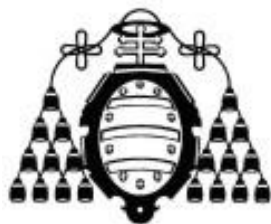


UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

AUTOMATIZACIÓN DE LA MEDICIÓN DE LA USABILIDAD WEB
MEDIANTE LA MÉTRICA DE IVORY

DIRECTORA: M^a del Carmen Suárez Torrente

CODIRECTORA: Ana Belén Martínez Prieto

Firma manuscrita en tinta azul, que parece ser la del autor, Cristina Alonso Sierra.

**Vº Bº del Director del
Proyecto**

AUTOR: Cristina Alonso Sierra

Agradecimientos

En primer lugar quiero dar las gracias a mi directora, M^a del Carmen Suárez Torrente, y a mi codirectora, Ana Belén Martínez Prieto, por toda la ayuda y consejos que me han dado y por su dedicación y apoyo a este proyecto.

A Javier Oyón Fernández y al resto de alumnos de la Escuela que han participado de alguna forma en Atenea, por todo el trabajo que han realizado.

También quiero dar las gracias a una serie de personas que desde uno u otro ámbito han hecho que sea posible que haya llegado hasta aquí:

A mis padres, especialmente a mi madre, y a mi abuela, por permitirme estudiar lo que he querido, por ayudarme siempre que lo he necesitado, por su apoyo incondicional, por animarme siempre a seguir hacia delante y por querer que “llegue a ser algo en la vida”.

A Jose, por todo su apoyo durante el transcurso de este Máster y por hacerme las horas de prácticas y proyecto mucho más felices.

A mis amigos.

Muchas gracias a todos.

Resumen

La usabilidad es uno de los factores más importantes a tener en cuenta a la hora de diseñar una interfaz de usuario, por lo que las herramientas automáticas para su evaluación son de gran ayuda en el desarrollo de sitios web usables.

El presente proyecto tiene como objetivo construir una herramienta que permita realizar un análisis cuantitativo de la usabilidad de un sitio web de manera automática. Para ello se realizará una implementación de la métrica propuesta por Ivory para el análisis cuantitativo de la usabilidad web.

Cabe destacar que la herramienta desarrollada se integrará en el sistema ya existente Atenea 3.0, que ofrece la funcionalidad necesaria para evaluar la accesibilidad y usabilidad en los sitios web de forma cuantitativa utilizando diferentes métricas.

Palabras Clave

Usabilidad, web, evaluación de la usabilidad, Ivory, Atenea 3.0.

Índice General

CAPÍTULO 1. MEMORIA DEL PROYECTO.....	19
1.1 RESUMEN DE LA MOTIVACIÓN, OBJETIVOS Y ALCANCE DEL PROYECTO	19
1.2 RESUMEN DE TODOS LOS ASPECTOS.....	20
CAPÍTULO 2. INTRODUCCIÓN.....	23
2.1 JUSTIFICACIÓN DEL PROYECTO.....	23
2.2 ESTUDIO DE LA SITUACIÓN ACTUAL	25
2.2.1 <i>Herramientas existentes para evaluar la usabilidad web</i>	25
2.2.2 <i>Evaluación de Alternativas</i>	30
PARTE I. INVESTIGACIÓN SOBRE LA PROPUESTA DE IVORY PARA LA EVALUACIÓN CUANTITATIVA DE LA USABILIDAD EN SITIOS WEB	31
CAPÍTULO 3. MÉTRICA DE IVORY	32
3.1 INTRODUCCIÓN Y OBJETIVOS	32
3.2 ASPECTOS EXAMINADOS Y MEDIDAS A TOMAR	33
3.2.1 <i>Texto</i>	33
3.2.2 <i>Enlaces</i>	34
3.2.3 <i>Imágenes</i>	35
3.2.4 <i>Formato de texto</i>	35
3.2.5 <i>Formato de enlaces</i>	36
3.2.6 <i>Formato de imágenes</i>	36
3.2.7 <i>Formato de página</i>	36
3.2.8 <i>Función de la página</i>	37
3.2.9 <i>Rendimiento</i>	37
3.2.10 <i>Arquitectura del sitio</i>	37
3.3 PERFILES Y MODELOS ESTADÍSTICOS OBTENIDOS	38
3.4 MÉTRICA DE IVORY	39
3.4.1 <i>Contador de enlaces (link count)</i>	39
3.4.2 <i>Complejidad de lectura (Reading complexity)</i>	39
3.4.3 <i>Contador de posición de texto (Text positioning count)</i>	41
3.4.4 <i>Contador de colores (Color count)</i>	41
3.4.5 <i>Tamaño de la página (Page size)</i>	42
3.4.6 <i>Porcentaje de texto en el cuerpo (Body text percentage)</i>	42
3.5 CONCLUSIÓN DEL ESTUDIO.....	44
PARTE II. DESARROLLO DE LA AUTOMATIZACIÓN DE LA MEDICIÓN DE LA USABILIDAD MEDIANTE LA MÉTRICA DE IVORY	45
CAPÍTULO 4. ASPECTOS TEÓRICOS.....	46
4.1 USABILIDAD.....	46
4.1.1 <i>Métodos de evaluación de la usabilidad</i>	47
4.2 ATENEA 3.0	48
4.3 JSTYLEPARSER.....	49
CAPÍTULO 5. PLANIFICACIÓN Y RESUMEN DE PRESUPUESTOS	50
5.1 PLANIFICACIÓN.....	50
5.2 RESUMEN DEL PRESUPUESTO	51

CAPÍTULO 6. ANÁLISIS.....	53
6.1 DEFINICIÓN DEL SISTEMA.....	53
6.1.1 <i>Determinación del Alcance del Sistema</i>	53
6.2 REQUISITOS DEL SISTEMA.....	54
6.2.1 <i>Obtención de los Requisitos del Sistema</i>	54
6.2.2 <i>Identificación de Actores del Sistema</i>	55
6.2.3 <i>Especificación de Casos de Uso</i>	55
6.3 IDENTIFICACIÓN DE LOS SUBSISTEMAS EN LA FASE DE ANÁLISIS.....	56
6.3.1 <i>Descripción de los Subsistemas</i>	56
6.3.2 <i>Descripción de los Interfaces entre Subsistemas</i>	56
6.4 DIAGRAMA DE CLASES PRELIMINAR DEL ANÁLISIS	57
6.4.1 <i>Diagrama de Clases</i>	57
6.4.2 <i>Descripción de las Clases</i>	58
6.5 ANÁLISIS DE CASOS DE USO Y ESCENARIOS	60
6.5.1 <i>Caso de Uso 1.1</i>	60
6.6 ESPECIFICACIÓN DEL PLAN DE PRUEBAS	61
CAPÍTULO 7. DISEÑO DEL SISTEMA.....	63
7.1 ARQUITECTURA DEL SISTEMA.....	63
7.1.1 <i>Diagramas de Paquetes</i>	63
7.1.2 <i>Diagramas de Componentes</i>	66
7.1.3 <i>Diagramas de Despliegue</i>	67
7.2 DISEÑO DE CLASES	68
7.2.1 <i>Diagrama de Clases del paquete IVORY.metrics</i>	68
7.2.2 <i>Diagrama de Clases del paquete IVORY.checkpoints</i>	69
7.2.3 <i>Diagrama de Clases del paquete IVORY.util</i>	70
7.2.4 <i>Diagrama de Clases del paquete IVORY.test</i>	70
7.3 DIAGRAMAS DE INTERACCIÓN Y ESTADOS.....	71
7.3.1 <i>Diagrama de secuencia</i>	71
7.4 ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS	73
7.4.1 <i>Pruebas Unitarias</i>	73
7.4.2 <i>Pruebas de Integración y del Sistema</i>	75
CAPÍTULO 8. IMPLEMENTACIÓN DEL SISTEMA	77
8.1 ESTÁNDARES Y NORMAS SEGUIDOS	77
8.1.1 <i>API Atenea3.0</i>	77
8.2 LENGUAJES DE PROGRAMACIÓN.....	78
8.2.1 <i>Java</i>	78
8.2.2 <i>HTML</i>	78
8.2.3 <i>CSS</i>	78
8.3 HERRAMIENTAS Y PROGRAMAS USADOS PARA EL DESARROLLO	79
8.3.1 <i>Eclipse Java EE IDE for Web Developers</i>	79
8.3.2 <i>AppServ</i>	79
8.3.3 <i>Tomcat</i>	79
8.3.4 <i>Notepad++</i>	79
8.4 CREACIÓN DEL SISTEMA.....	80
8.4.1 <i>Problemas Encontrados</i>	80
8.4.2 <i>Descripción Detallada de las Clases</i>	80
CAPÍTULO 9. DESARROLLO DE LAS PRUEBAS	91
9.1 PRUEBAS UNITARIAS.....	91

9.1.1	<i>MetricIVORYTest</i>	91
9.1.2	<i>Checkpoint109Test</i>	91
9.1.3	<i>Checkpoint110Test</i>	91
9.1.4	<i>Checkpoint111Test</i>	93
9.1.5	<i>Checkpoint112Test</i>	93
9.1.6	<i>Checkpoint113Test</i>	93
9.2	PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA	94
CAPÍTULO 10. ESTUDIOS EMPÍRICOS APLICANDO LA MÉTRICA DE IVORY.....		95
10.1	ANÁLISIS DEL NIVEL DE USABILIDAD EN ESPAÑA.....	95
10.1.1	<i>La usabilidad en los ayuntamientos</i>	95
10.1.2	<i>La usabilidad en las universidades</i>	98
10.1.3	<i>La usabilidad en las empresas</i>	100
10.1.4	<i>Comparación del nivel de usabilidad en webs de ayuntamientos, empresas y universidades</i>	102
10.2	RELACIÓN ENTRE LA USABILIDAD Y LA ACCESIBILIDAD.....	105
CAPÍTULO 11. MANUALES DEL SISTEMA.....		111
11.1	MANUAL DE INSTALACIÓN	111
11.1.1	<i>Requisitos del sistema</i>	111
11.1.2	<i>Elementos necesarios para la instalación del sistema</i>	111
11.1.3	<i>Instalación de AppServ</i>	111
11.1.4	<i>Instalación de Tomcat</i>	117
11.1.5	<i>Instalación de Atenea</i>	122
11.2	MANUAL DE EJECUCIÓN.....	129
11.2.1	<i>Arranque del sistema</i>	129
11.2.2	<i>Parada del sistema</i>	129
11.3	MANUAL DE USUARIO	130
11.3.1	<i>Realizar análisis desde la parte pública de Atenea 3.0</i>	130
11.3.2	<i>Realizar análisis desde la parte privada de Atenea 3.0</i>	132
11.4	MANUAL DEL PROGRAMADOR.....	140
11.4.1	<i>Creación de nuevos checkpoints</i>	140
11.4.2	<i>Creación de nuevas métricas</i>	140
11.4.3	<i>Empaquetado</i>	140
CAPÍTULO 12. CONCLUSIONES Y AMPLIACIONES.....		143
12.1	CONCLUSIONES	143
12.2	AMPLIACIONES.....	144
CAPÍTULO 13. PRESUPUESTO.....		145
13.1	PRESUPUESTO DE COSTES	145
13.2	PRESUPUESTO DEL CLIENTE	147
CAPÍTULO 14. REFERENCIAS BIBLIOGRÁFICAS		149
14.1	LIBROS Y ARTÍCULOS.....	149
14.2	REFERENCIAS EN INTERNET	150
CAPÍTULO 15. APÉNDICES.....		153
15.1	CONTENIDO ENTREGADO EN EL CD-ROM.....	153
15.1.1	<i>Contenidos</i>	153
15.1.2	<i>Código Ejecutable e Instalación</i>	154
15.2	ÍNDICE ALFABÉTICO	155

15.3	CÓDIGO FUENTE	156
15.3.1	<i>Paquete IVORY.metrics</i>	156
15.3.2	<i>Paquete IVORY.checkpoints</i>	158
15.3.3	<i>Paquete IVORY.util</i>	167
15.3.4	<i>Paquete IVORY.test</i>	170

Índice de Figuras

Ilustración 2.1 – Herramienta Navflow.....	26
Ilustración 2.2 – UserPlus.....	27
Ilustración 2.3 – ClickHeat.....	27
Ilustración 2.4 – Mapa de calor elaborado por ClickDensity.....	28
Ilustración 2.5 – Mapa de calor elaborado por Crazy Egg.....	29
Figura 3.1 - Elementos que componen el diseño de una página web según Ivory.....	33
Ilustración 3.1 – Ejemplo de la medida “link count”.....	39
Ilustración 3.2 – Ejemplo de la medida “Reading complexity”.....	40
Ilustración 3.3 – Ejemplo de la medida “Text positioning count”.....	41
Ilustración 3.4 – Ejemplo de la medida “Color count”.....	42
Ilustración 3.5 – Ejemplo I de la medida “Body Text Percentage”.....	43
Ilustración 3.6 – Ejemplo II para la medida “Body Text Percentage”.....	43
Ilustración 4.1 - Página principal de Atenea 3.0.....	48
Diagrama 5.5.1 – Diagrama de Gantt: planificación del proyecto.....	50
Tabla 5.1 – Resumen del presupuesto.....	51
Tabla 6.1 - Requisitos funcionales.....	¡Error! Marcador no definido.
Tabla 6.2 - Requisitos no funcionales.....	¡Error! Marcador no definido.
Diagrama 6.6.1 – Diagrama de casos de uso.....	55
Tabla 6.3 – Definición del caso de uso 1.1.....	55
Diagrama 6.2 - Diagrama de clases del análisis.....	57
Tabla 6.4 – Descripción de la clase MetricIvory.....	58
Tabla 6.5 – Descripción de la tabla Checkpoint109.....	58
Tabla 6.6 – Descripción de la clase Checkpoint110.....	59
Tabla 6.7 – Descripción de la clase Checkpoint111.....	59
Tabla 6.8 – Descripción de la clase Checkpoint112.....	59
Tabla 6.9 – Descripción de la clase Checkpoint113.....	59
Diagrama 6.3 – Caso de uso 1.1: diagrama de robustez.....	60
Tabla 6.10 – Caso de uso 1.1: explicación del diagrama de robustez.....	60
Diagrama 7.1 – Relaciones entre los paquetes.....	63
Diagrama 7.2 – Diagrama del paquete metrics.....	64
Diagrama 7.3 – Diagrama del paquete checkpoints.....	64
Diagrama 7.4 - Diagrama del paquete test.....	65
Diagrama 7.5 – Diagrama de componentes.....	66
Diagrama 7.6 – Diagrama de despliegue.....	67
Diagrama 7.7 – Diagrama de clases del paquete IVORY.metrics.....	68
Diagrama 7.8 – Diagrama de clases del paquete IVORY.checkpoints.....	69
Diagrama 7.9 - Diagrama de clases del paquete IVORY.util.....	70
Diagrama 7.10 - Diagrama de clases del paquete IVORY.test.....	70
Diagrama 7.11 – Diagrama de Secuencia.....	71
Tabla 7.1 – Test unitarios de la clase MetricIVORY.....	73
Tabla 7.2 – Test unitarios de la clase Checkpoint109.....	74
Tabla 7.3 – Test unitarios de la clase Checkpoint110.....	74
Tabla 7.4 – Test unitarios de la clase Checkpoint111.....	74
Tabla 7.5 – Test unitarios de la clase Checkpoint112.....	75
Tabla 7.6 – Test unitarios para la clase Checkpoint113.....	75
Tabla 7.7 – Pruebas de integración.....	76

Tabla 8.1 – Descripción detallada de la clase MetricIvory	81
Tabla 8.2 – Descripción detallada de la clase Checkpoint109	82
Tabla 8.3 – Descripción detallada de la clase Checkpoint110	82
Tabla 8.4 – Descripción detallada de la clase Checkpoint111	83
Tabla 8.5 – Descripción detallada de la clase Checkpoint112	83
Tabla 8.6 – Descripción detallada de la clase Checkpoint113	83
Tabla 8.7 – Descripción detallada de la clase MetricModel	84
Tabla 8.8 – Descripción detallada de CheckpointsNamesEnum	85
Tabla 8.9 – Descripción detallada de CssPropertiesEnum	85
Tabla 8.10 – Descripción detallada de HtmlTagsEnum	86
Tabla 8.11 – Descripción detallada de MeasuresEnum	86
Tabla 8.12 – Descripción detallada de la clase Checkpoint109Test	87
Tabla 8.13 – Descripción detallada de la clase Checkpoint110Test	87
Tabla 8.14 – Descripción detallada de la clase Checkpoint111Test	88
Tabla 8.15 – Descripción detallada de la clase Checkpoint112Test	88
Tabla 8.16 – Descripción detallada de la clase Checkpoint113Test	89
Tabla 8.17 – Descripción detallada de la clase MetricIVORYTest	89
Tabla 8.18 – Descripción detallada de la clase CheckpointModel	90
Ilustración 9.1 – Test unitarios de la clase MetricIVORY	91
Ilustración 9.2 – Test unitarios de la clase Checkpoint109	91
Tabla 9.1 – Test unitarios de la clase Checkpoint110: comprobaciones en caso de fallo	92
Ilustración 9.3 – Test unitarios de la clase Checkpoint110	92
Ilustración 9.4 – Test unitarios de la clase Checkpoint111	93
Ilustración 9.5 – Test unitarios de la clase Checkpoint112	93
Tabla 9.2 – Test unitarios de la clase Checkpoint113: comprobaciones en caso de fallo	93
Ilustración 9.6 – Test unitarios de la clase Checkpoint113	94
Tabla 9.3 – Resultado de las pruebas de integración	94
Tabla 10.1 – Niveles de usabilidad	95
Tabla 10.2 – Resultados de la evaluación de la usabilidad con la métrica Ivory en páginas de ayuntamientos	96
Gráfico 10.1 - Evaluación de usabilidad en ayuntamientos: nube de puntos	97
Gráfico 10.2 – Páginas web de ayuntamientos clasificadas por nivel de usabilidad	97
Tabla 10.3 – Resultados de la evaluación de la usabilidad con la métrica Ivory en páginas de universidades	99
Gráfico 10.3 – Evaluación de la usabilidad en universidades: nube de puntos	100
Gráfico 10.4 . Páginas web de universidades clasificadas por nivel de usabilidad	100
Tabla 10.4 - Resultados de la evaluación de la usabilidad con la métrica Ivory en páginas de empresas	101
Gráfico 10.5 – Evaluación de la usabilidad en empresas: nube de puntos	102
Gráfico 10.6 – Páginas web de empresas clasificadas por su nivel de usabilidad	102
Gráfico 10.7 – Comparación de niveles de usabilidad en páginas web de ayuntamientos, empresas y universidades	103
Gráfico 10.8 – Comparación de la puntuación media de usabilidad en ayuntamientos, universidades y empresas	104
Gráfico 10.9 – Clasificación por nivel de usabilidad del total de las páginas analizadas de ayuntamientos, universidades y empresas	104
Tabla 10.5 – Evaluación de la accesibilidad con la métrica WAB y evaluación de la usabilidad con la métrica de Ivory en webs de ayuntamientos de España	106
Tabla 10.6 - Evaluación de la accesibilidad con la métrica WAB y evaluación de la usabilidad con la métrica de Ivory en webs de universidades de España	108

Tabla 10.7 - Evaluación de la accesibilidad con la métrica WAB y evaluación de la usabilidad con la métrica de Ivory en webs de empresas de España	108
Ilustración 10.9 – Cálculo del coeficiente de correlación de Pearson con el paquete estadístico SPSS	109
Ilustración 10.10 – Cálculo de la Rho de Spearman con el paquete estadístico SPSS	110
Ilustración 11.1 – Instalación de AppServ: paso 1.....	112
Ilustración 11.2 – Instalación de AppServ: paso2.	112
Ilustración 11.3 – Instalación de AppServ: paso 3.....	113
Ilustración 11.4 – Instalación de AppServ: paso 4.....	113
Ilustración 11.5 – Instalación de AppServ: paso 5.....	114
Ilustración 11.6 – Instalación de AppServ: paso 6.....	115
Ilustración 11.7 – Instalación de AppServ: paso 7.....	115
Ilustración 11.8 – Instalación de AppServ: paso 8.....	116
Ilustración 11.9 – Instalación de AppServ: paso 9.....	116
Ilustración 11.10 – Instalación de AppServ: paso 10.....	117
Ilustración 11.11 – Instalación de Tomcat: página oficial.....	117
Ilustración 11.12 – Instalación de Tomcat: selección de la distribución.	118
Ilustración 11.13 – Instalación de Tomcat: paso 1.	118
Ilustración 11.14 – Instalación de Tomcat: paso 2.	119
Ilustración 11.15 – Instalación de Tomcat: paso 3.	119
Ilustración 11.16 – Instalación de Tomcat: paso 4.	120
Ilustración 11.17 – Instalación de Tomcat: paso 5.	120
Ilustración 11.18 – Instalación de Tomcat: paso 6	121
Ilustración 11.19 – Instalación de Tomcat: paso 7	121
Ilustración 11.20 – Instalación de Tomcat: paso 8.	122
Ilustración 11.21 – Instalación de Tomcat: paso 9.	122
Ilustración 11.22 – Instalación de AppServ: paso 1.....	123
Ilustración 11.23 – Creación de la base de datos: paso 2.....	123
Ilustración 11.24 – Creación de la base de datos: paso 3.....	124
Ilustración 11.25 – Creación de la base de datos: paso 4.....	124
Ilustración 11.26 – Creación de la base de datos: paso 5.....	125
Ilustración 11.27 – Creación de la base de datos: paso 6.....	125
Ilustración 11.28 – Creación del almacén de claves: paso 3.....	126
Ilustración 11.29 – Creación del almacén de claves: paso 4.....	127
Ilustración 11.30 – Configuración de Tomcat y despliegue de la aplicación web: paso 6.	128
Ilustración 11.31 – Página de inicio de Atenea 3.0.....	130
Ilustración 11.32 – Realizar análisis desde la parte pública: introducir datos	131
Ilustración 11.33 – Realizar análisis desde la parte pública: análisis en curso.....	131
Ilustración 11.34 – Realizar análisis desde la parte pública: resultado final.....	132
Ilustración 11.35 – Realizar análisis desde la parte privada: acceso a la parte privada	133
Ilustración 11.36 – Realizar análisis desde la parte privada: inicio de sesión	133
Ilustración 11.37 – Realizar análisis desde la parte privada: pantalla de bienvenida	134
Ilustración 11.38 – Realizar análisis desde la parte privada: crear proyecto nuevo.	135
Ilustración 11.39 – Realizar análisis desde la parte privada: configurar crawler.	135
Ilustración 11.40 – Realizar análisis desde la parte privada: proyecto creado correctamente	136
Ilustración 11.41 – Realizar análisis desde la parte privada: analizar una única página web.....	137
Ilustración 11.42 – Realizar análisis desde la parte privada: analizar un conjunto de páginas web.....	137
Ilustración 11.43 – Realizar análisis desde la parte privada: estado del análisis en curso	138
Ilustración 11.44 – Realizar análisis desde la parte privada: visualizar resultados	139
Tabla 13.1 – Presupuesto de costes.....	145
Tabla 13.2 – Cálculo de costes	146
Tabla 13.3 – Presupuesto del cliente	147

Tabla 15.1 – Estructura general de directorios del cd..... 153
Tabla 15.2 – Estructura de directorios de desarrollo 154

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

La usabilidad es uno de los factores más importantes a tener en cuenta a la hora de construir una web. Existen numerosos estudios sobre la usabilidad, tanto en la web como en aplicaciones de escritorio, que ofrecen pautas y guías a seguir por los desarrolladores, diseñadores y arquitectos para conseguir sitios web usables. Sin embargo, dado que en la mayoría de los casos, los métodos de evaluación de usabilidad propuestos incluyen la evaluación por parte de evaluadores expertos y/o usuarios potenciales del sitio web, se debe dedicar una parte importante de tiempo en los proyectos a realizar pruebas de usabilidad para conseguir determinar si el producto desarrollado es usable o no, y en qué medida. Es por este motivo por el que se han propuesto algunos modelos de evaluación que únicamente consideran atributos de usabilidad que pueden ser medidos automáticamente y que además proporcionan un valor cuantitativo que permite estimar el nivel de usabilidad alcanzado en el sitio evaluado.

Una de estas propuestas es la desarrollada por Melody Ivory en su tesis doctoral “An Empirical Foundation for Automated Web Interface Evaluation” en la Universidad de California en 2001. El objetivo principal, por tanto, de este Trabajo Fin de Máster es automatizar la medición de la usabilidad mediante la métrica propuesta por Ivory e integrarla en la herramienta Atenea 3.0, desarrollada por Javier Oyón en su Trabajo Fin de Máster en Ingeniería Web de la Universidad de Oviedo titulado “Atenea 3.0: Reingeniería del sistema para la evaluación cuantitativa de sitios web” en 2012.

Para completar el trabajo, se aplicará la métrica en diferentes estudios de usabilidad.

1.2 Resumen de Todos los Aspectos

Este documento comienza con una introducción tras la cual se divide en dos partes que, a su vez, están divididas en varios capítulos. A continuación se resume lo que incluye cada una de ellas.

- **Introducción.** Incluye la justificación del proyecto y un estudio de la situación actual en el que se habla de las alternativas existentes para evaluar la usabilidad de sitios web de manera automática.
- **Parte I: Investigación sobre la propuesta de Ivory para la evaluación cuantitativa de la usabilidad en sitios web.** Esta parte incluye el capítulo destinado a explicar la propuesta desarrollada por Ivory para evaluar la usabilidad de manera cuantitativa en las páginas web.
- **Parte II: Desarrollo de la automatización de la medición de la usabilidad mediante la métrica de Ivory.** Esta parte incluye los capítulos relativos al desarrollo de la aplicación así como los referentes al estudio empírico realizado. Dichos capítulos son los siguientes:
 - **Capítulo 4. Aspectos teóricos.** Se presenta qué es el concepto de usabilidad y su importancia en las aplicaciones haciendo énfasis en la web y se hace un resumen de las técnicas existentes para la evaluación de la usabilidad. También se explica brevemente qué es la herramienta Atenea 3.0 y las funcionalidades que presenta. Finalmente, se documenta de forma breve la librería JStyleParser utilizada en la implementación de la solución.
 - **Capítulo 5. Planificación y resumen de presupuestos.** Incluye un diagrama de Gantt con la planificación del proyecto y un resumen del presupuesto.
 - **Capítulo 6. Análisis.** En este apartado se define lo que va a hacer el sistema implementado. Incluye la identificación y descripción de los requisitos, casos de uso y subsistemas. También se definen los tipos de pruebas que se van a realizar.
 - **Capítulo 7. Diseño del sistema.** Se explica cómo se va a construir lo que se ha especificado en el análisis. Incluye la definición de la arquitectura del sistema y el diseño detallado de las clases, así como la especificación técnica del plan de pruebas.
 - **Capítulo 8. Implementación del sistema.** En primer lugar se enumeran los estándares y normas seguidos y los lenguajes de programación utilizados para implementar el sistema. A continuación se incluyen las herramientas que se han utilizado durante el desarrollo y se hace un análisis de los problemas encontrados. Finalmente se incluye una descripción detallada de las clases que conforman el sistema desarrollado.
 - **Capítulo 9. Desarrollo de las pruebas.** En este apartado se documentan los resultados obtenidos al llevar a cabo el plan de pruebas planteado en el diseño y las soluciones adoptadas en caso de haber encontrado errores.
 - **Capítulo 10. Estudios empíricos aplicando la métrica de Ivory.** Este capítulo comprende los estudios empíricos realizados para analizar el nivel de usabilidad en España y comprobar si se puede demostrar la existencia de una

relación entre la usabilidad y la accesibilidad con los resultados obtenidos al evaluar la usabilidad de diferentes sitios web con la métrica de Ivory.

- **Capítulo 11. Manuales del Sistema.** Incluye los manuales de instalación y ejecución y los manuales de usuario y del programador.
- **Capítulo 12. Conclusiones y ampliaciones.** En este apartado se encuentran las conclusiones a las que se ha llegado una vez finalizado el proyecto y las posibles ampliaciones que podrían hacerse.
- **Capítulo 13. Presupuesto.** En esta sección se encuentra el presupuesto de gastos y el presupuesto final para el cliente.
- **Capítulo 14. Referencias bibliográficas.** Incluye todas las referencias bibliográficas consultadas durante la realización del proyecto, tanto de artículos de investigación como de páginas web.
- **Capítulo 15. Apéndices.** En este apartado se encuentra una descripción del contenido entregado en el cd, un índice alfabético y el código fuente.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

Tal y como se mencionó anteriormente, la usabilidad de una aplicación es un aspecto importante a tener en cuenta a la hora de diseñar una aplicación web. Así, el diseñador debe prestar atención a aspectos como la calidad de los contenidos, la buena distribución de la información o la calidad del servicio prestado. Un buen diseño debe ser comprensible, fácil de usar, claro, intuitivo y de fácil aprendizaje para el usuario. Podemos tener una aplicación con una lógica muy compleja y que realice operaciones complejas, sin embargo, si al usuario no le resulta cómoda y sencilla de manejar y los procesos que tiene que aprender a realizar no son intuitivos, probablemente quedará relegada a un segundo plano y se convertirá más en un estorbo que en una ayuda para realizar un trabajo. De la misma forma, si tenemos una web centrada en la información que resulte difícil de leer, la información no esté debidamente estructurada o tenga una navegación compleja, el usuario no podrá encontrar lo que busca y dejará de visitarla.

En la actualidad, existen numerosas herramientas, tanto gratuitas como de pago, para ayudar a evaluar la usabilidad de un diseño web. Estas herramientas permiten diseñar test de usuario, enviar test a un número de usuarios de la comunidad para que los realicen y después poder estudiar los resultados obtenidos y elaborar mapas de calor de las páginas web (imágenes que muestran las zonas de la página en las que el usuario ha hecho más clics o en las que ha estado el cursor durante más tiempo). Si bien estas herramientas suponen una gran ayuda al diseñador y permiten reducir el tiempo y coste de realizar test de usuario, han surgido algunas propuestas que tratan de acortar estos tiempos proponiendo modelos de evaluación automáticos.

Conociendo los estudios realizados por M. Ivory sobre la medición cuantitativa automática de la usabilidad web surge la idea de integrar la fórmula propuesta por esta investigadora en la herramienta Atenea 3.0 desarrollada en la Universidad de Oviedo. Esta herramienta, permite medir la accesibilidad y usabilidad de las páginas web mediante el uso de diferentes métricas.

Por lo tanto, los objetivos del proyecto se concretan en los siguientes:

- **Estudio y análisis previo de la propuesta de Ivory para la evaluación automática de la usabilidad en las páginas web:** analizar los estudios realizados por Melody Ivory sobre la evaluación automática de la usabilidad [Ivory01] [IvoryHearst02] [IvoryRashmiHearst00] [IvoryRashmiHearst01].
- **Estudio de la herramienta Atenea 3.0:** estudiar el funcionamiento de Atenea 3.0 y las facilidades que ofrece para integrar nuevas métricas de evaluación de usabilidad.
- **Desarrollo de la métrica e integración de la misma en Atenea 3.0:** desarrollar una implementación de la métrica de Ivory para la evaluación automática de la usabilidad en sitios web que se pueda integrar en la herramienta Atenea3.0.

- **Aplicación de la métrica:** realizar un análisis empírico de la usabilidad en la web en España utilizando la herramienta desarrollada sobre una muestra de páginas web de ayuntamientos, universidades y empresas.

2.2 Estudio de la Situación Actual

Actualmente existe un gran número de aplicaciones web, gratuitas y de pago, que ayudan a los diseñadores a evaluar la usabilidad de sus diseños. Atendiendo a su funcionalidad se puede distinguir entre las que ayudan a realizar test de usuario y las que monitorizan el comportamiento de los usuarios.

Las herramientas para realizar test sobre usuarios reales se basan en la existencia de una comunidad colaborativa. Permiten diseñar diferentes test de usuario que después serán realizados por otros usuarios de la comunidad. La persona que realiza el test puede decidir un determinado perfil de usuario para ser destinatario de sus test (por ejemplo, estableciendo rangos de edad).

Estas herramientas permiten realizar diferentes tipos de test. Por ejemplo, se pueden diseñar test que consisten en mostrar una captura de la página al usuario durante unos segundos y después hacer preguntas sobre lo que ha visto, establecer un conjunto de tareas que el usuario tiene que realizar en la aplicación y después preguntarle sobre su experiencia. Los resultados de los test realizados se muestran al diseñador para que los estudie y decida qué aspectos de la página web se deberían mejorar.

En cuanto a las herramientas que se basan en monitorizar la navegación de los usuarios por la página web, se encargan de guardar valores como tiempos de visualización de cada página, número y zonas de clics del ratón, tiempo de posicionamiento del cursor en determinadas zonas de la pantalla, etc. y finalmente los muestran al diseñador en forma de gráficos o mapas de calor para que él pueda interpretarlos y determinar si su página es usable o debería realizar modificaciones.

Este tipo de herramientas, si bien son de gran ayuda para que el diseñador pueda ver si su página es usable o no, no dan un resultado determinado, sino que se limitan a almacenar y mostrar los resultados y es el propio diseñador el que tiene que deducir a partir de ellos si la página es usable o no.

A continuación se describen algunas de las herramientas existentes en la actualidad para ayudar a evaluar la usabilidad de las páginas web.

2.2.1 Herramientas existentes para evaluar la usabilidad web

2.2.1.1 *Navflow*

Herramienta online colaborativa que permite analizar la navegación de los usuarios por un sitio web. Para utilizar la versión gratuita basta con registrarse y subir los diseños que se quieran analizar. La aplicación creará un test que se publicará para que pueda ser realizado por otros usuarios.

Mediante Navflow se pueden crear test de diferentes tipos, como por ejemplo de imagen o de clics. Los primeros consisten en subir una captura del sitio que se quiere analizar y formular una serie de preguntas sobre él que el usuario que realiza el test responderá una vez que haya visualizado la imagen durante unos cinco segundos. Los test de clics se basan en el modo en el que el usuario navega por el sitio web y los puntos en los que hace clic.

Cuando los test hayan sido realizados por los usuarios, el creador del test podrá visualizar los resultados en gráficos y mapas de calor, lo que le permitirá conocer qué partes del diseño son las que llaman más la atención de los usuarios.

También existen diferentes modalidades de pago de la aplicación que permiten realizar test privados que solo resolverán los usuarios que designe el creador del test y obtener los resultados con mayor rapidez. **[Navflow10]**

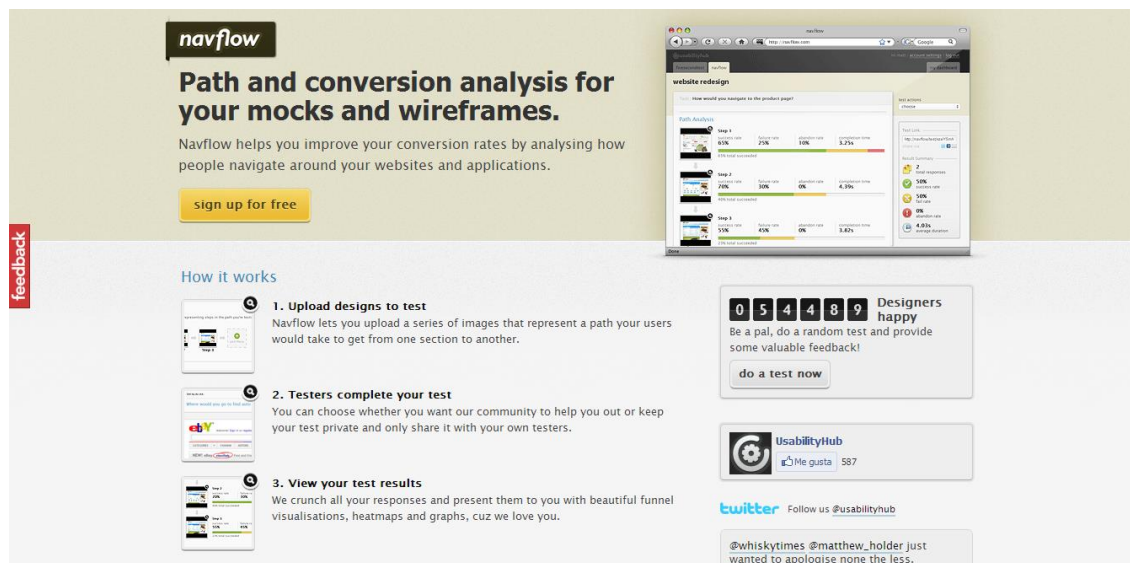


Ilustración 2.1 – Herramienta Navflow

2.2.1.2 UserPlus

Presenta dos herramientas: Tester y Advisor, con diferentes modalidades (gratuitas y de pago) para cada una de ellas.

Tester permite crear un test definiendo diferentes tareas a realizar sobre el sitio web y publicarlo para que otros usuarios (*testers*) puedan hacerlo. La aplicación recoge los clics y la navegación de los *testers* y finalmente presenta los resultados indicando aspectos como el tiempo medio para realizar cada tarea, el número de personas que consiguieron finalizar cada tarea con éxito, el grado de satisfacción del usuario o los lugares donde han hecho clic.

Advisor proporciona un checklist de evaluación de la usabilidad. Tras responder las preguntas del test, se obtiene una puntuación que indica el grado de usabilidad de la web. También muestra los aspectos que se deberían mejorar.

En la web de la herramienta, también se muestran diferentes patrones y buenas prácticas de diseño para orientar a los diseñadores. **[Userplus]**

The screenshot shows the UserPlus website. At the top right, there is a 'Customer login' link. The navigation menu includes 'Our Tools', 'About us', 'News', 'Contact', 'Design Patterns', and 'Best Practices'. The main banner features a computer monitor displaying two stylized human figures and the text 'Tester: Remote test on real users...'. The headline reads 'We help to make your site User Friendly with Tools & Advice'. Three main service areas are highlighted: 'The Advisor Tool' (described as 'A Usability expert at your fingertips!!'), 'The Tester Tool' (described as 'Test your site with real people!'), and 'Services: Let us help you out' (described as 'Need advice or a helping hand?'). Each service area includes a list of bullet points detailing its capabilities. A yellow arrow button labeled 'Check out the Tools Start Measuring usability' is positioned to the right of the main banner.

Ilustración 2.2 – UserPlus

2.2.1.3 ClickHeat

Se trata de una herramienta gratuita y Open Source que se instala en el servidor de la aplicación web que se quiere evaluar y muestra un mapa de calor que indica las zonas exactas de las páginas donde los usuarios han hecho clic. Para tenerlo en funcionamiento solo es necesario un servidor Windows o Linux con Apache y la librería gráfica GD2 para soportar imágenes PNG. [ClickHeat13]

The screenshot shows the ClickHeat website. The top right corner has a 'Login - Sign in' link. The main content area is titled 'ClickHeat | Clicks heatmap'. It includes a description: 'ClickHeat is a visual heatmap of clicks on a HTML page, showing hot and cold click zones. ClickHeat is an OpenSource software, released under GPL licence, and free of charge.' A central image shows a heatmap visualization of a webpage. Below this, there are sections for 'Requirements', 'Features', and 'Demo'. The 'Features' section lists: '- Low logging activity: a very few function calls to log a click, no server load rise should be noticed (have a look at Performance & optimization)', '- A keyword is used to define the page upon Javascript code load, allowing you to group same pages.', and '- Screen sizes and browsers are logged, making possible the tracking of liquid CSS layouts (100% used width)'. The 'Demo' section states: 'A ClickHeat demo is available (with real data from this website's clicks). Sign in with the user demo and its password demo.' On the right side, there is a 'More information:' section with links to 'Frequently Asked Questions', 'Heatmap Class', 'Installation and upgrade of ClickHeat', 'Performance and optimization', and 'Thanks'. Below that, there is a 'Resources' section with links to 'Download Demo (user:demo, pass:demo)', 'Latest news:', 'AIML Bot released', and 'MySQL Database Checker'. On the left side, there is a 'Projects/News' section with links to '3C.CMS', 'AIML Bot', 'BuzzRiver', 'ClickHeat', 'Database Checker', 'News', and 'Wordmap'.

Ilustración 2.3 – ClickHeat

2.2.1.4 ClickDensity

ClickDensity proporciona mapas de calor y mapas de clics de las páginas del sitio web que se está evaluando que muestran los lugares que más atraen a los usuarios y los lugares en los que hacen clic. Para utilizar la herramienta es necesario añadir un fragmento de Javascript en el

Automatización de la medición de la usabilidad web mediante la métrica de Ivory | Introducción

código de la aplicación web que se quiere evaluar. Este código se encargará de almacenar la navegación del usuario en el servidor de ClickDensity sin que éste lo perciba.

Finalmente, el usuario podrá ver los resultados del estudio a través de la herramienta.

Existe una versión gratuita y diferentes versiones de pago. **[ClickDensity11]**

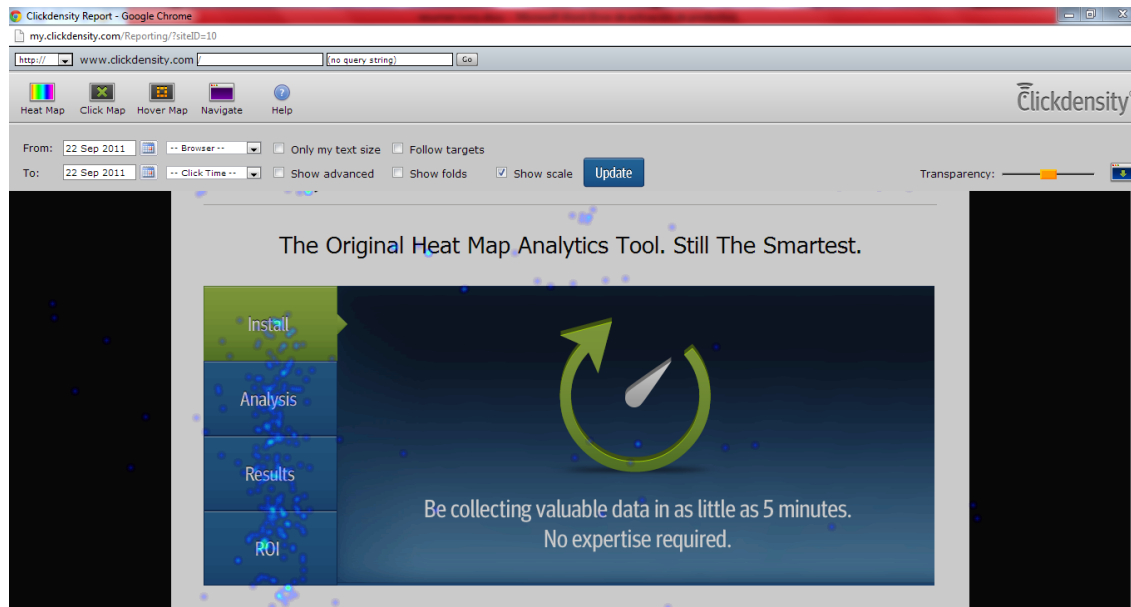


Ilustración 2.4 –Mapa de calor elaborado por ClickDensity

2.2.1.5 CrazyEgg

Herramienta web que guarda la navegación de los usuarios en la página a evaluar y ofrece los resultados en cuatro formatos diferentes:

- HeatMap: mapa de calor que muestra las zonas de la página en las que los usuarios han hecho clic.
- ScrollMap: imagen que muestra hasta donde han llegado los usuarios haciendo scroll y ayuda a determinar dónde han abandonado la página.
- Overlay: muestra el número de clics realizados sobre cada elemento de la página.
- Confetti: muestra los clics distinguiéndolos por diferentes aspectos como el sitio desde el que llegó el usuario, país de origen, sistema operativo, navegador, etc.

Ofrece una versión de prueba gratuita. **[CrEgg12]**



Ilustración 2.5 – Mapa de calor elaborado por Crazy Egg

2.2.1.6 ClickTale

Esta herramienta incluye muchas funcionalidades entre las que destacan las siguientes:

- Permite grabar y reproducir la experiencia íntegra de los usuarios en la página.
- Proporciona información en tiempo real sobre los usuarios que están visitando la página (desde dónde han llegado, las páginas de la aplicación que han visitado, qué están viendo en ese momento...).
- Realiza un análisis demográfico de los usuarios que visitan la página.

También proporciona mapas de calor, de clics y de scroll similares a los de CrazyEgg y ClickDensity. [CITale13]

2.2.1.7 Chalkmark

Permite realizar un test sobre un sitio web estableciendo una tarea que el usuario debe completar. La aplicación va guiando al usuario para informarle de lo que debe hacer y guarda su navegación.

Los resultados de los test se muestran en forma de mapas de calor que muestran los lugares en los que los usuarios han hecho clic para completar la tarea. [CIMark12]

2.2.2 Evaluación de Alternativas

El objetivo principal del proyecto es crear una herramienta que implemente la métrica Ivory para evaluar la usabilidad de los sitios web. Se busca una herramienta fácil de utilizar, cómoda, rápida y sencilla, por lo que una aplicación web es la mejor opción ya que no requiere ninguna instalación por parte del usuario final.

Dado que ya existe una herramienta web desarrollada por varios alumnos de la Universidad de Oviedo que permite evaluar la accesibilidad de sitios web mediante diferentes métricas (Atenea) se ha decidido integrar la implementación de la métrica Ivory en dicha herramienta en lugar de crear una nueva aplicación desde cero. Esta elección supone varias ventajas como el ahorro de código (ya que por ejemplo no habría que implementar partes de la aplicación como la interfaz gráfica) y la rapidez en el desarrollo, implantación y puesta en marcha de la métrica, ya que la herramienta permite añadir nuevas métricas de forma sencilla.

Para implementar la métrica de Ivory es necesario analizar el código de los documentos HTML evaluando diferentes aspectos que se definen, o bien en el propio HTML, o bien mediante hojas de estilo (CSS). Para este caso, se barajó la posibilidad de implementar un analizador propio de CSS o de utilizar alguno de los ya disponibles gratuitamente implementados en Java. Entre los estudiados se encuentran:

- **CSSParser**: librería que recibe una hoja de estilos CSS y devuelve un árbol DOM que la representa. Existe una versión gratuita y otra versión de pago que presenta más funcionalidades. **[CSSParser04]**
- **SimpleCSSParser**: herramienta que recibe una hoja de estilos y crea una representación visual de la misma en forma de árbol. **[SCSSParser11]**
- **JStyleParser**: librería que analiza los estilos de un documento HTML representado mediante un árbol DOM y los mapea en dicho árbol relacionando cada elemento del HTML con los estilos que le afectan. **[JStyleP13]**

De estas herramientas disponibles gratuitamente SimpleCSSParser se ha descartado porque está pensada para mostrar el documento CSS de forma visual e interactuar con el árbol manualmente y no como herramienta para realizar consultas concretas sobre los estilos utilizados en determinados elementos del HTML, que es lo que nos interesa. Entre CSSParser y JStyleParser la segunda supone una opción mejor dado que tiene en cuenta tanto los estilos descritos en un documento CSS como los que se especifican entre el código del propio documento HTML. Además, ofrece la posibilidad de relacionar cada estilo utilizado con los elementos concretos del documento sobre los que se aplica, lo que será de gran utilidad para realizar el análisis necesario para implementar la fórmula Ivory.

Dado que la librería JStyleParser se puede utilizar gratuitamente y proporciona todas las funcionalidades que se necesitan para implementar la métrica se ha escogido utilizar esta librería en lugar de desarrollar una propia.

Parte I. Investigación sobre la propuesta de Ivory para la evaluación cuantitativa de la usabilidad en sitios web

Capítulo 3. Métrica de Ivory

3.1 Introducción y objetivos

En su investigación sobre la medición de la usabilidad [Ivory01] [IvoryHearst02] [IvoryRashmiHearst00] [IvoryRashmiHearst01], Ivory pretende establecer modelos estadísticos que definan los perfiles de páginas web buenas y malas desde el punto de vista de la usabilidad. Para ello comienza por realizar un estudio profundo de las diferentes pautas y guías de usabilidad existentes incluyendo el trabajo de Nielsen, Spool, Roosenfield o Morville y a partir de ellas establece una lista con los aspectos de una página web que resultan determinantes para que una sea usable. Entre estos aspectos se encuentran, por ejemplo, el número de enlaces, imágenes y palabras que tenga la página, el número de bloques de texto y sus cambios de posición, el número de colores utilizados y el uso de los mismos, el formato de los encabezados y la agrupación de los enlaces y los elementos interactivos. Con todo esto, establece las métricas que va a utilizar para evaluar las páginas y desarrolla un crawler y una herramienta para efectuar las medidas.

Para poder establecer con qué medidas se identifican las páginas buenas, Ivory efectúa las medidas sobre el conjunto de páginas web evaluadas para los Webby Awards en el año 2000 y hace una comparación de sus resultados con las puntuaciones que recibieron las páginas en los premios.

Los Webby Awards son los premios que da cada año la International Academy of Digital Arts and Sciences a las mejores páginas web. Los organizadores de los premios distribuyen las páginas web en 27 categorías incluyendo noticias, webs personales, finanzas, deportes, servicios, moda y viajes. Un grupo de jueces evalúa las webs y selecciona a las finalistas. Los jueces son profesionales de Internet y están familiarizados con la categoría de la que son revisores. A cada uno se le da un conjunto de sitios a revisar y se les prohíbe revisar cualquier sitio con el que tengan relación personal o profesional. Además, la academia inspecciona el trabajo de cada juez.

Los sitios se valoran de acuerdo a seis criterios: contenido, estructura y navegación, diseño visual, funcionalidad, interactividad y experiencia global. Los resultados de este estudio se detallan en el siguiente apartado.

