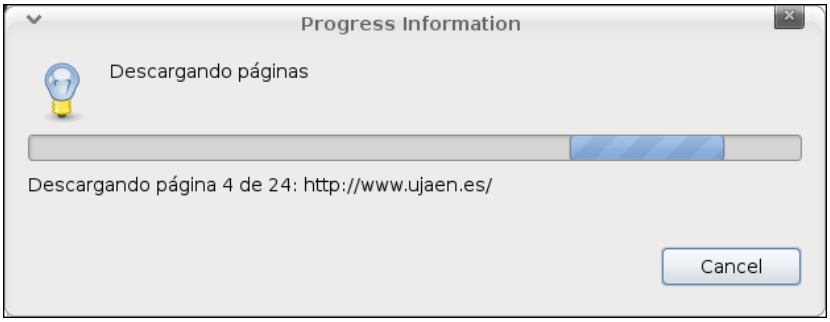


Figura B.38: Descarga de páginas

Cuando se complete la descarga se iniciará el análisis. Durante esta fase se le mostrará una barra de progreso con la que seguir la evolución del análisis (figura B.39).

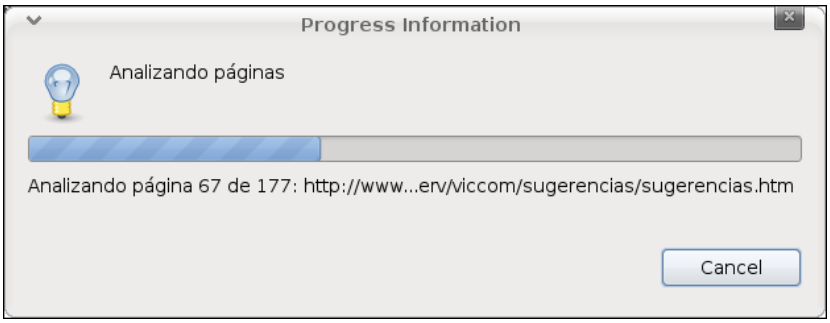


Figura B.39: Análisis de páginas

Una vez analizadas todas las páginas se lleva a cabo la última etapa del análisis la cual consiste en una agregación de los resultados obtenidos (figura B.40).

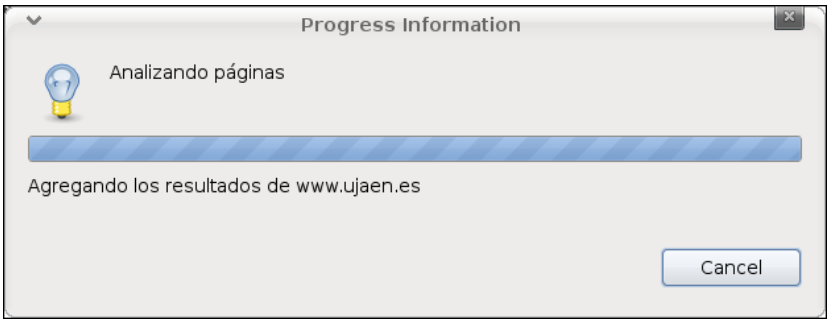


Figura B.40: Agregación de resultados

Finalizada la agregación de resultados podrá consultar el informe generado.

B.5.2. Generador de reglas

Con el generador de reglas podrá modificar los paquetes de reglas que guían el análisis.

Para abrir el generador de reglas seleccione dentro del menú 'Herramientas' la opción 'Generador de reglas' (figura B.41) o bien pulse 'Ctrl+R'.

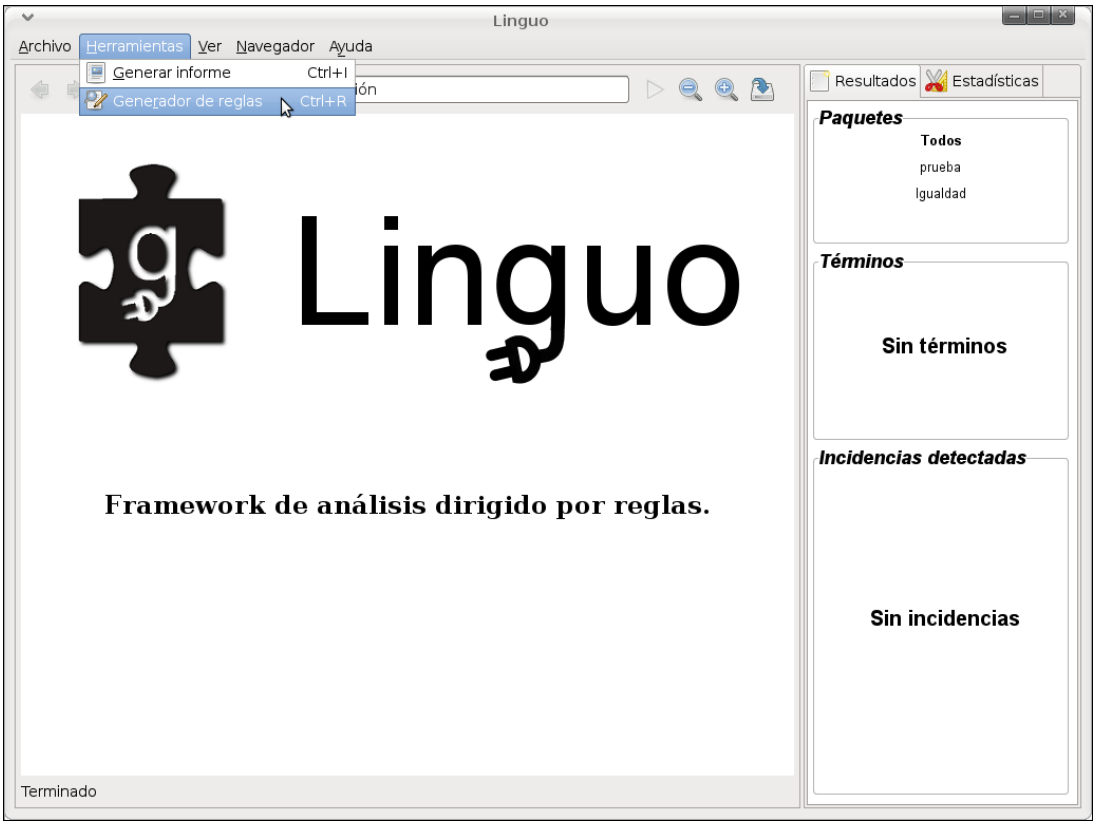


Figura B.41: Acceso al generador de reglas

Mientras se analiza el contenido de los visores el generador de reglas se desactiva temporalmente. Si encuentra el generador de reglas deshabilitado espere unos segundos a que concluya el análisis en curso.