3.2 Aspectos examinados y medidas a tomar

Tras los estudios llevados a cabo, Ivory considera que los elementos de una página web que deben ser considerados para determinar su nivel de usabilidad son los **elementos de texto**, los **enlaces**, las **imágenes**, su **formato**, el formato y **rendimiento** de las propias páginas y la **arquitectura** del sitio. Además, como se puede observar en la figura, considera que los elementos de texto, los enlaces y las imágenes son los pilares de la interfaz web y el resto de aspectos se fundamentan en ellos. Así, el siguiente nivel lo ocupa el formato de estos elementos y a continuación el formato a nivel de página. Los niveles más altos son para el rendimiento y la arquitectura del sitio.



Figura 3.1 - Elementos que componen el diseño de una página web según Ivory.

En resumen, los tres niveles más bajos se asocian con la información, navegación y actividades de diseño gráfico, mientras que los dos niveles más altos se asocian con la experiencia de navegación.

En los siguientes apartados se describirán los aspectos que Ivory ha estudiado de cada uno de los elementos mencionados para llegar a establecer una métrica de usabilidad.

3.2.1 Texto

Este aspecto trata de determinar cuánto texto hay en la página, qué tipo de texto contiene y lo bueno que es o cuál es su complejidad. Los elementos estudiados son los siguientes:

- **Número de palabras.** Se cuenta el número total de palabras de la página. Solo se tienen en cuenta las palabras en HTML estático; las palabras que aparecen en imágenes, applets u otros objetos se ignoran.
- **Número de palabras del title.**
- **Número de palabras en los meta-tags description y keywords.**
- **Porcentaje de texto que pertenece al cuerpo de la página.** Para calcularlo es necesario cuantificar la cantidad total de texto existente en la página (contador de palabras) y el número de palabras que pertenecen a encabezados.
- **Número de palabras de los enlaces.** Se cuenta el número de palabras totales en los enlaces y el número de palabras totales de los enlaces que son, a su vez, cabecera.
- **Longitud de los enlaces.** Mide el número de palabras que hay que en cada enlace.
- **Porcentaje de la página dedicado al contenido.** Para medirlo se utilizan técnicas de procesamiento de imagen.
- **Porcentaje de la página dedicado a la navegación.** También se mide con técnicas de procesamiento de imágenes.
- **Uso de signos de exclamación.** Se cuenta el número total de signos de exclamación que se encuentran en el cuerpo, cabeceras y enlaces.
- **Errores tipográficos.** Los errores tipográficos restan credibilidad a las páginas web. Para contar el número de errores tipográficos Ivory utiliza diccionarios, bases de datos lingüísticas, diccionarios de términos médicos, acrónimos, abreviaturas, etc. Sin embargo existen muchas limitaciones para calcular la medida exacta, como por ejemplo el uso de nombres propios y palabras el argot, por lo que es muy difícil obtener un resultado exacto.
- **Complejidad de lectura.** Se refiere a la complejidad de lectura de una página expresada mediante el GFI (Gunnig Fog Index).
- **Calidad de la información.** Se cuenta el número de palabras buenas, palabras en los enlaces, en las cabeceras, en el cuerpo, en el title y en los meta-tags. Se considera que una palabra es buena si no pertenece al grupo de “stop words”. Las stop words son palabras cuyo significado no es importante al realizar una búsqueda. Algunos ejemplos en español serían “el”, “la”, “a”, “en” y “y”. No existe una lista oficial que contenga todas las stop words, sino que en cada estudio se elabora una lista adecuada a las necesidades del mismo. La lista elaborada por Ivory para evaluar la calidad de la información en webs en inglés está formada por 525 palabras.

3.2.2 Enlaces

Este aspecto determina cuántos enlaces hay en la página y de qué tipo son. Las medidas estudiadas son las siguientes:

- **Número de enlaces.** Se cuenta el número total de enlaces (gráficos y de texto) que hay en una página. Mide la anchura de la página, no la profundidad.
- **Número de enlaces de texto.** Se cuenta el número total de enlaces de texto que hay en una página.

- **Número de enlaces gráficos.** Se cuenta el número total de enlaces de imagen que hay en una página.
- **Uso de anclas.** Se cuenta el número de anclas que hay en una página. Un ancla es un enlace que lleva a otro punto de la misma página.
- **Uso de enlaces internos.** Entendemos por enlace interno aquel enlace que apunta a algún punto dentro del mismo dominio. Se cuenta el número total de enlaces internos existentes en la página. De esta forma, si quisiésemos saber el número de enlaces externos, bastaría con calcular la diferencia entre el contador de enlaces y el contador de enlaces internos.
- **Uso de enlaces embebidos.** Son aquellos enlaces que se encuentran dentro de un texto. Para detectarlos se utilizan técnicas de procesamiento de imagen.
- **Uso de enlaces redundantes.** Se cuenta el número de enlaces que existen apuntando a la misma dirección.

3.2.3 Imágenes

Para evaluar los elementos gráficos, Ivory estudió los siguientes aspectos:

- **Número de enlaces de imágenes.**
- **Número de imágenes que son anuncios.** Para saber cuándo una imagen es un anuncio Ivory utiliza diferentes heurísticos.
- **Número de animaciones.**

3.2.4 Formato de texto

Este aspecto trata de determinar de qué manera se enfatiza el texto, si hay texto subrayado que no sea un enlace, qué fuentes se utilizan, de qué tamaño, cuántos colores de texto diferentes hay, cuántas veces se cambia de sitio el texto y cómo se destacan áreas de texto.

Los elementos estudiados son los siguientes:

- **Número de palabras enfatizadas.** Se cuenta el número de palabras en negrita, cursiva o enfatizadas de cualquier otra manera. También se mide por separado el número de palabras en negrita, el número de palabras en cursiva, el número de palabras coloreadas, el número de palabras subrayadas y el porcentaje que representan las palabras enfatizadas frente al total de palabras de la página.
- **Fuentes.** Se cuenta el número de palabras formateadas con serif, sans-serif y otros estilos para determinar cuál es el predominante.
- **Tamaño de fuente.** Se calcula el tamaño mínimo, máximo y medio de fuente utilizada en la página.
- **Colores del texto.** Se cuenta el número de colores diferentes utilizados en el texto del cuerpo de la página y en las cabeceras.
- **Posición del texto.** Se utiliza un contador de posiciones de texto para cuantificar el número de veces que el texto cambia de posición respecto al margen izquierdo de la página. También se cuenta el número de columnas de texto que tiene la página.

- **Agrupación de texto.** Entendemos como agrupación de texto un área de la página que contiene texto y que se resalta de alguna manera, por ejemplo, poniéndole un borde o cambiando el color de fondo. Se cuenta el número de agrupaciones de texto que hay en la página distinguiendo aquellas que son de enlaces.
- **Uso de listas.** Se cuenta el número de listas que hay en la página.

3.2.5 Formato de enlaces

Analiza mediante este aspecto si todos los enlaces del texto están subrayados o qué colores se utilizan para los enlaces.

Los elementos evaluados son los siguientes:

- **Uso de enlaces no subrayados.** Cuenta el número de enlaces de texto que no aparecen subrayados.
- **Uso de colores en los enlaces.** Se cuenta el número de colores diferentes que se utilizan para los enlaces. No se tiene en cuenta el cambio de color entre enlaces visitados y no visitados.

3.2.6 Formato de imágenes

Determina el ancho y alto de las imágenes o qué porcentaje de la página representan.

Los elementos evaluados son los siguientes:

- **Tamaño máximo, mínimo y medio de las imágenes.**
- **Área total cubierta por imágenes.**

3.2.7 Formato de página

Mediante este aspecto se determina de qué forma se utilizan los colores en la página, qué fuentes se utilizan, qué dimensiones tienen las páginas, cuántos elementos interactivos contienen o cómo se controla el estilo.

Los elementos evaluados son los siguientes:

- **Uso del color.** Se cuenta el número de colores diferentes que se utilizan en la página así como el número de veces que se utiliza cada uno de ellos. También se cuenta el número de “colores buenos” que se utilizan, entendiendo como un “color bueno” aquel cuyos valores RGB son divisibles por 51.
- **Combinaciones de colores.** En base a estudios previos realizados que clasifican las combinaciones de colores como buenas, malas y neutras, Ivory desarrolla seis medidas para evaluar las combinaciones de color utilizadas en una página web. Para ello evalúa los colores utilizados en el foreground y background de la página así como los empleados en textos y en imágenes, utilizando para éstas últimas técnicas de procesamiento de imagen.

- **Fuentes.** Como ya se ha comentado anteriormente, se desarrollaron contadores para el número de fuentes diferentes utilizadas en la página y para el número de familias de fuentes diferentes.
- **Uso de elementos interactivos.** Se cuenta el número total de elementos (botones, áreas de texto, combo boxes, checkboxes, etc.) utilizados. También se cuenta el número de elementos que están destinados a la búsqueda de información en la página.
- **Uso de scrolls.** Se cuenta el número de barras de scroll (horizontales y verticales) que son necesarias para visualizar la página completa en una resolución de 800 x 600.
- **Uso de hojas de estilo.** Se cuenta el número de hojas de estilo externas e internas que utiliza la página.

3.2.8 Función de la página

Según su función, Ivory clasifica las páginas web en los siguientes tipos:

- **Páginas de inicio:** página principal del sitio que se muestra al entrar en él. Normalmente su función es proporcionar una vista global del sitio.
- **Páginas de enlaces:** son aquellas cuya función principal es mostrar una o más listas de enlaces.
- **Páginas de contenidos:** su función principal es proporcionar información en forma de texto.
- **Otras páginas:** Cualquier otra página que no pueda ser incluida en las categorías anteriores.

3.2.9 Rendimiento

Se mide cuánto tarda en visualizarse la página, si la página es accesible o si hay errores HTML en la página.

Los elementos evaluados son los siguientes:

- **Tamaño de la página en bytes:** se mide el total de bytes de la página incluyendo el código HTML, las hojas estilos, objetos e imágenes; el total de bytes de las imágenes y la velocidad de descarga de la página.
- **Accesibilidad:** evalúa la accesibilidad de la página utilizando diferentes herramientas online.
- **Errores HTML:** evalúan las páginas utilizando herramientas online.

3.2.10 Arquitectura del sitio

Mediante este aspecto se determina la consistencia de las páginas de un sitio web. Los elementos estudiados son los siguientes:

- **Consistencia:** se cuenta las veces que cambia el title de las páginas, el formato de texto, el formato de las imágenes y el formato de la propia página.
- **Tamaño:** se cuenta el número total de páginas que componen el sitio y establece la máxima profundidad y anchura que tiene el sitio.

3.3 Perfiles y modelos estadísticos obtenidos

Tras tomar las medidas descritas en el punto 3.2 sobre las páginas web evaluadas durante los Webby Awards, Ivory estableció distintos perfiles para las webs mejor y peor puntuadas tanto a nivel de página como a nivel de sitio.

En un primer estudio, Ivory determinó que atendiendo solamente a seis de las medidas realizadas se podía determinar el nivel de usabilidad de una página web con una precisión del 63%. Estas medidas son:

- Contador de agrupaciones de texto
- Contador de enlaces
- Tamaño de la página
- Contador de imágenes
- Contador de colores
- Complejidad de lectura.

Una vez establecidos estos factores determinantes, tras el análisis de los mismos en los estudios llevados a cabo, descartó el factor de complejidad de lectura debido a que para medirla correctamente se necesitan textos de al menos 200 palabras y, muchas páginas web no llegaban a este número.

El análisis realizado sobre los valores obtenidos tuvo los siguientes objetivos:

1. Desarrollar un modelo para clasificar las páginas en buenas, malas y medias.
2. Identificar grupos de páginas con aspectos en común entre las páginas calificadas como buenas.
3. Examinar las relaciones entre las medidas en cada uno de los grupos.

El análisis de los resultados se realizó a nivel de página y de sitio, comparando todas las páginas o dividiéndolas por categoría de contenido (por ejemplo educación, medicina, información, etc.) o tipo de página (página de inicio, página de contenido, formulario, etc.)

Des estos estudios se dedujo una fórmula matemática que puntúa el nivel de usabilidad de una página web y que se describe a continuación.

3.4 Métrica de Ivory

Ivory enunció la siguiente fórmula para evaluar el grado de usabilidad de una página web con una precisión del 63%. Si el resultado es cercano a cero, se trata de una página poco usable y, si es cercano a 1, se trata de una página usable.

$$\begin{aligned} Rank = & 0.129 + 0.002 * Link\ Count - 0.003 * Reading\ Complexity - 0.011 \\ & * Text\ Positioning\ Count + 0.020 * Color\ Count + 8.734 * 10^{-7} \\ & * Page\ Size + 0.002 * Body\ Text\ \% \end{aligned}$$

A continuación se explica en qué significa cada uno de los factores de la ecuación. Para ilustrar cada uno de ellos se utilizan ejemplos extraídos de los estudios de Ivory. [IvoryPpt01]

3.4.1 Contador de enlaces (link count)

Determina el número total de enlaces existentes en una página, teniendo en cuenta tanto enlaces de texto como gráficos. En la siguiente imagen se marcan en rojo los enlaces que cuenta este factor de la métrica.

Link Count

Description
Total number of links on the page.

Computed Value: 34 **Actual Value:** 35
Comments: The blank link at the top is a clear image.

Aspects Assessed
InfoDes NavDes GrapDes ExpDes

Overall Accuracy

Hit	Miss	Avg.
--	--	99.8%

Related Measures

- *Text Link Count
- *Link Graphic Count
- *Page Link Count
- *Internal Link Count
- *Redundant Link Count

42

Ilustración 3.1 – Ejemplo de la medida “link count”.

3.4.2 Complejidad de lectura (Reading complexity)

Para evaluar la complejidad de lectura de una página Ivory utiliza el GFI (Gunning Fox Index). Este coeficiente, indica los años de educación que necesita una persona de habla inglesa para entender un texto escrito en inglés en una primera lectura. Se considera que, para que pueda

ser comprendido por la mayor parte de la gente, un texto debe tener un índice menor que 12 y, para que sea comprendido por casi todos los angloparlantes, el índice debe ser menor que 8.

Para calcular el GFI de un texto se toma un fragmento de, al menos, cien palabras y se cuenta el número de palabras (Fog Word Count), el número de palabras con más de dos sílabas (Fog Big Word Count) y el número de frases (Fog Sentence Count) que contiene. Una vez tomadas estas medidas se obtiene el valor del GFI aplicando la siguiente fórmula:

$$GFI = \left(\frac{\text{Fog Word Count}}{\text{Fog Sentence Count}} + \frac{\text{Fog Big Word Count}}{\text{Fog Word Count}} * 100 \right) * 0.4$$

En la siguiente imagen se muestra un ejemplo marcando con diferentes colores las palabras que se contarían en cada uno de los factores de la fórmula del GFI.

Reading Complexity

Description
Gunning Fog Index (GFI) computed over prose body text. Requires at least 100 fog words and 1 fog sentence to compute.
 $GFI = \left(\frac{\text{Fog Word Count}}{\text{Fog Sentence Count}} + \frac{\text{Fog Big Word Count}}{\text{Fog Word Count}} * 100 \right) * 0.4$

Aspects Assessed
InfoDes NavDes GrapDes ExpDes

Overall Accuracy

Hit	Miss	Avg.
--	--	97.6%

Related Measures
• Overall Reading Complexity
• Fog Word Count
• Fog Big Word Count
• Fog Sentence Count

Computed Value: 10.4 **Actual Value:** 10.4
Comments:

drkoop.com Nutrition Center
Fog Word Count = 100
Fog Big Word Count = 15
Fog Sentence Count = 9

fitguide
Spotting Nutrition Quackery
Additional Stories
Take a BOLD Step...

Ilustración 3.2 – Ejemplo de la medida “Reading complexity”

Tras la consideración inicial de este factor en diversos estudios, Ivory finalmente determinó que se trata de un factor prescindible por dos motivos:

- La precisión de los resultados obtenidos no varía significativamente al excluir de la métrica la complejidad de lectura.
- Para que la medida realizada sea válida, el texto sobre el que se aplica debe contener al menos 200 palabras y existe un gran número de webs que no cumplen este requisito.

3.4.3 Contador de posición de texto (Text positioning count)

El contador de posiciones de texto se utiliza para contabilizar el número de áreas de texto en las cuales éste cambia de posición respecto al margen izquierdo de la página, es decir el número de áreas que tienen el texto centrado.

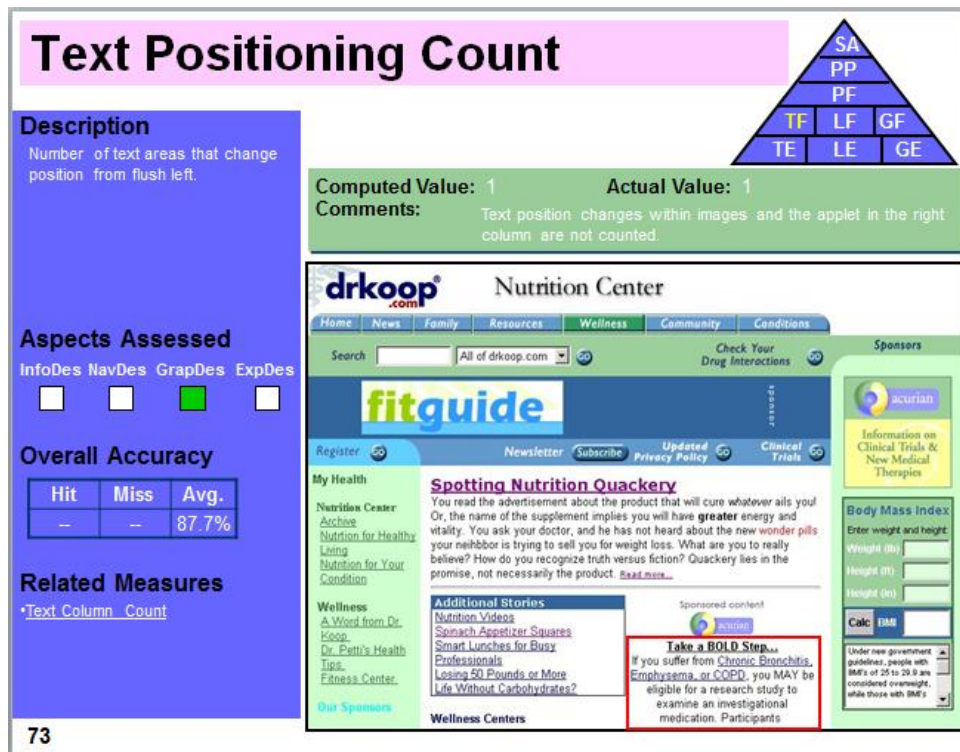


Ilustración 3.3 – Ejemplo de la medida “Text positioning count”

En el ejemplo anterior se muestran marcadas en rojo las áreas de texto que se cuentan con esta medida.

3.4.4 Contador de colores (Color count)

Mide el número de colores diferentes que se utilizan en texto, títulos, enlaces, fondos, etc.

En la siguiente imagen aparecen marcados en rojo los diferentes colores que cuenta la medida.

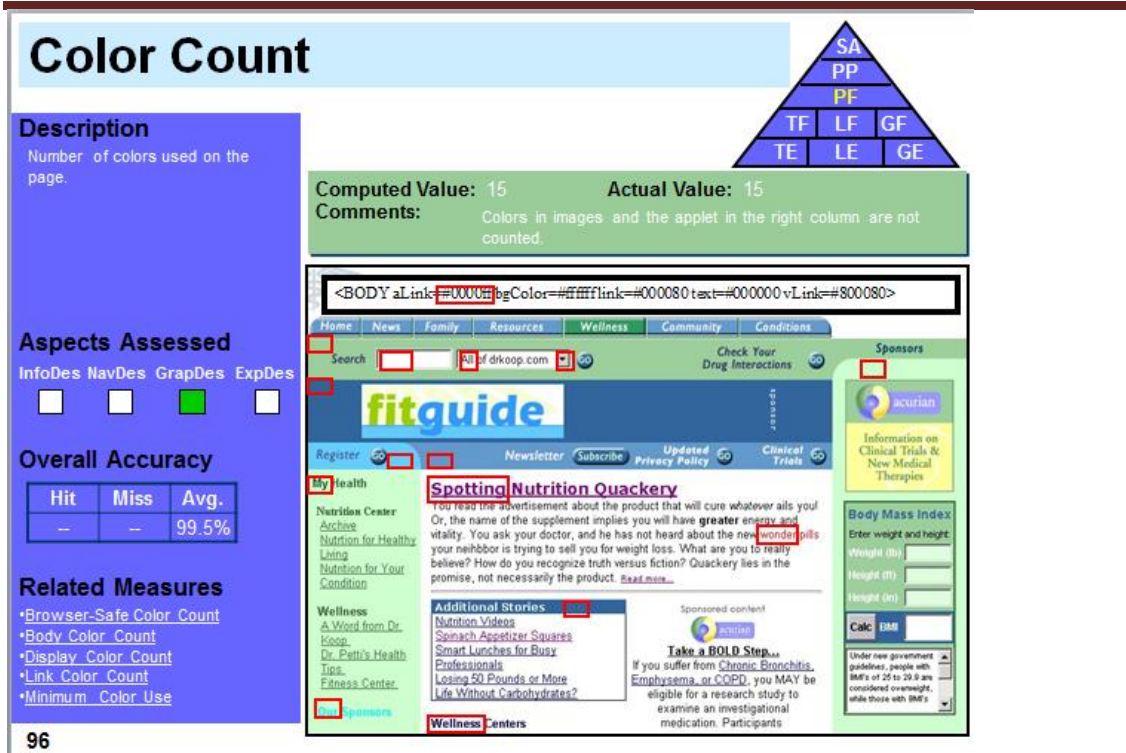


Ilustración 3.4 – Ejemplo de la medida “Color count”

3.4.5 Tamaño de la página (Page size)

Mide el tamaño total de la página en bytes incluyendo las imágenes.

3.4.6 Porcentaje de texto en el cuerpo (Body text percentage)

Representa el porcentaje del texto total de la página que pertenece al cuerpo. Para calcularlo es necesario medir el número de palabras totales que hay en una página y el número de palabras que pertenecen al cuerpo.

En la siguiente imagen se muestran marcadas en rojo las áreas con las palabras pertenecientes al cuerpo de la página.

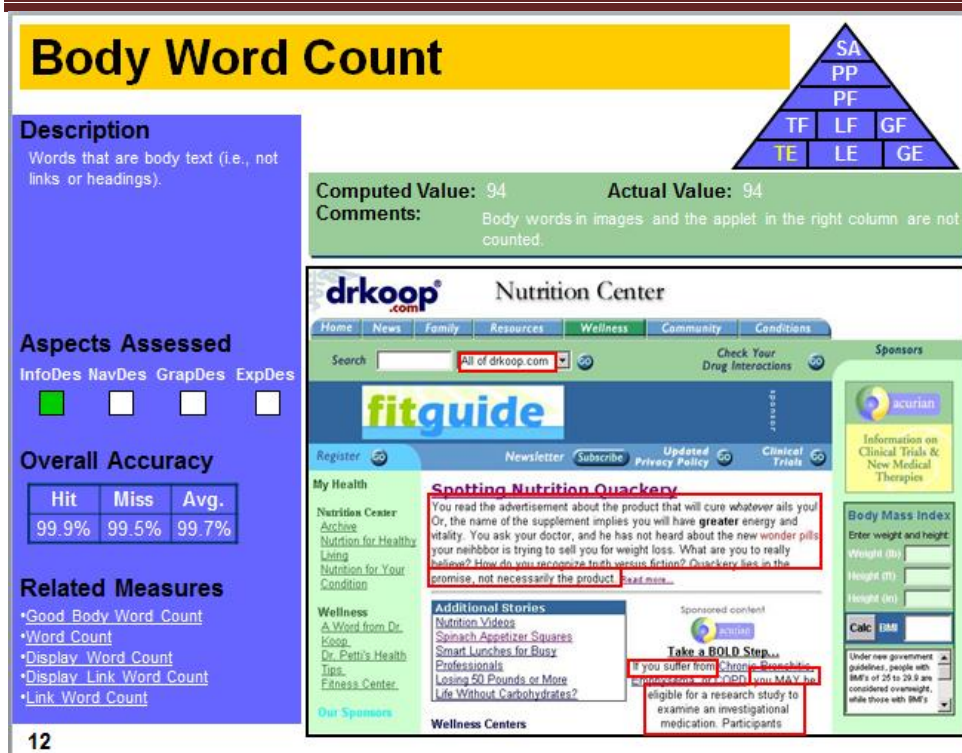


Ilustración 3.5 – Ejemplo I de la medida “Body Text Percentage”

En la siguiente imagen, se muestran marcadas en rojo las áreas que contienen el texto de toda la página.

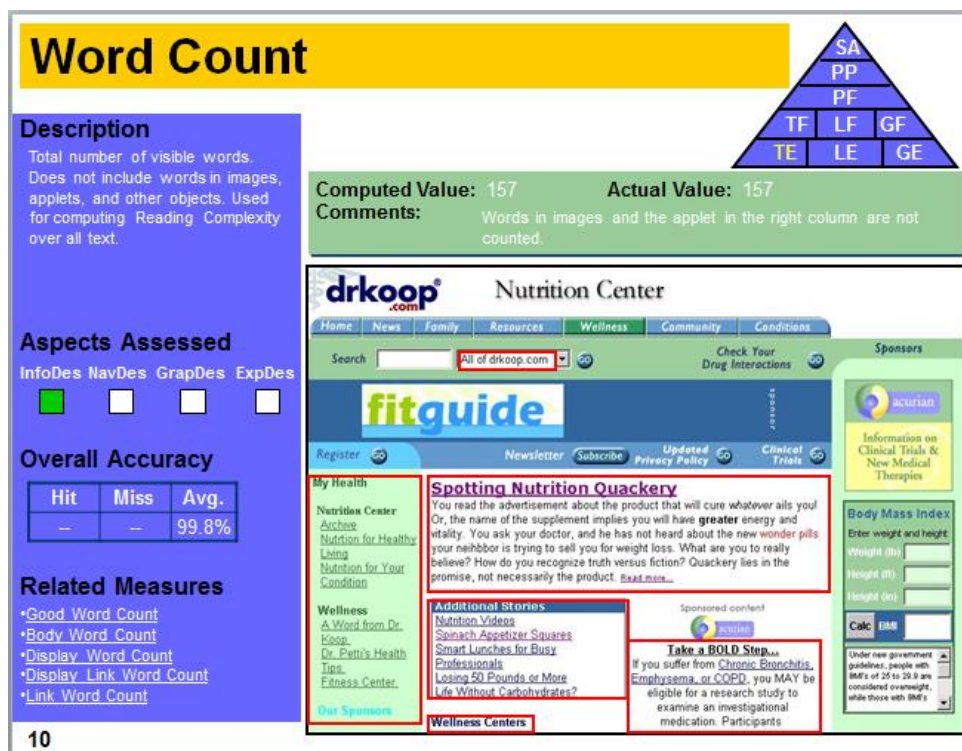


Ilustración 3.6 – Ejemplo II para la medida “Body Text Percentage”

3.5 Conclusión del estudio

Tras el análisis de los artículos en los que Melody Ivory desarrolla su propuesta de medición automática de la usabilidad se concluye que, aunque con una precisión del 63%, resulta factible estimar en base a 6 factores si una página web resulta o no usable. Además, estos 6 factores son factibles de ser automatizados con lo que esta estimación del nivel de usabilidad se puede realizar sin un incremento significativo en el período de desarrollo de un proyecto web. En los siguientes capítulos se describe el proceso seguido para implementar esta métrica que, finalmente enunció de la siguiente forma:

$$\text{Rank} = 0.129 + 0.002 * \text{Link Count} - 0.011 * \text{Text Positioning Count} + 0.020 \\ * \text{Color Count} + 8.734 * 10^{-7} * \text{Page Size} + 0.002 * \text{Body Text \%}$$

Parte II. Desarrollo de la automatización de la medición de la usabilidad mediante la métrica de Ivory

Capítulo 4. Aspectos teóricos

4.1 Usabilidad

Según Jakob Nielsen, la usabilidad es un atributo de calidad que evalúa la facilidad de uso de las interfaces. La palabra “usabilidad” también se refiere a los métodos para mejorar la facilidad de uso durante el proceso de diseño [Nielsen12]. Según este autor, hay cinco aspectos que definen la usabilidad:

- Facilidad con la que los usuarios realizan tareas básicas la primera vez que se encuentran con el diseño (learnability).
- Rapidez con la que los usuarios llevan a cabo las tareas una vez que conocen el diseño (efficiency).
- Facilidad de los usuarios para volver a dominar un diseño al usarlo tras un largo periodo de tiempo sin tener contacto con él (memorability).
- Número de errores que cometen los usuarios, gravedad de los mismos y facilidad para recuperarse después (errors).
- Utilizar el diseño resulta agradable (satisfaction).

Cuando un usuario navega por la red buscando información, si llega a una página que es difícil de utilizar, tiene una mala estructura, no encuentra lo que le interesa rápidamente, se produce algún tipo de error o tarda mucho tiempo en cargar simplemente el usuario la abandona y busca otra alternativa. Dada la inmensa cantidad de páginas web existentes actualmente encontrar una página que ofrezca lo que busca, resulta rápido y sencillo.

Lo mismo ocurre en el caso de las tiendas online. Si un usuario desea comprar un artículo que ofertamos en nuestra tienda pero no es capaz de encontrarlo en nuestra página o el proceso de compra le resulta muy complicado, simplemente buscará otra tienda online en el que lo encuentre fácilmente. Cientos de miles usuarios de nuestra tienda online tendrán el mismo problema con lo que se perderá un gran número de clientes potenciales, pues probablemente la siguiente vez que deseen comprar un artículo ni siquiera se planteen entrar en nuestra tienda.

En el caso de aplicaciones web corporativas que se utilizan para gestionar distintos procesos de negocio, una aplicación poco usable puede suponer grandes pérdidas económicas para la empresa pues el tiempo de realización de las tareas por parte de los empleados puede aumentar significativamente.

Por estos motivos, la usabilidad supone un factor muy importante en el diseño de las aplicaciones web pues una página web con muchos contenidos, o con grandes funcionalidades pero poco usable, estará abocada al fracaso.

4.1.1 Métodos de evaluación de la usabilidad

Un método de evaluación de la usabilidad es un procedimiento sistemático que tiene como función recoger datos relacionados con la interacción del usuario final con una aplicación. Los datos recogidos se analizan y evalúan para determinar la usabilidad de la aplicación.

En base a cómo se recogen y analizan los datos, podemos establecer una primera clasificación de los métodos de evaluación de la usabilidad en manuales y automáticos. Los métodos manuales son aquellos en los que los datos son analizados por personas; en los automáticos, se utilizan procedimientos computarizados para realizar la evaluación de la usabilidad.

Los métodos de evaluación de la usabilidad pueden ser clasificados en base a otros muchos criterios, como por ejemplo el grado de implicación del usuario o el objetivo de la evaluación. En la actualidad no existe un acuerdo para clasificar los métodos de evaluación de la usabilidad, sino que existen varias clasificaciones propuestas por diferentes autores como Nielsen y Molich, Wixon y Wilson o Baecker.

Los métodos más considerados son los de inspección, encuesta y evaluación, que se pasarán a explicar con más detalle a continuación.

4.1.1.1 Encuesta (*inquiry*)

El objetivo de estos métodos es obtener información subjetiva sobre diferentes aspectos de la interfaz de usuario. Para ello se utilizan encuestas, cuestionarios y entrevistas. Estos métodos se pueden llevar a cabo durante diseño de la interfaz de usuario, pero lo más común es que se realicen una vez que se ha liberado la primera versión del sistema con el fin de incluir mejoras en la interfaz en las siguientes versiones.

Hay diferentes tipos de técnicas según cómo el evaluador interactúe con los entrevistados, por ejemplo, se pueden realizar reuniones para compartir las experiencias verbalmente, realizar entrevistas escritas o utilizar los logs de la aplicación para averiguar información. También se pueden utilizar herramientas para monitorizar las acciones del usuario y almacenar los datos que resulten interesantes.

4.1.1.2 Inspección (*inspection*)

Una inspección consiste en que un evaluador experto examine los aspectos relativos a la usabilidad de una interfaz de usuario conforme a un grupo de pautas. Estas pautas pueden ser tanto líneas generales como aspectos muy específicos. Los métodos de inspección más comunes son la evaluación heurística y el seguimiento cognitivo.

Durante una evaluación heurística, un equipo de expertos en usabilidad analiza la interfaz de usuario de una aplicación y la evalúan contra una lista de principios heurísticos mayoritariamente aceptados. Una de las listas de heurísticos más conocidas y aceptadas es la propuesta por Nielsen **[Nielsen95]**.

Realizar un seguimiento cognitivo implica que varios evaluadores expertos creen una serie de escenarios de tareas y las realicen asumiendo el rol de usuarios finales. El fin es detectar pasos

que resulten complicados o que supongan algún tipo de problema para que el usuario llegue a terminar una tarea.

4.1.1.3 Evaluación (testing)

La realización de test con usuarios reales es uno de los métodos más importantes de evaluación de la usabilidad. Consiste en elegir un grupo de usuarios para que utilicen el sistema, o un prototipo del mismo, para completar un conjunto de tareas predeterminadas. El evaluador registrará los resultados del trabajo de los usuarios y los utilizará para determinar el grado de usabilidad del software evaluado basándose en aspectos como el número de tareas que se han completado correctamente, el número de errores cometidos o el tiempo de realización de una tarea.

Estos métodos pueden ser automatizados utilizando un software específico para grabar las acciones de los usuarios o para analizar los resultados obtenidos en base a una métrica o a un modelo.

4.2 Atenea 3.0

Atenea es una herramienta desarrollada por Javier Oyón Fernández en su Trabajo Fin de Máster titulado “Atenea 3.0: Reingeniería del Sistema para la evaluación cuantitativa de sitios web”. Permite evaluar de manera cuantitativa la accesibilidad de sitios web de forma automatizada utilizando diferentes métricas y crawlers que pueden integrarse fácilmente en la aplicación en forma de librerías.

La herramienta dispone de diversas utilidades para realizar los análisis como por ejemplo crear proyectos que aglutinen diferentes análisis, determinar la profundidad y el número de páginas analizadas o analizar un conjunto de URL almacenadas en un archivo. Existe tanto en versión web como de escritorio.

Se ha utilizado Atenea como aplicación base para integrar en ella la implementación realizada de la métrica de Ivory y evaluar de forma sencilla conjuntos de páginas web utilizando las facilidades que presenta. [Atenea12]



Ilustración 4.1 - Página principal de Atenea 3.0

4.3 JStyleParser

JStyleParser es una librería escrita en Java, gratuita y de código abierto, que analiza el estilo de los documentos HTML teniendo en cuenta tanto los estilos recogidos en hojas CSS como los estilos embebidos dentro del HTML mediante la etiqueta *style*. Sigue las especificaciones CSS2 y CSS3.

Una de las funcionalidades más importantes que ofrece es la posibilidad de mapear los estilos obtenidos del análisis en los elementos del árbol DOM que define el documento HTML. **[JStyleP13]**

Capítulo 5. Planificación y Resumen de Presupuestos

5.1 Planificación

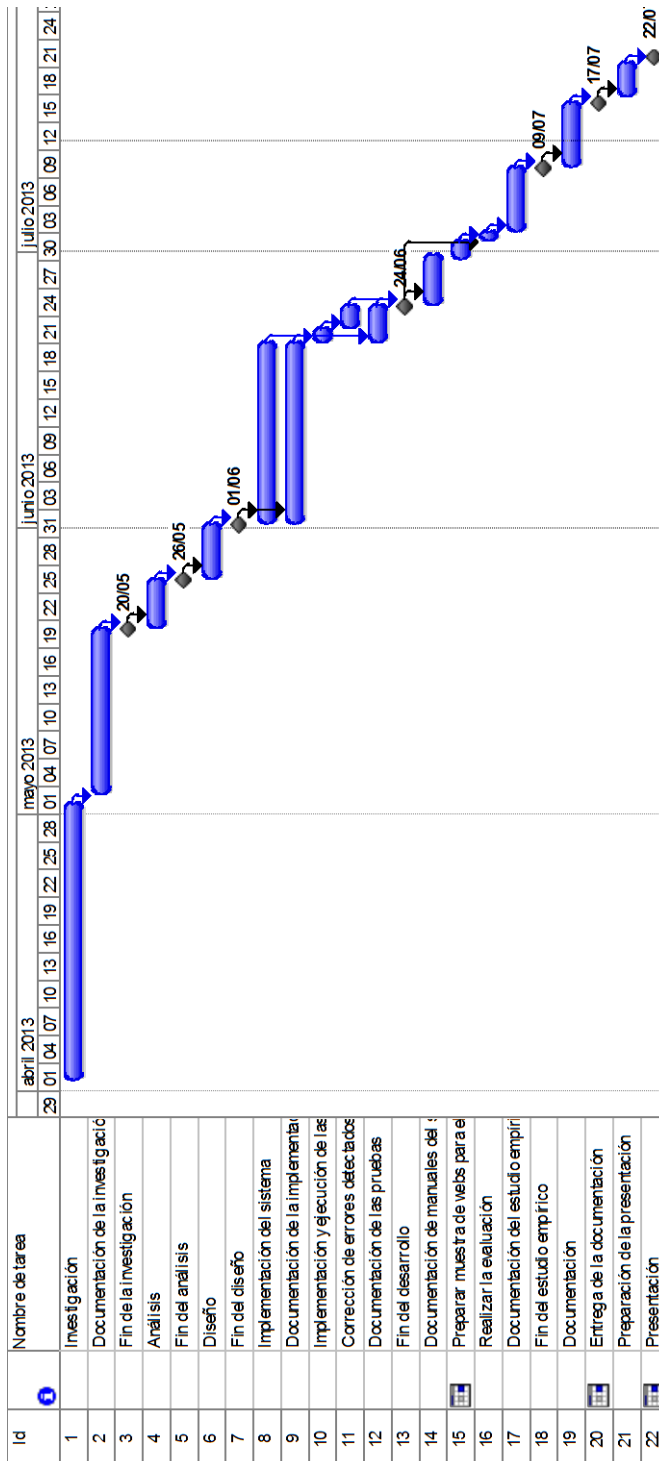


Diagrama 5.5.1 – Diagrama de Gantt: planificación del proyecto

5.2 Resumen del Presupuesto

A continuación se muestra el resumen del presupuesto en base a la planificación anterior.

Ítem	Subítem	Concepto	Horas	Precio/unidad	Total
1		Investigación			9.800,00 €
2		Desarrollo de aplicación			9.794,00 €
3		Estudio empírico de la usabilidad			2.650,00 €
Ítem	Subítem	Concepto	Unidades	Precio/unidad	Total
4		Software utilizado			222,53 €
5		Hardware			125,00 €
				Subtotal	22.591,53 €
				IVA 21%	4.744,22 €
				TOTAL	27.335,75 €

Tabla 5.1 – Resumen del presupuesto

Capítulo 6. Análisis

6.1 Definición del Sistema

6.1.1 Determinación del Alcance del Sistema

Se trata de implementar la métrica propuesta por Ivory a través de una librería que se integre en la herramienta Atenea 3.0. La librería desarrollada debe permitir evaluar la usabilidad de una página web mediante la métrica Ivory. Por este motivo el diseño de la librería vendrá determinado por la especificación definida en Atenea 3.0 para integrar nuevas librerías. **[Atenea12]**

Se implementará la fórmula de Ivory descrita en el capítulo 3.4. Se implementarán métodos para realizar las mediciones de los diferentes factores implicados en la fórmula.

La librería desarrollada recibirá un documento DOM que representará la página a analizar, realizará los análisis convenientes sobre el mismo y devolverá la puntuación final obtenida de aplicar la fórmula.

6.2 Requisitos del Sistema

6.2.1 Obtención de los Requisitos del Sistema

6.2.1.1 *Requisitos funcionales*

Código	Nombre del Requisito	Descripción del Requisito
R1.1	Evaluar un sitio web con la métrica Ivory	El sistema Atenea enviará una petición de evaluar un sitio web mediante la métrica Ivory y recibirá una respuesta con el valor obtenido.
R1.1.1	Contar el número de enlaces existentes en una página web	El sistema recibirá un documento DOM que representa la página web a analizar. A partir de él debe calcular el número de enlaces que hay en dicha página web.
R1.1.2	Contar el número de bloques de texto con el texto centrado que hay en una página web	El sistema recibirá un documento DOM que representa la página web a analizar. A partir de él, debe calcular el número de bloques de texto con el texto centrado que hay en dicha página web.
R1.1.3	Contar el número de colores diferentes que se utilizan en una página web	El sistema recibirá un documento DOM que representa la página web a analizar. A partir de él, debe calcular el número de colores diferentes que se utilizan en dicha página web.
R1.1.4	Medir el tamaño de una página web en bytes incluyendo las imágenes que contenga	El sistema recibirá un documento DOM que representa la página web a analizar. A partir de él, debe ser capaz de calcular el tamaño en bytes de dicha página web incluyendo también el tamaño de las imágenes que haya en ella.
R1.1.5	Calcular el porcentaje de texto de una página web que pertenece al cuerpo de la misma	El sistema recibirá un documento DOM que representa la página web a analizar. A partir de él, debe ser capaz de calcular qué porcentaje del texto de dicha página pertenece al cuerpo.
R1.1.6	Aplicar la fórmula de la métrica IVORY para calcular la usabilidad de una página web	El sistema recibirá los resultados de las medidas y aplicará la fórmula de la métrica de Ivory para obtener el grado de usabilidad de la página web analizada.

Tabla 6.1 – Requisitos funcionales

6.2.1.2 Requisitos no funcionales

Requisitos tecnológicos

Código	Nombre del Requisito	Descripción del requisito
R2.1	Integración con Atenea 3.0	Debe integrarse en la herramienta Atenea 3.0

Tabla 6.2 – Requisitos no funcionales

6.2.2 Identificación de Actores del Sistema

6.2.2.1 Atenea 3.0

Atenea 3.0 solicitará a la librería IVORY la evaluación de una página web y recibirá los resultados de la evaluación.

6.2.3 Especificación de Casos de Uso

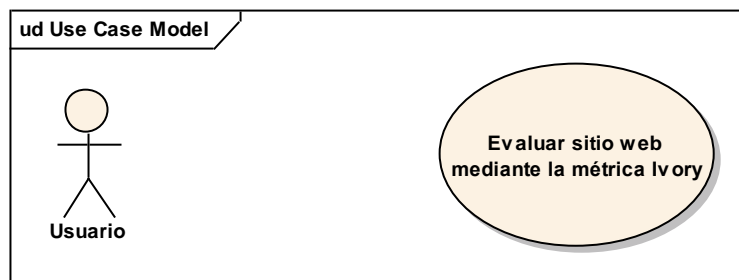


Diagrama 6.6.1 – Diagrama de casos de uso

Caso de Uso 1.1

Evaluar sitio web mediante la métrica Ivory

Descripción

El usuario solicita evaluar un sitio web mediante la métrica de Ivory. El sistema recibe la solicitud, realiza los análisis oportunos y devuelve la puntuación final de la página analizada.

Tabla 6.3 – Definición del caso de uso 1.1

6.3 Identificación de los Subsistemas en la Fase de Análisis

6.3.1 Descripción de los Subsistemas

El subsistema IVORY se integrará en un sistema ya existente, Atenea 3.0. A su vez, en el subsistema IVORY podemos distinguir los siguientes:

- **Checkpoints:** implementa cada una de las medidas que se necesitan para aplicar la fórmula Ivory.
- **Metrics:** implementa la fórmula Ivory y proporciona el resultado de la evaluación.

Los sistemas IVORY y Atenea 3.0 se verán entre ellos como cajas negras:

- Para IVORY Atenea será un sistema que realiza una petición de análisis y recibe un resultado.
- Para Atenea, IVORY será un sistema al que envía una solicitud de análisis y le devuelve un resultado.

6.3.2 Descripción de los Interfaces entre Subsistemas

La librería IVORY se comunicará con Atenea 3.0 a través de la API que ofrece esta última herramienta. Ambos sistemas se alojarán en la misma máquina.

6.4 Diagrama de Clases Preliminar del Análisis

Para poder integrar la librería IVORY en la herramienta Atenea 3.0, es necesario seguir unas directrices definidas en la API de Atenea 3.0. **[Atenea12]** A continuación se describen las pautas seguidas:

- La librería que implemente una métrica debe tener como nombre el nombre de la métrica en letras mayúsculas.
- Dentro de la librería existirán, al menos, los paquetes metrics y checkpoints.
- El paquete metrics albergará las clases de las métricas.
- El paquete checkpoints albergará las clases que implementan cada medida.
- Toda clase del paquete metric debe llamarse MetricXXX, siendo XXX el nombre de la métrica en letras mayúsculas y extenderá la clase MetricModel ubicada en el paquete model.
- Toda clase del paquete checkpoints debe llamarse CheckpointXXX donde XXX es el número de checkpoint que le corresponda. Los checkpoints deben numerarse consecutivamente y a partir del número 109. Cada clase extenderá de CheckpointModel, ubicada en el paquete model.

Para una información más detallada, consultar la referencia **[Atenea12]**.

6.4.1 Diagrama de Clases

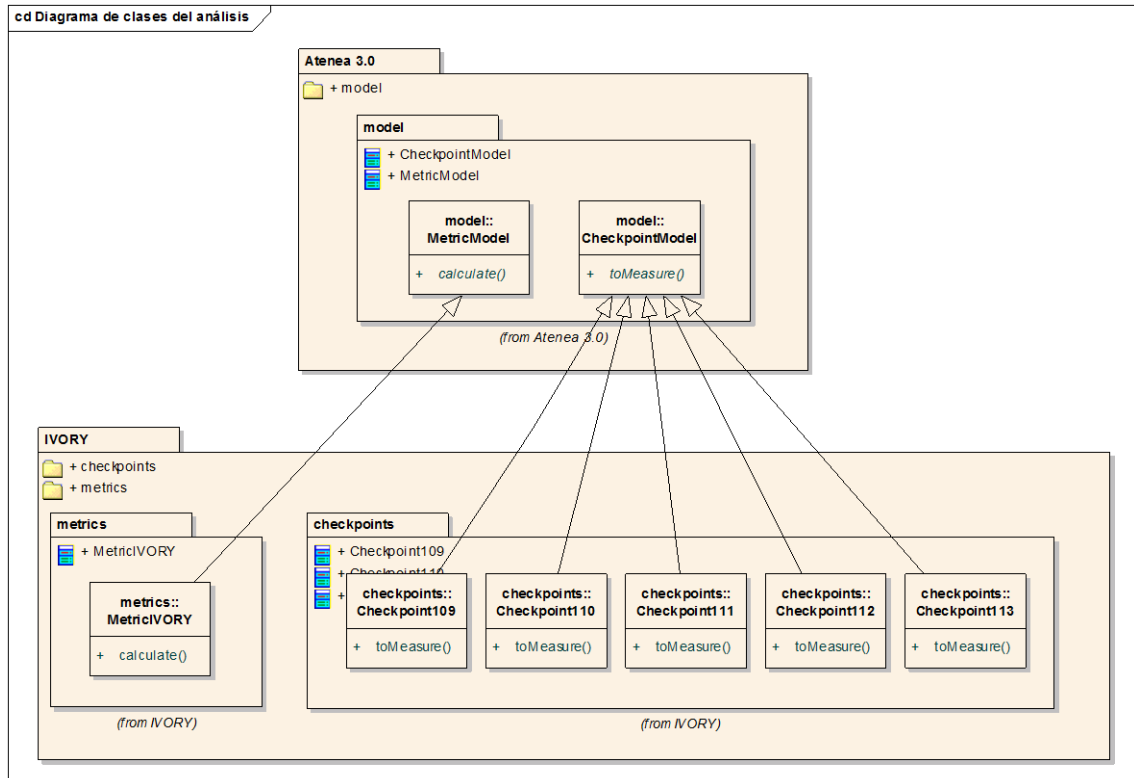


Diagrama 6.2 - Diagrama de clases del análisis

6.4.2 Descripción de las Clases

6.4.2.1 *IVORY.metrics*

Nombre de la Clase
MetricIvory
Descripción
Hereda de la clase MetricModel y define la métrica Ivory, su nombre y los checkpoints de los que se sirve. Recoge los resultados obtenidos por cada checkpoint y calcula el resultado final aplicando la fórmula Ivory.
Responsabilidades
<ul style="list-style-type: none"> Definir el nombre de la métrica Definir los checkpoints que pertenecen a la métrica Recoger los resultados de la evaluación realizada en cada checkpoint Aplicar la fórmula y calcular el resultado final Devolver el resultado final
Atributos Propuestos
name: define el nombre de la métrica
checkpoints: define los checkpoints que pertenecen a la métrica
Métodos Propuestos
calculate: aplica la fórmula y obtiene el resultado final

Tabla 6.4 – Descripción de la clase MetricIvory

6.4.2.2 *IVORY.checkpoints*

Nombre de la Clase
Checkpoint109
Descripción
Hereda de la clase CheckpointModel e implementa el checkpoint que se encarga de contar el número de enlaces que hay en un documento HTML.
Responsabilidades
<ul style="list-style-type: none"> Contar el número de enlaces que hay en un documento HTML.
Atributos Propuestos
name: nombre del checkpoint
description: descripción del checkpoint
Métodos Propuestos
toMeasure: analiza el documento para obtener la medida

Tabla 6.5 – Descripción de la tabla Checkpoint109

Nombre de la Clase
Checkpoint110
Descripción
Hereda de la clase CheckpointModel e implementa el checkpoint que se encarga de contar el número de bloques de texto con el texto centrado que hay en un documento HTML.
Responsabilidades
<ul style="list-style-type: none"> Contar el número de bloques de texto con el texto centrado que hay en un documento HTML.
Atributos Propuestos

name: nombre del checkpoint

description: descripción del checkpoint

Métodos Propuestos

toMeasure: analiza el documento para obtener la medida

Tabla 6.6 – Descripción de la clase Checkpoint110

Nombre de la Clase

Checkpoint111

Descripción

Hereda de la clase CheckpointModel e implementa el checkpoint que se encarga de contar el número de colores diferentes que se utilizan en un documento HTML.