Una vez inicie el generador podrá realizar cualquier modificación sobre los paquetes de reglas.

Dentro de la ventana del generador observará dos regiones diferenciadas. La superior se corresponde con el visor de recursos disponibles, la inferior con el visor de prueba. Para

más información de estos elementos consulte el punto B.3.7.

Para operar sobre los recursos disponibles sitúe el ratón encima de cualquiera de ellos y pulse el botón derecho del mismo. Aparecerá un menú contextual que le permitirá realizar distintas opciones.

Existen una serie de opciones comunes a la mayoría de recursos tales como crear un nuevo elemento, borrarlo, expandir para mostrar todos sus elementos o contraer para ocultarlos. Además, cada elemento presenta una serie de opciones concretas propias del mismo las cuales se verán con detalle en los siguientes puntos.

En la región de recursos se distinguen distintos tipos de iconos. Cada uno diferencia un recurso concreto siendo la correspondencia entre icono y recurso la que aparece en la tabla B.3.

Icono	Tipo de recurso
	Paquetes de reglas
	Paquete de reglas
	Conjunto de términos
	Regla

Cuadro B.3: Tipos de recursos

B.5.2.1. Paquete de reglas

Los paquetes de reglas se representan como elementos del contenedor 'Paquetes de reglas' y se identifican por el nombre del paquete. En la figura B.42 se ve un árbol con dos paquetes de reglas, 'prueba' e 'Igualdad'.

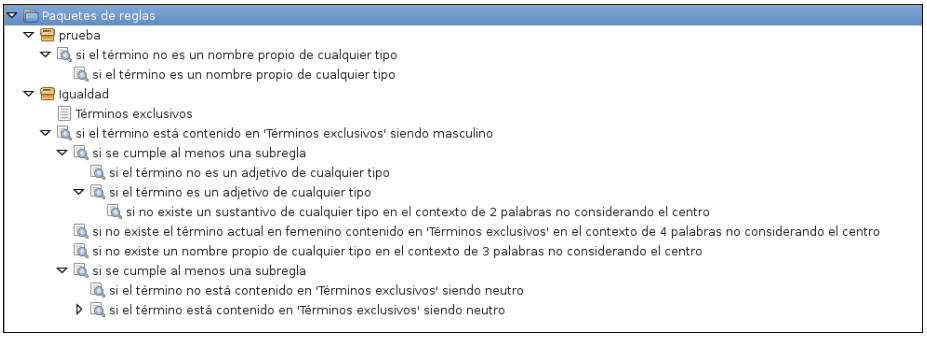


Figura B.42: Paquetes de reglas existentes

Para crear un nuevo paquete de reglas pulse con el botón derecho del ratón sobre 'Paquetes de reglas' y seleccione 'Nuevo paquete de reglas'. Aparecerá una ventana como la de la figura B.43.

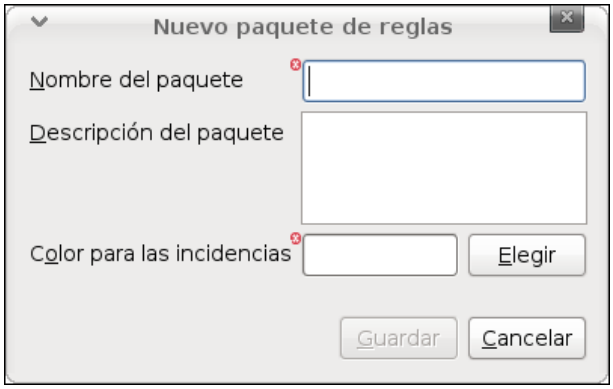


Figura B.43: Nuevo paquete de reglas

En esta ventana podrá introducir los valores para el nuevo paquete de reglas los cuales son (con * los valores obligatorios):

- **Nombre del paquete***: Nombre que recibirá el paquete. Deberá ser único.
- **Descripción del paquete**: Una descripción del tipo de análisis realizado con el paquete.
- **Color para las incidencias***: Color empleado para marcar las incidencias. El color se especifica según su valor hexadecimal.

Si pulsa en 'Guardar' se creará el nuevo paquete de reglas.

Una vez guardados, los paquetes de reglas pueden ser modificados seleccionando sobre ellos con el botón derecho del ratón y pulsando en 'Modificar el paquete de reglas'.

También puede borrar cualquier paquete seleccionando sobre el mismo y pulsando en 'Borrar'. Se eliminará el paquete y cualquier recurso contenido dentro de él.

B.5.2.2. Conjunto de términos

Los conjuntos de términos se muestran dentro de los paquetes de reglas y se identifican por el nombre del conjunto. Para crear un nuevo conjunto de términos situase sobre el paquete de reglas que vaya a contenerlo y pulse con el botón derecho del ratón. En el menú contextual que aparecerá seleccione 'Nuevo conjunto de términos'. Una ventana como la de la figura B.44 aparecerá en pantalla.

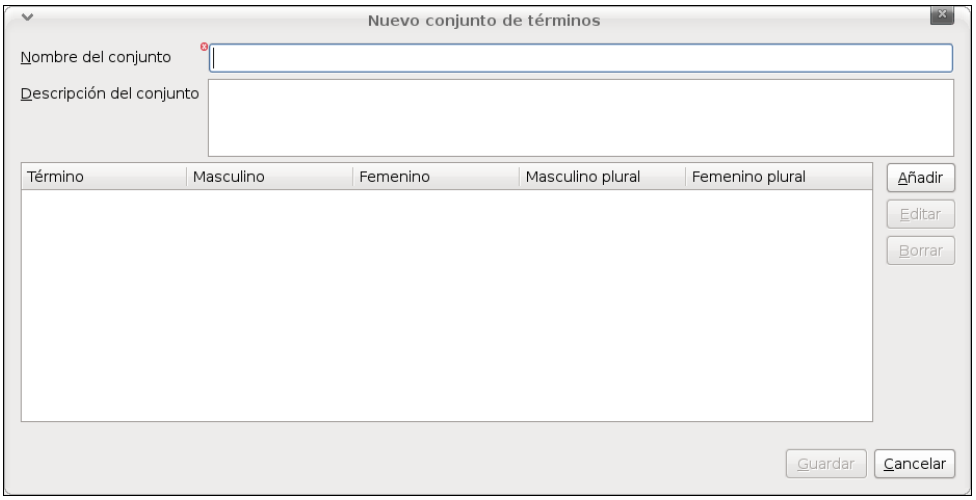


Figura B.44: Nuevo conjunto de términos

En esta ventana podrá introducir los valores para el nuevo conjunto de términos los cuales son (con * los valores obligatorios):

- **Nombre del conjunto***: Nombre que recibirá el conjunto. Deberá ser único.
- **Descripción del conjunto**: Una descripción de los términos contenidos en el conjunto.
- **Listado de términos**: Términos contenidos en el conjunto, se verá como trabajar con estos en el punto B.5.2.3.

Si pulsa en 'Guardar' se creará el nuevo conjunto de términos.

Los conjuntos de términos pueden ser modificados seleccionando sobre los mismos con el botón derecho del ratón y pulsando en 'Modificar el conjunto de términos'.

También puede eliminar un conjunto de términos si lo desea pulsando con el botón derecho del ratón y seleccionando 'Borrar'. Si el conjunto de términos esta siendo usado por cualquier regla deberá eliminar o modificar esta regla antes de poder eliminarlo, en caso contrario el generador no le dejará eliminar el conjunto y obtendrá un mensaje como el de la figura B.45.

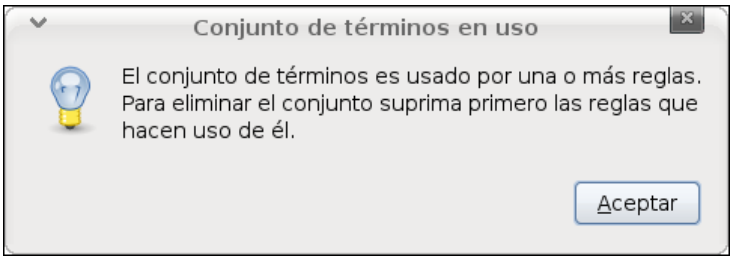


Figura B.45: Eliminar conjunto de términos en uso

B.5.2.3. Términos

Puede modificar los términos de un conjunto de términos desde la ventana de modificación del conjunto. Para añadir un nuevo término pulse en el botón 'Añadir' y aparecerá una ventana como la de la figura B.46.

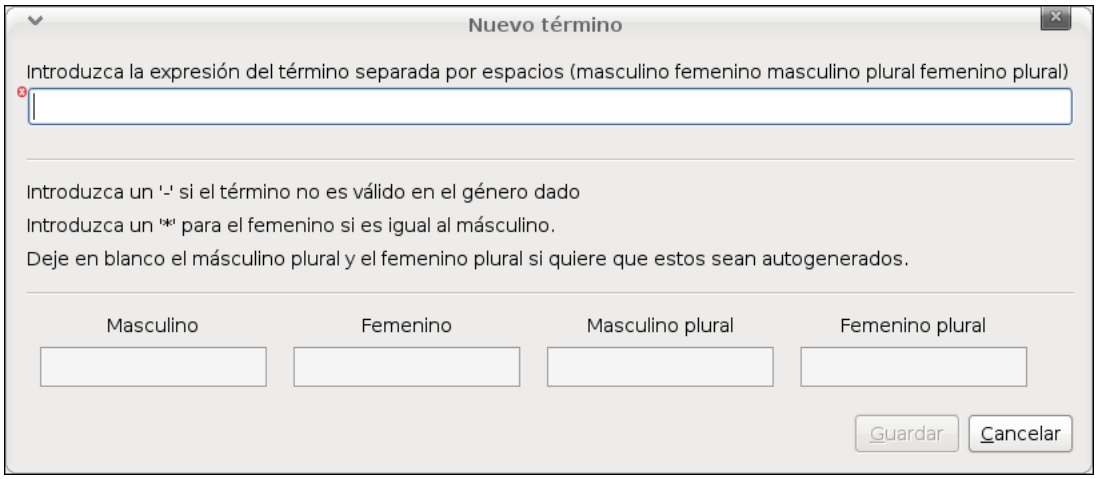


Figura B.46: Nuevo término

En esta ventana deberá introducir el término tal y como se le indica en la parte superior de la misma, esto es, 'masculino femenino masculino singular femenino singular'.

Dispone de una serie de ayudas pensadas para hacer este proceso lo más eficiente posible. Dichas ayudas se le muestran en el propio cuadro de creación.

Cuando finalice (figura B.47) y pulse en 'Guardar' el nuevo término se mostrará en la tabla del conjunto (figura B.48).