Responsabilidades

- Contar el número de colores diferentes que se utilizan en un documento HTML.

Atributos Propuestos

name: nombre del checkpoint

description: descripción del checkpoint

Métodos Propuestos

toMeasure: analiza el documento para obtener la medida

Tabla 6.7 – Descripción de la clase Checkpoint111

Nombre de la Clase

Checkpoint112

Descripción

Hereda de la clase CheckpointModel e implementa el checkpoint que se encarga de medir el tamaño en bytes de un documento HTML incluyendo los elementos de imagen.

Responsabilidades

- Medir el tamaño en bytes de un documento HTML incluyendo los elementos de imagen.

Atributos Propuestos

name: nombre del checkpoint

description: descripción del checkpoint

Métodos Propuestos

toMeasure: analiza el documento para obtener la medida

Tabla 6.8 – Descripción de la clase Checkpoint112

Nombre de la Clase

Checkpoint113

Descripción

Hereda de la clase CheckpointModel e implementa el checkpoint que se encarga de determinar qué porcentaje del texto de un documento HTML pertenece al cuerpo de la página.

Responsabilidades

- Determinar qué porcentaje del texto de un documento HTML pertenece al cuerpo.

Atributos Propuestos

name: nombre del checkpoint

description: descripción del checkpoint

Métodos Propuestos

toMeasure: analiza el documento para obtener la medida

Tabla 6.9 – Descripción de la clase Checkpoint113

6.5 Análisis de Casos de Uso y Escenarios

6.5.1 Caso de Uso 1.1

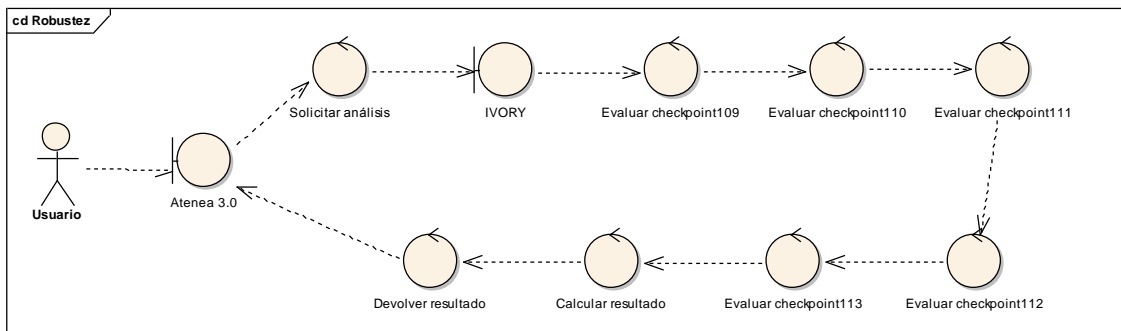


Diagrama 6.3 – Caso de uso 1.1: diagrama de robustez

Evaluar sitio web mediante la métrica Ivory	
Precondiciones	El usuario final solicita un análisis de una página web con Ivory desde Atenea 3.0.
Poscondiciones	Debe existir una nueva evaluación en el sistema.
Actores	Iniciado y terminado por el usuario final
Descripción	<ol style="list-style-type: none"> 1. El usuario inicia un análisis de una página web con la métrica Ivory. 2. Atenea 3.0. hace la solicitud de análisis a la librería IVORY. 3. Para cada página a evaluar: <ul style="list-style-type: none"> ○ Se evalúa cada uno de los checkpoints que forman la métrica. 4. Se recogen los valores obtenidos por los checkpoints y se calcula el resultado final. 5. Se envía el resultado final a Atenea 3.0.
Notas	No se diferencia entre usuario registrado o no registrado ni entre análisis de una o varias páginas debido a que el comportamiento de IVORY será el mismo en cualquiera de los casos.

Tabla 6.10 – Caso de uso 1.1: explicación del diagrama de robustez

6.6 Especificación del Plan de Pruebas

Para comprobar el buen funcionamiento de la aplicación y detectar y corregir errores se realizarán los siguientes tipos de pruebas:

- **Pruebas unitarias:** una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código o una clase individual que cumple con una función concreta. Esto sirve para asegurar que cada uno de ellos funcione correctamente por separado.
- **Pruebas de integración:** verifican que los distintos grupos de componentes funcionan correctamente cuando son ensamblados para cumplir con una función concreta.

Capítulo 7. Diseño del Sistema

7.1 Arquitectura del Sistema

7.1.1 Diagramas de Paquetes

En el siguiente diagrama se muestra la relación existente entre los paquetes IVORY.metrics e IVORY.checkpoints con el paquete model de Atenea. También se muestra la relación con los paquetes IVORY.test e IVORY.util.

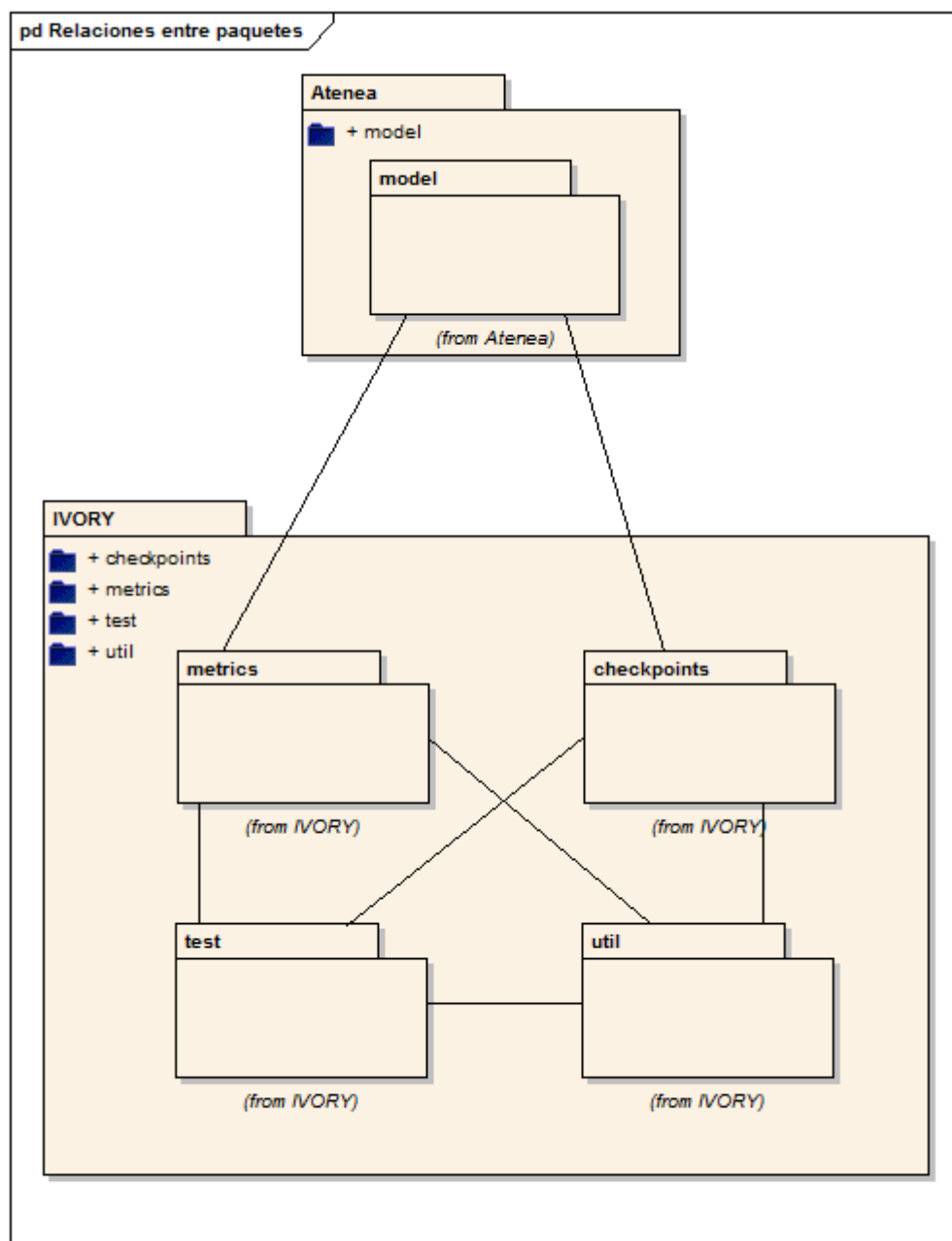


Diagrama 7.1 – Relaciones entre los paquetes

7.1.1.1 *IVORY.metrics*

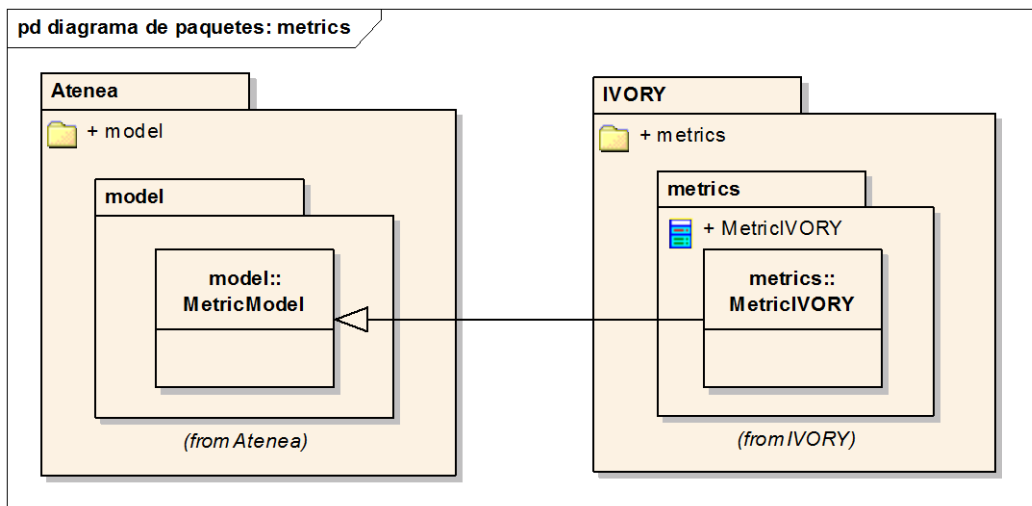


Diagrama 7.2 – Diagrama del paquete metrics

Como se ilustra en el diagrama, la clase MetricIVORY del paquete IVORY.metrics hereda de la clase MetricModel ubicada en el paquete model de Atenea.

7.1.1.2 *IVORY.checkpoints*

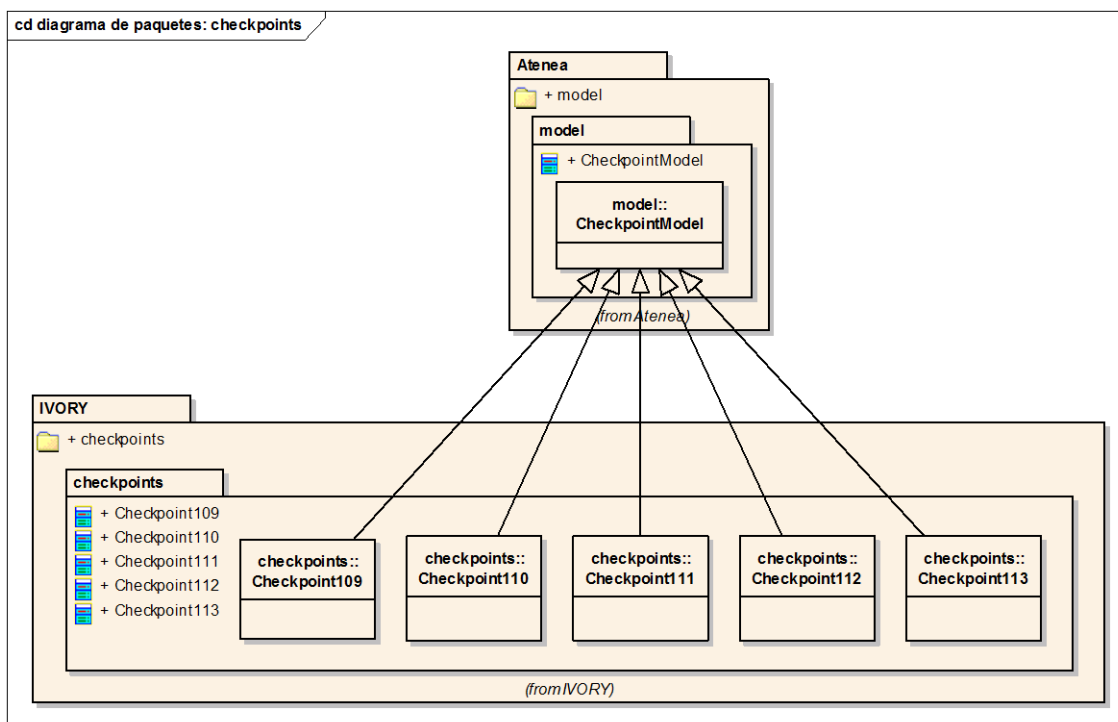


Diagrama 7.3 – Diagrama del paquete checkpoints

En el diagrama se puede ver como cada clase del paquete IVORY.checkpoints extiende la clase CheckpointModel del paquete model de Atenea.

7.1.1.3 IVORY.test

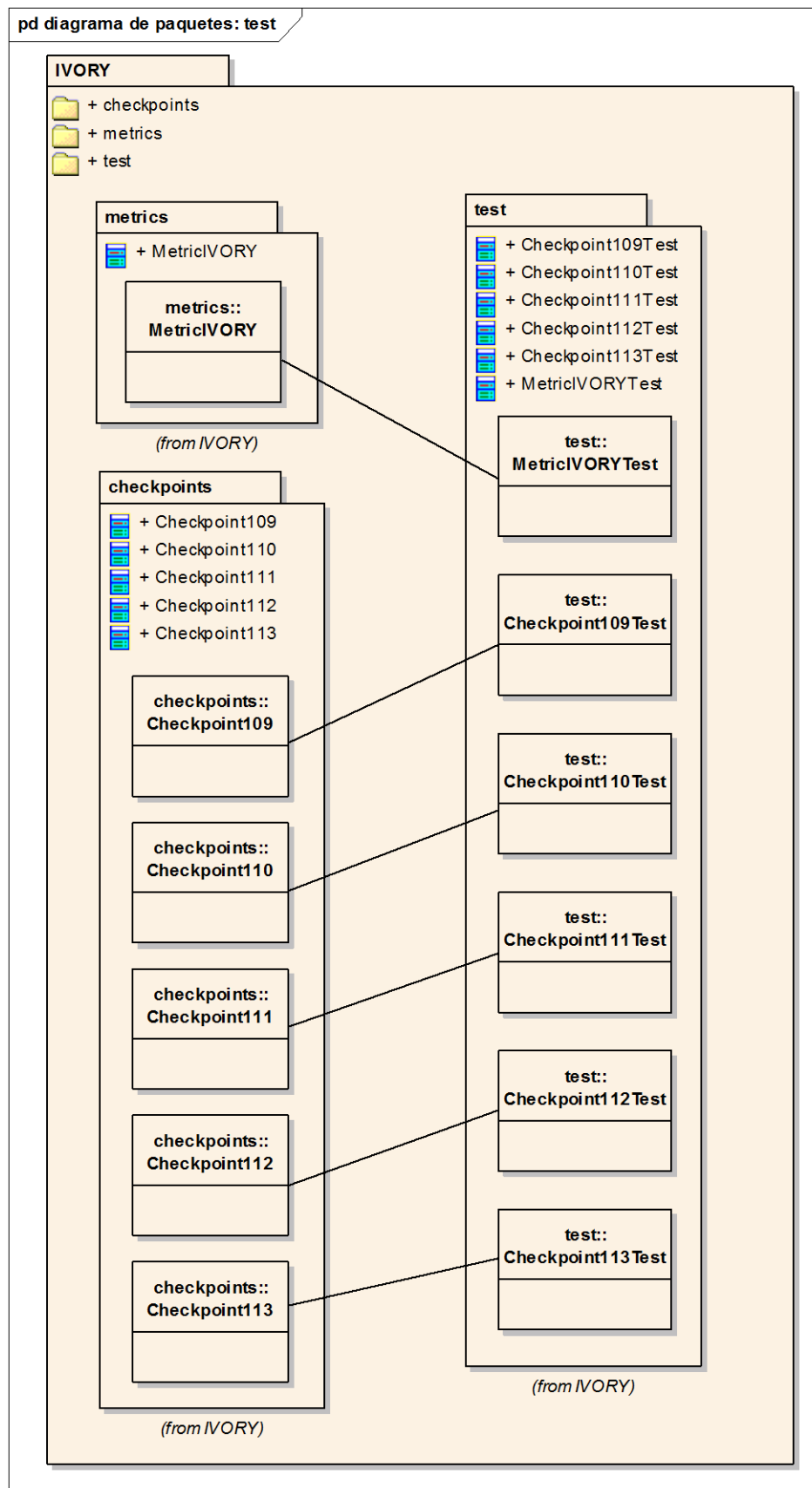


Diagrama 7.4 - Diagrama del paquete test

El paquete IVORY.test incluye las clases que implementan los test unitarios de las clases de los paquetes IVORY.metrics e IVORY.checkpoints. Dentro del paquete test existe una clase por cada clase de estos paquetes.

7.1.1.4 IVORY.util

Este paquete incluye enumeraciones que se utilizan desde el resto de los paquetes de la librería IVORY.

7.1.2 Diagramas de Componentes

La librería IVORY interactúa con la aplicación Atenea comportándose como una caja negra. Atenea recibe la petición del usuario final de realizar un análisis de una web mediante la métrica Ivory y delega la petición en la librería IVORY. La librería atenderá la petición y devolverá el resultado a Atenea, quien se encargará de mostrarlo al usuario final. De esta forma, las operaciones que realiza Atenea son totalmente ajenas a IVORY y viceversa.

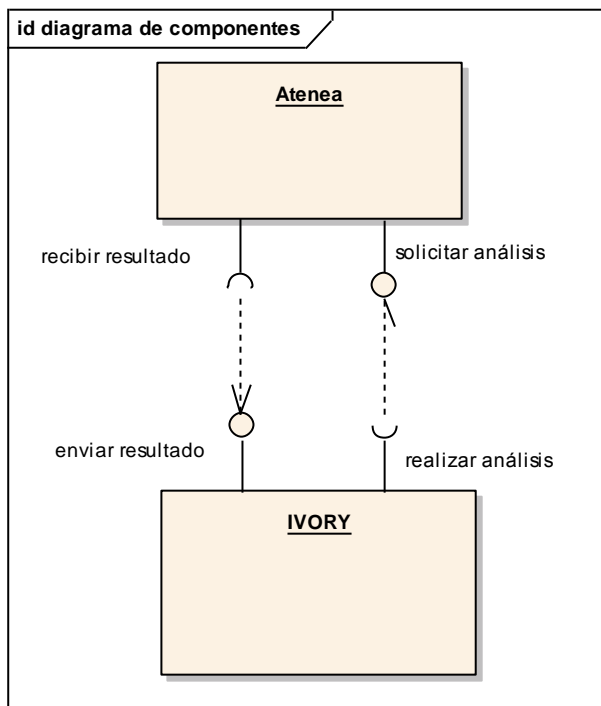


Diagrama 7.5 – Diagrama de componentes

7.1.3 Diagramas de Despliegue

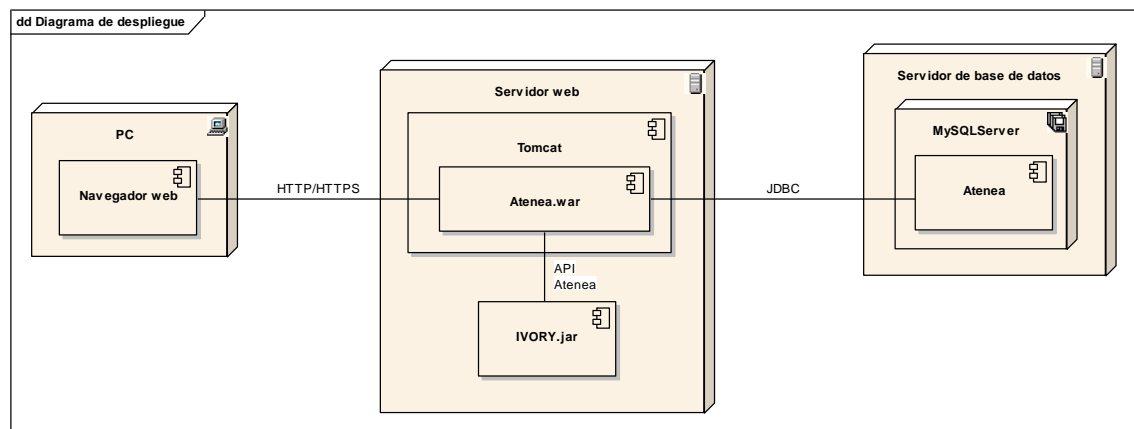


Diagrama 7.6 – Diagrama de despliegue

El diagrama de despliegue ilustra cómo encaja la librería IVORY dentro del despliegue necesario para poner en funcionamiento Atenea.

Los usuarios finales se acceden a Atenea desde su navegador web a través de una conexión HTTP o HTTPS, según se trate desde una conexión desde la parte pública o de una conexión desde la parte privada de Atenea. La aplicación Atenea se despliega en un Tomcat y se comunica con el servidor de base de datos MySQLServer mediante JDBC. A su vez, en el servidor donde está instalado Tomcat se encuentra la librería IVORY.jar que se comunicará con Atenea a través de la API definida por esta herramienta.

7.1.3.1 PC

Es el equipo desde el que el usuario accede a la aplicación web Atenea 3.0. Debe tener instalado un navegador web.

7.1.3.2 Servidor web

Es la máquina en la que la aplicación web Atenea 3.0 y la librería IVORY están alojadas. Se encarga de dar el soporte necesario para mantener la aplicación en la red y su ejecución.

7.1.3.3 Servidor de bases de datos

Da el soporte necesario para albergar la base de datos de la aplicación Atenea 3.0.

7.2 Diseño de Clases

7.2.1 Diagrama de Clases del paquete IVORY.metrics

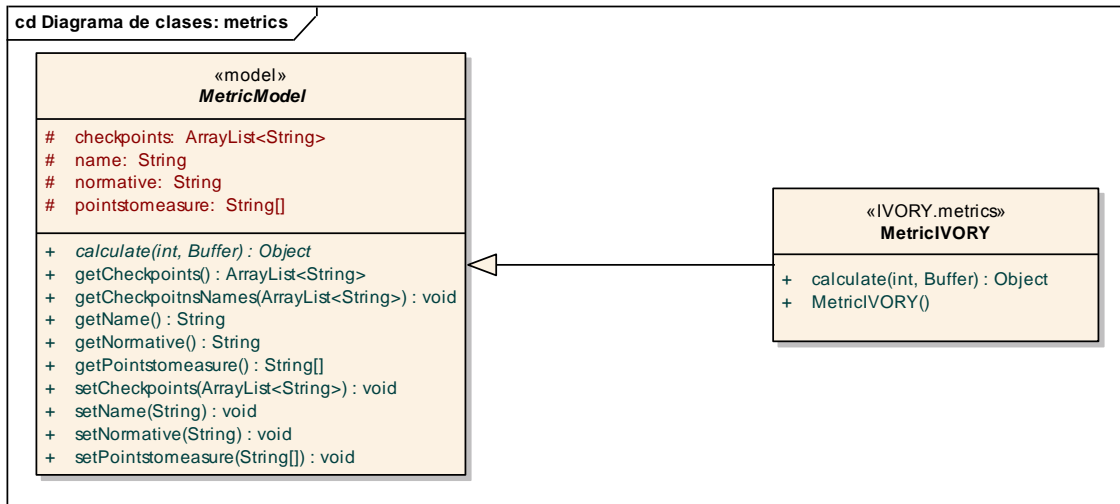


Diagrama 7.7 – Diagrama de clases del paquete IVORY.metrics

7.2.2 Diagrama de Clases del paquete IVORY.checkpoints

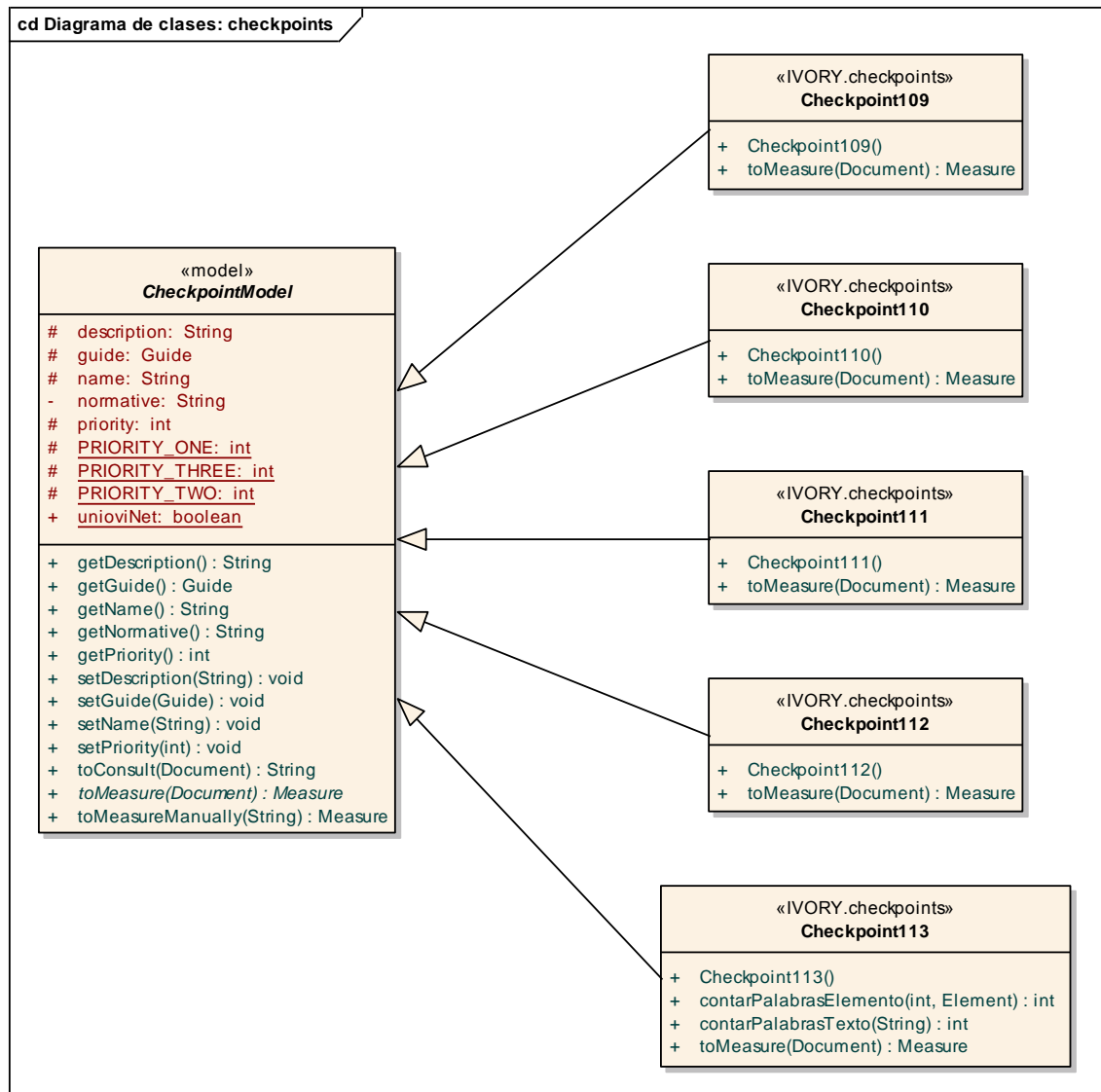


Diagrama 7.8 – Diagrama de clases del paquete IVORY.checkpoints

7.2.3 Diagrama de Clases del paquete IVORY.util

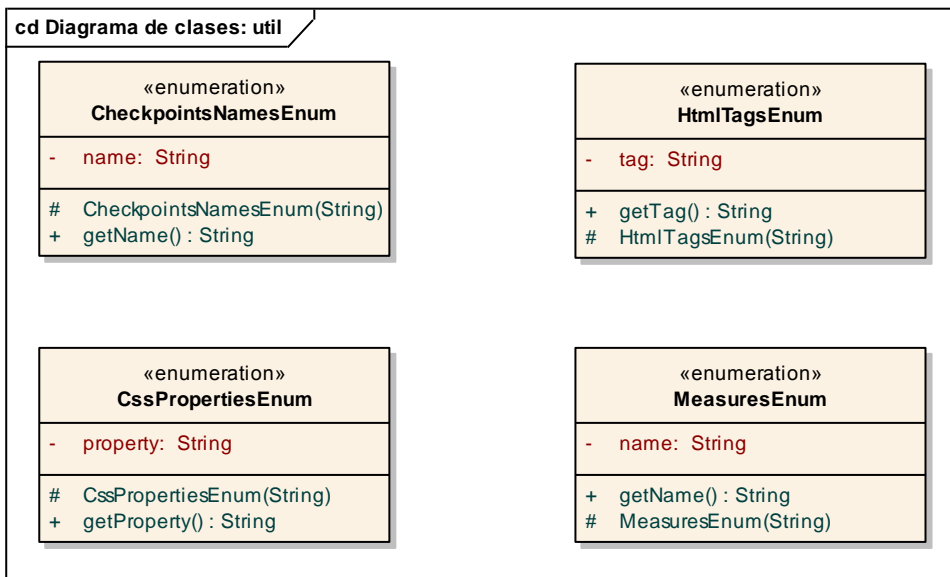


Diagrama 7.9 - Diagrama de clases del paquete IVORY.util

7.2.4 Diagrama de Clases del paquete IVORY.test

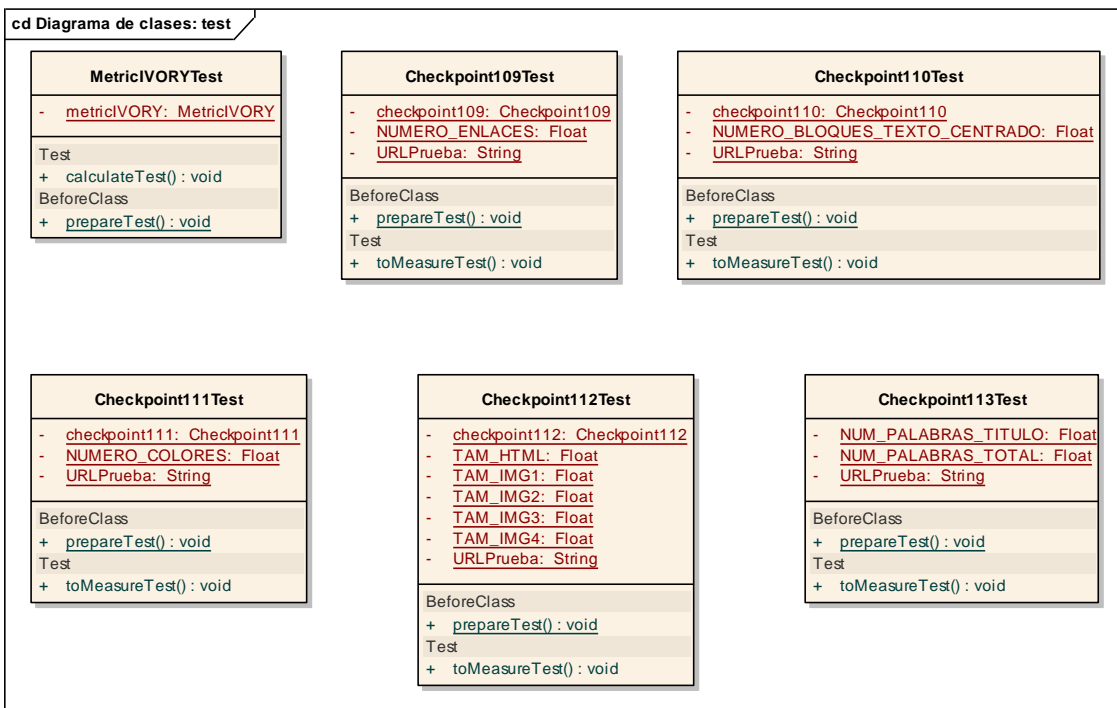


Diagrama 7.10 - Diagrama de clases del paquete IVORY.test

7.3 Diagramas de Interacción y Estados

7.3.1 Diagrama de secuencia

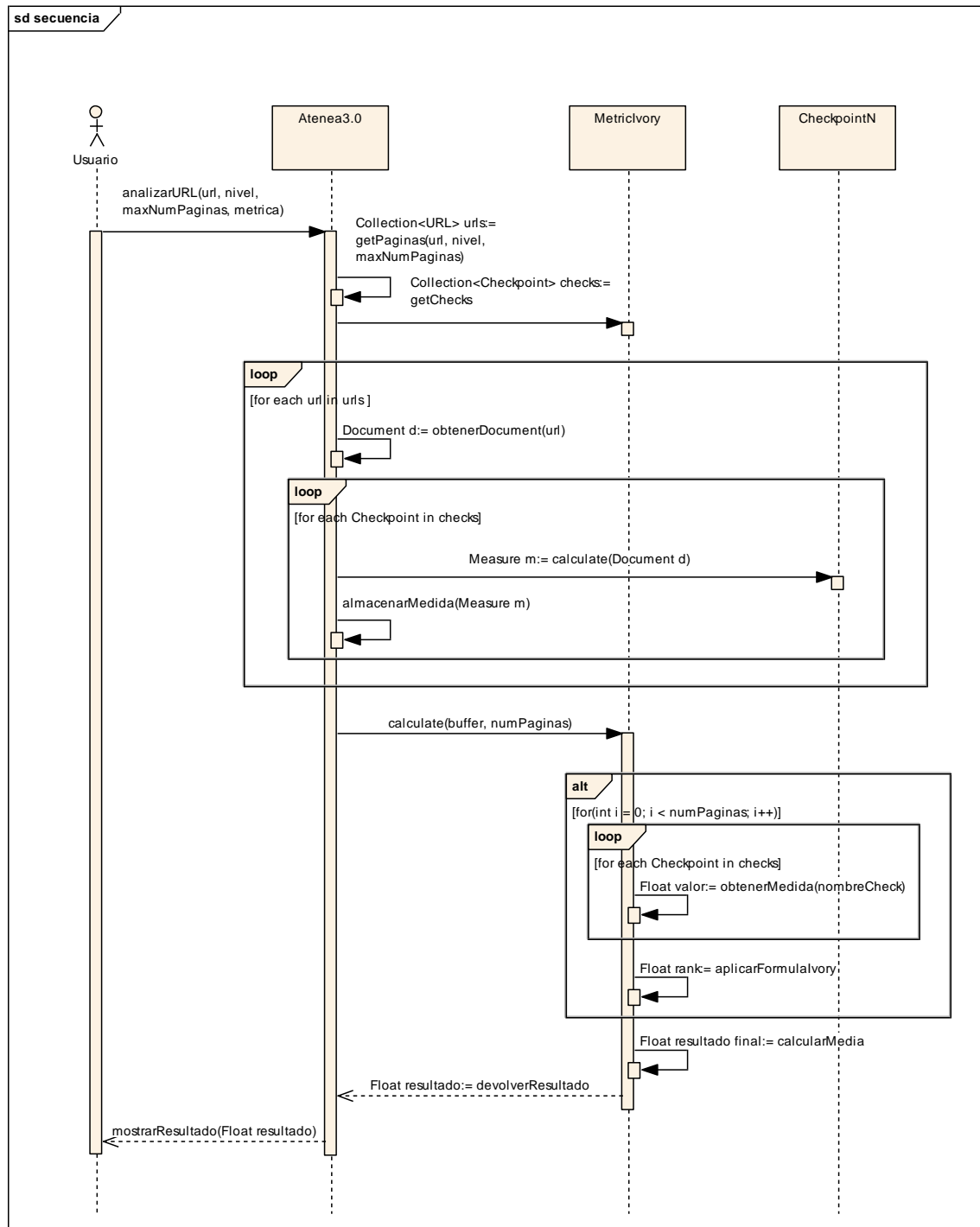


Diagrama 7.11 – Diagrama de Secuencia

El diagrama de secuencia explica el proceso mediante el que la librería Ivory trabaja para Atenea 3.0. Para simplificar el proceso, se ha considerado que el usuario solicita que se analice

una URL con la métrica Ivory y se ha puesto una línea de vida llamada “CheckpointN” que representa a un Checkpoint genérico.

El proceso descrito se resume en los siguientes pasos:

1. El usuario solicita el análisis de una URL con la métrica Ivory.
2. Atenea3.0 obtiene las páginas a analizar teniendo en cuenta la profundidad y anchura del análisis que ha solicitado el usuario.
3. Después, busca los checkpoints que pertenecen a la métrica seleccionada.
4. Para cada una de las páginas a analizar:
 - a. Obtiene el documento DOM que la representa.
 - b. Llama a la librería Ivory para que haga la evaluación de cada uno de los checkpoints sobre el documento obtenido.
 - c. Ivory devuelve la medida realizada y Atenea la almacena en un Buffer.
5. Atenea3.0 hace la llamada a Ivory para que calcule el resultado final y le envía el Buffer con los resultados obtenidos de la evaluación de cada checkpoint y el número de páginas analizadas.
6. Para cada página, Ivory:
 - a. Obtiene el valor de cada checkpoint.
 - b. Aplica la fórmula.
7. Finalmente, se calcula la media aritmética de los resultados obtenidos y se devuelve a Atenea.
8. Se muestra el resultado en pantalla.

7.4 Especificación Técnica del Plan de Pruebas

7.4.1 Pruebas Unitarias

Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código que cumple una función concreta. Mediante la realización de pruebas unitarias se busca saber que todos los módulos del código funcionan correctamente por separado.

Para realizar las pruebas unitarias se ha utilizado JUnit, un framework, creado por Erich Gamma y Kent Beck, que permite realizar las pruebas unitarias de manera controlada. Su funcionamiento es sencillo: evalúa una expresión para un valor de entrada concreto, si se obtiene el valor esperado, informará de que el método se ejecutó correctamente; en caso contrario, indicará que no se ha obtenido el resultado esperado. [JUnit4]

Las pruebas unitarias se realizarán sobre una página web creada para este fin de la que se conocerán todos los valores que la aplicación mide automáticamente.

A continuación se especifican las pruebas unitarias implementadas para cada una de las clases de la librería IVORY.

7.4.1.1 Clase MetricIVORY

Test	Resultado esperado	Si falla...
calculateTest	La puntuación de usabilidad calculada es la esperada	<ul style="list-style-type: none"> Comprobar que se utilizan los nombres correctos de los checkpoints. Comprobar que se buscan las medidas en el buffer con los nombres correctos. Comprobar que el tipo de dato de los valores es Float. Comprobar que las medidas tienen los nombres correctos. Comprobar que la métrica tiene asignados los checkpoints correctos. Comprobar que se utiliza la fórmula correcta para realizar el cálculo final.

Tabla 7.1 – Test unitarios de la clase MetricIVORY

7.4.1.2 Clase Checkpoint109

Test	Resultado esperado	Si falla...
toMeasureTest	El número de enlaces calculado por el checkpoint es el esperado	<ul style="list-style-type: none"> Comprobar que el nombre asignado a la medida es el correcto. Comprobar que se buscan los nodos del documento con la etiqueta HTML correcta.

- Comprobar que el valor obtenido se almacena en la medida con el nombre correcto.

Tabla 7.2 – Test unitarios de la clase Checkpoint109

7.4.1.3 Clase Chekcpont110

Test	Resultado esperado	Si falla...
toMeasureTest	El número calculado de bloques con el texto centrado es el esperado.	<ul style="list-style-type: none"> • Comprobar que la medida se crea con el nombre correcto. • Comprobar que se crea el mapa de estilos. • Comprobar que se buscan las propiedades de estilo correctas. • Comprobar que se hace la búsqueda para todos los elementos de párrafo y encabezados. • Comprobar que el resultado obtenido se almacena en la medida con el nombre correcto.

Tabla 7.3 – Test unitarios de la clase Checkpoint110

7.4.1.4 Clase Checkpoint111

Test	Resultado esperado	Si falla...
calculateTest	El número de colores diferentes contados es el esperado	<ul style="list-style-type: none"> • Comprobar que la medida se crea con el nombre correcto. • Comprobar que se crea el mapa de estilos. • Comprobar que se buscan todas las propiedades de estilo que se encargan del color. • Comprobar que los colores obtenidos de los estilos se convierten a Color correctamente. • Comprobar que el resultado obtenido se almacena en la medida con el nombre correcto.

Tabla 7.4 – Test unitarios de la clase Checkpoint111

7.4.1.5 Clase Checkpoint112

Test	Resultado esperado	Si falla...
calculateTest	El tamaño de la página y sus imágenes calculado es el esperado.	<ul style="list-style-type: none"> • Comprobar que la medida se crea con el nombre correcto. • Comprobar que se tienen en cuenta las imágenes para hacer el cálculo.

- Comprobar que el resultado obtenido se almacena en la medida con el nombre correcto.

Tabla 7.5 – Test unitarios de la clase Checkpoint112

7.4.1.6 Clase Checkpoint113

Test	Resultado esperado	Si falla...
calculateTest	El porcentaje de texto de la página que pertenece al cuerpo es el esperado.	<ul style="list-style-type: none"> • Comprobar que la medida se crea con el nombre correcto. • Comprobar que se obtiene correctamente el texto de los encabezados. • Comprobar que se obtiene correctamente el texto del cuerpo. • Comprobar que se tienen en cuenta todos los niveles de encabezado que pueden existir. • Comprobar que el resultado se almacena en la medida con el nombre correcto.

Tabla 7.6 – Test unitarios para la clase Checkpoint113

7.4.2 Pruebas de Integración y del Sistema

Las pruebas de integración se asocian a grupos de componentes. Consisten en comprobar que éstos funcionan correctamente cuando son ensamblados con una función concreta.

Las pruebas realizadas se resumen en la siguiente tabla.

Prueba	Resultado esperado	Si falla...
Seleccionar Ivory como métrica para realizar un análisis a través de Atenea 3.0	Entre las métricas disponibles para realizar el análisis debe aparecer la métrica Ivory y poder ser seleccionada	<ul style="list-style-type: none"> • Comprobar que la librería que contiene la métrica se encuentra instalada correctamente. • Comprobar que los componentes de la librería se han empaquetado correctamente. • Comprobar que los paquetes y clases de la librería siguen las especificaciones del API de Atenea 3.0
Realizar un análisis de una página web con la métrica Ivory desde Atenea 3.0	Se muestra el resultado del análisis en una tabla.	<ul style="list-style-type: none"> • Comprobar que la URL existe. • Comprobar que los componentes de la librería se han empaquetado correctamente. • Comprobar que la métrica tiene los checkpoints correctos. • Comprobar que los valores

medidos por los checkpoints se guardan correctamente.

- Comprobar que realmente se ha evaluado cada uno de los checkpoints.
- Comprobar que se ha aplicado la fórmula de IVORY con los valores correctos calculados por los checkpoints

Tabla 7.7 – Pruebas de integración

Capítulo 8. Implementación del Sistema

8.1 Estándares y Normas Seguidos

8.1.1 API Atenea3.0

Para poder integrar la métrica de Ivory en la aplicación Atenea3.0 el diseño y desarrollo se ha realizado siguiendo la API definida en la documentación de la herramienta. A continuación se resumen las pautas seguidas para la implementación de la métrica, para más información se puede consultar la referencia **[Atenea12]**.

Se ha implementado la métrica y sus checkpoints y se ha empaquetarlo todo en un archivo .jar. Atenea define los checkpoints como características de una página web. En nuestro caso, las características a implementar son los factores de la fórmula de Ivory. Cada checkpoint se implementa en una clase diferente y sigue las siguientes normas:

- Se ubica en un paquete llamado IVORY.checkpoints.
- Hereda de CheckpointModel y sobrescribe el método *toMeasure*.
- El nombre de la clase es CheckpointXXX, donde xxx es un número a partir de 109.
- Se inicializa el atributo *description* con una descripción del checkpoint.
- Se inicializa en atributo *name* con el nombre de la clase.

La métrica se encarga de calcular el resultado final de la evaluación de la página web. La clase que la implementa sigue las siguientes normas:

- Pertenece al paquete IVORY.metrics
- Hereda de MetricModel y sobrescribe el método *calculate*.
- El nombre de la clase es MetricIVORY.
- Contiene una *Collection* con los nombres de los checkpoints que le pertenecen.
- Se inicializa el atributo *name*.

8.2 Lenguajes de Programación

8.2.1 Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a mediados de los 90. Destaca por ser independiente de la plataforma, es decir, las aplicaciones escritas en este lenguaje pueden ejecutarse sobre una máquina virtual de Java (JVM) independientemente de la arquitectura del ordenador en el que esté instalada.

Todo el código de la implementación de la métrica Ivory ha sido escrito en Java. La versión utilizada ha sido Java 6.

8.2.2 HTML

HTML (Hyper Text Markup Language) es el lenguaje con el que se escriben las páginas web. Se dice que es un lenguaje de marcas porque se basa en un conjunto de etiquetas que describen el contenido del documento. Los documentos HTML contienen etiquetas HTML y texto plano.

Se ha utilizado HTML para escribir las páginas web estáticas que se han creado para probar la métrica Ivory y para realizar el estudio empírico. También ha sido necesario conocer este lenguaje para implementar la métrica Ivory. **[W3CSchHTML13]**

8.2.3 CSS

CSS (Cascading Style Sheets) se utiliza para definir la apariencia con la que se mostrarán los elementos HTML en un navegador web. Esta información puede estar embebida en los documentos HTML o escrita en un documento externo que se referenciará desde la cabecera del documento HTML.

En este proyecto se ha utilizado CSS para describir el estilo de las páginas web estáticas creadas para probar la métrica y realizar el estudio empírico. Además, para implementar la métrica ha sido necesario conocer cómo se utiliza CSS y su sintaxis. **[W3CSchCSS13]**

8.3 Herramientas y Programas Usados para el Desarrollo

8.3.1 Eclipse Java EE IDE for Web Developers

Eclipse es un Entorno de Desarrollo Integrado (IDE) gratuito y Open Source. Se ha utilizado la distribución Java EE IDE for Web Developers que está pensada para el desarrollo de aplicaciones web utilizando Java EE, JPA, JSF, etc. La versión utilizada ha sido la Helios Service Release 1.

El entorno puede descargarse de la página web oficial del proyecto Eclipse: <http://www.eclipse.org/downloads/>. [Eclipse13]

8.3.2 AppServ

AppServ es una herramienta gratuita que permite instalar de forma rápida y sencilla PHP, Apache y MySQL. Además, integra una herramienta para gestionar la base de datos MySQL llamada PhpMyAdmin.

Se ha utilizado para alojar y gestionar la base de datos que utiliza la aplicación web. La versión utilizada ha sido la 2.5.10 que incluye Apache 2.2.8, PHP 5.2.6 MySQL 5.0.51b y phpMyAdmin 2.10.3. La herramienta puede descargarse desde la web <http://www.appservnetwork.com/index.php>. [AppServ13]

8.3.3 Tomcat

Tomcat es un contenedor de servlets desarrollado por los componentes del proyecto Jakarta de la Apache Software Foundation. Es gratuito y de código abierto e implementa las especificaciones de servlets y JSP de Sun Microsystems.

Se ha utilizado la versión Tomcat 7.0.41 para desplegar la aplicación web Atenea. Puede descargarse desde la página web oficial del proyecto: <http://tomcat.apache.org/>. [Tomcat13]

8.3.4 Notepad++

Notepad++ es un editor de texto gratuito que soporta múltiples lenguajes como Java, SQL, HTML, CSS. Se ha utilizado para escribir y editar de forma rápida los documentos HTML y CSS.

Puede descargarse desde la web <http://notepad-plus-plus.org/>. [Notepad13]

8.4 Creación del Sistema

8.4.1 Problemas Encontrados

El principal problema encontrado durante el desarrollo ha sido la falta de conocimiento sobre el funcionamiento de la herramienta Atenea 3.0 en la que se integra la métrica implementada. Es necesario entender cómo funciona esta aplicación para comprender cómo se realizan los análisis de las páginas web y optimizar el funcionamiento de la nueva métrica y no siempre es sencillo comprender el código y la documentación que ha hecho otra persona.

Otro de los problemas encontrados radica en que en Atenea, cuando una página no podía ser analizada por algún motivo, se mostraba como código de error un -1.0 en lugar del valor correspondiente al análisis realizado. Sin embargo, dado que la métrica de Ivory puede proporcionar como resultado el valor -1.00000000 este hecho podría conducir a error. Por lo tanto, se ha solucionado el problema modificando el código de error de Atenea para que sea -1000.0.