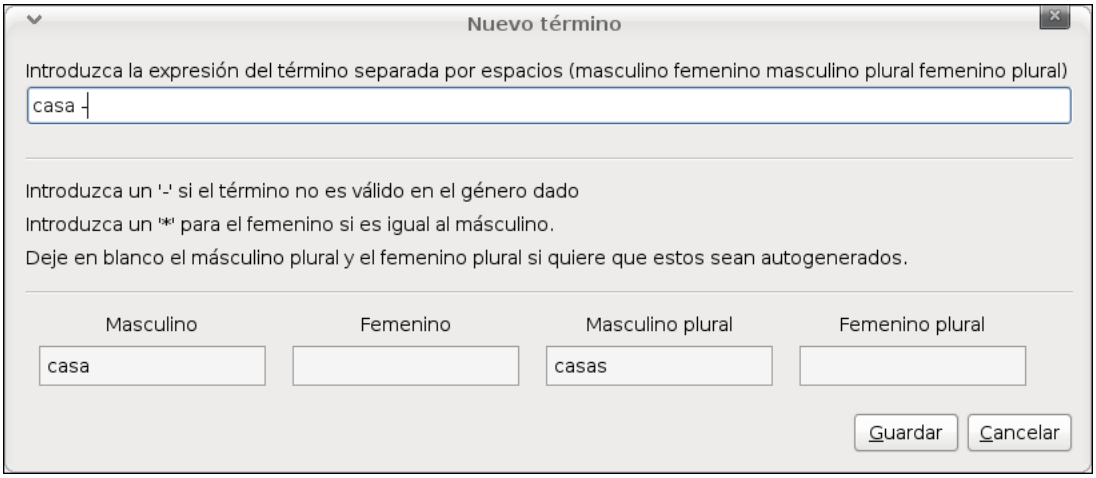


Figura B.47: Introducción de un término

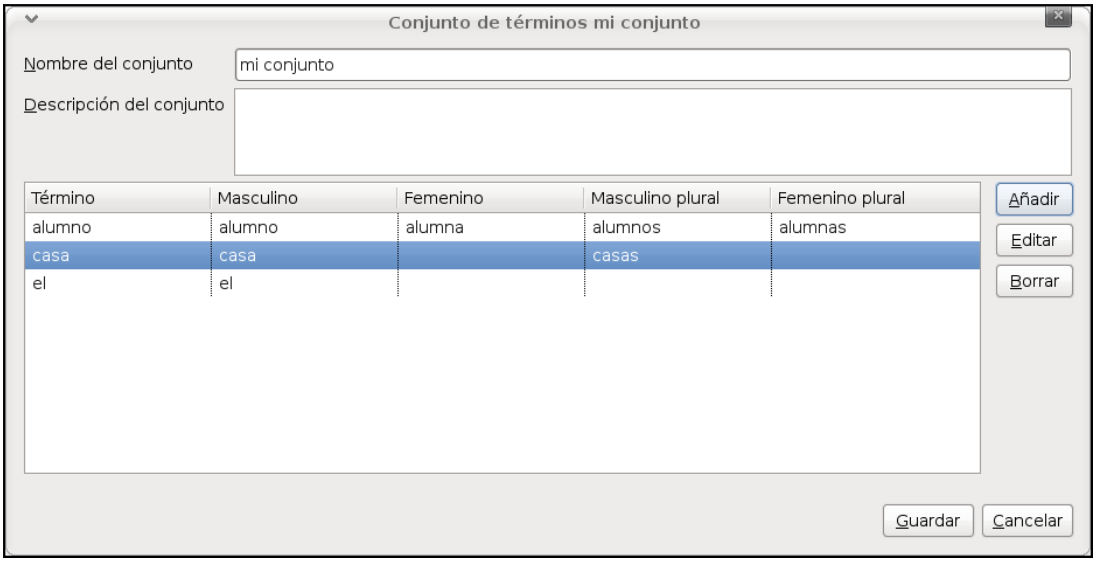


Figura B.48: Término añadido al conjunto de términos

Puede modificar o eliminar cualquier término del conjunto seleccionando los mismos y pulsando en los respectivos controles habilitados.

B.5.2.4. Reglas

Las reglas se muestran dentro de los paquetes de reglas o dentro de otras reglas. Cada regla se identifica por su forma compilada (el resultado de sustituir los valores asignados para todos los elementos). Para crear una nueva regla sitúese encima de un paquete de reglas o de una regla y pulse el botón derecho del ratón, en el menú contextual que aparecerá seleccione la opción 'Nueva regla'. Se le mostrará una ventana como la de la figura B.49.

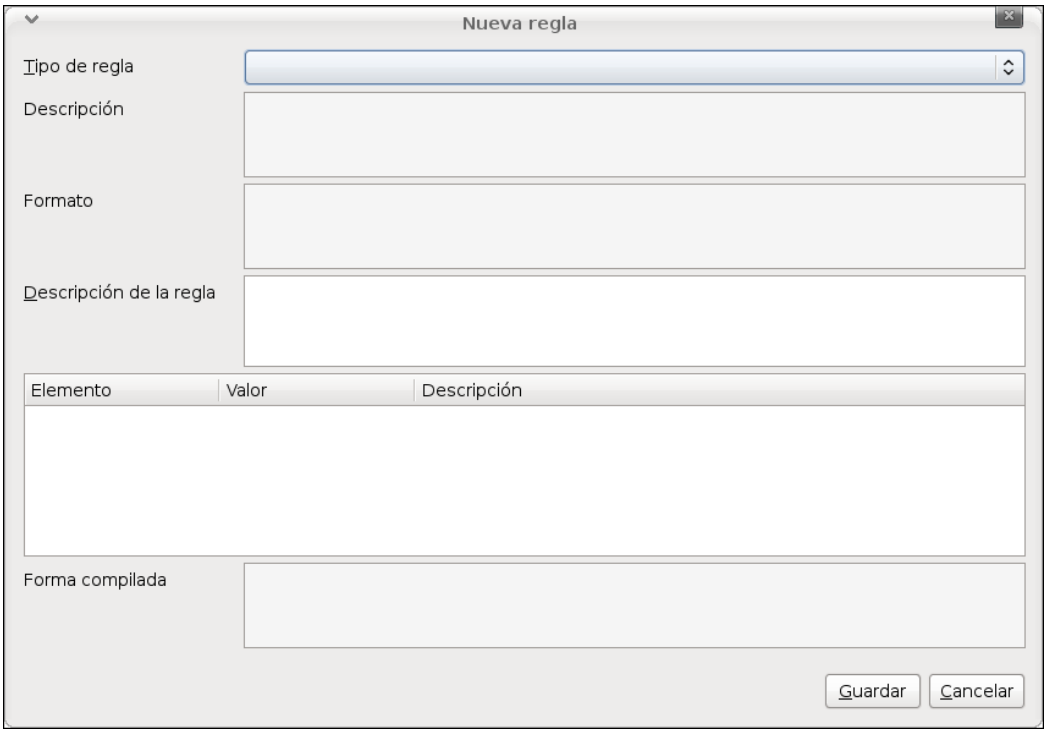


Figura B.49: Nueva regla

En la ventana dispone de una serie de campos, la mayor parte de ellos de carácter informativo, cuya función es la siguiente:

- **Tipo de regla**: Tipo de regla a crear.
- **Descripción**: Descripción del tipo de regla.
- **Formato**: Formato del tipo de regla antes de ser compilada.

- **Descripción de la regla**: Descripción de la regla creada.
- **Elementos**: Elementos del tipo de regla junto con su valor asignado.
- **Forma compilada**: Forma de la regla una vez asignados los valores de los elementos.

El tipo de regla determinará el contenido de la mayor parte de los campos, que se rellenarán de forma automática una vez seleccione el tipo de regla (figura B.50).

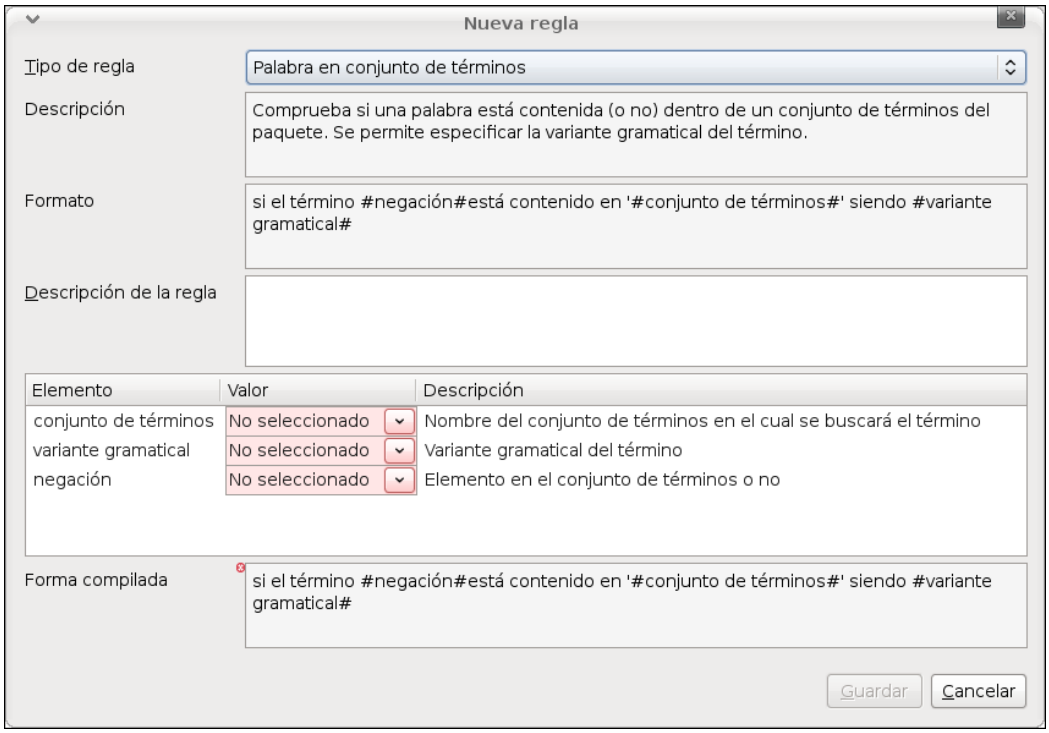


Figura B.50: Selección del tipo de regla

Como se aprecia en la figura B.50, en la tabla de elementos aparecen los valores asignados en color rojo. Esto indica que el valor del elemento no es válido. Si establece un valor válido para los elementos estos cambiarán su color de fondo pasando a ser verde (figura B.51). Una vez haya introducido todos los elementos de la regla pulse en guardar y se creará la nueva regla.

Puede modificar posteriormente cualquier regla pulsando sobre la misma con el botón derecho del ratón y seleccionando 'Modificar regla'. También es posible alterar el orden

de evaluación de una regla mediante las opciones 'Subir regla', 'Bajar regla', 'Aumentar subnivel' y 'Disminuir subnivel' que aparecerán en el menú contextual de la misma.

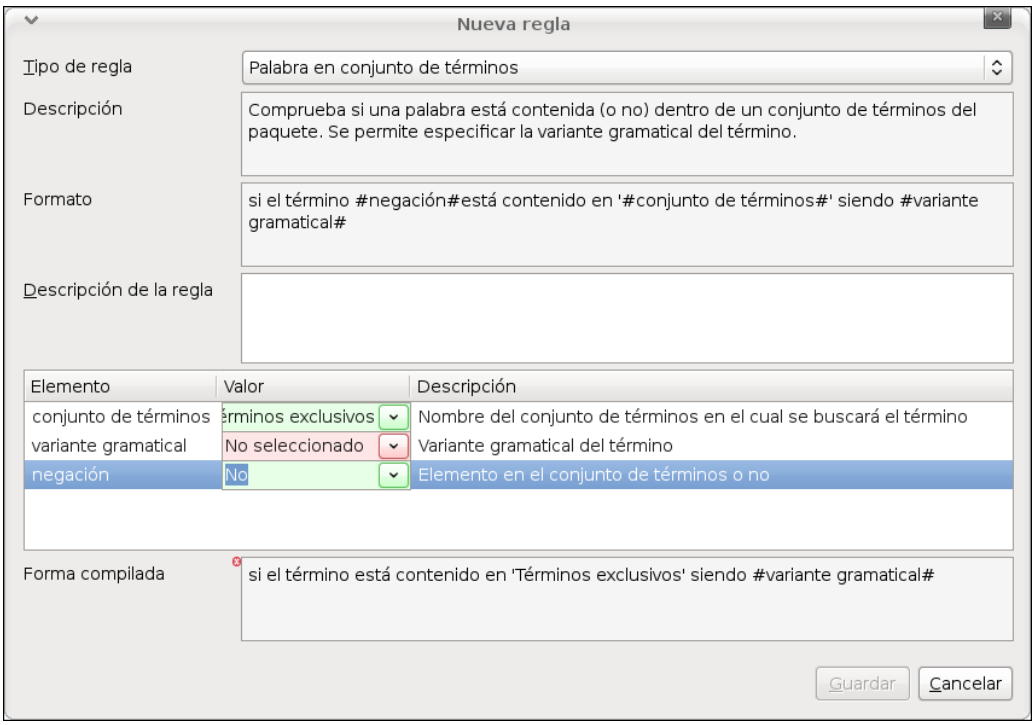


Figura B.51: Elementos de regla válidos

La eliminación de las reglas es igual que el resto de los recursos (seleccione la regla y pulse borrar en el menú contextual), sin embargo, tendrá que confirmar la eliminación de una regla si contiene alguna regla hija en su interior (figura B.52).