8.4.2 Descripción Detallada de las Clases

8.4.2.1 MetricIvory

Nombre	Tipo	Descripción	Hereda de...
MetricIvory		Implementa la fórmula Ivory	MetricModel
<i>Responsabilidades</i>			
Número	Descripción		
1	Definir la métrica Ivory y determinar cuáles son sus checkpoints.		
2	Obtener los valores calculados por los checkpoints para cada página analizada.		
3	Aplicar la fórmula Ivory para cada página analizada y obtener el resultado.		
4	Calcular el valor final haciendo una media aritmética de los valores obtenidos para cada página.		
<i>Métodos</i>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público		MetricIVORY	
Público	Object	calculate	Buffer buffer, int pagenum
<i>Atributos</i>			
Acceso	Modo	Tipo o Clase	Nombre
(Público, Privado)	Protegido, (Estático, Final)		
Observaciones			

Tabla 8.1 – Descripción detallada de la clase MetricIvory

8.4.2.2 Checkpoint109

Nombre	Tipo	Descripción	Hereda de...
Checkpoint109		Cuenta el número de enlaces que hay en una página	CheckpointModel
<i>Responsabilidades</i>			
Número		Descripción	
1		Definir un nuevo checkpoint.	
2		Contar el número de enlaces que hay en un documento DOM.	
<i>Métodos</i>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público		Checkpoint109	
Público	Measure	toMeasure	Document d

Tabla 8.2 – Descripción detallada de la clase Checkpoint109

8.4.2.3 Checkpoint110

Nombre	Tipo	Descripción	Hereda de...
Checkpoint110		Cuenta el número de bloques de texto con el texto centrado de una página	CheckpointModel
<i>Responsabilidades</i>			
Número		Descripción	
1		Definir un nuevo checkpoint.	
2		Contar el número de bloques de texto que hay en un documento DOM que tienen el texto centrado.	
<i>Métodos</i>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público		Checkpoint110	
Público	Measure	toMeasure	Document d

Tabla 8.3 – Descripción detallada de la clase Checkpoint110

8.4.2.4 Checkpoint111

Nombre	Tipo	Descripción	Hereda de...
Checkpoint111		Cuenta el número de colores diferentes que se utilizan en una página web	CheckpointModel
<i>Responsabilidades</i>			
Número		Descripción	
1		Definir un nuevo checkpoint.	
2		Determinar cuántos colores diferentes se utilizan en un documento DOM.	
<i>Métodos</i>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público		Checkpoint111	

Público	Measure	toMeasure	Document d
---------	---------	-----------	------------

Tabla 8.4 – Descripción detallada de la clase Checkpoint111

8.4.2.5 Checkpoint112

Nombre	Tipo	Descripción	Hereda de...
Checkpoint112		Mide el tamaño en bytes de una página web	CheckpointModel
<i>Responsabilidades</i>			
Número	Descripción		
1	Definir un nuevo checkpoint.		
2	Medir el tamaño en bytes de la página web representada por un documento DOM incluyendo las imágenes que pudiera contener.		
<i>Métodos</i>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público		Checkpoint112	
Público	Measure	toMeasure	Document d

Tabla 8.5 – Descripción detallada de la clase Checkpoint112

8.4.2.6 Checkpoint113

Nombre	Tipo	Descripción	Hereda de...
Checkpoint113		Calcula el porcentaje del texto de una página web que pertenece al cuerpo.	CheckpointModel
<i>Responsabilidades</i>			
Número	Descripción		
1	Define un nuevo checkpoint.		
2	Calcula que porcentaje del texto de una página web pertenece al cuerpo de la misma.		
<i>Métodos</i>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público		Checkpoint113	
Público	Measure	toMeasure	Document d
Privado	int	contarPalabrasTexto	String texto
Privado	Int	contarPalabrasElemento	Element e, int contador

Tabla 8.6 – Descripción detallada de la clase Checkpoint113

8.4.2.7 MetricModel

Nombre	Tipo	Descripción	Hereda de...
MetricModel	Abstracta	Define una métrica que se puede integrar en Atenea3.0	
<u>Responsabilidades</u>			
Número	Descripción		
1	Definir los métodos y atributos que tendrá una métrica de Atenea3.0		
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público Abstracto	Object	calculate	Buffer buffer, int pagenum
Público	String	getName	
Público		setName	String name
Público	ArrayList<String>	getCheckpoints	
Público		setCheckpoints	ArrayList<String> checkpoints
Público		setNormative	String normative
Público		setCheckPoints	ArrayList<String> checkpoints
Público	ArrayList<String>	getCheckpointsNames	
Público	String	getNormative	
Público	String[]	getPointstomeasure	
Público		setPointstomeasure	String[] checks
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
Privado		String	normative
Protegido		String	name
Protegido		ArrayList<String>	checkpoints
Protegido		String[]	pointstomeasure
Observaciones			
Esta clase ha sido desarrollada por Javier Oyón para la herramienta Atenea3.0. Se ha incluido en la descripción detallada de las clases porque se considera relevante para entender el funcionamiento de la librería Ivory, ya que la estructura de la clase MetricIvory y su comportamiento, vienen determinados directamente por esta clase. Para más información sobre la herramienta Atenea3.0 consultar la referencia [Atenea12] .			

Tabla 8.7 – Descripción detallada de la clase MetricModel

8.4.2.8 CheckpointsNamesEnum

Nombre	Tipo	Descripción	Hereda de...
CheckpointsNamesEnum	Enumeration	Define los nombres de los checkpoints de la métrica IVORY	
<u>Responsabilidades</u>			
Número		Descripción	
1	Definir los nombres de los checkpoints de la métrica IVORY		
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Protegido		CheckpointsNamesEnum	String name
Público	String	getName	
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
Privado		String	name

Tabla 8.8 – Descripción detallada de CheckpointsNamesEnum

8.4.2.9 CssPropertiesEnum

Nombre	Tipo	Descripción	Hereda de...
CssPropertiesEnum	Enumeración	Define las propiedades de CSS necesarias	
<u>Responsabilidades</u>			
Número		Descripción	
1	Definir las propiedades de CSS que utiliza el resto de clases		
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Protegido		CssPropertiesEnum	String property
Público	String	getProperty	
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
Privado		String	property

Tabla 8.9 – Descripción detallada de CssPropertiesEnum

8.4.2.10 HtmlTagsEnum

Nombre	Tipo	Descripción	Hereda de...
HtmlTagsEnum	Enumeración	Define las etiquetas HTML utilizadas	
<u>Responsabilidades</u>			
Número		Descripción	
1	Definir las etiquetas HTML utilizadas por las demás clases		
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Protegido		HtmlTagsEnum	String tag
Público	String	getTag	
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
Privado		String	tag

Tabla 8.10 – Descripción detallada de HtmlTagsEnum

8.4.2.11 MeasuresEnum

Nombre	Tipo	Descripción	Hereda de...
MeasuresEnum	Enumeración	Define los nombres de las propiedades medidas	
<u>Responsabilidades</u>			
Número		Descripción	
1	Definir los nombres de las propiedades medidas por los checkpoints		
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Protegido		MeasuresEnum	String name
Público	String	getName	
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
Privado		String	name

Tabla 8.11 – Descripción detallada de MeasuresEnum

8.4.2.12 Checkpoint109Test

Nombre	Tipo	Descripción	Hereda de...
Checkpoint109Test		Implementa los test unitarios para la clase Checkpoint109	
<u>Responsabilidades</u>			
Número	Descripción		
1	Implementar los test unitarios para los métodos de la clase Checkpoint109		
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público Estático		prepareTest	
Público		toMeasureTest	
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
Privado	Estático, final	String	URLPrueba
Privado	Estático, final	Float	NUMERO_ENLACES
Privado	Estático	Checkpoint109	checkpoint109

Tabla 8.12 – Descripción detallada de la clase Checkpoint109Test

8.4.2.13 Checkpoint110Test

Nombre	Tipo	Descripción	Hereda de...
Checkpoint110Test		Implementa los test unitarios de la clase Checkpoint110	
<u>Responsabilidades</u>			
Número	Descripción		
1	Implementar los test unitarios para los métodos de la clase Checkpoint110		
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público Estático		prepareTest	
Público		toMeasureTest	
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
Privado	Estático, final	String	URLPrueba
Privado	Estático, final	Float	NUMERO_BLOQUES_TEXTO_CENTRADO
Privado	Estático	Checkpoint110	checkpoint110

Tabla 8.13 – Descripción detallada de la clase Checkpoint110Test

8.4.2.14 Checkpoint111Test

Nombre	Tipo	Descripción	Hereda de...
Checkpoint111Test		Implementa los test unitarios para la clase Checkpoint111	
<u>Responsabilidades</u>			
Número	Descripción		
1	Implementar los test unitarios para los métodos de la clase Checkpoint111		
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público Estático		prepareTest	
Público		toMeasureTest	
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
Privado	Estático, final	String	URLPrueba
Privado	Estático, final	Float	NUMERO_COLORES
Privado	Estático	Checkpoint111	checkpoint111

Tabla 8.14 – Descripción detallada de la clase Checkpoint111Test

8.4.2.15 Checkpoint112Test

Nombre	Tipo	Descripción	Hereda de...
Checkpoint112Test		Implementa los test unitarios para la clase Checkpoint112	
<u>Responsabilidades</u>			
Número	Descripción		
1	Implementar los test unitarios para los métodos de la clase Checkpoint112		
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público Estático		prepareTest	
Público		toMeasureTest	
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
Privado	Estático, final	String	URLPrueba
Privado	Estático, final	Float	TAM_HTML
Privado	Estático, final	Float	TAM_IMG1
Privado	Estático, final	Float	TAM_IMG2
Privado	Estático, final	Float	TAM_IMG3
Privado	Estático, final	Float	TAM_IGM4
Privado	Estático	Checkpoint112	checkpoint112

Tabla 8.15 – Descripción detallada de la clase Checkpoint112Test

8.4.2.16 Checkpoint113Test

Nombre	Tipo	Descripción	Hereda de...
Checkpoint113Test		Implementa los test unitarios de la clase Checkpoint113	
<u>Responsabilidades</u>			
Número		Descripción	
1		Implementar los test unitarios de los métodos de la clase Checkpoint113	
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público Estático		prepareTest	
Público		toMeasureTest	
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
Privado	Estático, final	String	URLPrueba
Privado	Estático, final	Float	NUM_PALABRAS_TOTAL
Privado	Estático, final	Float	NUM_PALABRAS_TITULOS
Privado	Estático	Checkpoint113	checkpoint113

Tabla 8.16 – Descripción detallada de la clase Checkpoint113Test

8.4.2.17 MetricIVORYTest

Nombre	Tipo	Descripción	Hereda de...
MetricIVORYTest		Implementa los test unitarios para la clase MetricIVORY	
<u>Responsabilidades</u>			
Número		Descripción	
1		Implementar los test unitarios para los métodos de MetricIVORY	
<u>Métodos</u>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público Estático		prepareTest	
Público		calculateTest	
<u>Atributos</u>			
Acceso	Modo	Tipo o Clase	Nombre
Privado	Estático	MetricIVORY	metricIVORY

Tabla 8.17 – Descripción detallada de la clase MetricIVORYTest

8.4.2.18 CheckpointModel

Nombre	Tipo	Descripción	Hereda de...
CheckpointModel	Abstracta	Define un checkpoint de una métrica de Atenea 3.0	
<i>Responsabilidades</i>			
Número	Descripción		
1	Definir un checkpoint para una métrica incluida en Atenea3.0.		
<i>Métodos</i>			
Acceso Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Público	Guide	getGuide	
Público		setGuide	Guide guide
Público	int	getPriority	
Público		setPriority	int priority
Público		setName	String name
Público		setDescription	String description
Público	String	getDescription	
Público	String	getName	
Público	String	getNormative	
Público Abstracto	Measure	toMeasure	Document d
Público	String	toConsult	Document document
Público	Measure	toMeasureManually	String result
<i>Atributos</i>			
Acceso	Modo	Tipo o Clase	Nombre
Público	Estático	boolean	unioviNet
Protegido		String	name
Protegido		String	description
Protegido		Guide	guide
Protegido		int	priority
Protegido	Estático, final	int	PRIORITY_ONE
Protegido	Estático, final	int	PRIORITY_TWO
Protegido	Estático, final	int	PRIORITY_THREE
Privado		String	normative
Observaciones			
Esta clase ha sido desarrollada por Javier Oyón para la herramienta Atenea3.0. Se ha incluido en la descripción detallada de las clases porque se considera relevante para entender el funcionamiento de la librería Ivory, ya que la estructura de las clases Checkpoint109, Chekcpooint110, Checkpoint111, Checkpoint112 y Checkpoint113, así como su comportamiento, vienen determinados directamente por esta clase. Para más información sobre la herramienta Atenea3.0 consultar la referencia [Atenea12].			

Tabla 8.18 – Descripción detallada de la clase CheckpointModel

Capítulo 9. Desarrollo de las Pruebas

9.1 Pruebas Unitarias

En este apartado se muestran los resultados obtenidos al realizar las pruebas unitarias. Como ya se ha comentado en el diseño, para realizar las pruebas se ha utilizado JUnit, concretamente el plugin de JUnit4 para Eclipse.

Para la realización de las pruebas unitarias se ha creado una página web llamada “Más Cine” cuyo código se encuentra en el cd entregado (ver 15.1).

9.1.1 MetricIVORYTest

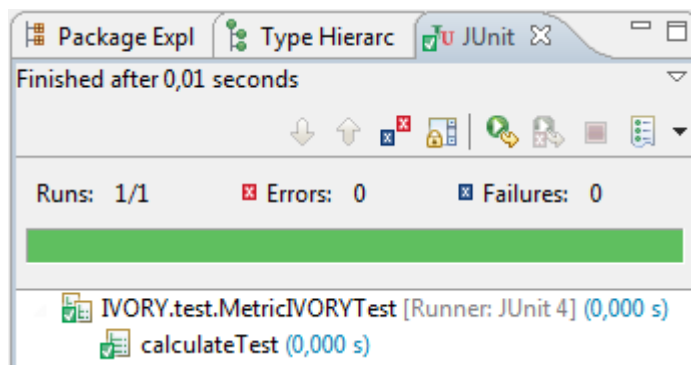


Ilustración 9.1 – Test unitarios de la clase MetricIVORY

9.1.2 Checkpoint109Test

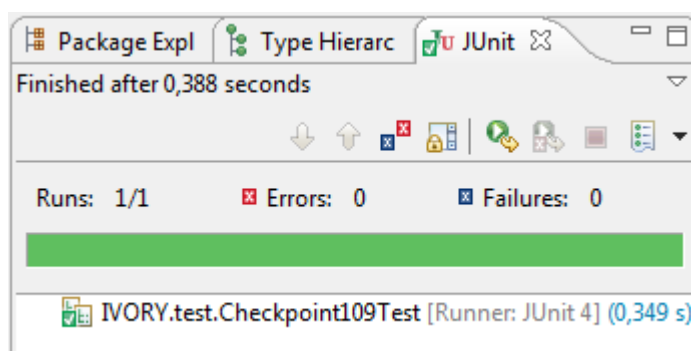


Ilustración 9.2 – Test unitarios de la clase Checkpoint109

9.1.3 Checkpoint110Test

La primera vez que se ejecutó este test el resultado fue fallido. Se hicieron las comprobaciones especificadas en la fase de diseño con los siguientes resultados:

Comprobación	Resultado
La medida se crea con el nombre correcto.	Correcto
Se crea el mapa de estilos.	Correcto
Se buscan las propiedades de estilo correctas.	Correcto
Se hace la búsqueda para todos los elementos de párrafo y encabezados.	INCORRECTO
El resultado obtenido se almacena en la medida con el nombre correcto.	Correcto

Tabla 9.1 – Test unitarios de la clase Checkpoint110: comprobaciones en caso de fallo

El problema encontrado consistía en que el sistema estaba contabilizando todos los tipos de elementos del HTML a los que afectaban las propiedades de estilo y no sólo a los párrafos y encabezados. Por ejemplo, imaginemos que el sistema analiza una página web con un fragmento de código como este:

```
<div id="bloque">
  <p>Párrafo de texto<p>
</div>
```

Y con una definición en su hoja de estilos como la siguiente:

```
#bloque{
  text-align: center;
}
```

El sistema incrementaría el contador al encontrarse con el elemento *div*, sobre el que recae la propiedad *text-align: center* y lo incrementaría de nuevo al encontrar el párrafo, sobre el que también recae la misma propiedad de estilo. De esta forma, el resultado del contador de bloques de texto centrado sería dos en lugar de uno.

Para solucionar el problema se ha restringido el algoritmo para que solo se tengan en cuenta los elementos marcados con las etiquetas de párrafo o encabezados.

Una vez subsanado el problema, el test pasa correctamente.

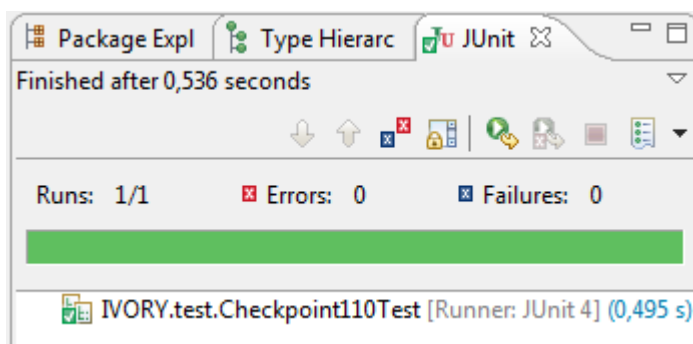


Ilustración 9.3 – Test unitarios de la clase Checkpoint110

9.1.4 Checkpoint111Test

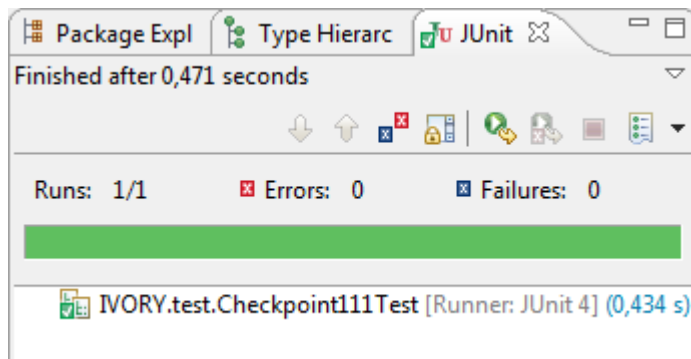


Ilustración 9.4 – Test unitarios de la clase Checkpoint111

9.1.5 Checkpoint112Test

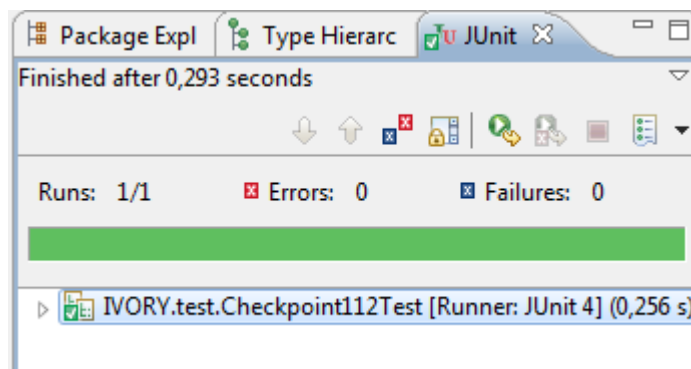


Ilustración 9.5 – Test unitarios de la clase Checkpoint112

9.1.6 Checkpoint113Test

La primera vez que se realizó este test el resultado fue fallido, por lo que se pasaron a realizar las comprobaciones definidas en la fase de diseño obteniendo los siguientes resultados:

Comprobación	Resultado
La medida se crea con el nombre correcto.	Correcto
Se crea el mapa de estilos.	Correcto
Se obtiene correctamente el texto de los encabezados.	Correcto
Se obtiene correctamente el texto del cuerpo.	Correcto
Se tienen en cuenta todos los niveles de encabezado que pueden existir.	Correcto
El resultado se almacena en la medida con el nombre correcto.	Correcto
El cálculo del porcentaje se realiza correctamente	INCORRECTO

Tabla 9.2 – Test unitarios de la clase Checkpoint113: comprobaciones en caso de fallo

El problema residía en que se utilizaban variables de tipo int para almacenar los contadores y calcular el resultado final, que una vez calculado se pasaba a Float. Al hacer esta operación se perdía precisión.

Una vez subsanado el problema, el test pasa correctamente.

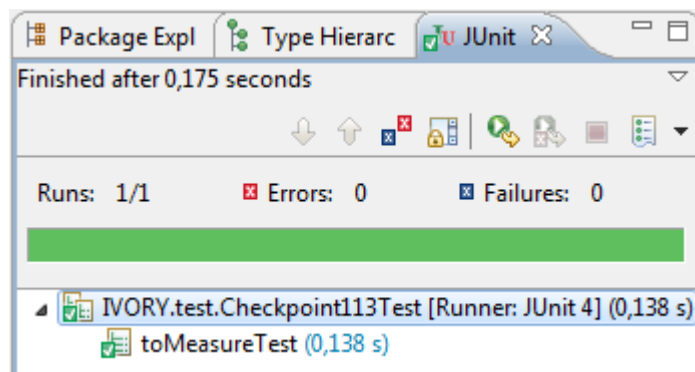


Ilustración 9.6 – Test unitarios de la clase Checkpoint113

9.2 Pruebas de Integración y del Sistema

Una vez ejecutadas las pruebas de integración definidas en el diseño se ha obtenido los resultados recogidos en la siguiente tabla.

Prueba	Resultado esperado	Resultado obtenido
Seleccionar Ivory como métrica para realizar un análisis a través de Atenea 3.0	Entre las métricas disponibles para realizar el análisis debe aparecer la métrica Ivory y poder ser seleccionada	Efectivamente, la métrica Ivory se encuentra entre las disponibles para realizar un análisis.
Realizar un análisis de una página web con la métrica Ivory desde Atenea 3.0	Se muestra el resultado del análisis en una tabla.	El análisis se realiza correctamente y una vez finalizado se muestra la página de resultados con el valor obtenido.

Tabla 9.3 – Resultado de las pruebas de integración

Capítulo 10. Estudios empíricos aplicando la métrica de Ivory

10.1 Análisis del nivel de usabilidad en España

Con el fin de conocer el nivel de usabilidad que tienen las páginas web en España según la métrica propuesta por Ivory, se ha tomado una muestra de webs de ayuntamientos, universidades y empresas y se han evaluado con dicha métrica utilizando la herramienta desarrollada.

Se han evaluado un total de 125 sitios web (32 ayuntamientos, 27 empresas y 67 universidades). De éstos, en 107 casos se han obtenido valores cercanos a 0 y 1, tal como enuncia Ivory. En cuanto al resto de casos, en la mayoría de ellos se han obtenido valores que se van alejando paulatinamente de estos extremos y los menos se alejan mucho del resto de las medidas tomadas. En términos estadísticos, estos valores son conocidos como “outliers”; en nuestro caso, pueden tratarse de sitios web que tengan características muy diferentes al grueso de los sitios analizados.

Para establecer una clasificación de los niveles de usabilidad de las páginas analizadas dividimos el intervalo 0..1, en el que se encuentran la mayor parte de los valores obtenidos en nuestro análisis, en 3 partes. Teniendo en cuenta que según lo enunciado por Ivory, los valores cercanos a 0 indican un pobre nivel de usabilidad y los cercanos a 1, un buen nivel, consideramos los siguientes valores:

Nivel malo	Resultado < 0,33
Nivel medio	0,33 <= Resultado <= 0,66
Nivel bueno	Resultado > 0,66

Tabla 10.1 – Niveles de usabilidad

En los siguientes apartados se recogen los resultados obtenidos y el análisis de los mismos.

10.1.1 La usabilidad en los ayuntamientos

Con el fin de evaluar la usabilidad en las páginas web de los ayuntamientos españoles se ha tomado una muestra de las páginas web de 32 ayuntamientos de capitales de provincia y se han evaluado con la métrica Ivory. El análisis se ha realizado con un nivel de profundidad 3 y un máximo de 75 páginas evaluadas por cada sitio web. Los resultados obtenidos se recogen en la siguiente tabla.

Nº	Nombre	Web	IVORY
1	Ayto. de Albacete	www.albacete.es	1,12773000
2	Ayto. de Alicante	www.alicante.es	0,16180500
3	Ayto. de Almería	www.ayotalmeria.es	-0,04987170

4	Ayto. de Avila	www.avila.es	0,32586700
5	Ayto. de Badajoz	www.aytobadajoz.es	0,90153000
6	Ayto. de Bilbao	www.bilbao.net	0,37682200
7	Ayto. de Burgos	www.aytoburgos.es	1,00970000
8	Ayto. de Castellón	www.castello.es	1,74693000
9	Ayto. de Ciudad Real	www.ciudadreal.es	-3,30502000
10	Ayto. de Córdoba	www.ayuncordoba.es	0,54560900
11	Ayto. de Granada	www.granada.org	0,52878900
12	Ayto. de Guadalajara	www.guadalajara.es	-1,47181000
13	Ayto. de Huelva	www.huelva.es/wps/portal	0,20508400
14	Ayto. de Huesca	www.huesca.es	0,65058700
15	Ayto. de Jaén	www.aytojaen.es	0,19287900
16	Ayto. de La Coruña	www.coruna.es	0,36514000
17	Ayto. de Lérida	www.paeria.es/cas/ajuntament/	0,64458600
18	Ayto. de Lugo	www.lugo.es	0,12923100
19	Ayto. de Madrid	www.madrid.es	0,13095400
20	Ayto. de Orense	www.ourense.es	0,85857900
21	Ayto. de Oviedo	www.oviedo.es	0,51403900
22	Ayto. de Palencia	www.aytopalencia.es	0,73977400
23	Ayto. de Pamplona	www.pamplona.es	0,74652600
24	Ayto. de San Sebastián	www.donostia.org	0,69181000
25	Ayto. de Santander	http://portal.ayto-santander.es/portal/page/portal/inet_santander	0,68210200
26	Ayto. de Segovia	www.segovia.es	0,61126600
27	Ayto. de Sevilla	www.sevilla.org	0,96440900
28	Ayto. de Soria	www.soria.es	0,22329900
29	Ayto. de Tenerife	www.tenerife.es	0,45088100
30	Ayto. de Valencia	www.valencia.es	2,83010000
31	Ayto. de Valladolid	www.valladolid.es	0,67333000
32	Ayto. de Zaragoza	www.zaragoza.es	-1,32648000

Tabla 10.2 – Resultados de la evaluación de la usabilidad con la métrica Ivory en páginas de ayuntamientos

En el siguiente gráfico de nube de puntos se ve cómo la mayor parte de las medidas están en la franja entre 0 y 1.

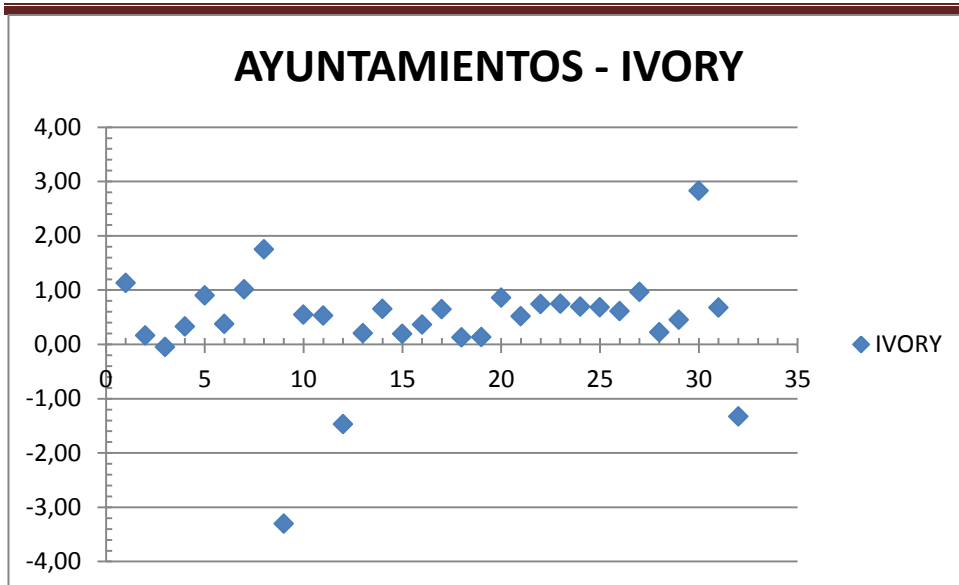


Gráfico 10.1 - Evaluación de usabilidad en ayuntamientos: nube de puntos

Al clasificar estos resultados utilizando los niveles de usabilidad especificados en la Tabla 10.1 obtenemos que el 34,38% de las páginas analizadas tiene un nivel de usabilidad malo; el 28,13% tiene un nivel de usabilidad medio y el 37,59% tiene un nivel de usabilidad bueno. Esta situación se ilustra en el siguiente gráfico, que muestra la cantidad de páginas web analizadas que pertenece a cada nivel.

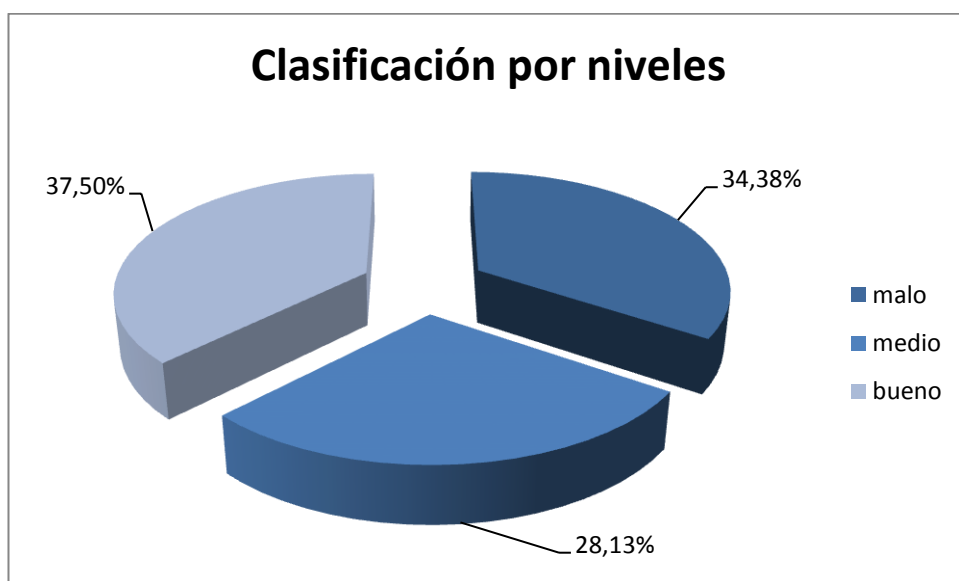


Gráfico 10.2 – Páginas web de ayuntamientos clasificadas por nivel de usabilidad

Como se puede ver en el gráfico, la mayor parte de las páginas analizadas tienen un nivel de usabilidad bueno, pero los porcentajes son muy parecidos en los tres casos. El nivel medio de usabilidad de las páginas web de ayuntamientos españoles es 0,40238051, que se corresponde con un nivel de usabilidad medio.

10.1.2 La usabilidad en las universidades

Para evaluar el nivel de usabilidad existente en las páginas web de las universidades españolas se ha estudiado una muestra de 67 páginas web de diferentes universidades, tanto públicas como privadas. Estas páginas se han evaluado con la métrica de Ivory utilizando la herramienta desarrollada. El análisis se ha realizado estableciendo un máximo de 75 páginas a analizar por sitio web y una profundidad máxima de tres niveles. La siguiente tabla recoge los datos obtenidos.

Nº	Universidad	Página web	IVORY
1	IE University	www.ie.edu	1,29959000
2	UNED	www.uned.es	0,12953800
3	Universidad a Distancia de Madrid	www.udima.es	-0,65236600
4	Universidad Abat Oliva CEU	www.uaoceu.es	0,84627400
5	Universidad Alfonso X el Sabio	m.uax.es	0,37513200
6	Universidad Camilo José Cela	www.ucjc.edu	1,42679000
7	Universidad Carlos III de Madrid	www.uc3m.es	0,60779300
8	Universidad Católica de Ávila	www.ucavila.es	1,31974000
9	Universidad Católica de Valencia	www.ucv.es	0,87153700
10	Universidad Católica San Antonio	www.ucam.edu	0,64280800
11	Universidad CEU Cardenal Herrera	www.uchceu.es	-0,99147200
12	Universidad CEU San Pablo	www.uspceu.com	0,12904900
13	Universidad Complutense	www.ucm.es	0,62699000
14	Universidad de Alcalá	www.uah.es	0,73149200
15	Universidad de Alicante	www.ua.es	0,55152800
16	Universidad de Almería	www.ual.es	1,00868000
17	Universidad de Barcelona	www.ub.edu	0,68992000
18	Universidad de Burgos	www.ubu.es	0,97284100
19	Universidad de Cádiz	www.uca.es	-1,61361000
20	Universidad de Cantabria	www.unican.es	0,70927000
21	Universidad de Castilla La Mancha	www.uclm.es	0,62530900
22	Universidad de Córdoba	www.uco.es	0,52079900
23	Universidad de Deusto	www.deusto.es	-0,30301800
24	Universidad de Extremadura	www.unex.es	0,49091800
25	Universidad de Gerona	www.udg.edu	0,63141800
26	Universidad de Granada	www.ugr.es	0,77886100
27	Universidad de Huelva	www.uhu.es	0,12926100
28	Universidad de La Coruña	www.udc.es	0,60071300
29	Universidad de La Laguna	www.ull.es	-0,25054400
30	Universidad de La Rioja	www.unirioja.es	0,51882000
31	Universidad de Las Islas Baleares	www.uib.es	0,58932800
32	Universidad de Las Palmas de Gran Canaria	www.ulpgc.es	0,93142800
33	Universidad de Lleida	www.udl.es	0,53985900
34	Universidad de Málaga	www.uma.es	0,61052500
35	Universidad de Mondragón	www.mondragon.edu/es	0,53295800
36	Universidad de Murcia	www.um.es	0,70789200

37	Universidad de Navarra	www.unav.edu	0,82806000
38	Universidad de Nebrija	www.nebrija.com	0,12415700
39	Universidad de Oviedo	www.uniovi.es	0,90767200
40	Universidad de Pamplona	www.unipamplona.edu.co	0,52575300
41	Universidad de Salamanca	www.usal.es	0,55466600
42	Universidad de Santiago de Compostela	www.usc.es	0,42246500
43	Universidad de Sevilla	www.us.es	0,49508200
44	Universidad de Valencia	www.uv.es	-1,68413000
45	Universidad de Valladolid	www.uva.es	1,36518000
46	Universidad de Vigo	www.uvigo.es	0,63020700
47	Universidad de Zaragoza	www.unizar.es	0,63507700
48	Universidad Europea de Madrid	www.uem.es	0,66901000
49	Universidad Francisco de Vitoria	www.ufv.es	0,39832400
50	Universidad Internacional de Andalucía	www.unia.es	0,58014100
51	Universidad Internacional de Cataluña	www.unica.edu	0,42351700
52	Universidad Internacional de La Rioja	www.unir.net	0,91253700
53	Universidad Miguel Hernández	www.umh.es	8,12911000
54	Universidad Oberta de Cataluña	www.uoc.edu	0,67623800
55	Universidad Pablo de Olavide	www.upo.es	1,42848000
56	Universidad Politécnica de Valencia	www.upv.es	0,87174700
57	Universidad Politécnica de Cartagena	www.upct.es	-0,90028100
58	Universidad Politécnica de Cataluña	www.upc.edu	0,33566600
59	Universidad Politécnica de Madrid	www.upm.es	0,83383000
60	Universidad Politécnica de Valencia	www.upv.es	0,87816000
61	Universidad Pompeu Fabra	www.upf.edu/es	0,63419400
62	Universidad Pontificia de Comillas	www.upcomillas.es	0,56466000
63	Universidad Pontificia de Salamanca	www.upsa.es	1,11619000
64	Universidad Ramón Llul	www.url.es	0,74628200
65	Universidad Rey Juan Carlos	www.urjc.es	0,49374000
66	Universidad Rovira y Virgili	www.urv.cat	0,56957400
67	Universidad San Jorge	www.usj.es	2,58504000

Tabla 10.3 – Resultados de la evaluación de la usabilidad con la métrica Ivory en páginas de universidades

Si calculamos la puntuación media de usabilidad de las páginas analizadas obtenemos un valor de 0,65800596, lo que se corresponde con un nivel de usabilidad medio-alto según la clasificación descrita en la Tabla 10.1. Este hecho se ilustra gráficamente en el siguiente gráfico que muestra todas las valoraciones de las páginas. En él se puede observar cómo la mayor parte de las puntuaciones tienen un valor comprendido entre cero y uno.

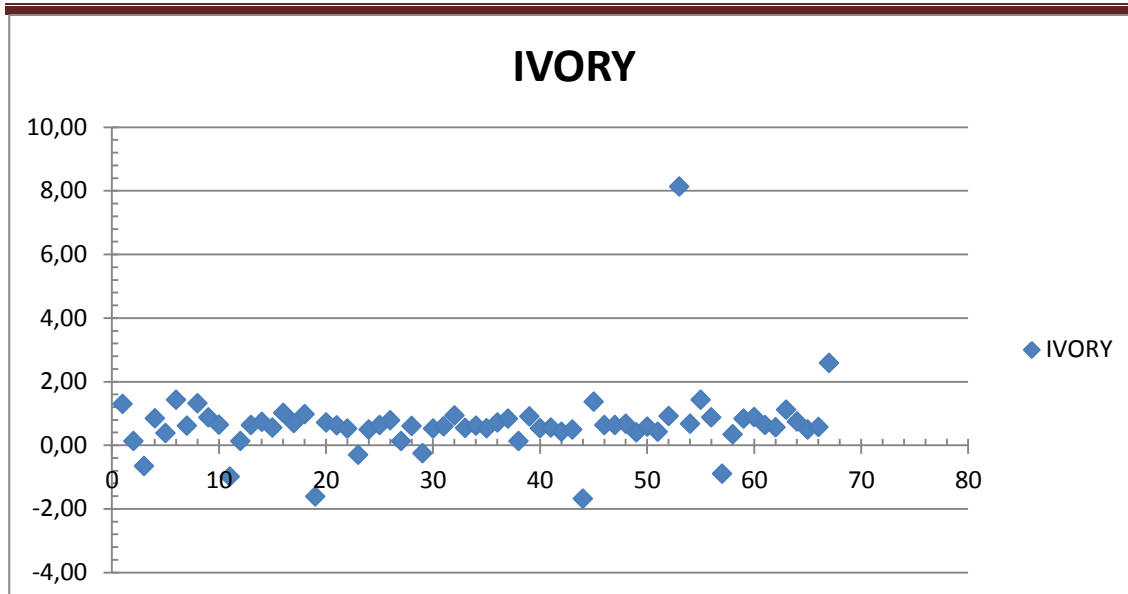


Gráfico 10.3 – Evaluación de la usabilidad en universidades: nube de puntos

Si clasificamos las páginas analizadas en buenas, malas y medias atendiendo a la puntuación obtenida con la métrica de Ivory, observamos que el 16,42% tienen un nivel de usabilidad malo, el 43,28% un nivel intermedio y el 40,30% tienen un nivel de usabilidad bueno. En el siguiente gráfico de sectores observamos como más de tres cuartas partes de las páginas de la muestra analizada tienen un nivel de usabilidad medio o bueno.

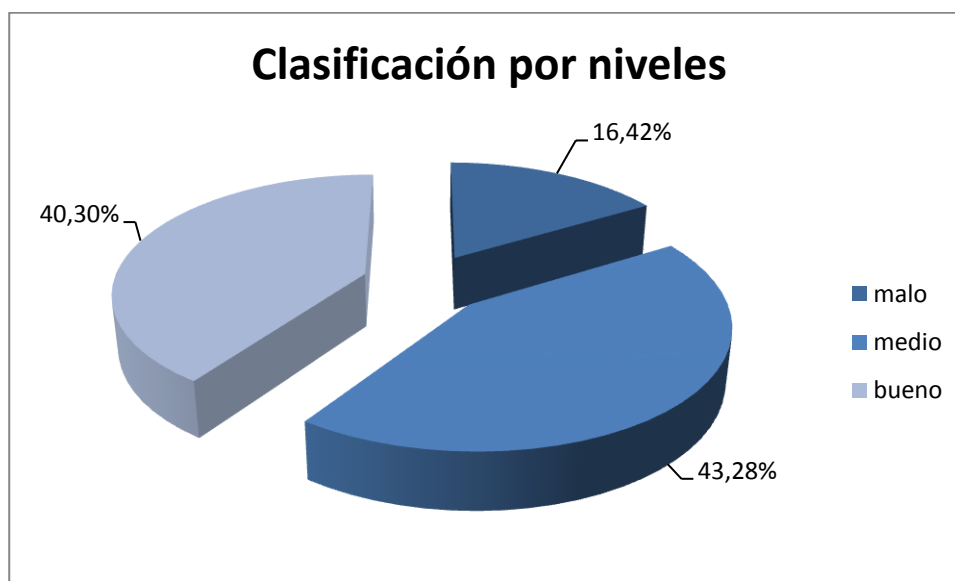


Gráfico 10.4 . Páginas web de universidades clasificadas por nivel de usabilidad

10.1.3 La usabilidad en las empresas

Se ha evaluado una muestra de 27 de las empresas que cotizan en el IBEX35 en julio de 2013 con la métrica de Ivory para analizar qué nivel de usabilidad poseen. El análisis se ha realizado con una profundidad máxima de tres niveles y una limitación de 75 páginas a analizar por sitio web. Los resultados obtenidos se recogen en la siguiente tabla.

Nº	Empresa	Web	IVORY
1	Abertis	www.abertis.com	-2,20311000
2	Acciona	www.acciona.es	0,63499200
3	Acerinox	www.acerinox.com	0,46695500
4	Amadeus	www.amadeus.net	0,12960700
5	Arcelor Mittal	www.arcelormittal.com	1,18789000
6	Banco Popular	www.bancopopular.es	0,55744900
7	Banco Sabadell	www.bancsabadell.com	0,12900100
8	Banco Santander	www.bancosantander.es	0,40162900
9	Bankinter	www.bankinter.com	0,12900100
10	BME	www.bolsasymercados.es	0,32976700
11	Caixabank	www.caixabank.com	-1,07726000
12	DIA	www.dia.es	-0,35929200
13	Ebro Foods	www.ebrofoods.es	0,43106300
14	Enagas	www.enagas.es	0,70096070
15	Endesa	www.endesa.com	0,93440500
16	Ferrovial	www.ferrovial.com	0,12984700
17	Grifols	www.grifols.com	0,64481000
18	Grupo ACS	www.grupoacs.com	0,44838500
19	Iberdrola	www.iberdrola.es	0,73294700
20	Inditex	www.inditex.es	0,38696900
21	Indra	www.indracompany.com	0,59969200
22	Jazztel	www.jazztel.com	0,89186200
23	Mapfre	www.mapfre.com	0,64037900
24	REE	www.ree.es	0,42617300
25	Repsol	www.repsol.com	-0,67150600
26	Telefónica	www.telefonica.es	0,35270600
27	Viscofan	www.viscofan.com	0,19787100

Tabla 10.4 - Resultados de la evaluación de la usabilidad con la métrica Ivory en páginas de empresas

Si calculamos la puntuación media de usabilidad de estas páginas obtenemos un valor de 0,26567380, que se corresponde con un nivel de usabilidad malo, atendiendo a la clasificación por niveles especificada en la Tabla 10.1. En el siguiente gráfico se observa como la mayor parte de los valores se encuentran comprendidos entre 0 y 0,5.

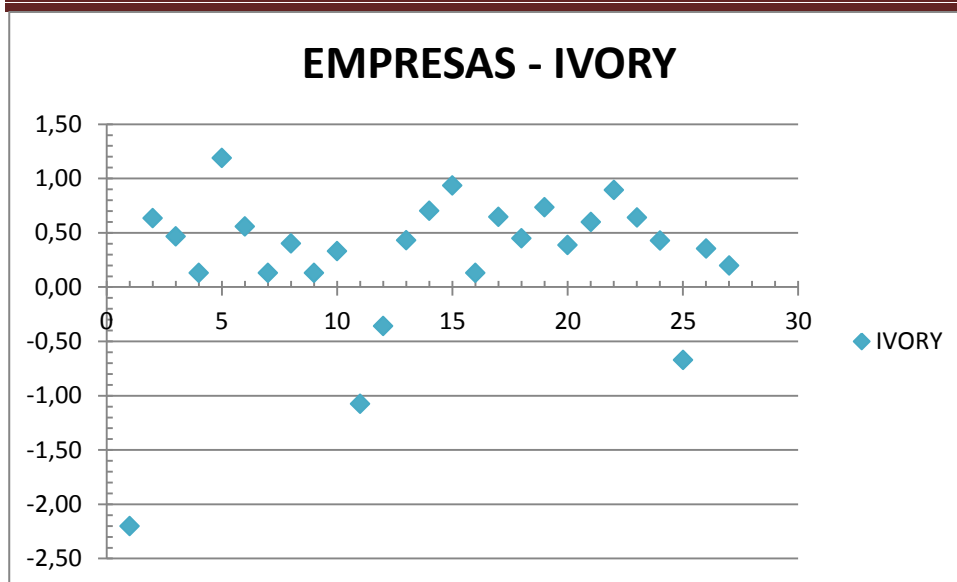


Gráfico 10.5 – Evaluación de la usabilidad en empresas: nube de puntos

Clasificando las páginas web por su nivel de usabilidad según la puntuación obtenida al ser evaluadas con la métrica de Ivory, obtenemos que un 37,04% tienen un nivel de usabilidad malo, un 44,44% tienen un nivel de usabilidad medio y un 18,52% tienen un nivel de usabilidad bueno.

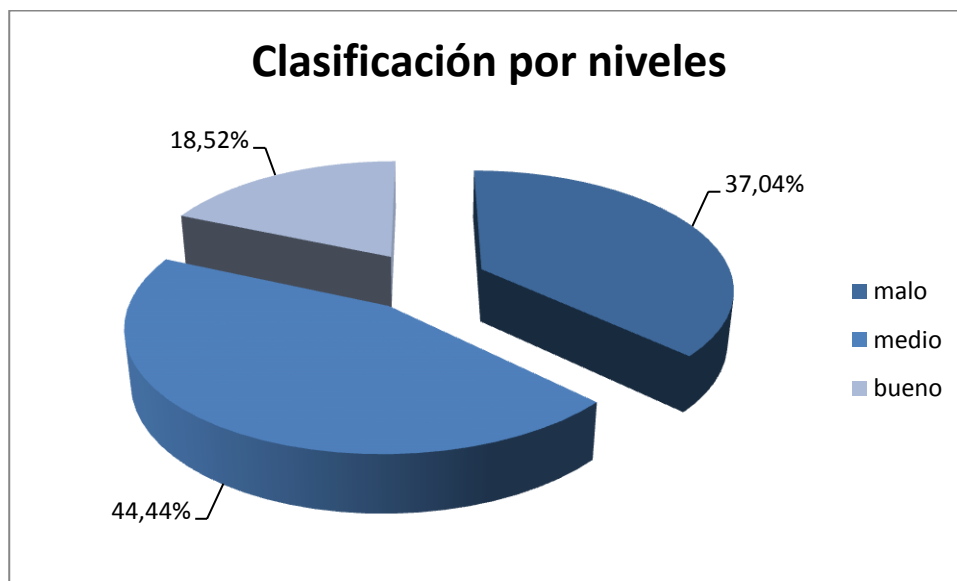


Gráfico 10.6 – Páginas web de empresas clasificadas por su nivel de usabilidad

10.1.4 Comparación del nivel de usabilidad en webs de ayuntamientos, empresas y universidades

Atendiendo a los resultados obtenidos tras evaluar la usabilidad de una muestra de 67 páginas web de universidades españolas, 32 ayuntamientos de capitales de provincia y 27 de las empresas que cotizan actualmente en el IBEX35 con la métrica de Ivory se realiza una serie de observaciones que se pasan a describir a continuación.

Las universidades son las entidades que presentan un mejor nivel de usabilidad en sus páginas web dado que el 40,30% de las páginas analizadas tienen un nivel de usabilidad bueno y el 43,28% tienen un nivel medio, lo que supone que más del 80% sean aceptables en cuanto a la usabilidad. En contraposición se encuentran las empresas, quienes presentan solamente un 18,52% de páginas web con un buen nivel de usabilidad, tienen un porcentaje similar al de las universidades en cuanto a páginas web con un nivel medio de usabilidad y más de un tercio tienen un nivel de usabilidad malo.

En cuanto a los ayuntamientos, la mayoría de las páginas analizadas tiene un nivel de usabilidad bueno, pero los porcentajes son bastante similares para los tres niveles.

Estos datos se representan en el siguiente gráfico.