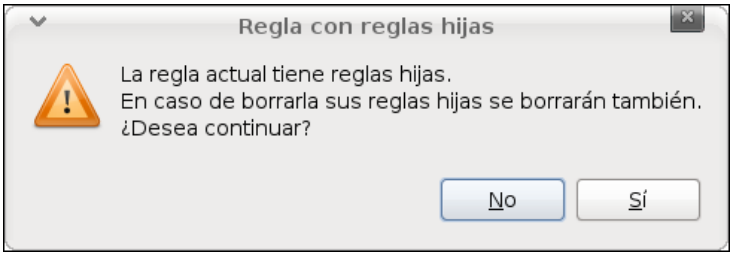


Figura B.52: Eliminar regla con reglas hijas

B.5.2.5. Importar y exportar paquetes de reglas

Linguo le permite importar y exportar paquetes de reglas. Para exportar un paquete pulse sobre él con el botón derecho del ratón y seleccione la opción 'Exportar paquete de reglas'. Se le pedirá que introduzca el nombre del archivo. Los paquetes exportados tendrán la extensión 'linguo'.

Para importar un paquete de reglas seleccione con el ratón sobre 'Paquetes de reglas' y marque 'Importar paquete de reglas'. Se le solicitará que seleccione el archivo y, si no está dañado, se importará.

En el caso de que un paquete a importar tenga un nombre igual a algún paquete existente, se modificará el nombre del nuevo paquete para que no afecte al almacenado. No existe por tanto riesgo de que se sobreescriban los paquetes.

La figura B.53 refleja esta situación, donde se ha importado un paquete de reglas de nombre 'prueba' y ha sido renombrado con el nombre 'prueba-1' para no sobreescribir el anterior.

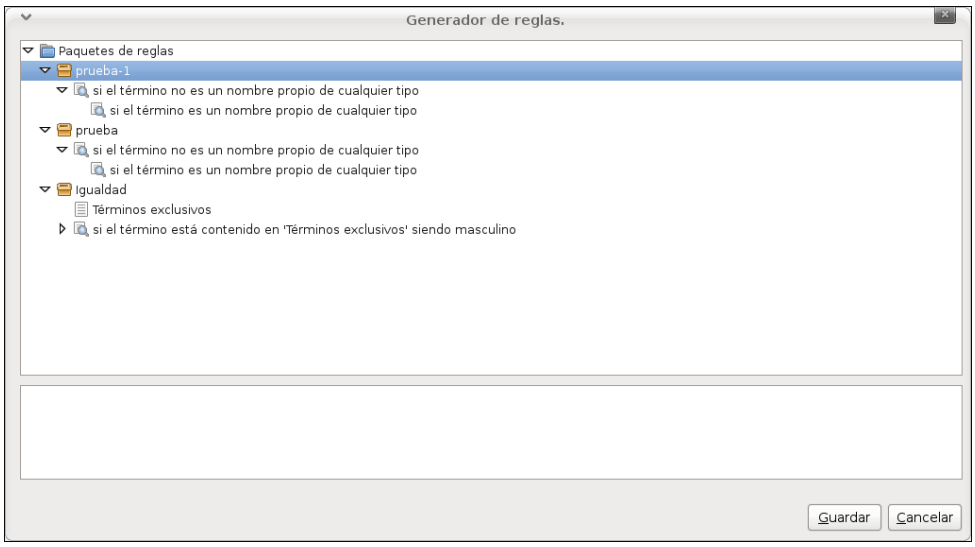


Figura B.53: Importación de paquete duplicado

B.5.2.6. Visor de prueba

El visor de prueba le permite introducir cualquier texto para comprobar si el análisis se adecua a lo esperado. Para emplear el visor de prueba solo escriba en él, el texto introducido se analizará inmediatamente. Cada uno de las incidencias detectadas se mostrarán subrayando la misma con una linea de color igual al establecido en el paquete de reglas (figura B.54).

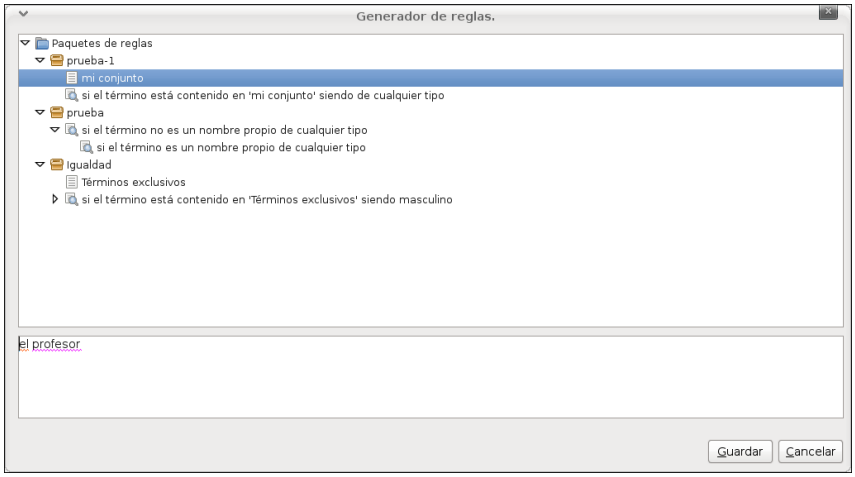


Figura B.54: Marcado de incidencias en visor de prueba

Si algún término es una incidencia en más de un paquete la linea que se empleará para marcar la incidencia será de color negro (figura B.55).

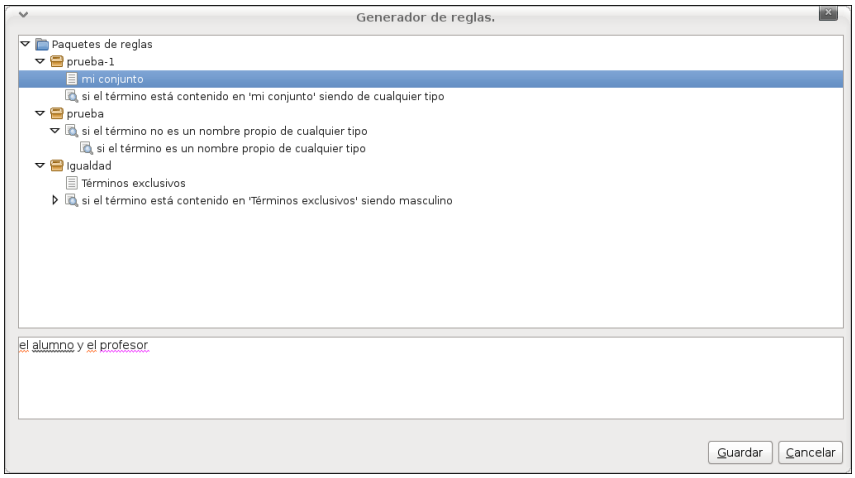


Figura B.55: Incidencia por múltiples paquetes

B.5.2.7. Almacenar los cambios realizados

Cualquier modificación realizada sobre los paquetes de reglas no tendrá ningún efecto hasta que no pulse en el botón guardar del generador. En el momento que pulse este botón los cambios se harán efectivos y automáticamente se volverá a analizar el contenido del visor activo.

B.6. Atajos de teclado

Comando	Atajo de teclado	Función
Anterior	Shift+Alt+Left	Cargar dirección anterior
Siguiente	Shift+Alt+Right	Cargar dirección siguiente
Parar	Esc	Detener la carga actual
Refrescar	Ctrl+F5	Refrescar la página
Inicio	Home	Ir a la página inicial
Ir	Shift+Enter	Ir a la página introducida
Reiniciar zoom	Ctrl+0	Restablecer zoom
Disminuir zoom	Ctrl+-	Disminuir zoom
Aumentar zoom	Ctrl++	Aumentar zoom
Guardar página	Ctrl+Alt+S	Guardar la página actual
Cambiar URL	Ctrl+L	Introducir una nueva URL
Nuevo archivo	Ctrl+Alt+N	Crear un nuevo archivo
Abrir archivo	Ctrl+O	Abrir un archivo
Salir	Ctrl+X	Abandonar el programa
Generar informe	Ctrl+I	Abrir el asistente de informes
Generador de reglas	Ctrl+R	Abrir el generador de reglas
Contenido de la ayuda	Ctrl+H	Abrir la ayuda

Cuadro B.4: Atajos de teclado

B.7. FAQ

B.7.1. Algunas incidencias detectadas no aparecen en el visor web

No todo el contenido que se muestra en una página web es el que existe realmente. Existen contenidos que los diseñadores de las páginas ocultan mientras realizan cambios, o que solo se muestran bajo determinadas condiciones (menús desplegables) o incluso que están escondidos intencionadamente.

Linguo en el peor de los casos podría no reconocer una palabra por cualquier circunstancia pero nunca podría reconocer una palabra si esta no se encontrase realmente en el documento.

B.7.2. El contenido del navegador web se muestra mal una vez analizado

Linguo hace uso de un sencillo mecanismo para notificar las incidencias, obtiene su posición inicial y final y asigna a este fragmento un estilo determinado. Lo anterior no debería provocar ningún error, sin embargo, puede que por defectos en la página web abierta se produzcan. Estas situaciones pueden darse en aquellos casos en los que las páginas no cumplan con los estándares web o estén mal diseñadas. En estos casos al modificar el código el resultado es inesperado.

B.7.3. El análisis falla con algunos documentos

Linguo solo trata de forma nativa con los documentos en texto plano, para el resto de documentos analizados hace uso de diversas librerías gratuitas, si alguna de estas no es capaz de trabajar adecuadamente con un documento dado, tampoco lo podrá hacer Linguo. Sin embargo, en la mayoría de los casos los problemas se deberán a que el documento se encuentra dañado o con una codificación que no es reconocida por el parser. Intente guardar el documento en un formato distinto y vuelva a repetir la operación, si nuevamente falla el análisis, modifique su codificación de caracteres.

B.7.4. El comportamiento del componente X no es correcto

Verifique si cumple con los requisitos software expuestos en el manual de instalación. Tenga en cuenta que la utilización de software de forma embebida es una cuestión delicada y que cualquier modificación en la API del programa puede volverlo inaccesible. Si cuenta con las versiones requeridas y sigue teniendo problemas, será con certeza culpa de Linguo.

Bibliografía

[1] El poder de las políticas lingüísticas. http://www.juntadeandalucia.es/institutodelamujer/IMG/pdf/12_opinion.pdf (Visitado en enero de 2011).

[2] Guía para un uso igualitario del lenguaje administrativo. http://www.mujeresenred.net/IMG/pdf/Guialenguaje.pdf (Visitado en enero de 2011).

[3] El lenguaje que pone primero a la persona. http://sitebuilder.redarrowgroup.com/client_files/7_page_files/pfl09spanish.pdf (Visitado en enero de 2011).

[4] Hudson, G. (2000) Essential Introductory Linguistics, Oxford: Blackwell.

[5] Lingüistics. http://en.wikipedia.org/wiki/Linguistics (Visitado en agosto de 2011).

[6] Introducción a la Lingüística. http://www.fundaciondoctordepando.com/GLOSARIOSyVOCABULARIOS/DiccionariodeLenguasyDialectosdelMundo-INTRODUCCION.htm (Visitado en agosto de 2011).

[7] WordSmith Tools. http://www.lexically.net/wordsmith/purchasing.htm (Visitado en agosto de 2011).

[8] The Plagiarism Checker. http://www.dustball.com/cs/plagiarism.checker/ (Visitado en agosto de 2011).

[9] Concordance. http://www.concordancesoftware.co.uk/ (Visitado en agosto de 2011).

[10] Bioclipse. http://bioclipse.net (Visitado en agosto de 2011).

[11] Azureus. http://azureus.sourceforge.net (Visitado en agosto de 2011).

[12] J3DWorkbench. http://eclectic3d.net (Visitado en agosto de 2011).

[13] CCTVnet. http://www.cctvnet.biz (Visitado en agosto de 2011).

[14] Kalypso. http://www.kalypso-simulation-platform.org (Visitado en agosto de 2011).

[15] BaZaa. http://www.bazaanetworks.com (Visitado en agosto de 2011).

[16] Global Vision Systems. http://www.global-vision-systems.com (Visitado en agosto de 2011).