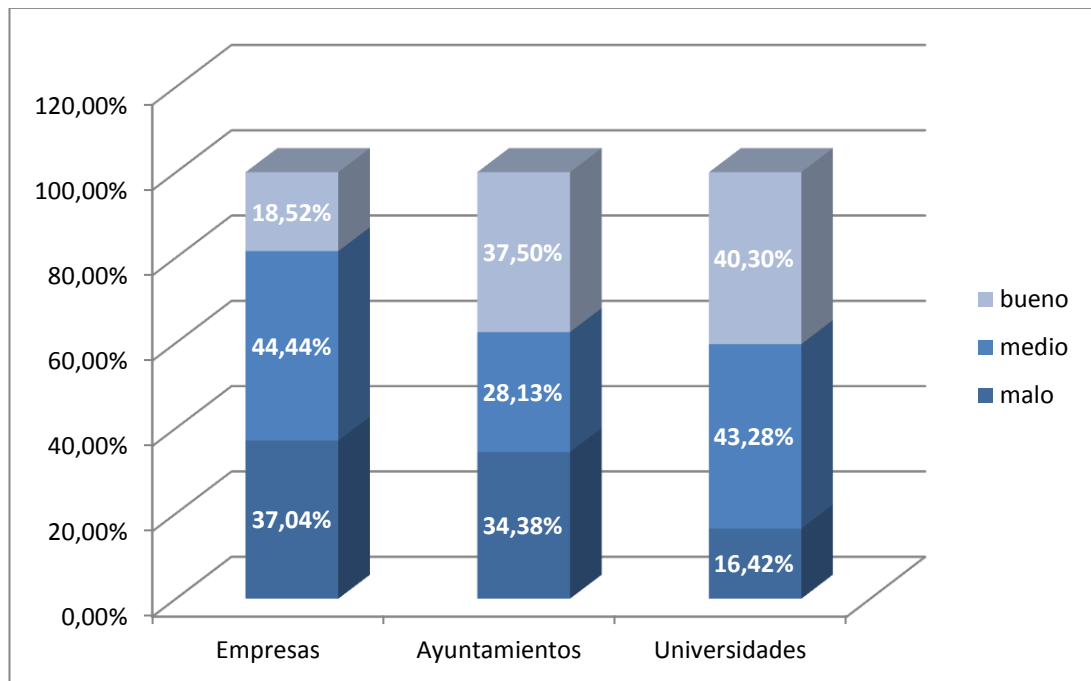


Gráfico 10.7 – Comparación de niveles de usabilidad en páginas web de ayuntamientos, empresas y universidades

Si comparamos las puntuaciones medias de usabilidad calculados para las páginas de los ayuntamientos, universidades y empresas vemos que son las universidades las que presentan una mayor puntuación media con un 0,65800596, que se corresponde con un nivel de usabilidad medio-alto.

A continuación, se sitúan las páginas web de los ayuntamientos, con una puntuación media de 0,40238051, lo que les sitúa también en un nivel de usabilidad medio, pero significativamente más bajo que en el caso de las universidades.

Finalmente, las empresas obtienen una valoración media de 0,265567380, que les confiere un nivel de usabilidad malo.

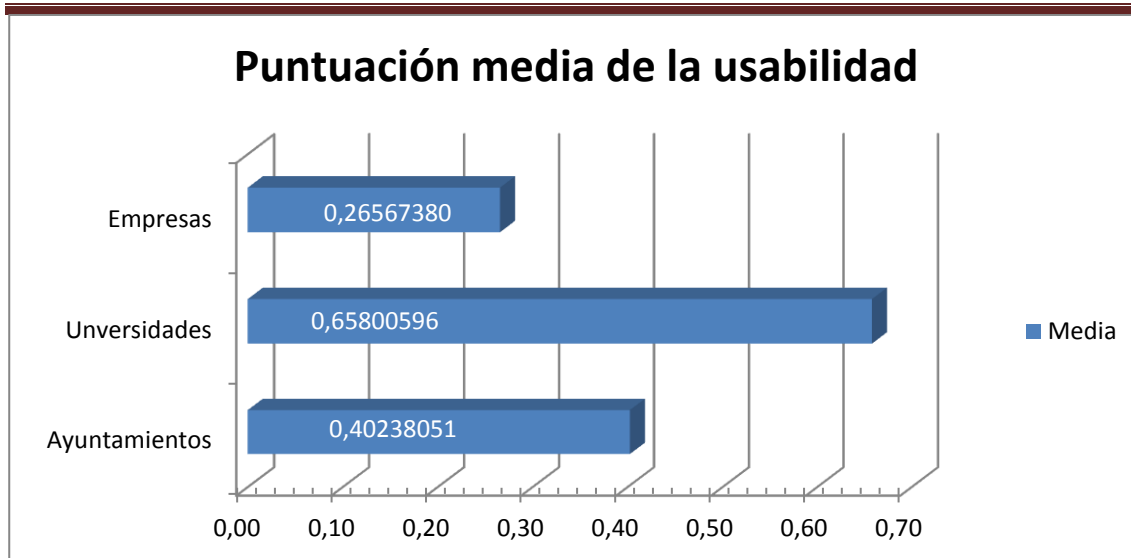


Gráfico 10.8 – Comparación de la puntuación media de usabilidad en ayuntamientos, universidades y empresas

En cuanto a la situación global, la mayor parte de las páginas web analizadas presentan un nivel de usabilidad medio y, aproximadamente una cuarta parte, tienen un nivel de usabilidad malo. El siguiente gráfico muestra los porcentajes de páginas con niveles de usabilidad buenos, medios y malos del total de las páginas analizadas de ayuntamientos, universidades y empresas.

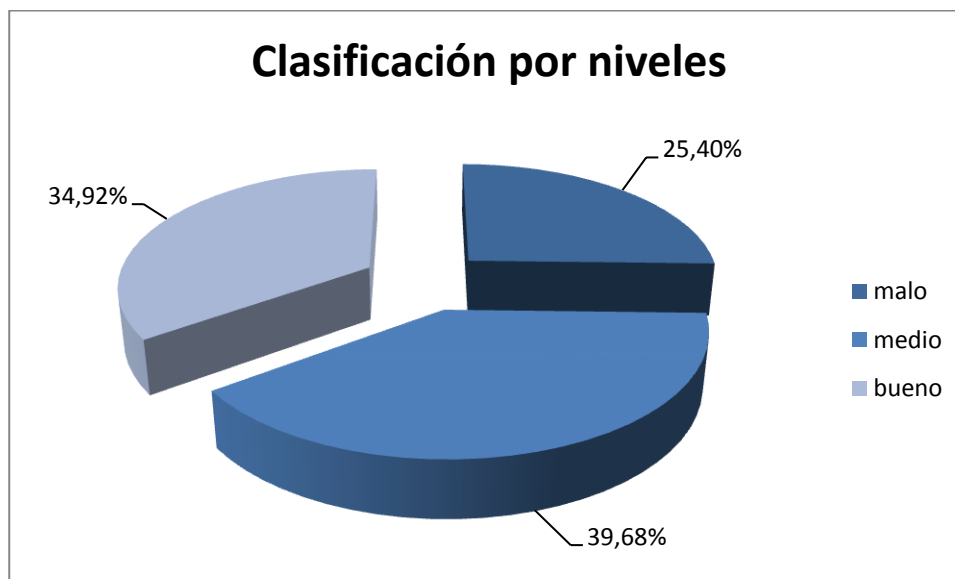


Gráfico 10.9 – Clasificación por nivel de usabilidad del total de las páginas analizadas de ayuntamientos, universidades y empresas

10.2 Relación entre la usabilidad y la accesibilidad

Dado que varios autores sostienen que existe una relación directa entre los niveles de accesibilidad y usabilidad [Medina10] [Petrie07] [Takay07], mediante este estudio se trata de determinar si con los valores proporcionados por Ivory y una métrica estándar de medición de la accesibilidad puede determinarse esta relación.

Como métrica de evaluación de la accesibilidad se utilizará WAB (Web Accessibility Barrier) [Parmant05]. Esta métrica puntúa las páginas web en cuanto a su accesibilidad en base a la evaluación de 25 de los puntos del conjunto de pautas WCAG1.0. Si se obtiene un valor igual a cero, el nivel de accesibilidad del sitio evaluado es aceptable; si el valor es menor o igual que 5,5, el sitio presenta pocas barreras de accesibilidad; y, si el valor WAB es mayor que 5,5, el sitio presenta muchas barreras de accesibilidad. Esta métrica ya se encuentra incluida en la herramienta Atenea 3.0, por lo que la utilizaremos para realizar la evaluación de la accesibilidad de los mismos sitios web de ayuntamientos, universidades y empresas a los que se ha evaluado la usabilidad con la métrica de Ivory.

En las siguientes tablas se muestran las puntuaciones de accesibilidad con la métrica WAB que reciben las páginas web de las muestras de ayuntamientos, universidades y empresas utilizadas para el estudio de la usabilidad en España (ver punto 10.1). Para mayor comodidad, también se incluye la puntuación de usabilidad que ha obtenido cada sitio al ser analizado con la métrica de Ivory.

Nº	Nombre	Web	IVORY	WAB
1	Ayto. de Albacete	www.albacete.es	1,12773000	2,64125000
2	Ayto. de Alicante	www.alicante.es	0,16180500	3,84890000
3	Ayto. de Almería	www.ayotalmeria.es	-0,04987170	9,19819000
4	Ayto. de Ávila	www.avila.es	0,32586700	5,14930000
5	Ayto. de Badajoz	www.aytobadajoz.es	0,90153000	0,90154300
6	Ayto. de Bilbao	www.bilbao.net	0,37682200	0,66666700
7	Ayto. de Burgos	www.aytoburgos.es	1,00970000	4,33022000
8	Ayto. de Castellón	www.castello.es	1,74693000	1,35326000
9	Ayto. de Ciudad Real	www.ciudadreal.es	-3,30502000	
10	Ayto. de Córdoba	www.ayuncordoba.es	0,54560900	5,60485000
11	Ayto. de Granada	www.granada.org	0,52878900	5,26441000
12	Ayto. de Guadalajara	www.guadalajara.es	-1,47181000	2,76239000
13	Ayto. de Huelva	www.huelva.es/wps/portal	0,20508400	6,31024000
14	Ayto. de Huesca	www.huesca.es	0,65058700	3,34818000
15	Ayto. de Jaén	www.aytojaen.es	0,19287900	4,00000000
16	Ayto. de La Coruña	www.coruna.es	0,36514000	2,14319000
17	Ayto. de Lérida	www.paeria.es/cas/ajuntament/	0,64458600	5,52685000
18	Ayto. de Lugo	www.lugo.es	0,12923100	7,00000000
19	Ayto. de Madrid	www.madrid.es	0,13095400	3,74840000
20	Ayto. de Orense	www.ourense.es	0,85857900	2,89218000
21	Ayto. de Oviedo	www.oviedo.es	0,51403900	4,04741000

22	Ayto. de Palencia	www.aytopalencia.es	0,73977400	0,88138100
23	Ayto. de Pamplona	www.pamplona.es	0,74652600	2,00952000
24	Ayto. de San Sebastián	www.donostia.org	0,69181000	3,02945000
25	Ayto. de Santander	http://portal.ayto-santander.es/portal/page/portal/inet_santander	0,68210200	1,81548000
26	Ayto. de Segovia	www.segovia.es	0,61126600	2,59462000
27	Ayto. de Sevilla	www.sevilla.org	0,96440900	3,59355000
28	Ayto. de Soria	www.soria.es	0,22329900	3,69751000
29	Ayto. de Tenerife	www.tenerife.es	0,45088100	1,54666000
30	Ayto. de Valencia	www.valencia.es	2,83010000	0,44252600
31	Ayto. de Valladolid	www.valladolid.es	0,67333000	4,67593000
32	Ayto. de Zaragoza	www.zaragoza.es	-1,32648000	4,44834000

Tabla 10.5 – Evaluación de la accesibilidad con la métrica WAB y evaluación de la usabilidad con la métrica de Ivory en webs de ayuntamientos de España

Nº	Universidad	Página web	IVORY	WAB
1	IE University	www.ie.edu	1,29959000	4,53201000
2	UNED	www.uned.es	0,12953800	3,00000000
3	Universidad a Distancia de Madrid	www.udima.es	-0,65236600	7,96556000
4	Universidad Abat Oliva CEU	www.uaoceu.es	0,84627400	7,61445000
5	Universidad Alfonso X el Sabio	m.uax.es	0,37513200	3,57932000
6	Universidad Camilo José Cela	www.ucjc.edu	1,42679000	8,79205000
7	Universidad Carlos III de Madrid	www.uc3m.es	0,60779300	6,47945000
8	Universidad Católica de Ávila	www.ucavila.es	1,31974000	6,33154000
9	Universidad Católica de Valencia	www.ucv.es	0,87153700	3,97756000
10	Universidad Católica San Antonio	www.ucam.edu	0,64280800	4,34462000
11	Universidad CEU Cardenal Herrera	www.uchceu.es	-0,99147200	6,22520000
12	Universidad CEU San Pablo	www.uspceu.com	0,12904900	5,00000000
13	Universidad Complutense	www.ucm.es	0,62699000	3,97016000
14	Universidad de Alcalá	www.uah.es	0,73149200	1,24574000
15	Universidad de Alicante	www.ua.es	0,55152800	1,98156000
16	Universidad de Almería	www.ual.es	1,00868000	3,93618000
17	Universidad de Barcelona	www.ub.edu	0,68992000	3,50357000
18	Universidad de Burgos	www.ubu.es	0,97284100	4,89253000
19	Universidad de Cádiz	www.uca.es	-1,61361000	2,04305000
20	Universidad de Cantabria	www.unican.es	0,70927000	11,24710000
21	Universidad de Castilla La Mancha	www.uclm.es	0,62530900	0,48486500
22	Universidad de Córdoba	www.uco.es	0,52079900	6,19727000
23	Universidad de Deusto	www.deusto.es	-0,30301800	5,44180000
24	Universidad de Extremadura	www.unex.es	0,49091800	3,22280000

Estudios empíricos aplicando la métrica de Ivory | **Automatización de la medición de la usabilidad web mediante la métrica de Ivory**

25	Universidad de Gerona	www.udg.edu	0,63141800	2,55093000
26	Universidad de Granada	www.ugr.es	0,77886100	5,10773000
27	Universidad de Huelva	www.uhu.es	0,12926100	5,00000000
28	Universidad de La Coruña	www.udc.es	0,60071300	1,33208000
29	Universidad de La Laguna	www.ull.es	-0,25054400	3,75047000
30	Universidad de La Rioja	www.unirioja.es	0,51882000	9,40173000
31	Universidad de Las Islas Baleares	www.uib.es	0,58932800	1,58266000
32	Universidad de Las Palmas de Gran Canaria	www.ulpgc.es	0,93142800	5,50737000
33	Universidad de Lleida	www.udl.es	0,53985900	6,16427000
34	Universidad de Málaga	www.uma.es	0,61052500	4,88056000
35	Universidad de Mondragón	www.mondragon.edu/es	0,53295800	6,49603000
36	Universidad de Murcia	www.um.es	0,70789200	2,20774000
37	Universidad de Navarra	www.unav.edu	0,82806000	6,45159000
38	Universidad de Nebrija	www.nebrija.com	0,12415700	4,70840000
39	Universidad de Oviedo	www.uniovi.es	0,90767200	6,33913000
40	Universidad de Pamplona	www.unipamplona.edu.co	0,52575300	6,10777000
41	Universidad de Salamanca	www.usal.es	0,55466600	2,45403000
42	Universidad de Santiago de Compostela	www.usc.es	0,42246500	0,68444700
43	Universidad de Sevilla	www.us.es	0,49508200	4,05420000
44	Universidad de Valencia	www.uv.es	-1,68413000	3,60855000
45	Universidad de Valladolid	www.uva.es	1,36518000	2,11587000
46	Universidad de Vigo	www.uvigo.es	0,63020700	1,82096000
47	Universidad de Zaragoza	www.unizar.es	0,63507700	3,30052000
48	Universidad Europea de Madrid	www.uem.es	0,66901000	5,90593000
49	Universidad Francisco de Vitoria	www.ufv.es	0,39832400	6,04543000
50	Universidad Internacional de Andalucía	www.unia.es	0,58014100	5,50852000
51	Universidad Internacional de Cataluña	www.unica.edu	0,42351700	6,19976000
52	Universidad Internacional de La Rioja	www.unir.net	0,91253700	7,37346000
53	Universidad Miguel Hernández	www.umh.es	8,12911000	6,58988000
54	Universidad Oberta de Cataluña	www.uoc.edu	0,67623800	2,71406000
55	Universidad Pablo de Olavide	www.upo.es	1,42848000	2,13747000
56	Universidad Politécncia de Valencia	www.upv.es	0,87174700	2,72263000
57	Universidad Politécnica de Cartagena	www.upct.es	-0,90028100	5,36049000
58	Universidad Politécnica de Cataluña	www.upc.edu	0,33566600	2,62501000
59	Universidad Politécnica de Madrid	www.upm.es	0,83383000	4,46050000

60	Universidad Politécnica de Valencia	www.upv.es	0,87816000	2,72235000
61	Universidad Pompeu Fabra	www.upf.edu/es	0,63419400	2,81784000
62	Universidad Pontificia de Comillas	www.upcomillas.es	0,56466000	6,90939000
63	Universidad Pontificia de Salamanca	www.upsa.es	1,11619000	4,65303000
64	Universidad Ramón Llul	www.url.es	0,74628200	6,19917000
65	Universidad Rey Juan Carlos	www.urjc.es	0,49374000	2,56437000
66	Universidad Rovira y Virgili	www.urv.cat	0,56957400	2,90515000
67	Universidad San Jorge	www.usj.es	2,58504000	3,08551000

Tabla 10.6 - Evaluación de la accesibilidad con la métrica WAB y evaluación de la usabilidad con la métrica de Ivory en webs de universidades de España

Nº	Empresa	Web	IVORY	WAB
1	Abertis	www.abertis.com	-2,20311000	1,56338000
2	Acciona	www.acciona.es	0,63499200	3,47805000
3	Acerinox	www.acerinox.com	0,46695500	8,90351000
4	Amadeus	www.amadeus.net	0,12960700	3,00000000
5	Arcelor Mittal	www.arcelormittal.com	1,18789000	7,43776000
6	Banco Popular	www.bancopopular.es	0,55744900	3,68933000
7	Banco Sabadell	www.bancsabadell.com	0,12900100	5,00000000
8	Banco Santander	www.bancosantander.es	0,40162900	2,50000000
9	Bankinter	www.bankinter.com	0,12900100	5,00000000
10	BME	www.bolsasymercados.es	0,32976700	1,00000000
11	Caixabank	www.caixabank.com	-1,07726000	9,89581000
12	DIA	www.dia.es	-0,35929200	4,39602000
13	Ebro Foods	www.ebrofoods.es	0,43106300	2,92034000
14	Enagas	www.enagas.es	0,70096070	0,65463100
15	Endesa	www.endesa.com	0,93440500	1,55348000
16	Ferrovial	www.ferrovial.com	0,12984700	5,00000000
17	Grifols	www.grifols.com	0,64481000	6,27328000
18	Grupo ACS	www.grupoacs.com	0,44838500	1,05402000
19	Iberdrola	www.iberdrola.es	0,73294700	4,02019000
20	Inditex	www.inditex.es	0,38696900	1,33397000
21	Indra	www.indracompany.com	0,59969200	1,54439000
22	Jazztel	www.jazztel.com	0,89186200	6,73902000
23	Mapfre	www.mapfre.com	0,64037900	3,49342000
24	REE	www.ree.es	0,42617300	4,15653000
25	Repsol	www.repsol.com	-0,67150600	4,86527000
26	Telefónica	www.telefonica.es	0,35270600	2,00000000
27	Viscofan	www.viscofan.com	0,19787100	0,00000000

Tabla 10.7 - Evaluación de la accesibilidad con la métrica WAB y evaluación de la usabilidad con la métrica de Ivory en webs de empresas de España

Para comprobar si existe o no correlación entre las puntuaciones de usabilidad obtenidas con la métrica Ivory y las de accesibilidad obtenidas con la métrica WAB, se ha realizado un análisis

de correlaciones calculando el coeficiente de Pearson y la Rho de Spearman. Para ello se ha utilizado el programa estadístico SPSS de IBM, concretamente la versión 19 del mismo.

El coeficiente de correlación de Pearson mide la relación lineal entre dos variables aleatorias cuantitativas. El valor de la correlación es igual a -1 ó 1 si la correlación es máxima. Cuanto más se acerque a cero, más pequeña será la correlación existente. [UV]

La Rho de Spearman mide la correlación entre dos variables aleatorias continuas. Al igual que el coeficiente de correlación de Pearson toma valores comprendidos en el intervalo [-1, 1] y se interpreta de la misma manera. [UCM]

El valor obtenido al calcular el coeficiente de correlación de Pearson para los valores estudiados es de 0,006, lo que indica que no existe correlación entre los valores medidos para la usabilidad y para la accesibilidad.

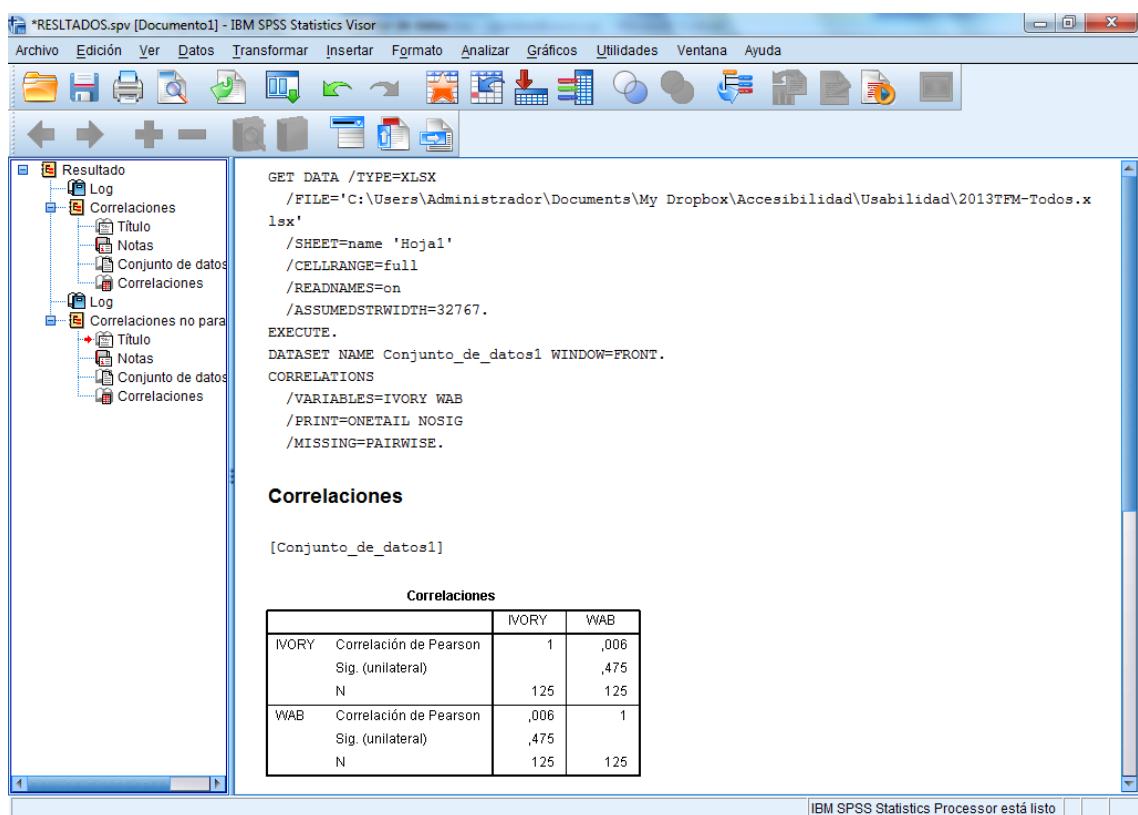


Ilustración 10.1 – Cálculo del coeficiente de correlación de Pearson con el paquete estadístico SPSS

En cuanto a la Rho de Spearman, se ha obtenido un valor de -0,050, lo que confirma la conclusión anterior: no existe correlación entre los valores de usabilidad obtenidos al aplicar la métrica de Ivory y los valores de accesibilidad obtenidos al aplicar la métrica WAB.

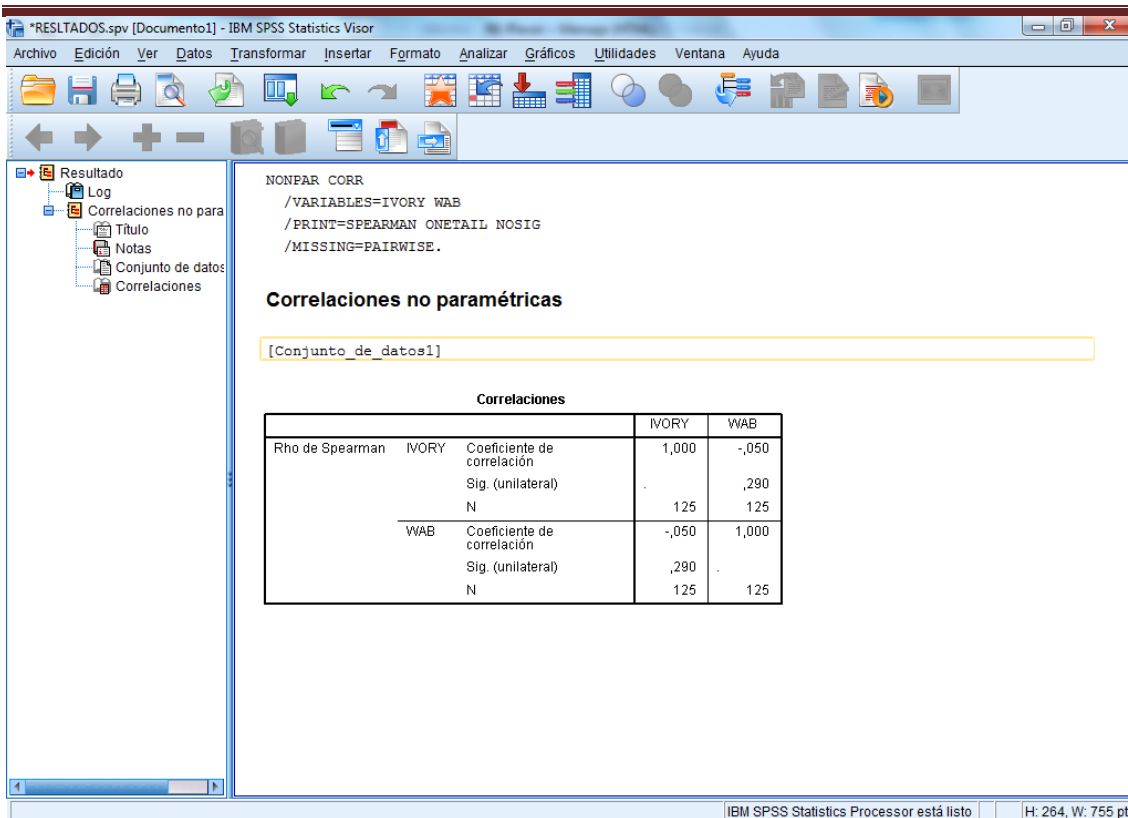


Ilustración 10.2 – Cálculo de la Rho de Spearman con el paquete estadístico SPSS

En conclusión, a partir de la valoración de la usabilidad de las páginas web obtenida al evaluarlas con la métrica de Ivory no se puede confirmar la existencia de una correlación entre la usabilidad y la accesibilidad en la web.

Capítulo 11. Manuales del Sistema

11.1 Manual de Instalación

11.1.1 Requisitos del sistema

Para poder instalar nuestra herramienta, el sistema debe disponer de los siguientes requisitos:

- Windows XP o superior
- Java6 o superior

11.1.2 Elementos necesarios para la instalación del sistema

Para llevar a cabo la instalación de la herramienta Atenea con la métrica Ivory integrada es necesario disponer de los archivos Atenea.zip, atenea.war y configuracionBBDD.sql.

El archivo Ateneena.zip debe contener las siguientes carpetas:

- classes: contiene los .class del proyecto Antenea.
- crawler_adapters: contiene la librería Adapter.jar.
- jars: contiene las librerías necesarias para integrar las diferentes métricas en Atenea.

El archivo atenea.war contiene la distribución de la aplicación web.

El archivo configuracionBBDD.sql es un script para construir el esquema de base de datos de MySQL sobre el que correrá la aplicación.

11.1.3 Instalación de AppServ

AppServ es una herramienta gratuita que permite instalar de una forma rápida y sencilla Apache, PHP y MySQL. En primer lugar descargamos la última versión de AppServ de su página oficial <http://www.appservnetwork.com/?newlang=spanish>.

Una vez descargado el archivo, lo ejecutamos y comenzamos la instalación:

1. Hacemos clic en “Next” para continuar con la instalación.



Ilustración 11.1 – Instalación de AppServ: paso 1.

2. Aceptamos el acuerdo de licencia haciendo clic en “I Agree”.

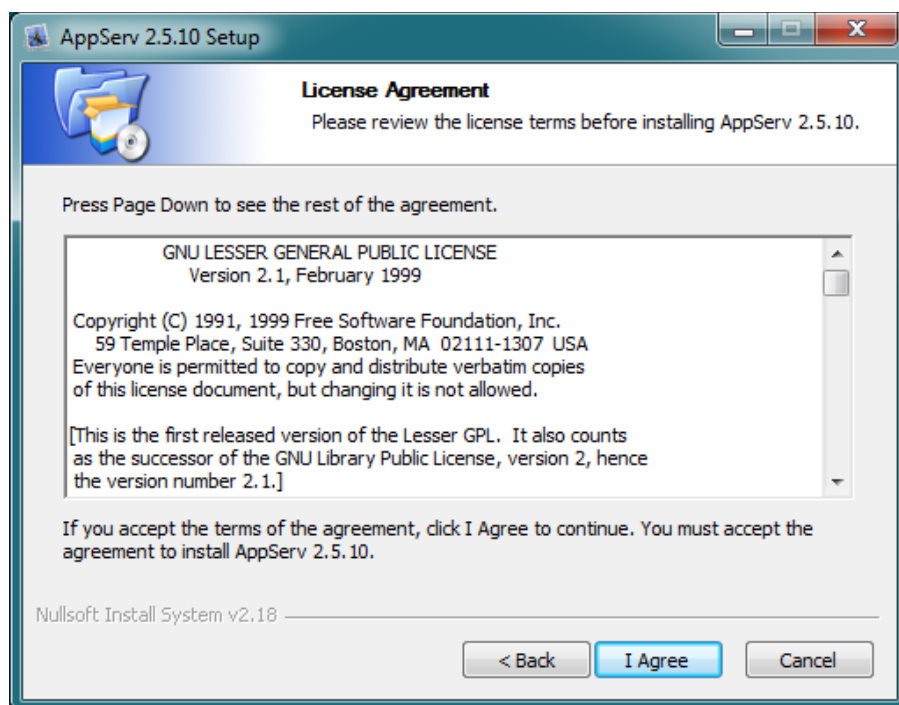


Ilustración 11.2 – Instalación de AppServ: paso2.

3. A continuación, el instalador nos preguntará en qué directorio queremos instalar la herramienta. En este caso dejaremos el directorio por defecto C:/AppServ y hacemos clic en “Next”.

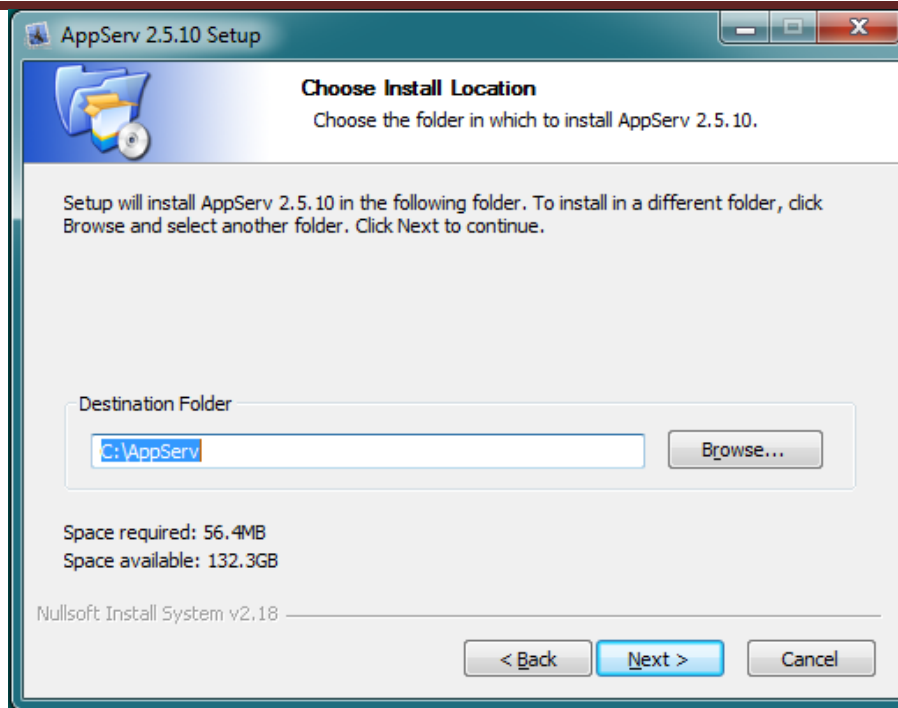


Ilustración 11.3 – Instalación de AppServ: paso 3.

- Ahora el instalador nos preguntará qué componentes queremos instalar. Dejamos marcados todos los componentes, tal y como viene por defecto, y hacemos clic en “Next”.

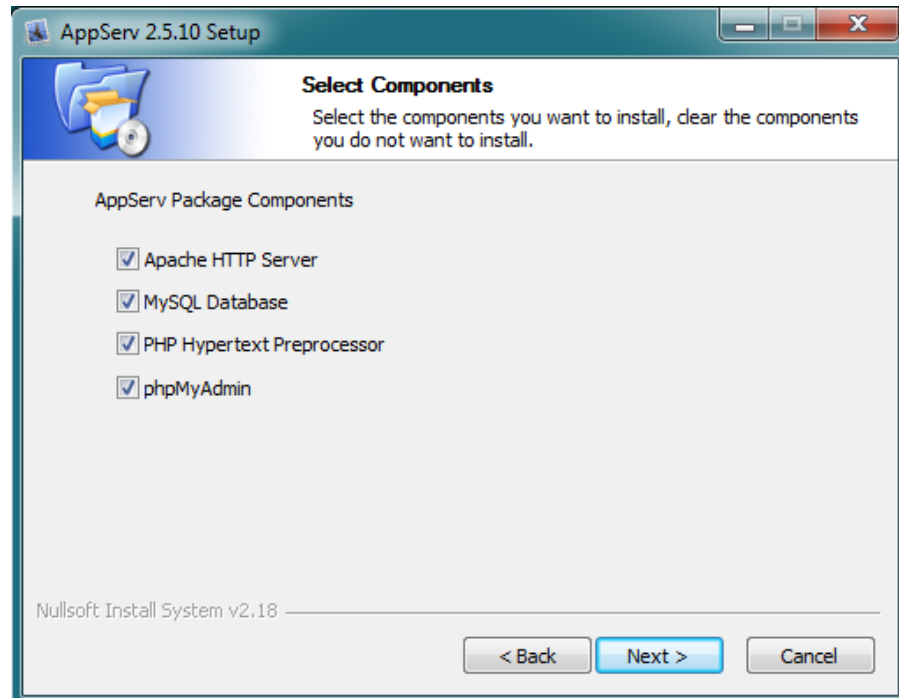


Ilustración 11.4 – Instalación de AppServ: paso 4.

- En la pantalla que se nos muestra ahora configuraremos el servidor Apache. Rellenaremos los campos con los siguientes valores:
 - Server Name (nombre del servidor): le damos el valor “localhost”.

- Administrator's email address (correo electrónico del administrador del servidor): le damos el valor "root@localhost".
- Apache HTTP port (puerto del servidor Apache): le damos el valor 8080. Es importante utilizar este puerto pues es donde Atenea buscará la conexión con la base de datos.

Una vez rellenos los campos con los valores indicados, hacemos clic en "Next" para continuar.

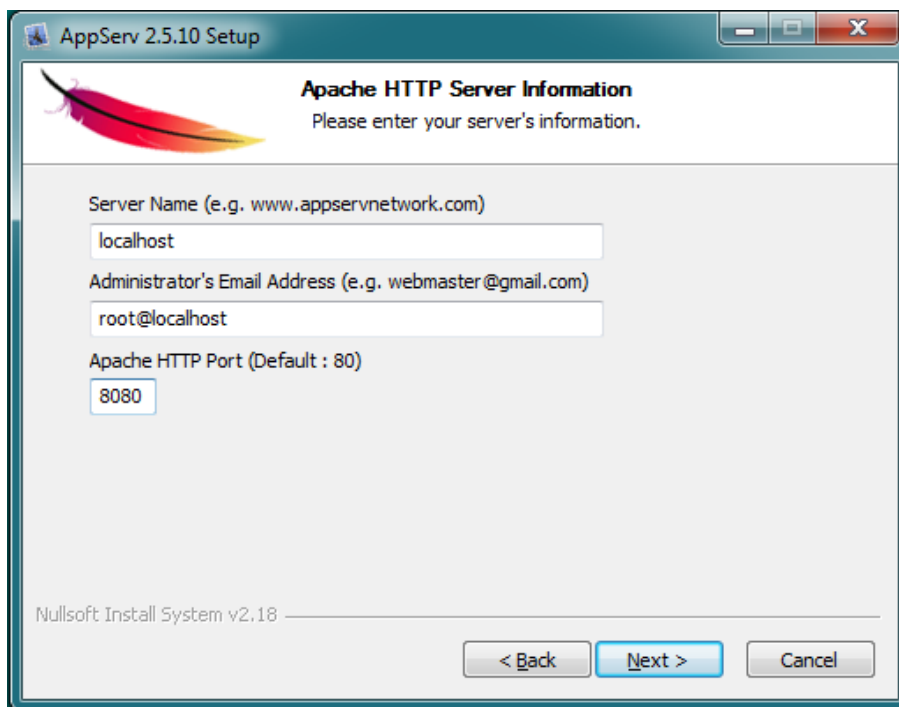


Ilustración 11.5 – Instalación de AppServ: paso 5.

6. En la pantalla actual, configuraremos MySQL Server. En el primer campo debemos introducir la contraseña que se utilizará posteriormente para conectarse al servidor de base de datos. En el segundo campo debe volver a introducirse de nuevo la misma contraseña para confirmarla. Es importante recordar la contraseña ya que se necesitará posteriormente. El usuario que MySQL crea por defecto es "root". El resto de los campos se dejan con los valores por defecto y se hace clic en "Install".

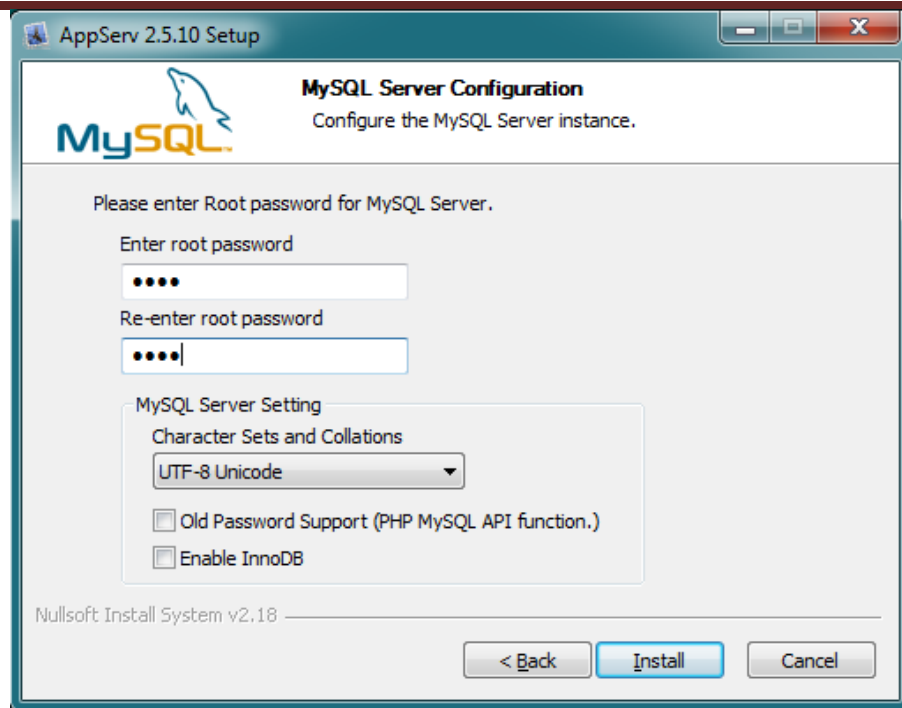


Ilustración 11.6 – Instalación de AppServ: paso 6.

7. El sistema proseguirá con la instalación de AppServ.

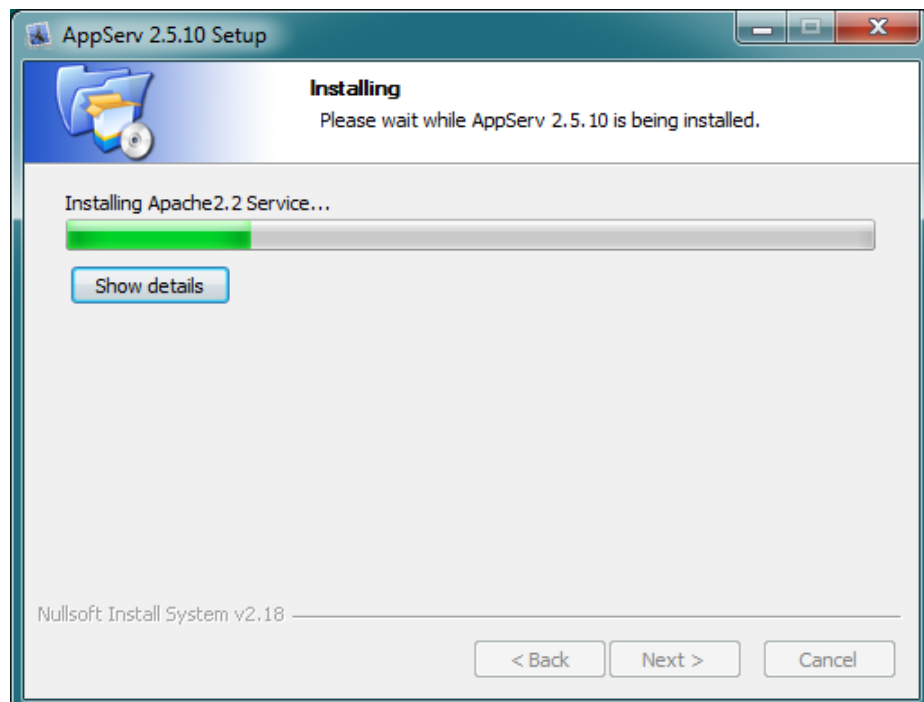


Ilustración 11.7 – Instalación de AppServ: paso 7.

8. En el transcurso del proceso puede ocurrir que se nos muestre un mensaje del Firewall de Windows pidiendo confirmación para permitir que se ejecute AppServ. Debemos dejar marcado el primer check. En caso de que queramos poder trabajar con nuestro

servidor en redes calificadas como públicas se debe marcar también el segundo check. Finalmente, se debe hacer clic en “Permitir acceso” para continuar.

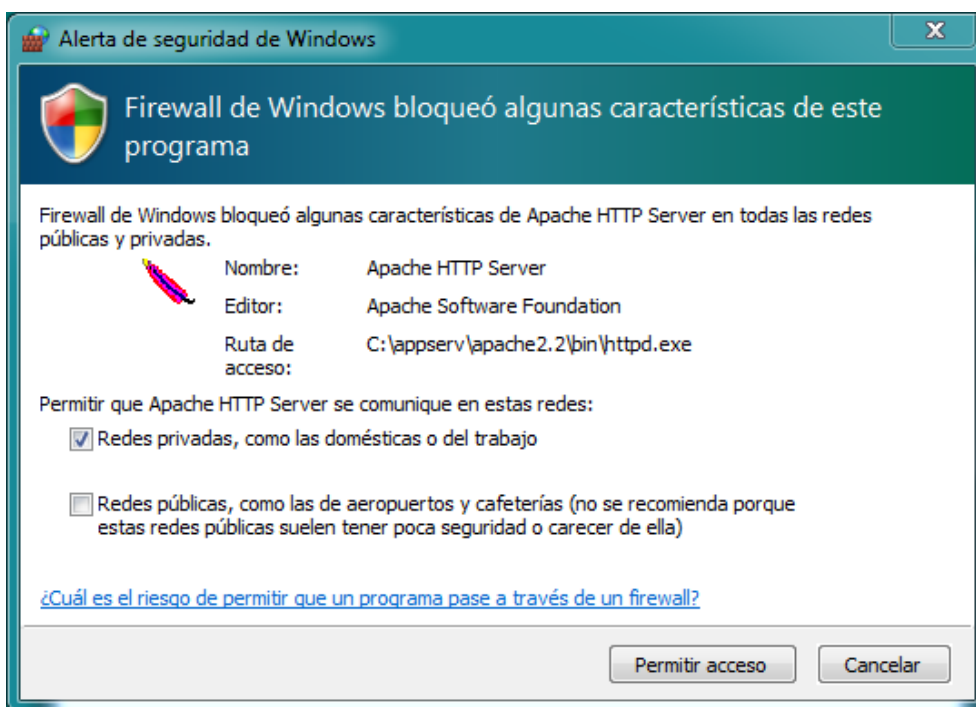


Ilustración 11.8 – Instalación de AppServ: paso 8.

9. Para finalizar la instalación hacemos clic en “Finish”. Si se dejan marcados los checks, Apache y MySQL arrancaran en ese momento.

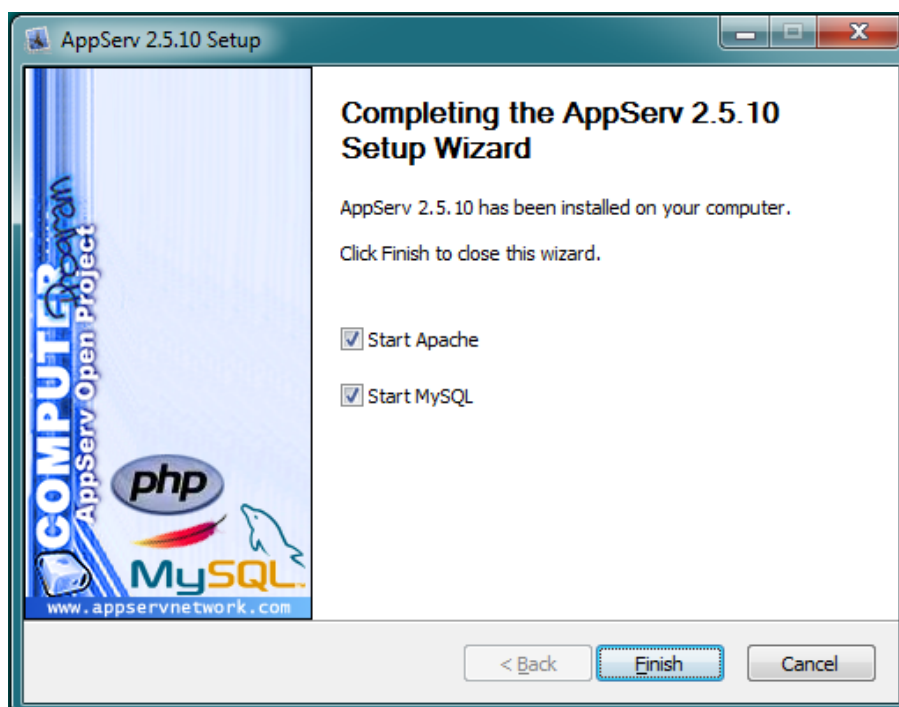


Ilustración 11.9 – Instalación de AppServ: paso 9.

10. Para comprobar que la instalación se ha realizado correctamente, abrir un navegador y poner la dirección <http://localhost:8080>. Debe mostrarse una pantalla similar a la siguiente:

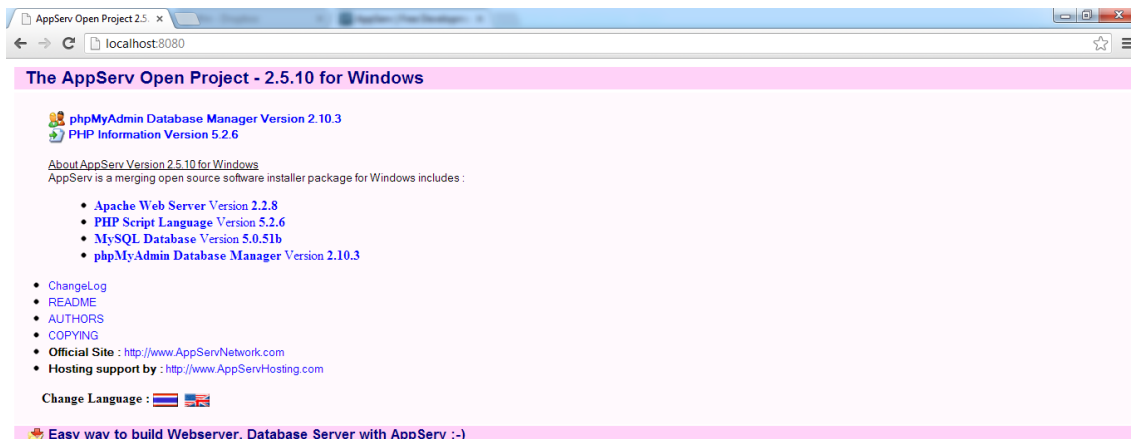


Ilustración 11.10 – Instalación de AppServ: paso 10.

11.1.4 Instalación de Tomcat

Tomcat es un servidor web gratuito que soporta servlets y JSP. Para descargar la última versión, accedemos a la página web del proyecto <http://tomcat.apache.org/>. En la sección de descargas, seleccionamos la última versión disponible, actualmente Tomcat 7.0.