[17] El procesamiento del lenguaje natural aplicado al análisis del contenido de los documentos. http://revistas.ucm.es/byd/11321873/articulos/RGID9696220205A.PDF (Visitado en enero de 2011).

[18] B. Pang and L. Lee. Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval, Vol. 2, Nos. 1-2, pp. 1-135, 2008.

[19] J. Wiebe Manual and Automatic Subjectivity and Sentiment Analysis [Presentation]. University of Pittsburgh, 2007

[20] Ingeniería social (seguridad informática). http://es.wikipedia.org/wiki/Ingeniería_social_(seguridad_informática) (Visitado en agosto de 2011).

[21] Spam. http://es.wikipedia.org/wiki/Spam (Visitado en agosto de 2011).

[22] López-Ibor Aliño, Juan J. & Valdés Miyar, Manuel. (2002) DSM-IV-TR. Manual diagnóstico y estadístico de los trastornos mentales. Texto revisado. Barcelona. Masson.

[23] Síndrome de Asperger: National Institute of Neurological Disorders and Stroke (NINDS). http://espanol.ninds.nih.gov/trastornos/el_Sindrome_de_Asperger.htm (Visitado en agosto de 2011).

[24] Remove.

[25] N. Bartlett. (2009) OSGi In Practice. (Auto editado).

[26] OSGi Core Concepts Overview. http://www.weiglewilczek.com/fileadmin/Publications/Poster_OSGi_web.pdf (Visitado en agosto de 2011).

[27] D. Rubel and E. Clayberg. (2009) Eclipse Plug-Ins, Addison-Wesley Proffesional.

[28] Rich Client Platforms. http://www.nadig.org/technology/rcp/rich-client-platforms.pdf (Visitado en enero de 2011).

[29] J. McAffer, J. Lemieux and C. Aniszczyk. (2010). Eclipse Rich Client Platform, Second Edition, Addison-Wesley Proffesional.

[30] Netbeans. http://www.netbeans.org (Visitado en agosto de 2011).

[31] Spring Framework. http://www.springsource.org/ (Visitado en agosto de 2011).

[32] Eclipse. http://www.eclipse.org/ (Visitado en agosto de 2011).

[33] Write once, run anywhere. http://en.wikipedia.org/wiki/Write_once,_run_anywhere (Visitado en agosto de 2011).

[34] TIOBE Programming Community Index for August 2011. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html (Visitado en agosto de 2011).

[35] HTML Parser. http://htmlparser.sourceforge.net/ (Visitado en agosto de 2011).

[36] SWTChart. http://www.swtchart.org/ (Visitado en agosto de 2011).

[37] NOA. http://ubion.ion.ag/solutions/004niceofficeaccess (Visitado en agosto de 2011).

[38] HyperSQL. http://hsqldb.org/ (Visitado en agosto de 2011).

[39] iText. http://www.itextpdf.com/ (Visitado en agosto de 2011).

[40] jFreeChart. http://www.jfree.org/jfreechart/ (Visitado en agosto de 2011).

[41] JUnit. http://www.junit.org/ (Visitado en agosto de 2011).

[42] Cobertura. http://cobertura.sourceforge.net/ (Visitado en agosto de 2011).

[43] Mockito. http://mockito.org/ (Visitado en agosto de 2011).

[44] SWTBot. http://www.eclipse.org/swtbot/ (Visitado en agosto de 2011).

[45] Git. http://git-scm.com/ (Visitado en agosto de 2011).

[46] ODFDOM. http://odftoolkit.org/projects/odfdom/pages/Home (Visitado en agosto de 2011).

[47] Máquina virtual Java. http://java.com/es/download/ (Visitado en agosto de 2011).

[48] XULRunner. https://developer.mozilla.org/en/xulrunner (Visitado en agosto de 2011).

[49] OpenOffice.org. http://www.openoffice.org/ (Visitado en agosto de 2011).

[50] M. Scarpino, S. Holder, NG. Stanford and L. Mihalkovic. (2005) SWT/JFace in Action, Manning Publications.

[51] IEEE, 1993.

[52] J. Rumbaugh, I. Jacobson and G. Booch. (2004) The Unified Modeling Language Reference Manual, 2nd Edition, Addison-Wesley.

[53] J. Arlow and I. Neustadt. (2005) UML 2 and the unified process, 2nd Edition, Addison-Wesley.

[54] T. Halton. (2004) SWT - A Developer's Notebook, O Reilly.

[55] Proyecto Tango!. http://tango.freedesktop.org/Tango_Desktop_Project (Visitado en agosto de 2011).

[56] C. Blé Jurado. (2010) Diseño ágil con TDD. (Auto editado).

[57] Apache POI. http://poi.apache.org/ (Visitado en agosto de 2011).

[58] Apache Aries. http://aries.apache.org/ (Visitado en agosto de 2011).

[59] DataNucleus. http://www.datanucleus.org/ (Visitado en agosto de 2011).

[60] Birt. http://www.eclipse.org/birt/ (Visitado en agosto de 2011).

[61] GX WebManager. http://www.gxsoftware.com/ (Visitado en agosto de 2011).

[62] Confluence. http://www.atlassian.com/software/confluence/ (Visitado en agosto de 2011).

[63] Weblogic. http://www.oracle.com/us/products/middleware/application-server/index.html (Visitado en agosto de 2011).

[64] Eclipse Virgo. http://www.eclipse.org/virgo/ (Visitado en agosto de 2011).

[65] EasyBeans. http://www.easybeans.org (Visitado en agosto de 2011).

[66] SAP BusinessObjects Event Insight. http://www.sap.com/solutions/sapbusinessobjects/large/eim/event-insight/index.epx (Visitado en agosto de 2011).

[67] JBoss. http://www.jboss.org (Visitado en agosto de 2011).

[68] GlassFish. http://glassfish.java.net (Visitado en agosto de 2011).

[69] JOnAS 5. http://jonas.ow2.org (Visitado en agosto de 2011).

[70] WebSphere Application Server. http://www-01.ibm.com/software/webservers/appserv/was/ (Visitado en agosto de 2011).

[71] WSO2 Carbon. http://wso2.com/products/carbon (Visitado en agosto de 2011).

[72] nombra.en.red. http://www.migualdad.es/mujer/descargas/ (Visitado en agosto de 2011).

[73] Themis. http://www.themis.es (Visitado en agosto de 2011).