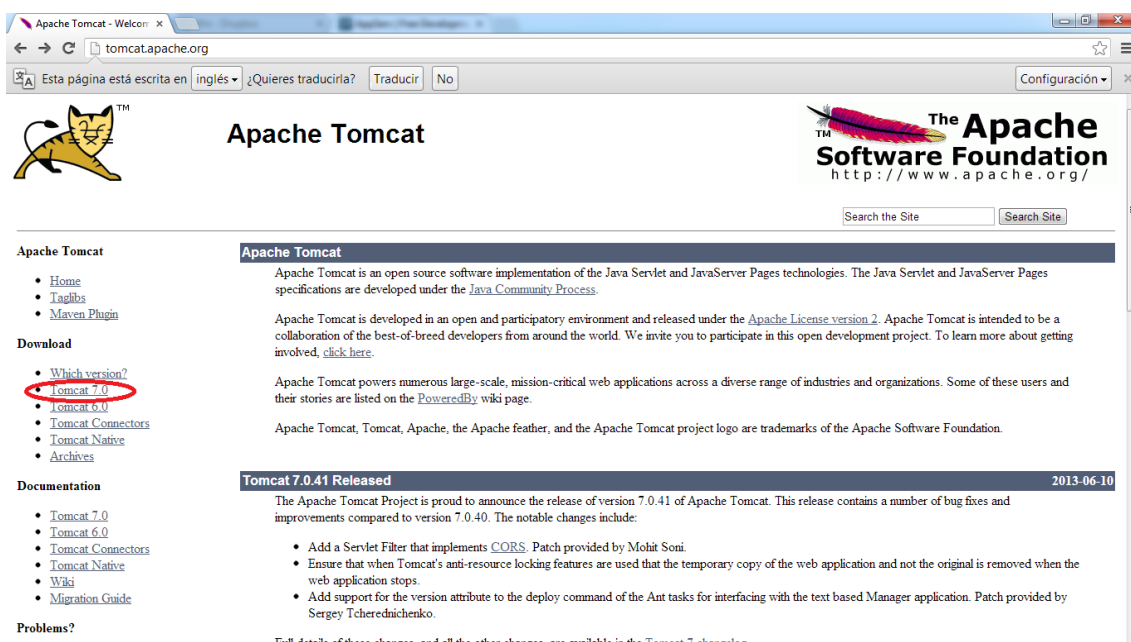


Ilustración 11.11 – Instalación de Tomcat: página oficial.

Buscamos la sección “Binary Distributions” y, por comodidad, descargamos la versión 32-bit/64-bit Windows Service Installer.

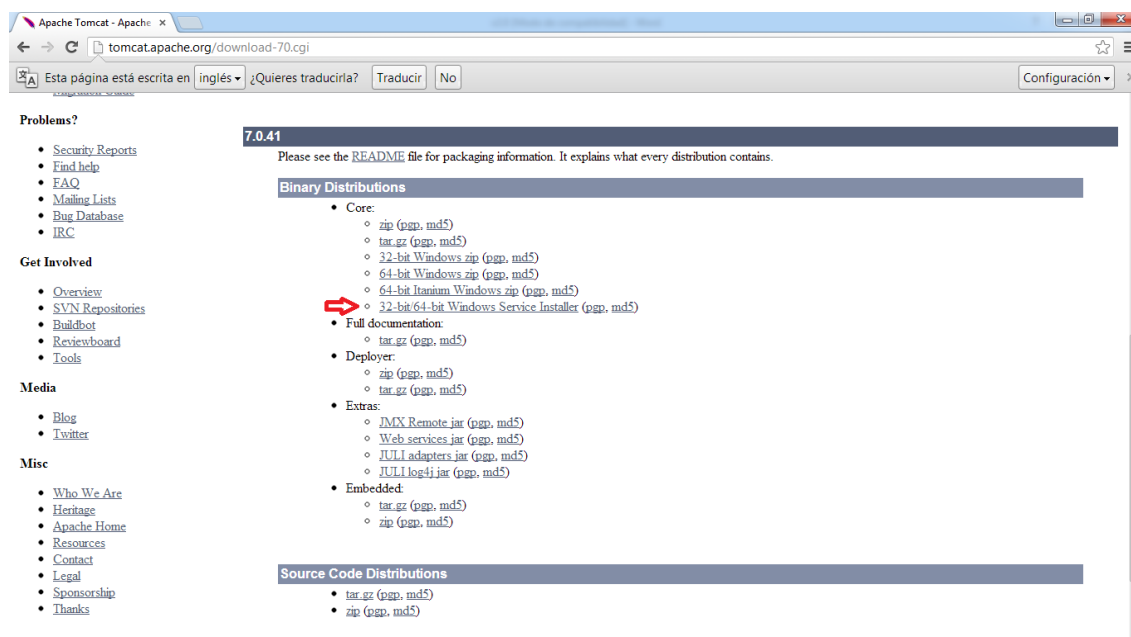


Ilustración 11.12 – Instalación de Tomcat: selección de la distribución.

Una vez finalizada la descarga, ejecutamos el archivo y comenzamos la instalación:

1. Hacemos clic en “Next”.

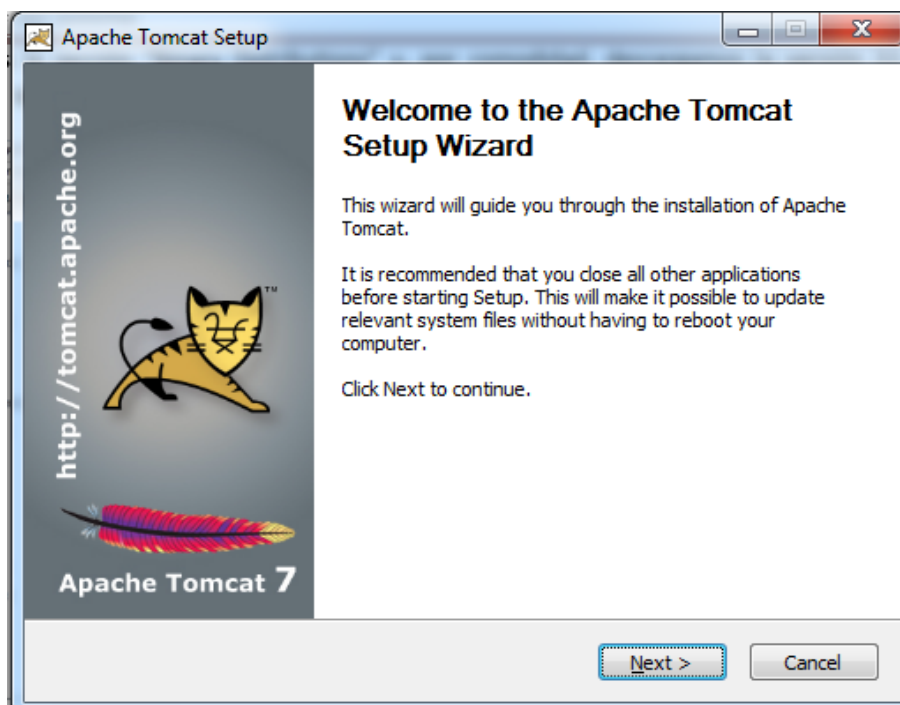


Ilustración 11.13 – Instalación de Tomcat: paso 1.

2. Aceptamos el acuerdo de licencia pulsando “I Agree”.

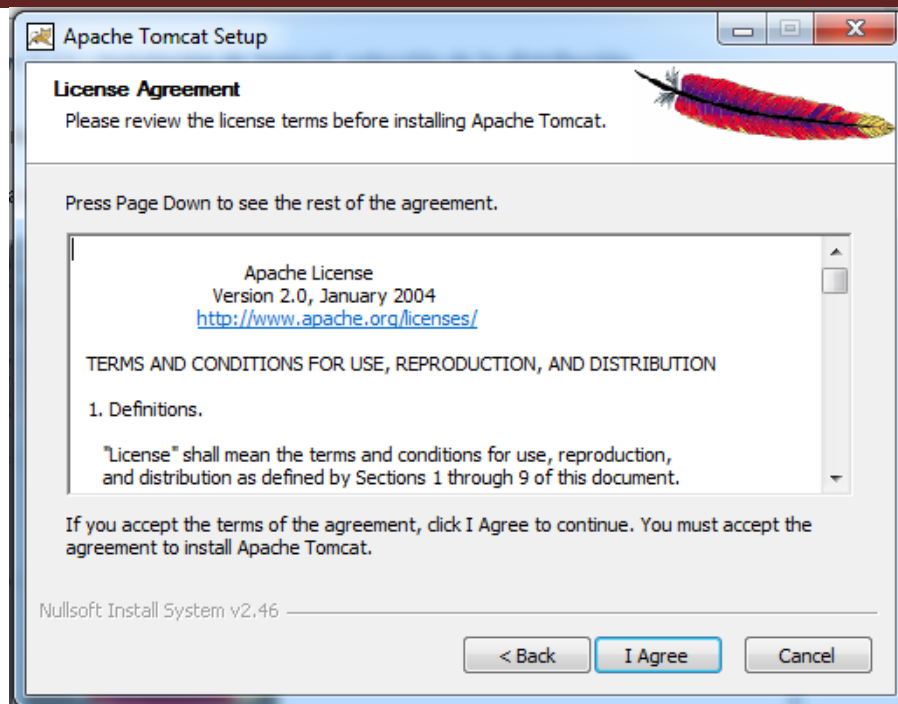


Ilustración 11.14 – Instalación de Tomcat: paso 2.

3. En la siguiente pantalla se nos preguntará qué componentes deseamos instalar. Dejamos marcada la selección por defecto y hacemos clic en “Next”.

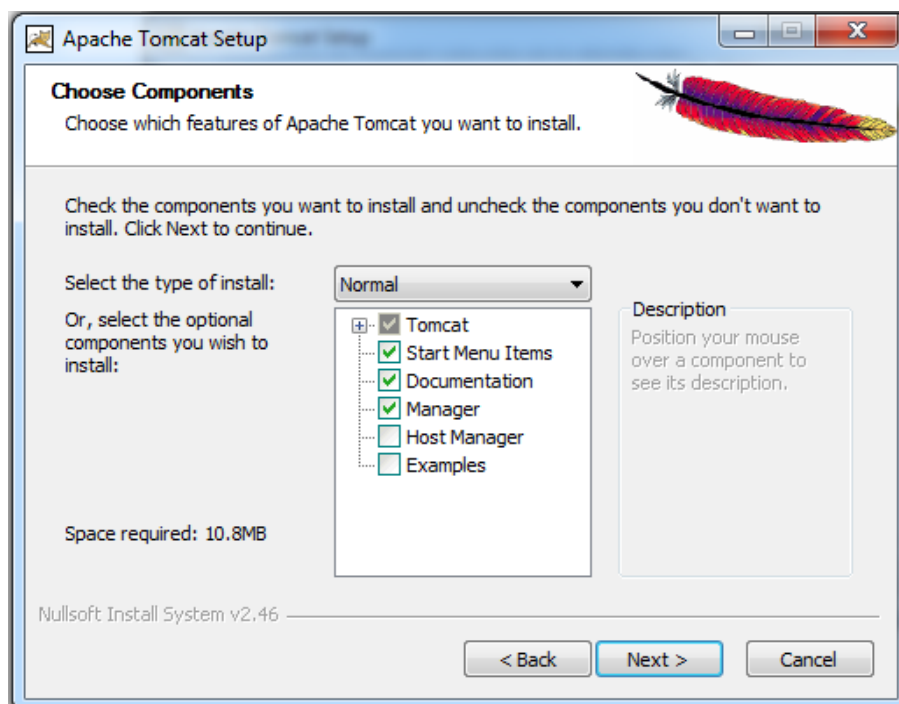


Ilustración 11.15 – Instalación de Tomcat: paso 3.

4. En esta pantalla configuraremos el servidor. Es muy importante darle el valor 80 al campo “HTTP/1.1. Conector Port”, pues es el puerto en el que Atenea está preparada para funcionar. Para el resto de campos dejamos los valores por defecto.

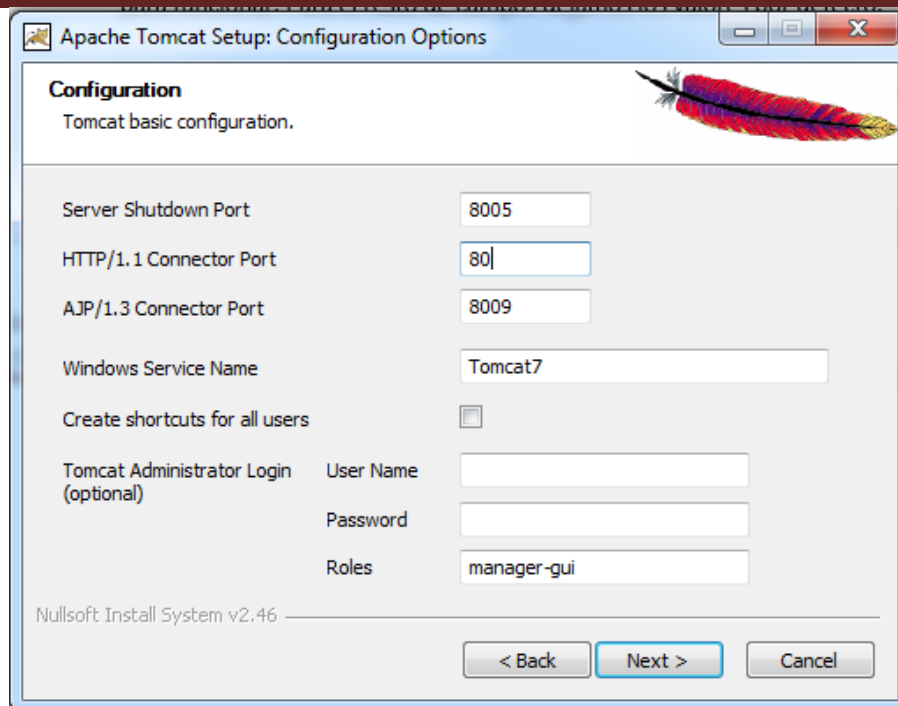


Ilustración 11.16 – Instalación de Tomcat: paso 4.

5. En la siguiente pantalla indicamos el directorio en el que está instalado Java.

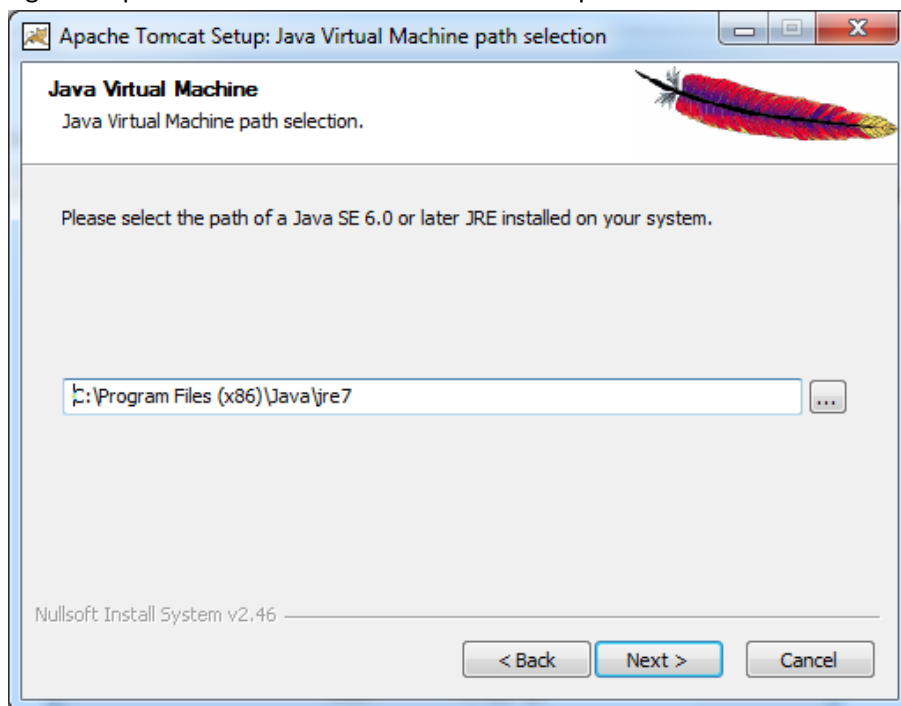


Ilustración 11.17 – Instalación de Tomcat: paso 5.

6. Finalmente, debemos indicar la carpeta en la que queremos instalar Tomcat y hacer clic en "Install".

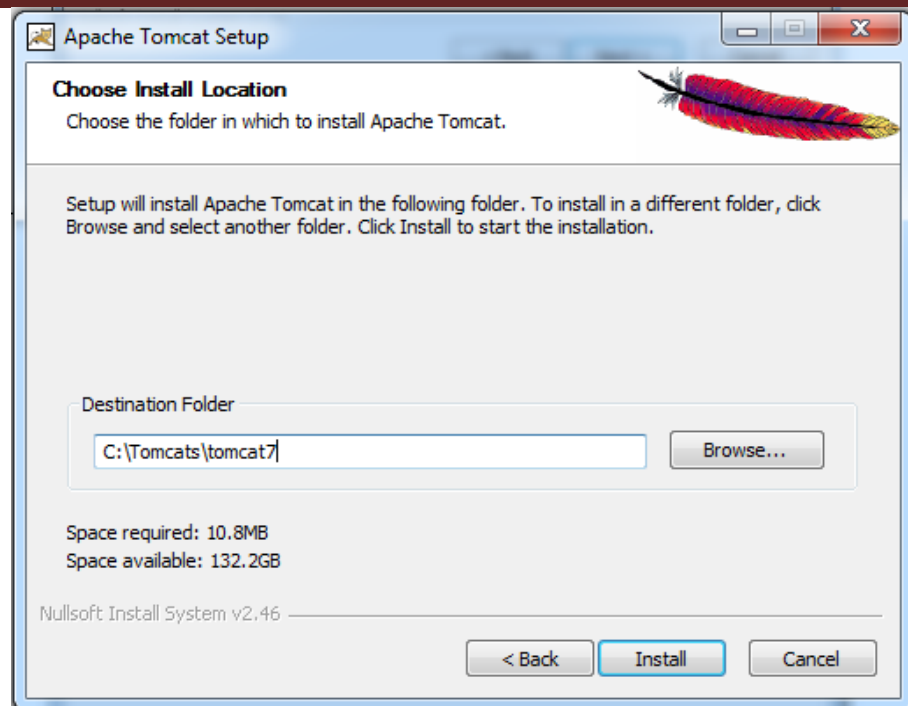


Ilustración 11.18 – Instalación de Tomcat: paso 6

7. Esperamos a que termine el proceso de instalación.

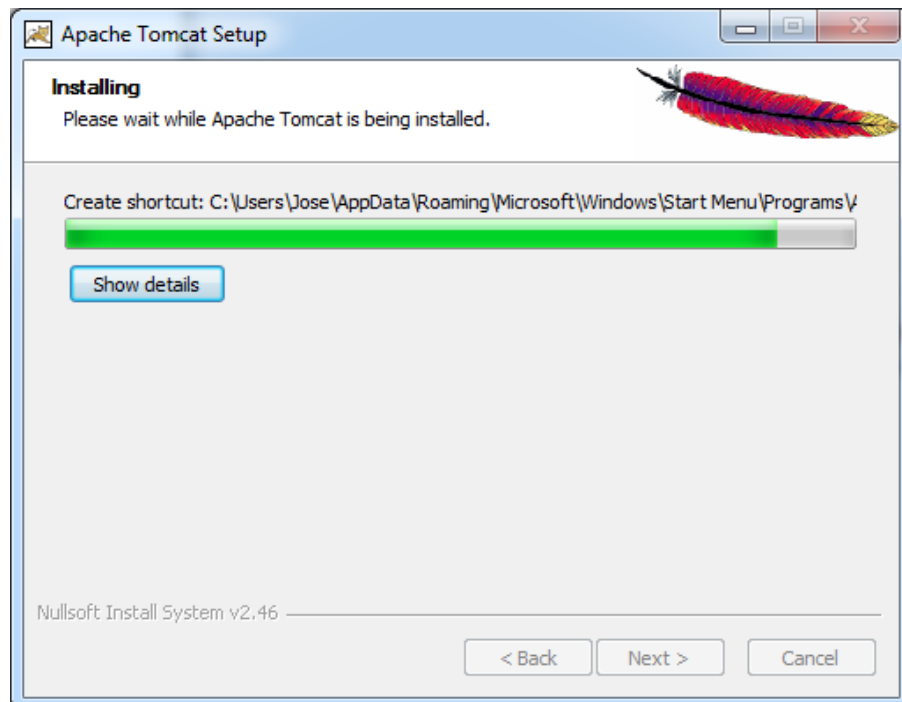


Ilustración 11.19 – Instalación de Tomcat: paso 7

8. Una vez finalizada la instalación, hacemos clic en "Finish".

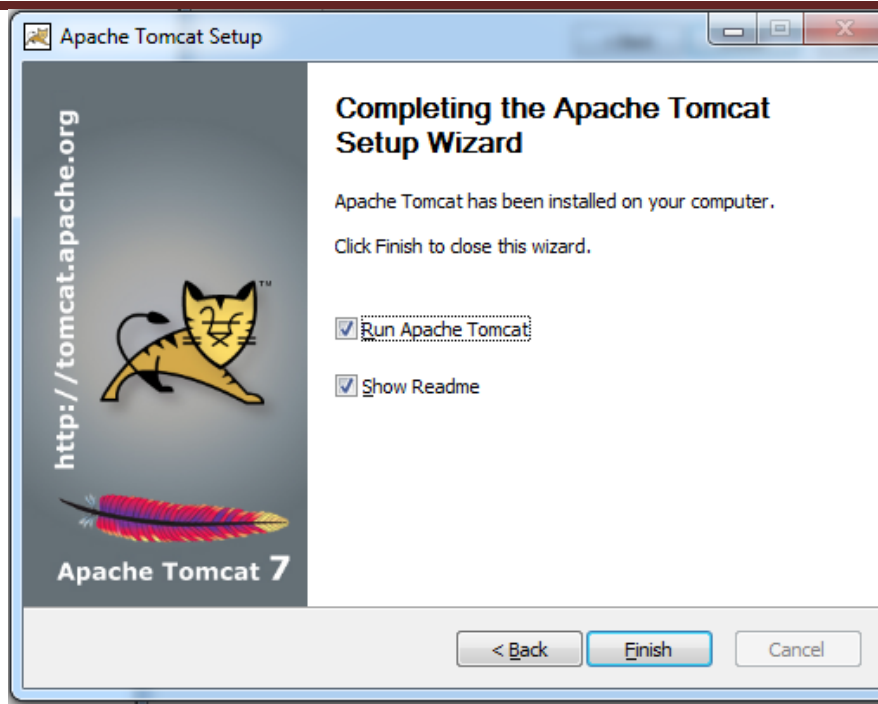


Ilustración 11.20 – Instalación de Tomcat: paso 8.

9. Para comprobar que la instalación se ha hecho correctamente, abrimos un navegador y escribimos la dirección <http://localhost>. En la pantalla debería aparecer una página similar a la siguiente:

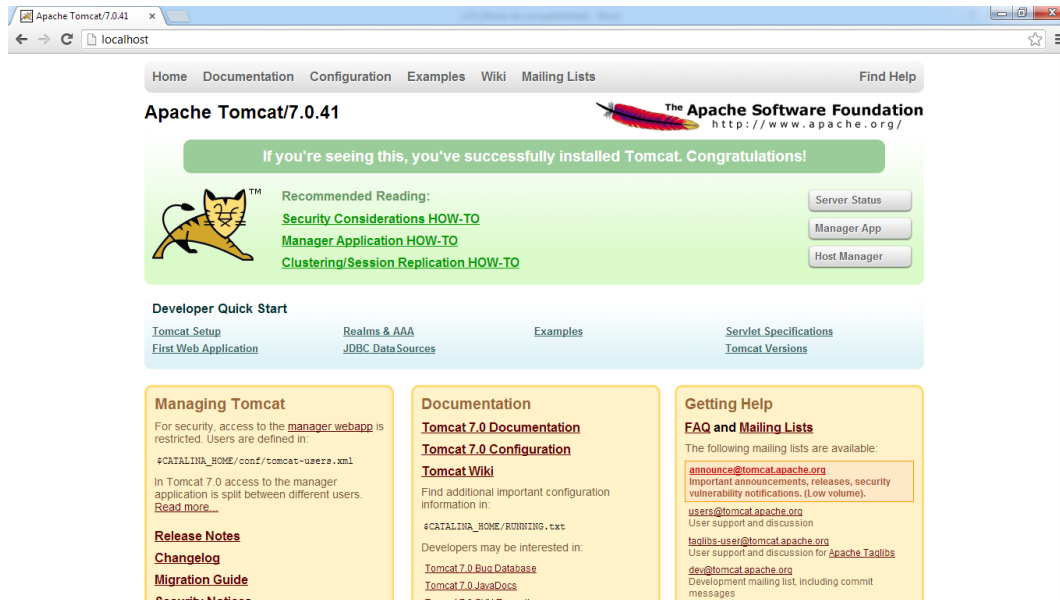


Ilustración 11.21 – Instalación de Tomcat: paso 9.

11.1.5 Instalación de Atenea

11.1.5.1 Creación de la base de datos

Para crear la base de datos que utiliza Atenea es necesario seguir los siguientes pasos:

1. Una vez hayamos instalado AppServ tal y como se explica en el capítulo 11.1.3 arrancamos los servidores desde Inicio -> Todos los programas -> AppServ -> Control Server By Manual -> Apache Start y MySQL Start. A continuación, abrimos un navegador y accedemos a la dirección <http://localhost:8080>. Se nos mostrará la siguiente página web:

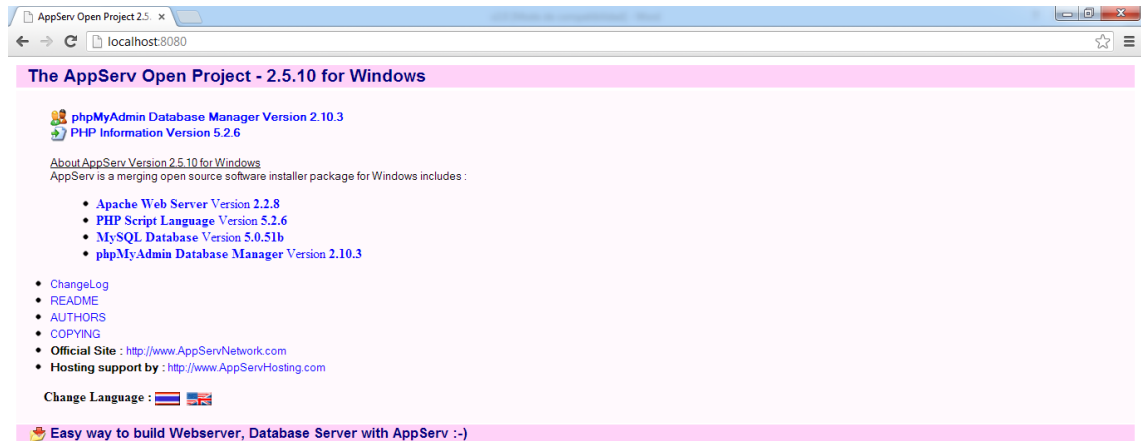


Ilustración 11.22 – Instalación de AppServ: paso 1.

2. Hacemos clic en el enlace phpMyAdmin Database Manager Version 2.10.3 para acceder al gestor de la base de datos. Nos aparecerá una ventana en la que debemos introducir el nombre de usuario y contraseña que establecimos durante la instalación de AppServ. El usuario por defecto de MySQL es “root”.

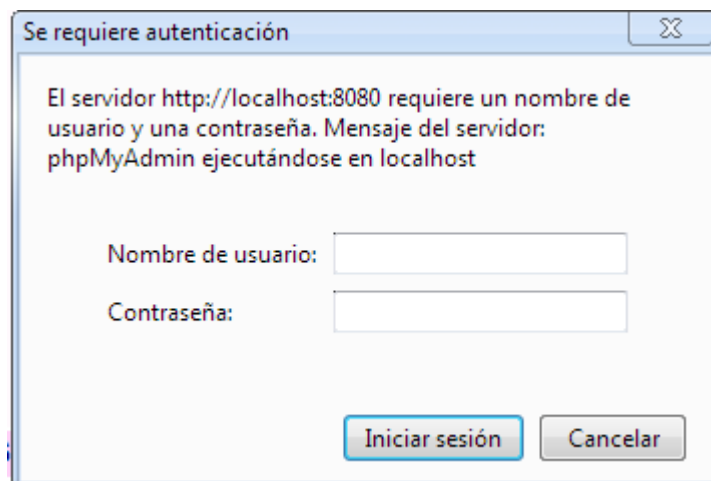


Ilustración 11.23 – Creación de la base de datos: paso 2.

3. Una vez hayamos iniciado sesión, se nos mostrará la página de la aplicación de gestión de la base de datos. En dicha página debemos hacer clic en el icono que abre la ventana de consultas a la base de datos.

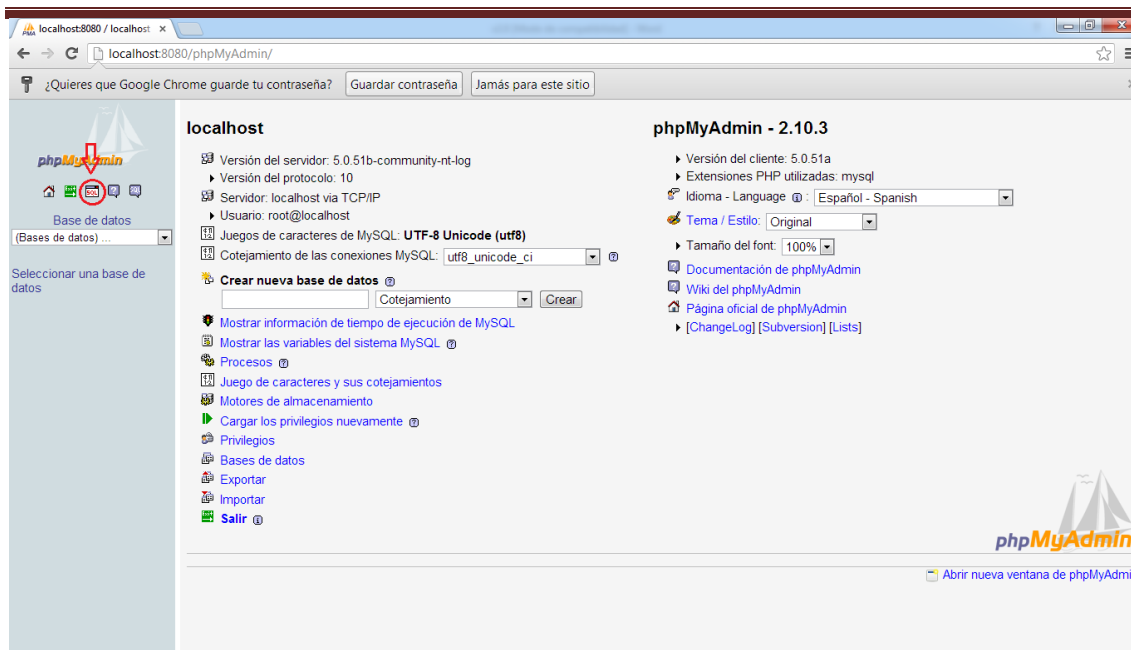


Ilustración 11.24 – Creación de la base de datos: paso 3.

4. Ahora debemos abrir el archivo configuracionBBDD.sql en un editor de texto, copiamos todo el contenido, lo pegamos en la ventana de consulta y hacemos clic en “Continuar”.

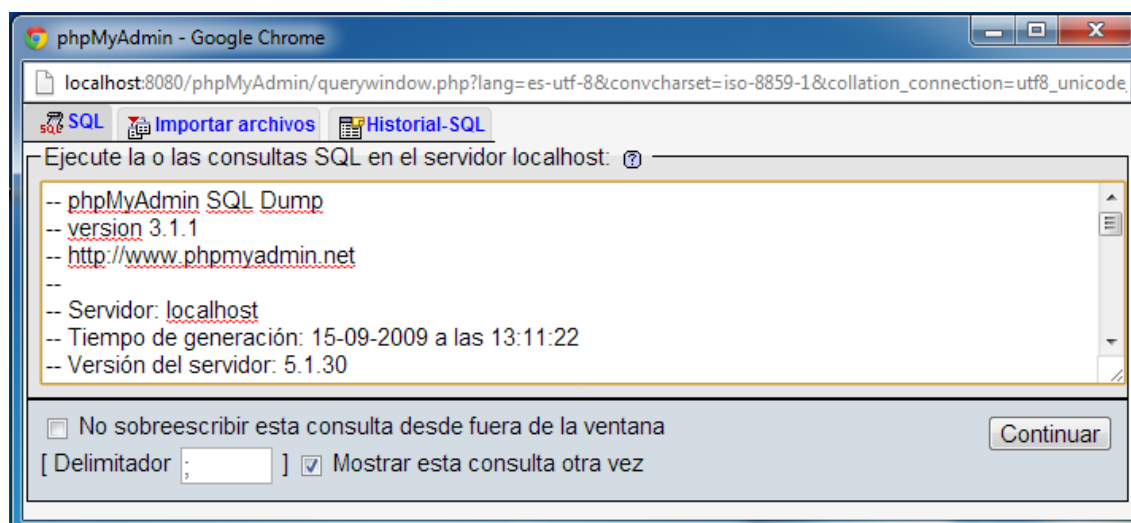


Ilustración 11.25 – Creación de la base de datos: paso 4

5. Nos aparecerá una pantalla indicando que el script se ha ejecutado correctamente.

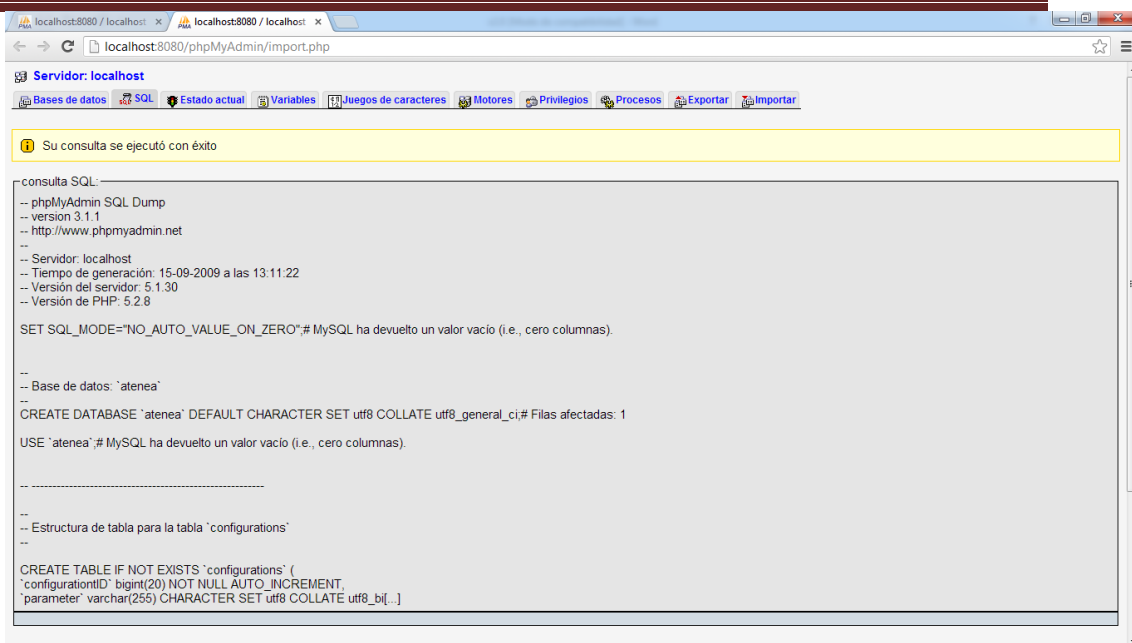


Ilustración 11.26 – Creación de la base de datos: paso 5.

6. En la pestaña “Bases de datos” podemos comprobar que se ha creado una nueva base de datos llamada “atenea”.

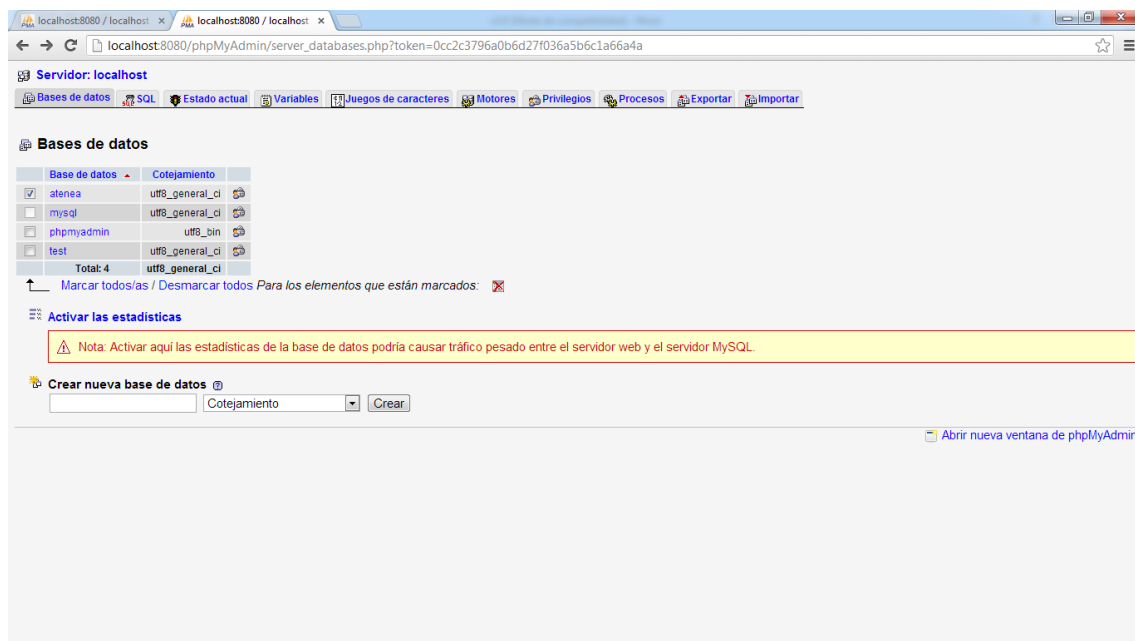


Ilustración 11.27 – Creación de la base de datos: paso 6.

11.1.5.2 Instalación de librerías de métricas y crawlers.

Para que Atenea funcione correctamente debemos crear una carpeta C:/Atenea en la que descomprimiremos el contenido del archivo Atenea.zip.

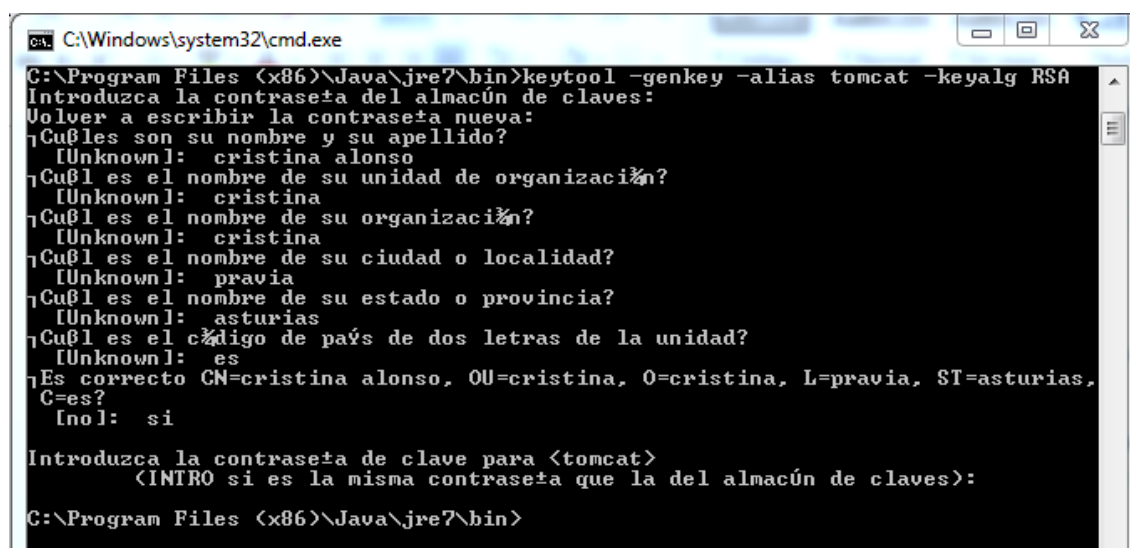
11.1.5.3 Creación del almacén de certificados.

Para navegar por la parte privada de Atenea se utiliza una conexión SSL por lo que debemos crear un almacén de claves que contendrá el certificado de conexión segura. Para ello seguiremos los siguientes pasos:

1. Abrimos una ventana de línea de comandos y nos colocamos en el directorio /bin de la carpeta donde tengamos instalado Java.
2. Una vez situados en el directorio ejecutamos el siguiente comando:

```
keytool -genkey -alias tomcat -keyalg RSA
```

3. Se nos irá pidiendo que introduzcamos una contraseña, nuestro nombre y apellidos, el nombre de la empresa, la provincial y las iniciales del país. Una vez introducidos todos estos datos, se nos preguntará si son correctos, a lo que se debe contestar con “sí”.



```
C:\Windows\system32\cmd.exe
C:\Program Files (x86)\Java\jre7\bin>keytool -genkey -alias tomcat -keyalg RSA
Introduzca la contraseña del almacén de claves:
Volver a escribir la contraseña nueva:
¿Cuáles son su nombre y su apellido?
[Unknown]: cristina alonso
¿Cuál es el nombre de su unidad de organización?
[Unknown]: cristina
¿Cuál es el nombre de su organización?
[Unknown]: cristina
¿Cuál es el nombre de su ciudad o localidad?
[Unknown]: pravia
¿Cuál es el nombre de su estado o provincia?
[Unknown]: asturias
¿Cuál es el código de país de dos letras de la unidad?
[Unknown]: es
¿Es correcto CN=cristina alonso, OU=cristina, O=cristina, L=pravia, ST=asturias,
C=es?
[no]: si
Introduzca la contraseña de clave para <tomcat>
(INTRO si es la misma contraseña que la del almacén de claves):
C:\Program Files (x86)\Java\jre7\bin>
```

Ilustración 11.28 – Creación del almacén de claves: paso 3.

4. La herramienta habrá creado el archivo de almacén de claves en la carpeta “home” del usuario actual. El nombre del archivo será “.keystore”.

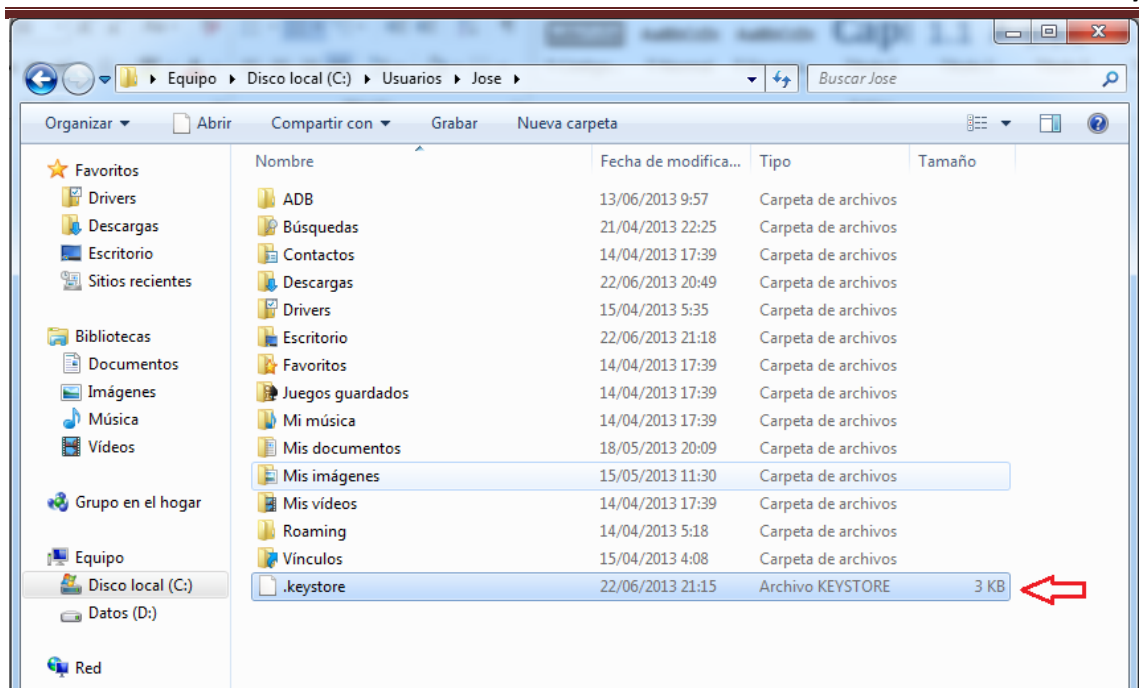


Ilustración 11.29 – Creación del almacén de claves: paso 4.

11.1.5.4 Configuración de Tomcat y despliegue de la aplicación web

Como se ha explicado en el punto anterior, Atenea utiliza una conexión web segura para la navegación por su parte privada. Debemos configurar el servidor Tomcat sobre el que vamos a desplegar la aplicación para que utilice SSL con el certificado que hemos creado según lo explicado en el punto 11.1.5.3.

1. Abrimos la carpeta en la que tengamos instalado el Tomcat, entramos en el directorio “/conf” y abrimos el archivo server.xml con un editor de texto.
2. Buscamos la sección que define una conexión SSL y añadimos las siguientes líneas:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
    protocol="HTTP/1.1"
    port="8443" maxThreads="200"
    scheme="https" secure="true" SSLEnabled="true"
    keystoreFile="C:/Users/Cris/.keystore" keystorePass="cris"
    clientAuth="false" sslProtocol="TLS"/>
```

En el parámetro “keystoreFile” se debe poner la ruta absoluta en la que se encuentra el archivo de almacén de claves y en el parámetro “keystorePass” la contraseña que hemos introducido al crearlo.

3. Guardamos los cambios y cerramos el archivo server.xml.
4. Ahora debemos copiar el archivo atenea.war en la carpeta “/webapps” que se encuentra en el directorio de instalación del Tomcat.
5. A continuación, entramos en la carpeta “/bin” y ejecutamos el archivo Tomcat7.exe que arrancará el servidor y desplegará la aplicación web.

- Una vez el servidor haya arrancado abrimos un navegador y accedemos a la dirección <http://localhost/atenea> y se nos mostrará la página principal de la aplicación web Atenea.

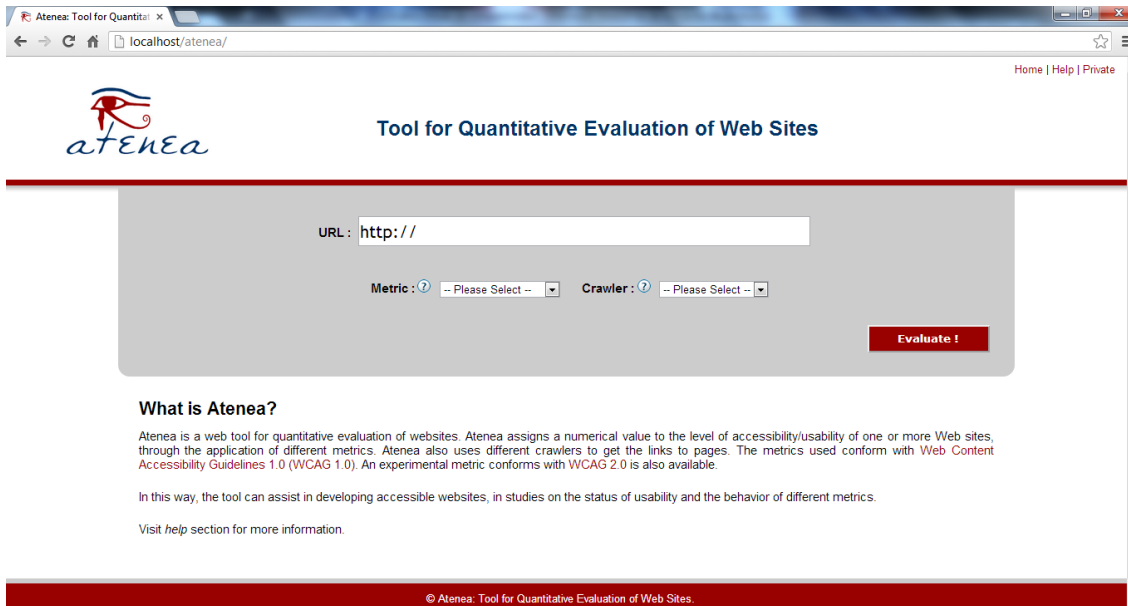


Ilustración 11.30 – Configuración de Tomcat y despliegue de la aplicación web: paso 6.

11.2 Manual de Ejecución

11.2.1 Arranque del sistema

Una vez realizada la instalación y configuración de todos los programas y componentes tal y como se explica en el capítulo 11.1 basta con seguir los siguientes pasos para arrancar la aplicación web:

1. Arrancar el servidor de base de datos desde Inicio -> Todos los programas -> AppServ -> Control Server By Manual: Apache Start y MySQL Start.
2. Arrancar el servidor Tomcat ejecutando el archivo Tomcat7.exe que se encuentra en la carpeta “/bin” situada en el directorio donde tengamos instalado el Tomcat.
3. Abrir un navegador web y acceder a la dirección <http://localhost/atenea>.

11.2.2 Parada del sistema

Para parar el sistema hay que seguir los siguientes pasos:

1. Parar el servidor Tomcat haciendo Ctrl+C desde la ventana de línea de comandos del servidor.
2. Parar la base de datos desde Inicio -> Todos los programas -> AppServ -> Control Server By Manual: Apache Stop y MySQL Stop.

11.3 Manual de Usuario

El usuario final utiliza la librería Ivory a través de la interfaz de la aplicación Atenea 3.0 por lo que se ha decidido incluir un pequeño manual de usuario que describa las funciones básicas de Atenea 3.0 que el usuario debe conocer para poder analizar la usabilidad de páginas web mediante la métrica Ivory. Para una información más detallada sobre el funcionamiento de Atenea 3.0, consultar la referencia [Atenea12].

11.3.1 Realizar análisis desde la parte pública de Atenea 3.0

Para acceder a la parte pública de la aplicación no es necesario estar registrado ni haber iniciado sesión en el sistema, basta con acceder a la herramienta desde un navegador web y se mostrará la página de inicio de Atenea 3.0.

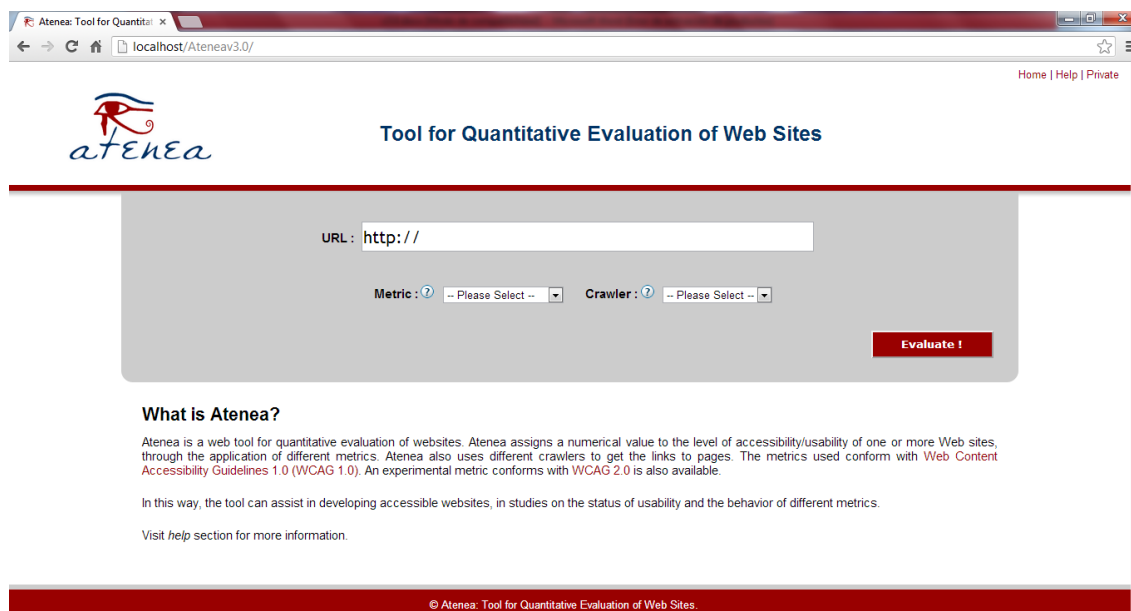


Ilustración 11.31 – Página de inicio de Atenea 3.0

Para analizar una página web con la métrica Ivory se debe introducir su dirección en el campo de texto “URL” y seleccionar “Ivory” en el campo “Metric” y “Jobbo” en el campo “Crawler”.

Para que comience el análisis es necesario hacer clic en el botón “Evaluate!”.

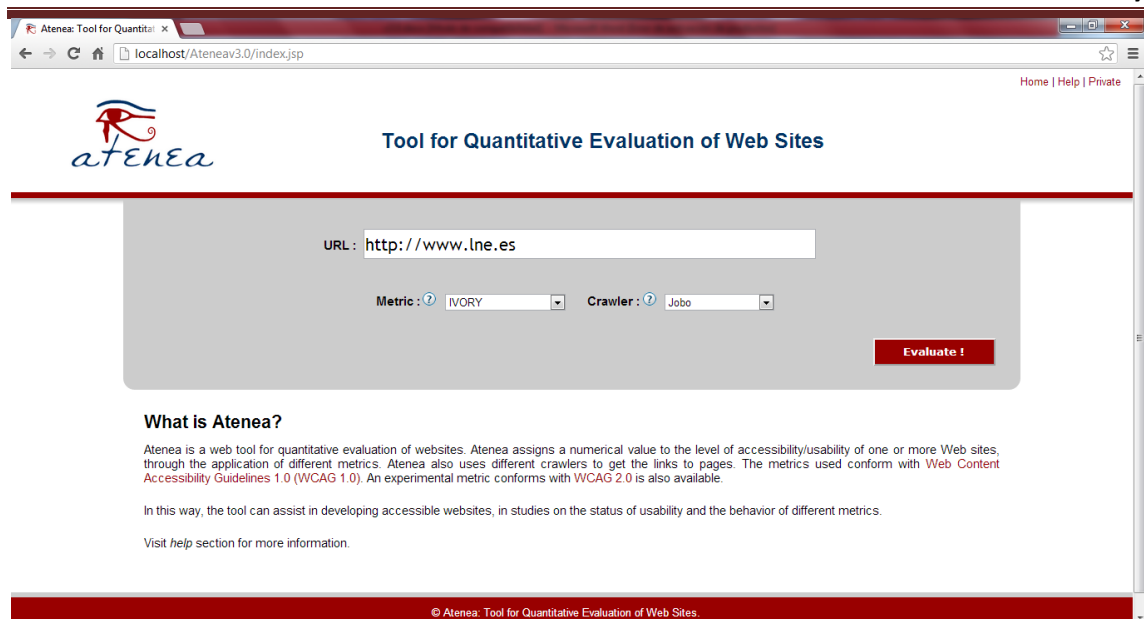


Ilustración 11.32 – Realizar análisis desde la parte pública: introducir datos

Es importante saber que al realizar un análisis de una URL desde la parte pública de la aplicación, éste solo se realizará sobre la página a la que corresponde dicha URL, no se explorará ninguno de sus enlaces ni en profundidad, ni en anchura.

Mientras se realiza el análisis se muestra un mensaje al usuario informando de que la operación está en curso.

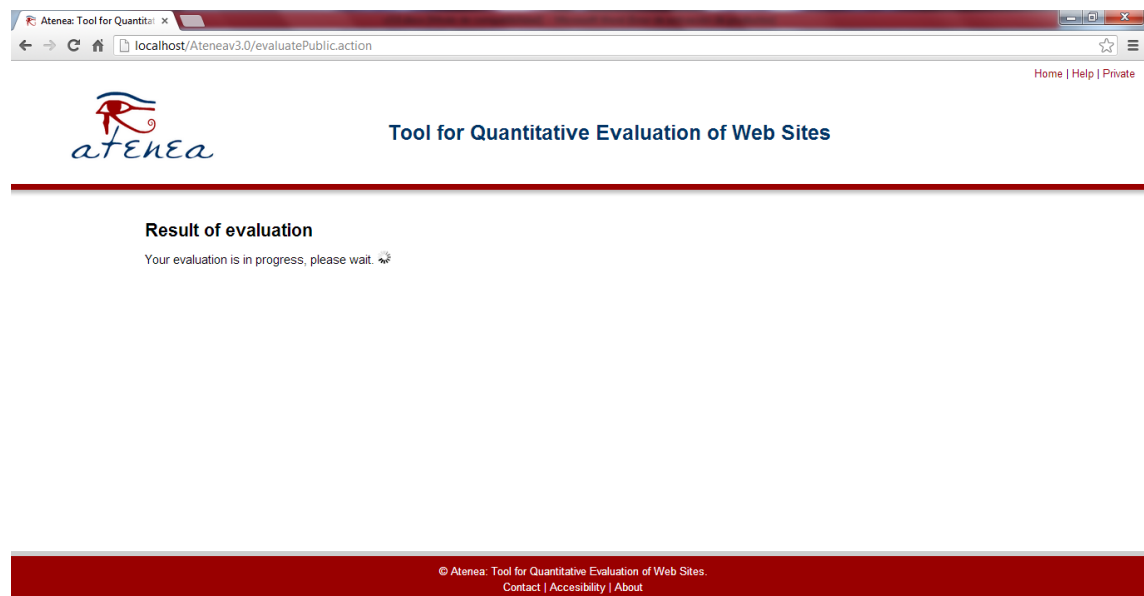
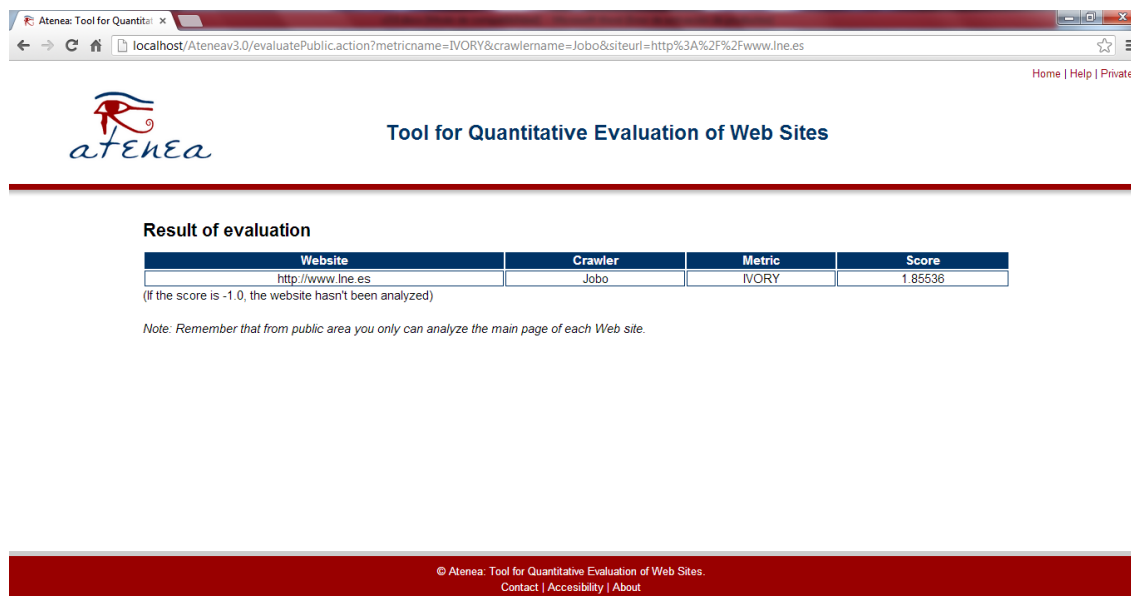


Ilustración 11.33 – Realizar análisis desde la parte pública: análisis en curso

Una vez se haya terminado de realizar el análisis, se mostrará una tabla con las siguientes columnas:

- Website: url analizada.
- Crawler: crawler utilizado para obtener la página.

- Métrica con la que se ha evaluado la página web.
- Result: valoración de la página con la métrica seleccionada.



The screenshot shows a web browser window with the URL `localhost/Ateneav3.0/evaluatePublic.action?metricname=IVORY&crawlername=Jobo&siteurl=http%3A%2F%2Fwww.lne.es`. The page title is "Tool for Quantitative Evaluation of Web Sites". Below the title, there is a section titled "Result of evaluation" containing a table with the following data:

Website	Crawler	Metric	Score
http://www.lne.es	Jobo	IVORY	1.85536

Below the table, there is a note: "(If the score is -1.0, the website hasn't been analyzed)". At the bottom of the page, there is a footer with the text: "© Atenea. Tool for Quantitative Evaluation of Web Sites. Contact | Accessibility | About".

Ilustración 11.34 – Realizar análisis desde la parte pública: resultado final

Si queremos realizar un nuevo análisis debemos volver a la página principal de Atenea 3.0 y volver a seguir el procedimiento descrito. Para regresar a la página principal en cualquier momento basta con hacer clic sobre el logotipo de la herramienta.

11.3.2 Realizar análisis desde la parte privada de Atenea 3.0.

11.3.2.1 Acceso a la parte privada de Atenea 3.0

Para acceder a la parte privada de la aplicación es necesario estar registrado en el sistema. Con estos datos podremos iniciar sesión en el sistema.

Haciendo clic en el enlace "Private" situado en la esquina superior derecha de la página principal de Atenea 3.0 accedemos a la pantalla de inicio de sesión.

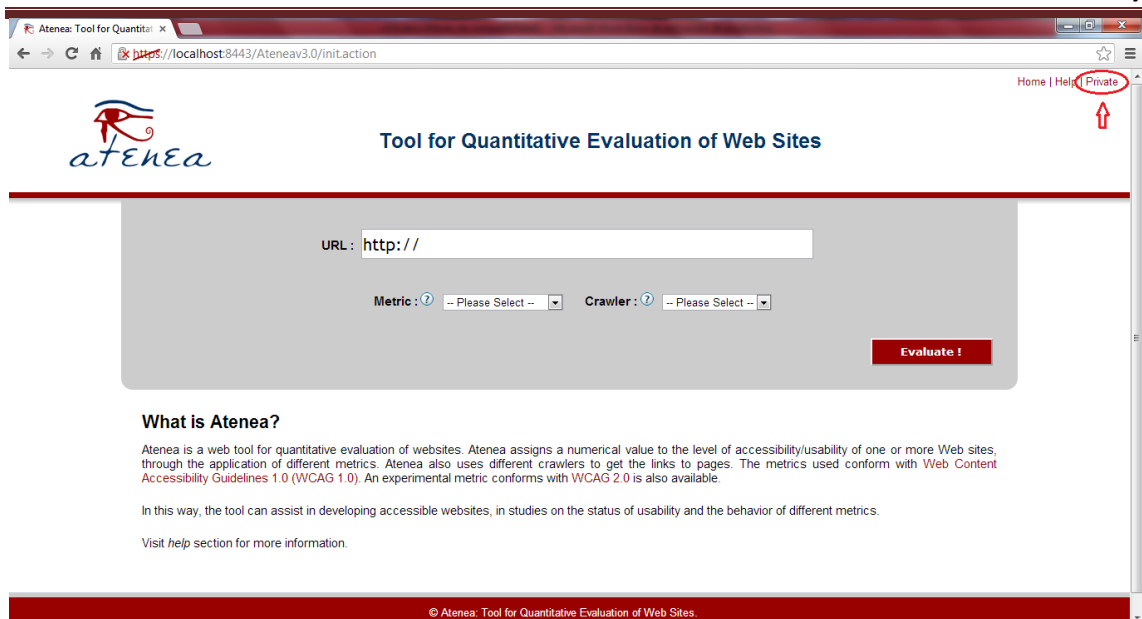


Ilustración 11.35 – Realizar análisis desde la parte privada: acceso a la parte privada

Una vez en la página de login rellenamos el campo “Login” con nuestro nombre de usuario y el campo “Password” con nuestra contraseña y pulsamos el botón “Enter” para iniciar sesión en el sistema.

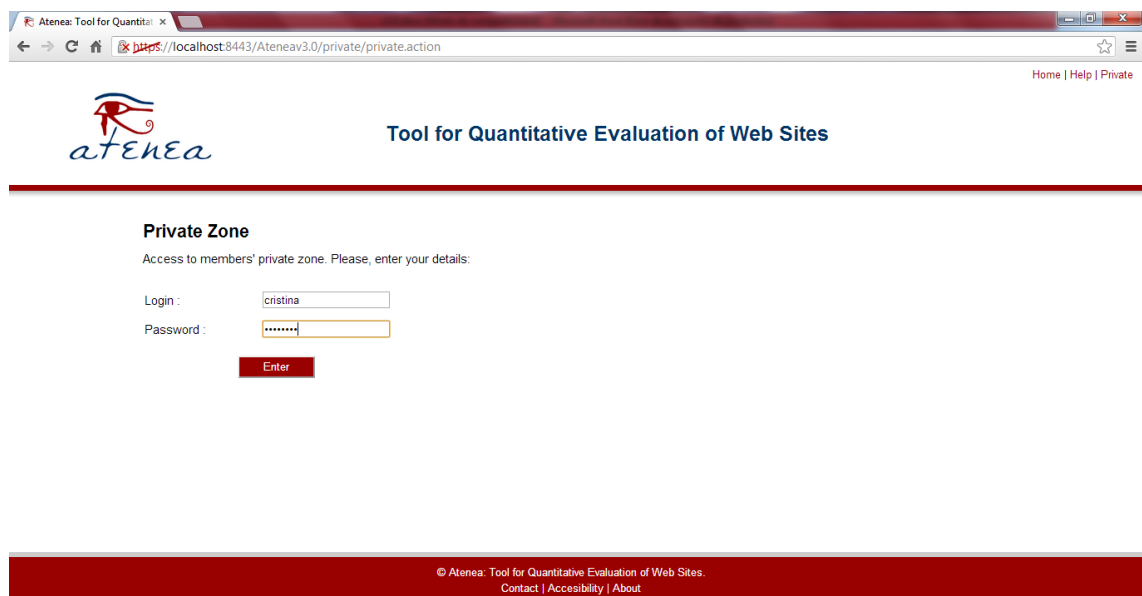


Ilustración 11.36 – Realizar análisis desde la parte privada: inicio de sesión

Si los datos introducidos son correctos se mostrará la pantalla de inicio de la parte privada.

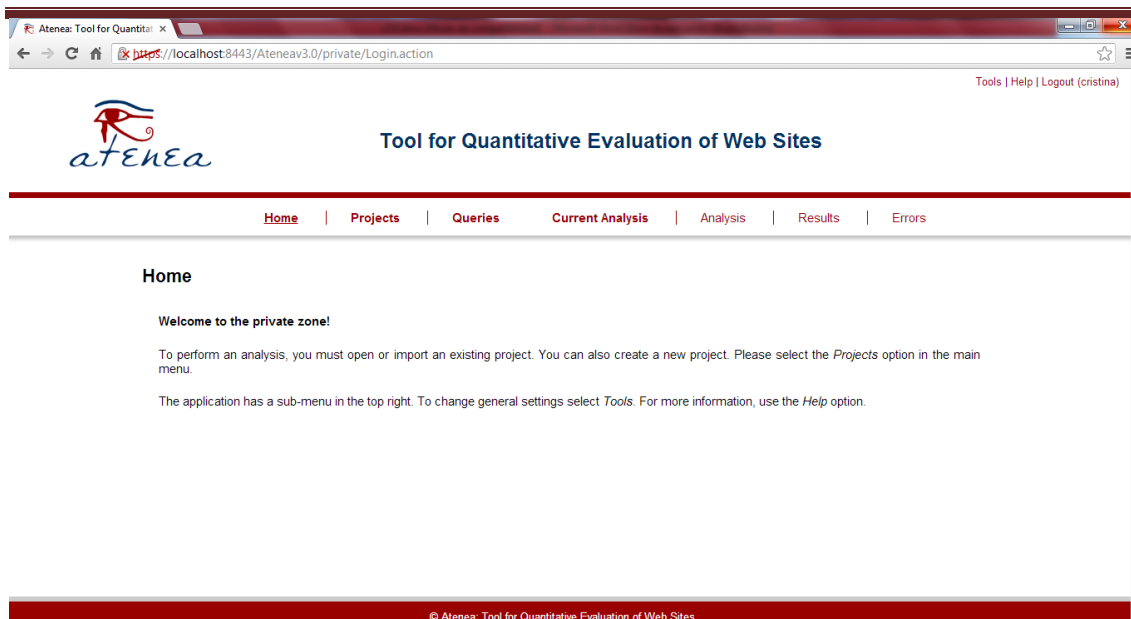


Ilustración 11.37 – Realizar análisis desde la parte privada: pantalla de bienvenida

11.3.2.2 Creación de un proyecto

Para poder realizar un análisis es necesario que creamos un nuevo proyecto en la aplicación. Para ello hacemos clic en la pestaña Projects y se nos mostrará un formulario en el que rellenaremos los siguientes campos:

- Name of Project: nombre que queramos darle a nuestro proyecto.
- Short description: descripción breve del proyecto.
- Metrics to use: seleccionaremos todas aquellas que queramos que se apliquen a las páginas que analicemos dentro de este proyecto.
- Depth level: nivel de profundidad que se aplicará al obtener las páginas.
- Maximun pages: número máximo de páginas que se analizarán.
- Crawler: crawler que se utilizará para obtener los documentos.

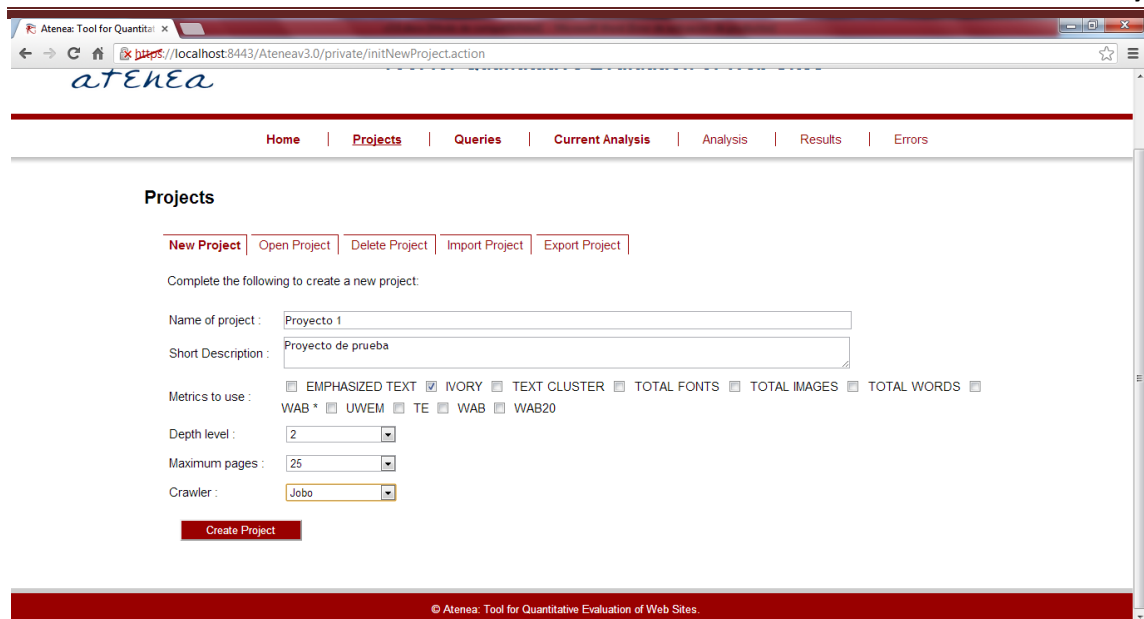


Ilustración 11.38 – Realizar análisis desde la parte privada: crear proyecto nuevo.

Una vez cubiertos los campos, pulsamos el botón “Create Project” y se nos mostrará una pantalla desde la que podremos configurar el crawler o dejarlo con la configuración por defecto. En este caso, dejamos la configuración por defecto pulsando el botón “Continue Without Configuration”.

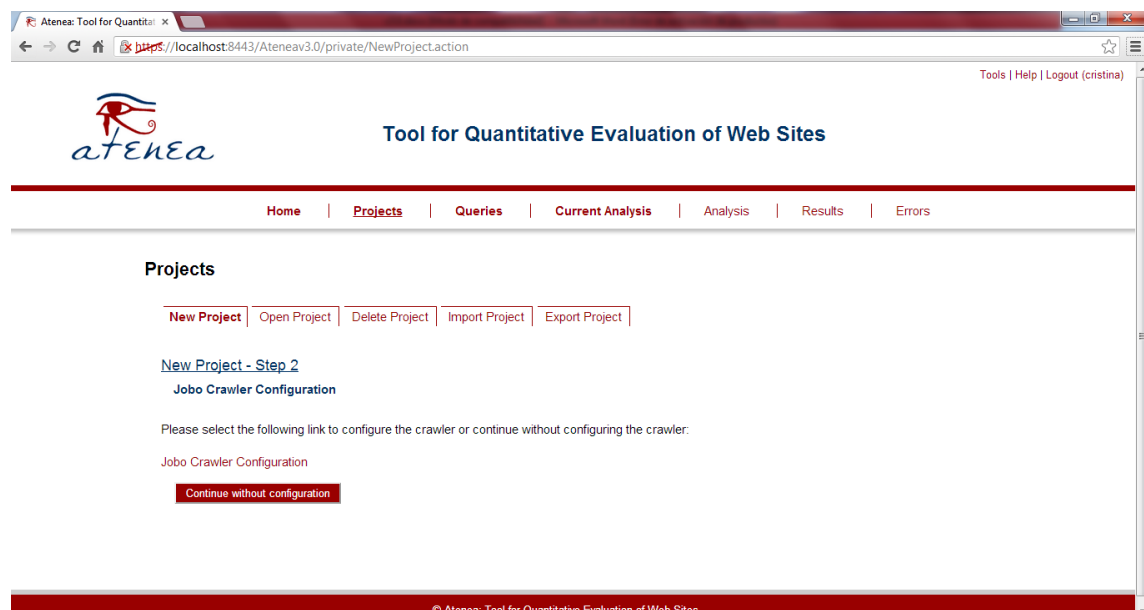


Ilustración 11.39 – Realizar análisis desde la parte privada: configurar crawler.

Se mostrará una pantalla indicando que el proyecto se ha creado correctamente.

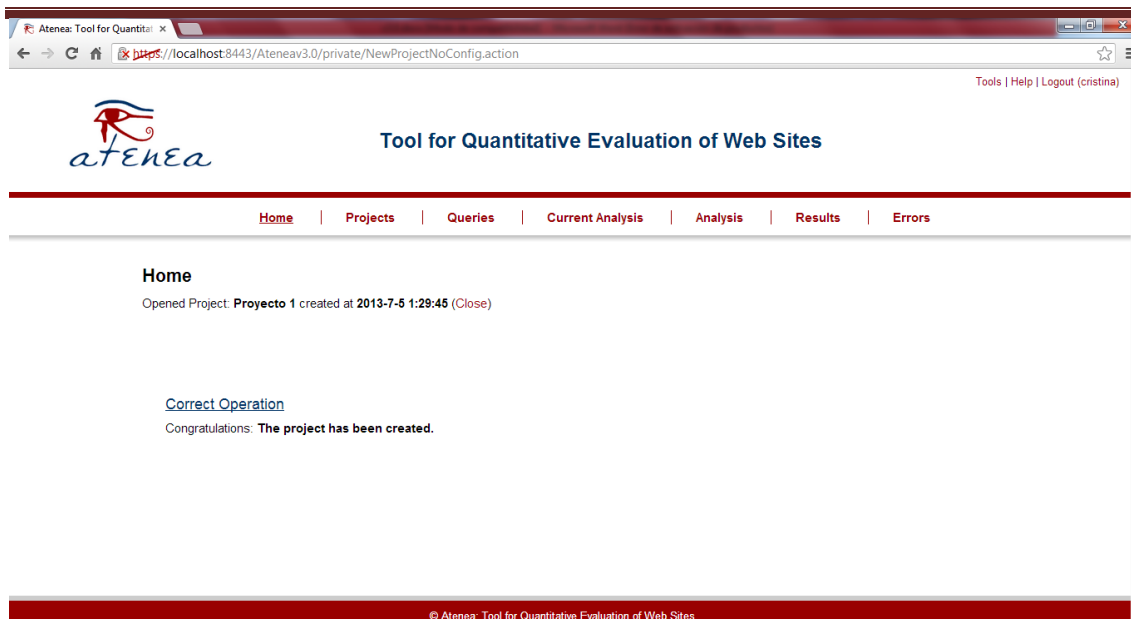


Ilustración 11.40 – Realizar análisis desde la parte privada: proyecto creado correctamente

11.3.2.3 Creación de un análisis

Una vez hayamos creado un proyecto nuevo, tal y como se describe en el punto 11.3.2.2, podremos añadirle un análisis. Para ello vamos a la pestaña *Analysis* donde se nos mostrarán varias opciones entre las que nos interesan dos:

- Evaluate One URL: analizará una sola dirección con los parámetros de profundidad, anchura, crawler y métricas especificadas durante la creación del proyecto.
- Evaluate URL's file: podremos realizar un análisis sobre un conjunto de webs escritas en un fichero .txt con los parámetros establecidos durante la creación del proyecto.

A continuación se describe el procedimiento a seguir para llevar a cabo el análisis con las diferentes opciones.

Evaluate One URL

Para evaluar una única página web basta con introducir la dirección en el campo "URL". El campo "Threads" nos indica el número máximo de análisis que se realizarán simultáneamente.

Para comenzar el análisis pulsamos el botón "Analyze".

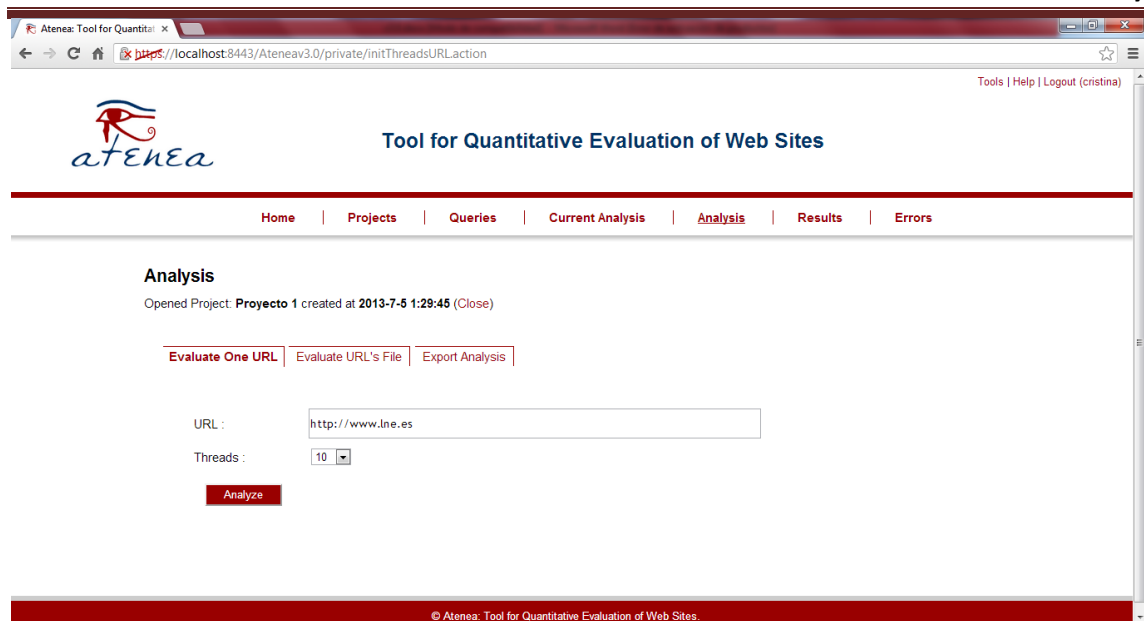


Ilustración 11.41 – Realizar análisis desde la parte privada: analizar una única página web

Evaluate URL's file

Para poder realizar esta tarea debemos tener en nuestro equipo un archivo .txt en el que habremos introducido todas las URL que queramos analizar. Es importante tener en cuenta que las URL deben estar separadas entre sí por un salto de línea.

Seleccionamos el archivo en el que tenemos las URL desde el campo "Urls File". El campo "Threads" nos indica el número máximo de análisis que se realizarán simultáneamente.

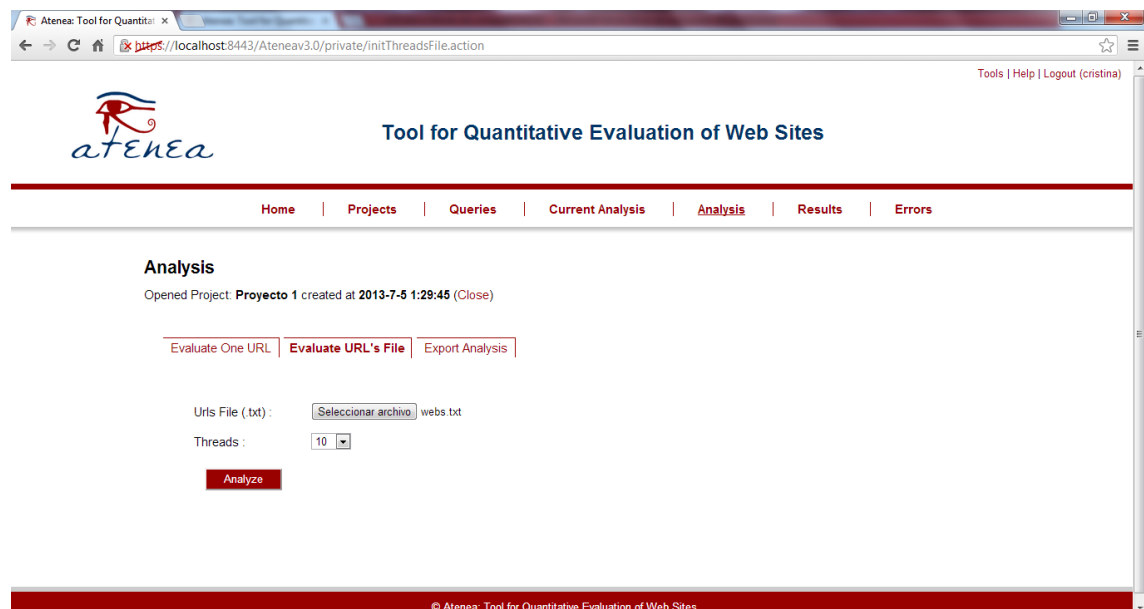


Ilustración 11.42 – Realizar análisis desde la parte privada: analizar un conjunto de páginas web

11.3.2.4 Estado del análisis en curso

Mientras se está realizando un análisis podemos ver el estado en el que se encuentra desde la pestaña “Current Analysis”.

Se mostrará una tabla con la siguiente información:

- Project: proyecto al que pertenece el análisis en curso.
- Date: fecha y hora en las que se inició el análisis.
- Completed: porcentaje completado.
- User: usuario que ha creado el análisis.

Para obtener más información, podemos hacer clic en “Trace” y se nos mostrará un área de texto con las URL del análisis. Aquellas que ya han sido analizadas aparecerán marcadas en rojo y las que están analizándose en ese momento aparecerán en verde.

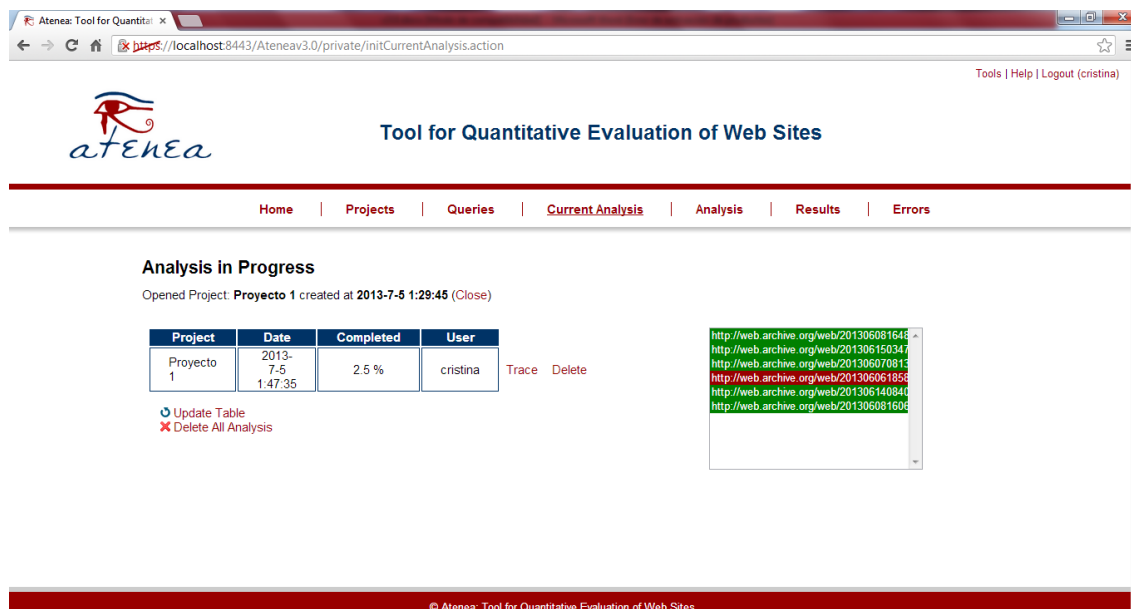
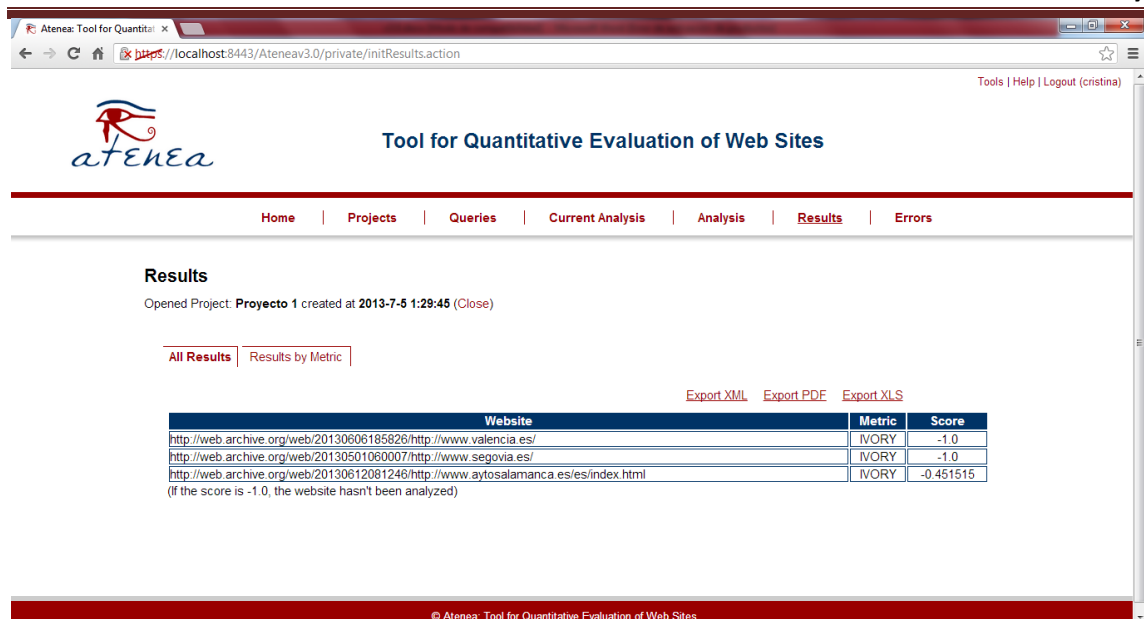


Ilustración 11.43 – Realizar análisis desde la parte privada: estado del análisis en curso

También podremos cancelar el análisis haciendo clic en “Delete”.

11.3.2.5 Resultados del análisis

Ya sea durante el transcurso de un análisis, o una vez éste haya finalizado, podremos visualizar una tabla con los resultados obtenidos para cada página y métrica evaluadas desde la pestaña “Results”.



The screenshot shows a web browser window with the URL `https://localhost:8443/Ateneav3.0/private/initResults.action`. The page title is "Tool for Quantitative Evaluation of Web Sites". The navigation menu includes Home, Projects, Queries, Current Analysis, Analysis, Results (selected), and Errors. The main content area displays "Results" for "Projecto 1" created on 2013-7-5 at 1:29:46. There are tabs for "All Results" (selected) and "Results by Metric". Above the table are links for "Export XML", "Export PDF", and "Export XLS". The table has three columns: "Website", "Metric", and "Score".

Website	Metric	Score
http://web.archive.org/web/20130606185826/http://www.valencia.es/	IVORY	-1.0
http://web.archive.org/web/20130501060007/http://www.segovia.es/	IVORY	-1.0
http://web.archive.org/web/20130612081246/http://www.aytosalamanca.es/es/index.html	IVORY	-0.451515

(If the score is -1.0, the website hasn't been analyzed)

© Atenea: Tool for Quantitative Evaluation of Web Sites

Ilustración 11.44 – Realizar análisis desde la parte privada: visualizar resultados

Desde esta misma pantalla podremos exportar los resultados en formato XML, PDF o XLS.

Es importante saber que si la puntuación obtenida por una página es -1000.0 la página en cuestión no ha sido analizada.

11.4 Manual del Programador

En esta sección se describen los aspectos más importantes a tener en cuenta a la hora de modificar o ampliar la librería Ivory e integrarla en Atenea 3.0.

11.4.1 Creación de nuevos checkpoints

En el caso de que se quisiesen crear nuevos checkpoints, bastaría con crear las clases para los mismos dentro del paquete `IVORY.checkpoints` siguiendo las pautas definidas en el punto 8.1.1 y en la referencia **[Atenea12]**. Además, en caso de que se utilizasen desde la métrica implementada en este trabajo, habría que añadirlos a la colección *checkpoints* de la clase `MetricIvory`.

Es importante tener en cuenta que, tanto para si se añaden nuevos checkpoints a la librería Ivory, como si se añaden dentro de otras librerías integradas en la aplicación Atenea3.0, el nombre de las clases debe ser `CheckpointXXX`, donde XXX es un número mayor que 113, debido a que éste es el número del último checkpoint implementado y el tener checkpoints con los mismos nombres podría generar problemas en el transcurso de los análisis.

11.4.2 Creación de nuevas métricas

Para añadir una nueva métrica a la librería, bastaría con crear la clase que la defina y sus checkpoints tal y como se describe en el apartado 8.1.1 y en la referencia **[Atenea12]**.

11.4.3 Empaquetado

Si se ha modificado algún aspecto de la librería o se han añadido nuevas funcionalidades, será necesario volver a generar el archivo `IVORY.jar` que contiene la métrica y se integra en Atenea 3.0. Para ello basta con seguir los siguientes pasos:

- Compilar todas las clases para obtener los `.class`.
- La librería Ivory utiliza a su vez otras librerías externas por lo que debemos crear un archivo `Manifest.txt` que contendrá el siguiente texto en la primera línea:

```
Class-Path: jStyleParser.jar antlr-runtime-3.1.jar junit-4.4.jar logback-classic-0.9.9.jar logback-core-0.9.9.jar nekohtml.jar slf4j-api-1.5.2.jar xercesImpl.jar xml-apis.jar
```

Estas son las librerías que necesita Ivory actualmente para funcionar correctamente. En caso de necesitar otras, bastará con añadirlas al final de la línea.

Es importante tener en cuenta que el archivo `Manifest.txt` debe estar codificado en UTF-8 sin BOOM y que la última línea debe tener solamente un salto de línea.

- Ahora hay que empaquetar los archivos en un `.jar`. Para hacer esto existen varias opciones, aquí se explica una de las más sencillas.

- Abrimos una ventana de línea de comandos y nos situamos en la carpeta donde tendremos la carpeta IVORY con los archivos .class y ejecutamos la siguiente orden:

```
jar cfm IVORY.jar Manifest.txt IVORY/
```

- Obtendremos un archivo IVORY.jar, en el mismo directorio en el que nos encontramos, que contendrá nuestras clases empaquetadas y el archivo Manifest con la información de las librerías externas.
- Finalmente, para integrar la librería en el sistema Atenea 3.0, basta con copiarla en la carpeta C:\Atenea\jars junto con el resto de librerías externas nuevas que hayamos incluido, si es el caso.

Capítulo 12. Conclusiones y Ampliaciones

y

12.1 Conclusiones

Una vez finalizado el proyecto “Automatización de la medición de la usabilidad web mediante la métrica de Ivory”, podemos decir que se han cumplido los objetivos planteados al inicio del mismo:

- Se ha realizado un estudio de la propuesta de Ivory para la evaluación cuantitativa de la usabilidad en sitios web y la posibilidad de automatizarla.
- Se ha realizado un estudio de la herramienta Atenea 3.0 atendiendo al a funcionalidad que proporciona y a la viabilidad y conveniencia de incorporar en ella la automatización de la métrica de Ivory.
- Se ha llevado a cabo una implementación de la métrica de Ivory y se ha puesto en funcionamiento integrándola en la herramienta Atenea 3.0.
- Se ha realizado un estudio empírico que ha permitido establecer los valores entre los que se mueve la métrica, además de proporcionar una visión global de estado de la usabilidad en la actualidad en páginas web de ayuntamientos, universidades y empresas españolas.

Durante la realización de este proyecto he tenido que trabajar sobre muchos temas que nunca había estudiado en profundidad, lo que me ha supuesto un gran trabajo de documentación e investigación. Esto me ha permitido aprender más sobre aspectos relevantes en la formación de un ingeniero web, como la usabilidad y su evaluación o conocer cómo se llega a enunciar una métrica.

Por último, quisiera destacar la realización del estudio y análisis empírico, tarea a la que me he enfrentado por primera vez y que considero que puede servir para clarificar e ilustrar de forma significativa el trabajo desarrollado.

12.2 Ampliaciones

A continuación se enumera una serie de posibles ampliaciones a realizar sobre este proyecto que no se han llevado a cabo por estar fuera de los objetivos del mismo:

- Ampliar el estudio empírico realizado aumentando el tamaño de las muestras e incluyendo factores como el número de alumnos de las universidades, el tamaño de las ciudades a las que pertenecen los ayuntamientos o el volumen de negocio de las empresas para comprobar si existe alguna relación entre estos factores y el nivel de usabilidad de las páginas web.
- Modificar la aplicación Atenea 3.0 para que sea ella quien realice el análisis de los estilos de las páginas web y no tenga que hacerse en los checkpoints que lo necesiten. De esta manera se mejorará el rendimiento de la aplicación.
- Buscar otra métrica de evaluación cuantitativa de la usabilidad y comparar los resultados que se obtienen al evaluar los mismos sitios web con ambas métricas.

Capítulo 13. Presupuesto

13.1 Presupuesto de costes

El presupuesto de costes se ha hecho en base al presupuesto presentado en la planificación (ver punto 5.2). Cabe destacar que en este presupuesto no se incluye el IVA, pues este impuesto es un porcentaje que debe aplicarse al precio final que se le va a cobrar al cliente.

Así, los costes que supone el desarrollo del proyecto se recogen en la siguiente tabla.

Ítem	Subítem	Concepto	Horas	Precio/unidad	Total
1		Investigación			9.800,00 €
	1	Investigación	245	40,00 €	9.800,00 €
2		Desarrollo de aplicación			9.794,00 €
	1	Análisis	45,5	60,00 €	2.730,00 €
	2	Diseño	32,5	60,00 €	1.950,00 €
	3	Implementación	95,5	38,00 €	3.629,00 €
	4	Pruebas	33	45,00 €	1.485,00 €
3		Estudio empírico			2.650,00 €
	1	Estudio empírico	53	50,00 €	2.650,00 €
Ítem	Subítem	Concepto	Unidades	Precio/unidad	Total
4		Software utilizado			222,53 €
	1	Microsoft Office Home and Business	0,125	237,99 €	29,75 €
	2	SPSS Statistics Professional	0,125	1.390,00 €	173,75 €
	3	Enterprise Architect Profesional	0,125	152,27 €	19,03 €
5		Hardware			125,00 €
	1	PC completo	0,125	1.000,00 €	125,00 €
Ítem	Subítem	Concepto	Meses	Precio/mes	Total
6		Gastos de oficina			483,60 €
		Luz	4	70,00 €	280,00 €
		Conexión a Internet	4	50,90 €	203,60 €
TOTAL					23.075,13 €

Tabla 13.1 – Presupuesto de costes

Hay que tener en cuenta que en el presupuesto que se entrega al cliente no deben aparecer los ítems 4, 5 y 6, con lo cual debemos hacer los cálculos oportunos para averiguar qué suponen los costes de dichos ítems por hora de trabajo. De la misma forma, al realizar el proyecto se debe obtener un beneficio, que se calculará como el 20% del total de costes del

Automatización de la medición de la usabilidad web mediante la métrica de Ivory | Presupuesto

proyecto. Esta cantidad tampoco debe aparecer en el presupuesto para el cliente, así que también calcularemos el dinero correspondiente por hora de trabajo.

Gastos	831,13 €
Software	222,53 €
Hardware	125,00 €
Oficina	483,60 €
	4.615,03 €
Beneficio	
Total horas	504,5
Coste/hora	10,80 €

Tabla 13.2 – Cálculo de costes

Una vez hechos estos cálculos podremos pasar a elaborar el presupuesto para el cliente incluyendo los gastos de manera proporcional en el importe de cada uno de los módulos.

13.2 Presupuesto del cliente

Finalmente, el presupuesto que se entregará al cliente será el siguiente.

Ítem	Subítem	Concepto	Total
1		Investigación	12.444,81 €
	1	Investigación	12.444,81 €
2		Desarrollo de aplicación	12.023,20 €
	1	Análisis	3.221,18 €
	2	Diseño	2.300,84 €
	3	Implementación	4.659,94 €
	4	Pruebas	1.841,24 €
3		Estudio empírico	3.222,14 €
	1	Estudio empírico	3.222,14 €
Subtotal			27.690,16 €
IVA 21%			5.814,93 €
TOTAL			33.505,09 €

Tabla 13.3 – Presupuesto del cliente

Capítulo 14. Referencias Bibliográficas

14.1 Libros y Artículos

[Atenea12] Oyón Fernández, Javier. “Atenea 3.0: Reingeniería del Sistema para la evaluación cuantitativa de Sitios Web”. Escuela de Ingeniería Informática, Universidad de Oviedo. 2012.

[Ivory01] Ivory, Melody Y. “An Empirical Foundation for Automated Web Interface Evaluation”. University of California at Berkeley, USA. 2001.

[IvoryHearst02] Ivory, Melody Y.; Hearst, Marti A. “Statitstical Profiles of Highly-Rated Web Sites”. Conference on Human Factors in Computing Systems, Minneapolis, Minnesota, USA. 2002.

[IvoryRashmiHearst00] Ivory, Melody Y.; Sinha, Rashmi R.; Hearst, Marti A. “Preliminary Findings on Quantitative Measures for Distinguishing Highly Rated Information-Centric Web Pages”. 6th Conference on Human Factors and the Web, Austin, Texas, USA. 2000.

[IvoryRashmiHearst01] Ivory, Melody Y.; Sinha, Rashmi R.; Hearst, Marti A. “Empirically Validated Web Page Design Metrics”. Conference on Human Factors in Computing Systems, Seattle, Washington, USA. 2001.

[Medina10] Medina, N.; Burella, J.; Rossi, G.; Grigera, J.;Luna, E. “An Incremental Approach for Building Accessible and Usable Web Applications”. Web Information Systems Engineering WISE. Springer Berlin Heidelberg. 564-577. 2010.

[Parmant05] B. Parmanto, X. Zeng, “Metric for web accessibility evaluation”. Journal of the American Society for Information Science and Technology. 2005.

[Petrie07] Petrie, H., Kheir, O. “The relationship between accessibility and usability of websites”. Proceedings of CHI. 2007.

[Takay07] Takayuki, W. “Experimental evaluation of usability and accessibility of heading elements”. Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A). Banff, Canada. 2007.

14.2 Referencias en Internet

- [AppServ13] AppServ Open Project. “¿Qué es Appserv?”.
<http://www.appservnetwork.com/modules.php?name=Content&pa=showpage&pid=21>. 2013.
- [Claros06] Claros Gómez, Iván Darío. “Lineamientos de Diseño para el Desarrollo de Aplicaciones Usables bajo Entornos Web”. “Capítulo III, Técnicas y Métricas para la evaluación de la usabilidad”.
http://artemisa.unicauca.edu.co/~iclaros/usabilidad/chapter3.htm#_Toc136093951. 2006.
- [ClickDensity11] ClickDensity; “The Original Heat Map Analytics Tool. Still the Smartest”.
<http://www.clickdensity.com/>. 2013
- [ClickHeat13] LabsMedia; “ClickHeat | Clicks heatmap”,
<http://www.labsmedia.com/clickheat/index.html>. 2013.
- [CIMark12] Optimal Workshop; “Chalkmark”.
<http://www.optimalworkshop.com/chalkmark.htm>. 2013.
- [CITale13] ClickTale; “Visualize Visitors’ Interactions”. <http://www.clicktale.com/>. 2013
- [CrEgg12] CrazyEgg; “The Astonishing Power of Eye Tracking Tecnology... Without the High Costs”. <http://www.crazyegg.com/>. 2013
- [CSSParser04] CSSParser; “CSSParser”. <http://cssparser.sourceforge.net/>. 2013.
- [DHHS13] U.S. Department of Health & Human Services. “Usability.gov: Improving the User Experience”. “How To & Tools”. <http://www.usability.gov/how-to-and-tools/methods/index.html>. 2013.
- [Eclipse13] The Eclipse Project. “Eclipse”. <http://www.eclipse.org/>. 2013.
- [EcuRed12] EcuRed; Evaluación de usabilidad.
http://www.ecured.cu/index.php/Evaluaci%C3%B3n_de_usabilidad. 2012.
- [Hom98] Hom, James; “The Usability Methods Toolbox”. <http://usability.jameshom.com/>. 1998.
- [IvoryPpt01] Ivory, Melody Y. “Appendix C: Web Interface Measures”.
<http://webtango.berkeley.edu/tools/metrics/appc.ppt> . 2001.
- [JMeter13] The Apache Software Foundation. “Apache JMeter”. <http://jmeter.apache.org/>. 2013.
- [JStyleP13] CSSBox. “JStyleParser”. <http://cssbox.sourceforge.net/jstyleparser/>. 2013.
- [JUnit4] JUnit, “JUnit. A programmer-oriented testing framework for Java.” 2013
- [Navflow10] Navflow; “Navflow. Path and conversion analysis for your mocks and wireframes”. <http://www.navflow.com/>. 2013.

[Nielsen95] Nielsen, Jakob; “NN/g Nielsen Norman Group. Evidence-Based User Experience Research, Training and Consulting”, “10 Usability Heuristics for User Interface Design”. <http://www.nngroup.com/articles/ten-usability-heuristics/>. 1995.

[Nielsen12] Nielsen, Jakob; “NN/g Nielsen Norman Group. Evidence-Based User Experience Research, Training and Consulting”, “Usability 101: Introduction to Usability”. <http://www.nngroup.com/articles/usability-101-introduction-to-usability/>. 2013.

[Notepad13] Notepad++; “About”. <http://notepad-plus-plus.org/>. 2013.

[SCSSParser11] Code Project. “Simple CSS Parser”. <http://www.codeproject.com/Articles/20450/Simple-CSS-Parser>. 2013.

[Tomcat13] The Apache Software Foundation. “Apache Tomcat”. <http://tomcat.apache.org/>. 2013.

[UCM] “Capítulo 17. Análisis de correlación lineal: Los procedimientos Correlaciones bivariadas y Correlaciones Parciales”. Apuntes de estadística descriptiva. Universidad Complutense de Madrid.

[Userplus] Userplus; “Userplus”. <http://www.userplus.org/>. 2013.

[UV] “Coeficiente de Pearson”. Apuntes de estadística descriptiva. Universidad de Valladolid. http://www.uv.es/webgid/Descriptiva/31_coeficiente_de_pearson.html

[W3CSchCSS13] W3CSchools; “CSS Introduction”. http://www.w3schools.com/css/css_intro.asp. 2013.

[W3CSchHTML13] W3CSchools; “HTML Introduction”. http://www.w3schools.com/html/html_intro.asp. 2013.

[WebUs] WebUsable; “Manual de las Técnicas de Evaluación y Testing de Usabilidad”; http://webusable.com/useTechniques_B.htm#heuristica

Capítulo 15. Apéndices

15.1 Contenido Entregado en el CD-ROM

15.1.1 Contenidos

15.1.1.1 Estructura general de directorios

Directorio	Contenido
<i>./ Directorio raíz del CD</i>	Contiene un fichero leeme.txt explicando toda esta estructura.
<i>./Ateneav3.0</i>	Contiene toda la estructura de directorios del proyecto para desarrollo.
<i>./instalacion</i>	Ficheros utilizados para la instalación del proyecto.
<i>./documentacion</i>	Contiene toda la documentación asociada al proyecto en formato .doc, .docx y .pdf.
<i>./documentacion/img</i>	Contiene las imágenes utilizadas en la documentación.
<i>./documentacion/uml</i>	Contiene el proyecto de Enterprise Architect con los diagramas realizados.
<i>./documentacion/planificacion</i>	Contiene el proyecto de Microsoft Project en el que se ha elaborado la planificación del proyecto.
<i>./documentacion/excel</i>	Contiene todos los documentos Excel que se han utilizado para realizar los presupuestos y los gráficos y tablas del estudio empírico.
<i>./documentacion/pruebas</i>	Contiene la página web creada para realizar las pruebas.
<i>./herramientas</i>	Contiene los ficheros de instalación de las herramientas utilizadas para el desarrollo o puesta en marcha del proyecto.
<i>./herramientas/desarrollo</i>	Ficheros de instalación de las herramientas utilizadas en el desarrollo
<i>./herramientas/explotacion</i>	Herramientas para la explotación.

Tabla 15.1 – Estructura general de directorios del cd

15.1.1.2 Estructura de Directorios de “desarrollo”

Directorio	Contenido
<i>./ Directorio raíz de “desarrollo”</i>	Contiene los ficheros de proyecto del IDE utilizado.
<i>./doc</i>	Contiene los archivos del javadoc del proyecto.
<i>./conf</i>	Contiene los diferentes ficheros de configuración del proyecto.

<code>./dist</code>	Directorio donde se sitúan los ficheros para la distribución del proyecto. Contiene el .war de Atenea y el .jar de la librería IVORY.
<code>./doc</code>	Contiene toda la documentación relativa al proyecto, incluyendo los ficheros generados por herramientas de generación de documentación automática como <i>Javadoc</i> o similar.
<code>./lib</code>	Bibliotecas externas de las que depende el proyecto.
<code>./src</code>	Ficheros fuente
<code>./web</code>	Este directorio contiene los ficheros (<i>.JSP</i> , <i>.ASPX</i> , <i>.HTML...</i>) de la Web.
<code>./web/images</code>	Contiene las imágenes utilizadas por los ficheros de la web del proyecto.
<code>./classes</code>	Directorio donde se guardan los ficheros compilados.

Tabla 15.2 – Estructura de directorios de desarrollo

15.1.2 Código Ejecutable e Instalación

La librería desarrollada se encuentra dentro del sistema Atenea3.0, por lo que, en primer lugar, es necesario instalar éste siguiendo estos pasos:

1. Instalar AppServ (para tener MySQLServer y PHPMyAdmin)
2. Instalar Tomcat
3. Copiar en la carpeta webapps del Tomcat el archivo Ateneav3.0.war que se encuentra en la carpeta `./instalacion`
4. Crear la base de datos desde PHPMyAdmin utilizando el script `configuracionBBDD.sql` que se encuentra en la carpeta `./instalación`

Si se desea utilizar la parte privada de Ateneav3.0 es necesario crear un archivo de almacén de claves y modificar el archivo `server.xml` de la configuración del Tomcat. Este paso se explica detalladamente en el punto 11.1.5. Cabe destacar, que el script que crea la base de datos introduce un usuario en la aplicación con nombre “cristina” y contraseña “cristina” para iniciar sesión en la parte privada.

Una vez instalado el sistema Atenea, para instalar la librería IVORY es necesario copiar la carpeta “Atenea”, que se encuentra en el directorio `./instalacion`, en `C:/`.

Una vez arranquemos el Tomcat, la aplicación estará lista para su uso.

15.2 Índice Alfabético

A

Atenea 3.0, 5, 7, 19, 20, 23, 48, 53, 54, 55, 56, 57, 60, 67, 71, 75, 80, 90, 94, 105, 130, 132, 140, 141, 143, 144, 149

C

Caso de Uso, 55, 60

Ch

checkpoints, 57, 58, 60, 63, 64, 66, 69, 72, 73, 75, 77, 80, 84, 85, 86, 140, 144

C

Contador de colores, 38, 41
Contador de enlaces, 38, 39
Contador de posición de texto, 41

E

evaluación automática, 23

I

Ivory, 5, 7, 19, 20, 23, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 45, 48, 53, 54, 55, 56, 58, 60, 66, 71, 72, 75, 77, 78, 80, 84, 90, 94, 96, 98, 99, 100, 101,

102, 105, 106, 108, 109, 110, 111, 130, 140, 143, 149, 150

M

métrica de Ivory, 21

N

Nielsen, 32, 46, 47, 151

P

Porcentaje de texto en el cuerpo, 42

R

Requisitos, 54, 55, 111

T

Tamaño de la página, 37, 38, 42
Test unitarios, 91, 92, 93, 94

U

usabilidad, 5, 7, 19, 20, 23, 24, 25, 26, 30, 31, 32, 33, 38, 39, 45, 46, 47, 48, 51, 53, 54, 73, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 108, 109, 110, 130, 143, 144, 150

15.3 Código Fuente

15.3.1 Paquete IVORY.metrics

15.3.1.1 Fichero MetricIVORY.java

```
package IVORY.metrics;

import java.util.ArrayList;

import IVORY.util.CheckpointsNamesEnum;
import IVORY.util.MeasuresEnum;

import utilities.measures.Measure;

import engine.Buffer;
import model.MetricModel;

/**
 * Implementa la fórmula de Ivory para medir la
 * usabilidad de una página web
 *
 * @author Cristina Alonso Sierra
 *
 */
public class MetricIVORY extends MetricModel{

    /**
     * Constructor
     */
    public MetricIVORY(){
        //Asignar el nombre de la métrica
        name = "IVORY";
        //Asignar los checkpoints a la métrica
        checkpoints = new ArrayList<String>();
        checkpoints.add(CheckpointsNamesEnum.CHECK_109.getName());
        checkpoints.add(CheckpointsNamesEnum.CHECK_110.getName());
        checkpoints.add(CheckpointsNamesEnum.CHECK_111.getName());
        checkpoints.add(CheckpointsNamesEnum.CHECK_112.getName());
        checkpoints.add(CheckpointsNamesEnum.CHECK_113.getName());
    }

    /**
     * Aplica la fórmula Ivory y utilizando los valores
     * calculados por los checkpoints y obtiene el valor final
     *
     * @param buffer almacena los valores calculados por los checkpoints
     * @param pagenum número de página
     *
     * @return resultado final
     */
    @Override
    public Object calculate(Buffer buffer, int pagenum) {
        float sumaRanks = 0;
        //Para cada página
        for(int i = 0; i<pagenum; i++){
            //número de enlaces
            Float linkCount = (float)0;
            //contador de posiciones de texto
            Float textPositionCount = (float)0;
            //contador de colores
            Float colorCount = (float)0;
            //tamaño de la página
            Float pageSize = (float)0;
            //porcentaje de texto del cuerpo
            Float bodyTextPercentage = (float)0;

            //Para cada check evaluado
            for(int j = 0; j < pointstomeasure.length; j++){
                String name = pointstomeasure[j];
```

```

        String nameSite = (new
StringBulder(String.valueOf(name))).append(i).toString();
        Measure m = buffer.getMeasure(nameSite);

        //si es el contador de enlaces
        if(m != null &&
name.equals(CheckpointsNamesEnum.CHECK_109.getName().concat("|"))){
            linkCount = (Float)m.getValor(MeasuresEnum.LINK_COUNT.getName());

            //si es el contador de textos centrados
            } else if(m != null &&
name.equals(CheckpointsNamesEnum.CHECK_110.getName().concat("|"))){
                textPositionCount =
(Float)m.getValor(MeasuresEnum.TEXT_POSITION_COUNT.getName());

                //si es el contador de colores
                } else if(m != null &&
name.equals(CheckpointsNamesEnum.CHECK_111.getName().concat("|"))){
                    colorCount =
(Float)m.getValor(MeasuresEnum.COLOR_COUNT.getName());

                    //si es el tamaño
                    } else if(m != null &&
name.equals(CheckpointsNamesEnum.CHECK_112.getName().concat("|"))){
                        pageSize = (Float)m.getValor(MeasuresEnum.TAMANIO.getName());

                        //si es el porcentaje de texto del cuerpo
                        } else if(m != null &&
name.equals(CheckpointsNamesEnum.CHECK_113.getName().concat("|"))){
                            bodyTextPercentage =
(Float)m.getValor(MeasuresEnum.BODY_TEXT.getName());
                        }
                    }

                    //calculamos el valor final para la página
                    Float rank = new Float(0.129)
                        + (new Float(0.002)*linkCount)
                        - (new Float(0.011)*textPositionCount)
                        + (new Float(0.02)*colorCount)
                        + (new Float(0.000000873)*pageSize)
                        + (new Float(0.002)*bodyTextPercentage);
                    sumaRanks += rank;
                }
            //calcula el valor final para el sitio
            return pagenum > 0 ? sumaRanks / pagenum : (float)-1000;
        }
    }
}

```

15.3.2 Paquete IVORY.checkpoints

15.3.2.1 Fichero Checkpoint109.java

```
package IVORY.checkpoints;

import org.w3c.dom.Document;
import org.w3c.dom.NodeList;

import IVORY.util.CheckpointsNamesEnum;
import IVORY.util.MeasuresEnum;

import utilities.measures.ComplexMeasure;
import utilities.measures.Measure;
import model.CheckpointModel;

/**
 * Calcula el número de enlaces que hay en una
 * página web
 *
 * @author Cristina Alonso Sierra
 */
public class Checkpoint109 extends CheckpointModel{

    /**
     * Constructor
     */
    public Checkpoint109(){
        //Asignamos el nombre del checkpoint
        this.name = CheckpointsNamesEnum.CHECK_109.getName();
        //Asignamos una descripción al checkpoint
        this.description = "Cuenta el número de enlaces que hay en una página";
    }

    /**
     * Cuenta el número de enlaces que hay en la página web
     * representada en el {@link Document} que se pasa por
     * parámetro
     *
     * @param d {@link Document} representación de la página web
     * que se está analizando
     * @return {@link Measure} contiene el valor obtenido
     */
    @Override
    public Measure toMeasure(Document d) {
        Measure measure = new
        ComplexMeasure(CheckpointsNamesEnum.CHECK_109.getName());
        NodeList listaNodos = d.getElementsByTagName("a");
        Float valor = new Float(listaNodos.getLength());
        measure.putValor(MeasuresEnum.LINK_COUNT.getName(), valor);
        return measure;
    }
}
```

15.3.2.2 Fichero Checkpoint110.java

```
package IVORY.checkpoints;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.List;
import java.util.Set;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import IVORY.util.CheckpointsNamesEnum;
```

```

import IVORY.util.HtmlTagsEnum;
import IVORY.util.MeasuresEnum;

import cz.vutbr.web.css.CSSFactory;
import cz.vutbr.web.css.NodeData;
import cz.vutbr.web.css.StyleSheet;
import cz.vutbr.web.domassign.Analyzer;
import cz.vutbr.web.domassign.StyleMap;

import utilities.measures.ComplexMeasure;
import utilities.measures.Measure;
import model.CheckpointModel;

/**
 * Cuenta el número de bloques de texto centrado
 * que hay en una página web
 *
 * @author Cristina Alonso Sierra
 *
 */
public class Checkpoint110 extends CheckpointModel{

    /**
     * Constructor
     */
    public Checkpoint110(){
        //Asingamos el nombre del Checkpoint
        this.name = CheckpointsNamesEnum.CHECK_110.getName();
        //Asignamos la descripción del checkpoint
        this.description = "Cuenta el número de bloques de texto con el texto
centrado";
    }

    /**
     * Cuenta el número de bloques de texto con el texto centrado
     * que hay en la página web representada en el {@link Document}
     * que se pasa por parámetro.
     *
     * @param d {@link Document} que representa la página web
     * @return {@link Measure} contiene el valor calculado
     */
    @Override
    public Measure toMeasure(Document d) {
        Measure measure = new
ComplexMeasure(CheckpointsNamesEnum.CHECK_110.getName());

        //Obtener la url del documento analizado
        String url = "";
        NodeList nl = d.getElementsByTagName("Url");
        Element e = (Element) nl.item(0);
        if (e != null) {
            url = e.getAttribute("url");
        }

        //Representación de la hoja de estilos
        StyleSheet css;
        //Mapa de estilos en el dom
        StyleMap map;
        //Contador del número de textos centrados
        Float contador = (float) 0;

        //Obtenemos todos los estilos que se aplican al documento, tanto de hoja
de estilos como con el atributo style
        try {
            css = CSSFactory.getUsedStyles(d, "utf-8", new URL(url),
"screen");

            //Analizar el documento y obtener el mapa
            Analyzer analyzer = new Analyzer(css);
            map = analyzer.evaluateDOM(d, "allowed", true);

            //Claves del mapa
            Set<Element> elementos = map.keySet();
            //Nodo actual
            NodeData nodeData;

            //Para cada elemento del mapa
            for(Element el : elementos){
                //Si se trata de un párrafo o un encabezado

```

```
        if (e1.getLocalName().equals(HtmlTagsEnum.P.getTag()) ||
el.getLocalName().equals(HtmlTagsEnum.H1.getTag()) ||
el.getLocalName().equals(HtmlTagsEnum.H2.getTag()) ||
el.getLocalName().equals(HtmlTagsEnum.H3.getTag()) ||
el.getLocalName().equals(HtmlTagsEnum.H4.getTag()) ||
el.getLocalName().equals(HtmlTagsEnum.H5.getTag()) ||
el.getLocalName().equals(HtmlTagsEnum.H6.getTag())) {
        // Obtener nodo actual
        nodeData = map.get(e1);
        // Obtener propiedades que se aplican al nodo
        List<String> propiedades = (List<String>) nodeData
            .getPropertyNames();

        // Si hay propiedades de estilo sobre el nodo
        if (propiedades.size() > 0) {
            // Buscamos propiedad text-align
            String terminos = nodeData.toString();
            if (terminos.contains("text-align: center")
                || terminos.contains("text-
align:center")
                || terminos.contains("TEXT-
ALIGN: CENTER")
                || terminos.contains("TEXT-
ALIGN:CENTER")) {
                contador++;
            }
        }
    } catch (MalformedURLException e1) {
        e1.printStackTrace();
    }

    measure.putValor(MeasuresEnum.TEXT_POSITION_COUNT.getName(), contador);
    return measure;
}
}
```

15.3.2.3 Fichero Checkpoint111.java

```
package IVORY.checkpoints;

import java.awt.Color;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import IVORY.util.CheckpointsNamesEnum;
import IVORY.util.CssPropertiesEnum;
import IVORY.util.MeasuresEnum;

import cz.vutbr.web.css.CSSFactory;
import cz.vutbr.web.css.NodeData;
import cz.vutbr.web.css.StyleSheet;
import cz.vutbr.web.css.TermColor;
import cz.vutbr.web.domassign.Analyzer;
import cz.vutbr.web.domassign.StyleMap;

import utilities.measures.ComplexMeasure;
import utilities.measures.Measure;
import model.CheckpointModel;
```



```

/**
 * Cuenta el número de colores diferentes que se utilizan
 * en una página web
 *
 * @author Cristina Alonso Sierra
 *
 */
public class Checkpoint111 extends CheckpointModel{

    /**
     * Constructor
     */
    public Checkpoint111(){
        //Asignamos el nombre del checkpoint
        this.name = CheckpointsNamesEnum.CHECK_111.getName();
        //Asignamos una descripción
        this.description = "Cuenta el número de colores diferentes que se
utilizan";
    }

    /**
     * Calcula en número de colores diferentes que se utilizan en la
     * página web que está representada en el {@link Document}
     * que se pasa por parámetro.
     *
     * @param d {@link Document} que representa la página a analizar
     * @return {@link Measure} con el valor obtenido
     */
    @Override
    public Measure toMeasure(Document d) {
        //Crear la medida
        Measure measure = new
ComplexMeasure (CheckpointsNamesEnum.CHECK_111.getName());

        //Obtener la url del documento analizado
        String url = "";
        NodeList nl = d.getElementsByTagName("Url");
        Element e = (Element) nl.item(0);
        if (e != null) {
            url = e.getAttribute("url");
        }

        //Representación de la hoja de estilos
        StyleSheet css;
        //Mapa de estilos en el dom
        StyleMap map;
        //Set con los diferentes colores utilizados en la página
        Set<Color> coloresUtilizados = new HashSet<Color>();

        try {
            //Obtenemos todos los estilos que se aplican al documento, tanto
de hoja de estilos como con el atributo style
            css = CSSFactory.getUsedStyles(d, "utf-8", new URL(url),
"screen");

            //Analizar el documento y obtener el mapa
            Analyzer analyzer = new Analyzer(css);
            map = analyzer.evaluateDOM(d, "allowed", true);

            //Claves del mapa
            Set<Element> elementos = map.keySet();
            //Nodo actual
            NodeData nodeData;

            //Para cada elemento del mapa
            for(Element el : elementos){
                //Obtener nodo actual
                nodeData = map.get(el);
                //Obtener propiedades que se aplican al nodo
                List<String> propiedades = (List<String>)
nodeData.getPropertyNames();

                //Si hay propiedades de estilo sobre el nodo
                if(propiedades.size() > 0){

                    //Buscamos propiedad color
                    TermColor color =
nodeData.getValue(TermColor.class, CssPropertiesEnum.COLOR.getProperty());

```

```

        if(color == null){
            color = nodeData.getValue(TermColor.class,
            CssPropertiesEnum.COLOR.getProperty().toUpperCase());
        }
        if(color != null && color.toString().trim() != ""){
            String colorHex =
color.toString().substring(1);
            coloresUtilizados.add(new
Color(Integer.parseInt(colorHex, 16)));
        }

        //Buscamos propiedad background-color
        TermColor backgroundColor =
nodeData.getValue(TermColor.class, CssPropertiesEnum.BACKGROUND_COLOR.getProperty());
        if(backgroundColor == null){
            backgroundColor =
nodeData.getValue(TermColor.class,
            CssPropertiesEnum.BACKGROUND_COLOR.getProperty().toUpperCase());
        }
        if(backgroundColor != null &&
backgroundColor.toString().trim() != ""){
            String backgroundColorHex =
backgroundColor.toString().substring(1);
            coloresUtilizados.add(new
Color(Integer.parseInt(backgroundColorHex, 16)));
        }

        //Buscamos propiedad border-color
        TermColor borderColor =
nodeData.getValue(TermColor.class, CssPropertiesEnum.BORDER_COLOR.getProperty());
        if(borderColor == null){
            borderColor =
nodeData.getValue(TermColor.class,
            CssPropertiesEnum.BORDER_COLOR.getProperty().toUpperCase());
        }
        if(borderColor != null &&
borderColor.toString().trim() != ""){
            String borderColorHex =
borderColor.toString().substring(1);
            coloresUtilizados.add(new
Color(Integer.parseInt(borderColorHex, 16)));
        }

        //Buscamos propiedad border-left-color
        TermColor borderLeftColor =
nodeData.getValue(TermColor.class, CssPropertiesEnum.BORDER_LEFT_COLOR.getProperty());
        if(borderLeftColor == null){
            borderLeftColor =
nodeData.getValue(TermColor.class,
            CssPropertiesEnum.BORDER_LEFT_COLOR.getProperty().toUpperCase());
        }
        if(borderLeftColor != null &&
borderLeftColor.toString().trim() != ""){
            String borderLeftColorHex =
borderLeftColor.toString().substring(1);
            coloresUtilizados.add(new
Color(Integer.parseInt(borderLeftColorHex, 16)));
        }

        //Buscamos propiedad border-right-color
        TermColor borderRightColor =
nodeData.getValue(TermColor.class, CssPropertiesEnum.BORDER_RIGHT_COLOR.getProperty());
        if(borderRightColor == null){
            borderRightColor =
nodeData.getValue(TermColor.class,
            CssPropertiesEnum.BORDER_RIGHT_COLOR.getProperty().toUpperCase());
        }
        if(borderRightColor != null &&
borderRightColor.toString().trim() != ""){
            String borderRightColorHex =
borderRightColor.toString().substring(1);
            coloresUtilizados.add(new
Color(Integer.parseInt(borderRightColorHex, 16)));
        }

        //Buscamos propiedad border-top-color
        TermColor borderTopColor =
nodeData.getValue(TermColor.class, CssPropertiesEnum.BORDER_TOP_COLOR.getProperty());

```

```

        if(borderTopColor == null){
            nodeData.getValue(TermColor.class,
                CssPropertiesEnum.BORDER_TOP_COLOR.getProperty().toUpperCase());
        }
        if(borderTopColor != null &&
borderTopColor.toString().trim() != ""){
            String borderTopColorHex =
borderTopColor.toString().substring(1);
            coloresUtilizados.add(new
Color(Integer.parseInt(borderTopColorHex, 16));
        }

        //Buscamos propiedad border-bottom-color
        TermColor borderBottomColor =
nodeData.getValue(TermColor.class, CssPropertiesEnum.BORDER_BOTTOM_COLOR.getProperty());
        if(borderBottomColor == null){
            borderBottomColor =
nodeData.getValue(TermColor.class,
                CssPropertiesEnum.BORDER_BOTTOM_COLOR.getProperty().toUpperCase());
        }
        if(borderBottomColor != null &&
borderBottomColor.toString().trim() != ""){
            String borderBottomColorHex =
borderBottomColor.toString().substring(1);
            coloresUtilizados.add(new
Color(Integer.parseInt(borderBottomColorHex, 16));
        }

        //Buscamos propiedad column-rule-color
        TermColor columnRuleColor =
nodeData.getValue(TermColor.class, CssPropertiesEnum.COLUMN_RULE_COLOR.getProperty());
        if(columnRuleColor == null){
            columnRuleColor =
nodeData.getValue(TermColor.class,
                CssPropertiesEnum.COLUMN_RULE_COLOR.getProperty().toUpperCase());
        }
        if(columnRuleColor != null &&
columnRuleColor.toString().trim() != ""){
            String columnRuleColorHex =
columnRuleColor.toString().substring(1);
            coloresUtilizados.add(new
Color(Integer.parseInt(columnRuleColorHex, 16));
        }

        //Buscamos propiedad outline-color
        TermColor outlineColor =
nodeData.getValue(TermColor.class, CssPropertiesEnum.OUTLINE_COLOR.getProperty());
        if(outlineColor == null){
            outlineColor =
nodeData.getValue(TermColor.class,
                CssPropertiesEnum.OUTLINE_COLOR.getProperty().toUpperCase());
        }
        if(outlineColor != null &&
outlineColor.toString().trim() != ""){
            String outlineColorHex =
outlineColor.toString().substring(1);
            coloresUtilizados.add(new
Color(Integer.parseInt(outlineColorHex, 16));
        }
    }
} catch (MalformedURLException e1) {
    e1.printStackTrace();
}

//Guardamos el número de colores utilizados en la medida
measure.putValor(MeasuresEnum.COLOR_COUNT.getName(), new
Float(coloresUtilizados.size()));
return measure;
}
}

```

15.3.2.4 Fichero Checkpoint112.java

```
package IVORY.checkpoints;
```

```

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import IVORY.util.CheckpointsNamesEnum;
import IVORY.util.HtmlTagsEnum;
import IVORY.util.MeasuresEnum;

import utilities.measures.ComplexMeasure;
import utilities.measures.Measure;
import model.CheckpointModel;

/**
 * Mide el tamaño de una página web en bytes
 * incluyendo las imágenes que contenga
 *
 * @author Cristina Alonso Sierra
 *
 */
public class Checkpoint112 extends CheckpointModel{

    /**
     * Constructor
     */
    public Checkpoint112(){
        //Asignamos el nombre del chekcpooint
        this.name = CheckpointsNamesEnum.CHECK_112.getName();
        //Asignamos una descripción
        this.description = "Mide el tamaño de la página en bytes";
    }

    /**
     * Mide el tamaño en bytes de la página web representada
     * en el {@link Document} que se pasa por parámetro
     *
     * @param d {@link Document} que representa la página que
     * se está analizando
     * @return {@link Measure} con el valor obtenido
     */
    @Override
    public Measure toMeasure(Document d) {
        Measure measure = new
ComplexMeasure(CheckpointsNamesEnum.CHECK_112.getName());

        Float bytes = (float) 0;
        //Obtener la url del documento analizado
        NodeList nl = d.getElementsByTagName("Url");
        Element e = (Element) nl.item(0);
        if (e != null) {
            String direccion = e.getAttribute("url");
            URL url;

            //Contar bytes del html
            try {
                url = new URL(direccion);
                InputStream is = url.openStream();
                int leido = 0;
                while(leido != -1){
                    leido = is.read();
                    bytes++;
                }
                is.close();

                //Contar bytes imágenes
                NodeList listaImágenes =
d.getElementsByTagName(HtmlTagsEnum.IMG.getTag());
                Node currentNode;
                URL urlImagen;
                InputStream isImagen;
                for(int i=0; i < listaImágenes.getLength(); i++){
                    currentNode = listaImágenes.item(i);

```

```

        if(currentNode.getAttributes().getNamedItem("src").getNodeValue().startsWith("http://")){
            urlImagen = new
            URL(currentNode.getAttributes().getNamedItem("src").getNodeValue());
        } else {
            urlImagen = new
            URL(direccion.concat("/").concat(currentNode.getAttributes().getNamedItem("src").getNodeValue()));
        }
        isImagen = urlImagen.openStream();
        int leidoImagen = 0;
        while(leidoImagen != -1){
            leidoImagen = isImagen.read();
            bytes++;
        }
        isImagen.close();
    }

    } catch (FileNotFoundException e1) {
        e1.printStackTrace();
    } catch (MalformedURLException e2) {
        e2.printStackTrace();
    } catch (IOException e3) {
        e3.printStackTrace();
    }

}

measure.putValor(MeasuresEnum.TAMANIO.getName(), bytes);
return measure;
}
}
}

```

15.3.2.5 Fichero Checkpoint113.java

```

package IVORY.checkpoints;

import java.util.StringTokenizer;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.w3c.dom.Text;

import IVORY.util.CheckpointsNamesEnum;
import IVORY.util.HtmlTagsEnum;
import IVORY.util.MeasuresEnum;

import utilities.measures.ComplexMeasure;
import utilities.measures.Measure;
import model.CheckpointModel;

/**
 * Calcula el porcentaje del texto de una página web
 * que pertenece al texto.
 *
 * @author Cristina Alonso Sierra
 *
 */
public class Checkpoint113 extends CheckpointModel{

    /**
     * Constructor
     */
    public Checkpoint113(){
        //Asignamos el nombre al checkpoint
        this.name = CheckpointsNamesEnum.CHECK_113.getName();
        //Asignamos una descripción
        this.description = "Porcentaje de texto que pertenece al cuerpo de la
página";
    }
}

```

```

/**
 * Calcula el porcentaje del texto de la página web
 * representada en el {@link Document} que se pasa
 * por parámetro que pertenece al cuerpo.
 *
 * @param d {@link Document} que representa la página
 * web que se está analizando
 * @return {@link Measure} con el valor calculado
 */
@Override
public Measure toMeasure(Document d) {
    Measure measure = new
ComplexMeasure(CheckpointsNamesEnum.CHECK_113.getName());
    float palabrasEncabezados = 0;

    //Texto de toda la página
    NodeList listaNodos = d.getElementsByTagName(HtmlTagsEnum.BODY.getTag());
    float totalPalabras = contarPalabrasElemento((Element)listaNodos.item(0),
0);

    Node currentNode;
    //Texto de los encabezados
    NodeList listaNodosh1 = d.getElementsByTagName(HtmlTagsEnum.H1.getTag());
    for(int i = 0; i < listaNodosh1.getLength(); i++){
        currentNode = listaNodosh1.item(i);
        if(currentNode instanceof Text){
            palabrasEncabezados +=
contarPalabrasTexto(((Text)currentNode).getData());
        } else if(listaNodosh1.item(i) instanceof Element){
            palabrasEncabezados +=
contarPalabrasElemento((Element)currentNode, 0);
        }
    }
    NodeList listaNodosh2 = d.getElementsByTagName(HtmlTagsEnum.H2.getTag());
    for(int i = 0; i < listaNodosh2.getLength(); i++){
        currentNode = listaNodosh2.item(i);
        if(currentNode instanceof Text){
            palabrasEncabezados +=
contarPalabrasTexto(((Text)currentNode).getData());
        } else if(currentNode instanceof Element){
            palabrasEncabezados +=
contarPalabrasElemento((Element)currentNode, 0);
        }
    }
    NodeList listaNodosh3 = d.getElementsByTagName(HtmlTagsEnum.H3.getTag());
    for(int i = 0; i < listaNodosh3.getLength(); i++){
        currentNode = listaNodosh3.item(i);
        if(currentNode instanceof Text){
            palabrasEncabezados +=
contarPalabrasTexto(((Text)currentNode).getData());
        } else if(currentNode instanceof Element){
            palabrasEncabezados +=
contarPalabrasElemento((Element)currentNode, 0);
        }
    }
    NodeList listaNodosh4 = d.getElementsByTagName(HtmlTagsEnum.H4.getTag());
    for(int i = 0; i < listaNodosh4.getLength(); i++){
        currentNode = listaNodosh4.item(i);
        if(currentNode instanceof Text) {
            palabrasEncabezados +=
contarPalabrasTexto(((Text)currentNode).getData());
        } else if(currentNode instanceof Element){
            palabrasEncabezados +=
contarPalabrasElemento((Element)currentNode, 0);
        }
    }
    NodeList listaNodosh5 = d.getElementsByTagName(HtmlTagsEnum.H5.getTag());
    for(int i = 0; i < listaNodosh5.getLength(); i++){
        currentNode = listaNodosh5.item(i);
        if(currentNode instanceof Text){
            palabrasEncabezados +=
contarPalabrasTexto(((Text)currentNode).getData());
        } else if(currentNode instanceof Element){
            palabrasEncabezados +=
contarPalabrasElemento((Element)currentNode, 0);
        }
    }
    NodeList listaNodosh6 = d.getElementsByTagName(HtmlTagsEnum.H6.getTag());

```

```

        for(int i = 0; i < listaNodosh6.getLength(); i++){
            currentNode = listaNodosh6.item(i);
            if(currentNode instanceof Text){
                palabrasEncabezados +=
contarPalabrasTexto(((Text)currentNode).getData());
            } else if(currentNode instanceof Element){
                palabrasEncabezados +=
contarPalabrasElemento((Element)currentNode, 0);
            }
        }
        float palabrasCuerpo = totalPalabras - palabrasEncabezados;

        Float valor;
        if(totalPalabras != 0){
            valor = palabrasCuerpo*(float)100/totalPalabras;
        } else {
            valor = new Float(0);
        }

        measure.putValor(MeasuresEnum.BODY_TEXT.getName(), valor);
        return measure;
    }

    /**
     * Cuenta el número de palabras que hay en el texto
     * que se pasa por parámetro
     *
     * @param texto texto para contar las palabras
     * @return número de palabras del texto
     */
    private int contarPalabrasTexto(String texto){
        if(texto != null && texto.trim() != ""){
            StringTokenizer tokenizer = new StringTokenizer(texto);
            return tokenizer.countTokens();
        }
        return 0;
    }

    /**
     * Cuenta el número de palabras que tiene el {@link Element}
     * que se pasa por parámetro y lo añade al contador
     *
     * @param e {@link Element} del que se cuentan las palabras
     * @param contador almacena las palabras contadas
     * @return palabras contadas
     */
    private int contarPalabrasElemento(Element e, int contador){
        Node currentNode;
        for(int i=0; i<e.getChildNodes().getLength(); i++){
            currentNode = e.getChildNodes().item(i);
            if(currentNode instanceof Text){
                contador +=
contarPalabrasTexto(((Text)currentNode).getData());
            } else if(currentNode instanceof Element) {
                contador = contarPalabrasElemento((Element)currentNode,
contador);
            }
        }
        return contador;
    }
}

```

15.3.3 Paquete IVORY.util

15.3.3.1 Fichero CheckpointNamesEnum.java

```

package IVORY.util;

/**
 * Enumeración que define los nombres de
 * los checkpoints de la métrica Ivory
 *
 * @author Cristina Alonso Sierra

```

```
*  
*/  
public enum CheckpointsNamesEnum {  
  
    CHECK_109("Checkpoint109"),  
    CHECK_110("Checkpoint110"),  
    CHECK_111("Checkpoint111"),  
    CHECK_112("Checkpoint112"),  
    CHECK_113("Checkpoint113");  
  
    //Nombre  
    private String name;  
  
    /**  
     * Constructor  
     * @param name nombre  
     */  
    CheckpointsNamesEnum(String name){  
        this.name = name;  
    }  
  
    /**  
     * Devuelve el nombre  
     * @return nombre  
     */  
    public String getName() {  
        return name;  
    }  
  
}
```

15.3.3.2 Fichero *CssPropertiesEnum.java*

```
package IVORY.util;  
  
/**  
 * Enumeración que define las propiedades  
 * CSS utilizadas  
 *  
 * @author Cristina Alonso Sierra  
 *  
 */  
public enum CssPropertiesEnum {  
  
    COLOR("color"),  
    BACKGROUND_COLOR("background-color"),  
    BORDER_COLOR("border-color"),  
    BORDER_LEFT_COLOR("border-left-color"),  
    BORDER_RIGHT_COLOR("border-right-color"),  
    BORDER_TOP_COLOR("border-top-color"),  
    BORDER_BOTTOM_COLOR("border-bottom-color"),  
    COLUMN_RULE_COLOR("column-rule-color"),  
    OUTLINE_COLOR("outline-color");  
  
    //valor de la propiedad CSS  
    private String property;  
  
    /**  
     * Constructor  
     * @param property valor de la propiedad  
     */  
    CssPropertiesEnum(String property){  
        this.property = property;  
    }  
  
    /**  
     * Devuelve la propiedad  
     * @return  
     */  
    public String getProperty(){  
        return this.property;  
    }  
  
}
```


15.3.3.3 Fichero *HtmlTagsEnum.java*

```
package IVORY.util;

/**
 * Enumeración que define las etiquetas
 * HTML utilizadas
 *
 * @author Cristina Alonso Sierra
 *
 */
public enum HtmlTagsEnum {

    P("p"),
    H1("h1"),
    H2("h2"),
    H3("h3"),
    H4("h4"),
    H5("h5"),
    H6("h6"),
    IMG("img"),
    BODY("body");

    /**
     * Etiqueta
     */
    private String tag;

    /**
     * Constructor
     * @param tag etiqueta
     */
    HtmlTagsEnum(String tag){
        this.tag = tag;
    }

    /**
     * Devuelve la etiqueta
     * @return etiqueta
     */
    public String getTag(){
        return tag;
    }
}
```

15.3.3.4 Fichero *MeasuresEnum.java*

```
package IVORY.util;

/**
 * Enumeración que contiene los nombres de
 * las propiedades medidas
 *
 * @author Cristina Alonso Sierra
 *
 */
public enum MeasuresEnum {

    LINK_COUNT("link_count"),
    TEXT_POSITION_COUNT("text_position_count"),
    COLOR_COUNT("color_count"),
    TAMANIO("tamaño_bytes"),
    BODY_TEXT("body_text_percentage");

    /**
     * Nombre de la medida
     */
    private String name;

    /**
     * Constructor
     * @param name nombre de la medida
     */
    MeasuresEnum(String name){
```

```
        this.name = name;
    }

    /**
     * Devuelve el nombre de la medida
     * @return nombre de la medida
     */
    public String getName() {
        return name;
    }
}
```

15.3.4 Paquete IVORY.test

15.3.4.1 Fichero Checkpoint109Test.java

```
package IVORY.test;

import static org.junit.Assert.assertTrue;

import org.junit.BeforeClass;
import org.junit.Test;

import IVORY.checkpoints.Checkpoint109;
import IVORY.util.MeasuresEnum;

import org.w3c.dom.Document;
import org.w3c.dom.Element;

import syntactic_analysis.Parser;
import utilities.measures.Measure;

/**
 * Implementa los test unitarios para la clase
 * {@link Checkpoint109}
 *
 * @author Cristina Alonso Sierra
 */
public class Checkpoint109Test {

    //Dirección de la página sobre la que se realizan las pruebas
    //private static final String URLPrueba = "http://localhost:8080/prueba1/";
    private static final String URLPrueba = "http://localhost:8080/cineMuyMaloV2/";
    //Valor esperado
    private static final Float NUMERO_ENLACES = (float)4;
    //Checkpoint que se prueba
    private static Checkpoint109 checkpoint109;

    /**
     * Prepara los datos necesarios para ejecutar los
     * test
     */
    @BeforeClass
    public static void prepareTest(){
        checkpoint109 = new Checkpoint109();
    }

    /**
     * Test para el método {@link Checkpoint109#toMeasure(Document)}
     */
    @Test
    public void toMeasureTest() {
        Parser parser = new Parser();
        Document d = parser.parseUrl(URLPrueba);
        Element e = d.createElement("Url");
        e.setAttribute("url", URLPrueba);
        if (d.getLastChild() != null) {
            d.getLastChild().appendChild(e);
        }

        Measure resultado = checkpoint109.toMeasure(d);
    }
}
```

```

        assertTrue("El número de enlaces no se ha contado correctamente",
            NUMERO_ENLACES.equals(resultado

                .getValor(MeasuresEnum.LINK_COUNT.getName())));
    }
}

```

15.3.4.2 Fichero Checkpoint110Test.java

```

package IVORY.test;

import static org.junit.Assert.assertTrue;

import org.junit.BeforeClass;
import org.junit.Test;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

import syntactic_analysis.Parser;
import utilities.measures.Measure;

import IVORY.checkpoints.Checkpoint110;
import IVORY.util.MeasuresEnum;

/**
 * Implementa los test unitarios para la clase
 * {@link Checkpoint110}
 *
 * @author Cristina Alonso Sierra
 *
 */
public class Checkpoint110Test {

    //Dirección de la página sobre la que se está probando
    //private static final String URLPrueba = "http://localhost:8080/prueba1/";
    private static final String URLPrueba = "http://localhost:8080/cineMuyMaloV2/";
    //Valor esperado
    private static final Float NUMERO_BLOQUES_TEXTO_CENTRADO = (float)4;
    //Checkpoint sobre el que se prueba
    private static Checkpoint110 checkpoint110;

    /**
     * Prepara los valores necesarios para
     * ejecutar los test
     */
    @BeforeClass
    public static void prepareTest(){
        checkpoint110 = new Checkpoint110();
    }

    /**
     * Test del método {@link Checkpoint110#toMeasure(Document)}
     */
    @Test
    public void toMeasureTest(){
        Parser parser = new Parser();
        Document d = parser.parseUrl(URLPrueba);
        Element e = d.createElement("Url");
        e.setAttribute("url", URLPrueba);
        if (d.getLastChild() != null) {
            d.getLastChild().appendChild(e);
        }

        Measure resultado = checkpoint110.toMeasure(d);

        assertTrue("El número de bloques de texto con el texto centrado no se ha
            contado correctamente",
            NUMERO_BLOQUES_TEXTO_CENTRADO.equals(resultado.getValor(MeasuresEnum.TEXT_POSITION_COUNT
                .getName())));
    }
}

```

15.3.4.3 Fichero Checkpoint111Test.java

```
package IVORY.test;

import static org.junit.Assert.assertTrue;

import org.junit.BeforeClass;
import org.junit.Test;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

import syntactic_analysis.Parser;
import utilities.measures.Measure;

import IVORY.checkpoints.Checkpoint111;
import IVORY.util.MeasuresEnum;

/**
 * Implementa los test unitarios para la clase
 * {@link Checkpoint111}
 *
 * @author Cristina Alonso Sierra
 *
 */
public class Checkpoint111Test {

    //Dirección de la página sobre la que se hacen las pruebas
    //private static final String URLPrueba = "http://localhost:8080/pruebal/";
    private static final String URLPrueba = "http://localhost:8080/cineMuyMaloV2/";
    //Valor esperado
    private static final Float NUMERO_COLORES = (float)4;
    //Checkpoint que se prueba
    private static Checkpoint111 checkpoint111;

    /**
     * Prepara los valores necesarios para
     * ejecutar los test
     */
    @BeforeClass
    public static void prepareTest(){
        checkpoint111 = new Checkpoint111();
    }

    /**
     * Test para el método {@link Checkpoint111#toMeasure(Document)}
     */
    @Test
    public void toMeasureTest(){
        Parser parser = new Parser();
        Document d = parser.parseUrl(URLPrueba);
        Element e = d.createElement("Url");
        e.setAttribute("url", URLPrueba);
        if (d.getLastChild() != null) {
            d.getLastChild().appendChild(e);
        }

        Measure resultado = checkpoint111.toMeasure(d);

        assertTrue("El número de colores diferentes utilizados no se ha calculado
correctamente",
NUMERO_COLORES.equals(resultado.getValor(MeasuresEnum.COLOR_COUNT.getName())));
    }
}
```

15.3.4.4 Fichero Checkpoint112Test.java

```
package IVORY.test;

import static org.junit.Assert.assertTrue;

import org.junit.BeforeClass;
import org.junit.Test;
import org.w3c.dom.Document;
```

```

import org.w3c.dom.Element;

import syntactic_analysis.Parser;
import utilities.measures.Measure;

import IVORY.checkpoints.Checkpoint112;
import IVORY.util.MeasuresEnum;

/**
 * Implementa los test unitarios para la clase
 * {@link Checkpoint112}
 *
 * @author Cristina Alonso Sierra
 *
 */
public class Checkpoint112Test {

    //Dirección de la página web sobre la que se hacen las pruebas
    //private static final String URLPrueba = "http://localhost:8080/prueba1/";
    private static final String URLPrueba = "http://localhost:8080/cineMuyMaloV2/";
    //Tamaño del html
    private static final Float TAM_HTML = (float)5720;
    //Tamaño de la imagen1
    private static final Float TAM_IMG1 = (float)37404;
    //Tamaño de la imagen2
    private static final Float TAM_IMG2 = (float)46814;
    //Tamaño de la imagen3
    private static final Float TAM_IMG3 = (float)59012;
    //Tamaño de la imagen4
    private static final Float TAM_IMG4 = (float)43505;
    //Checkpoint que se prueba
    private static Checkpoint112 checkpoint112;

    /**
     * Prepara los valores necesarios para
     * ejecutar los test
     */
    @BeforeClass
    public static void prepareTest(){
        checkpoint112 = new Checkpoint112();
    }

    /**
     * Test para el método {@link Checkpoint112#toMeasure(Document)}
     */
    @Test
    public void toMeasureTest(){
        Parser parser = new Parser();
        Document d = parser.parseUrl(URLPrueba);
        Element e = d.createElement("url");
        e.setAttribute("url", URLPrueba);
        if (d.getLastChild() != null) {
            d.getLastChild().appendChild(e);
        }

        Measure resultado = checkpoint112.toMeasure(d);

        Float valor = TAM_HTML + TAM_IMG1 + TAM_IMG2 + TAM_IMG3 + TAM_IMG4;

        assertTrue("El tamaño calculado no es correcto",
        valor.equals(resultado.getValor(MeasuresEnum.TAMANIO.getName())));
    }
}

```

15.3.4.5 Fichero Checkpoint113Test.java

```

package IVORY.test;

import static org.junit.Assert.assertTrue;

import org.junit.BeforeClass;
import org.junit.Test;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

```

```
import syntactic_analysis.Parser;
import utilities.measures.Measure;

import IVORY.checkpoints.Checkpoint113;
import IVORY.util.MeasuresEnum;

/**
 * Implementa los test unitarios para la clase
 * {@link Checkpoint113}
 *
 * @author Cristina Alonso Sierra
 *
 */
public class Checkpoint113Test {

    //Dirección de la página web sobre la que se prueba
    //private static final String URLPrueba = "http://localhost:8080/prueba1/";
    private static final String URLPrueba = "http://localhost:8080/cineMuyMaloV2/";
    //Número de palabras totales de la página de prueba
    private static final float NUM_PALABRAS_TOTAL = 562;
    //Número de palabras de los títulos de la página que se prueba
    private static final float NUM_PALABRAS_TITULOS = 34;
    //Checkpoint sobre el que se prueba
    private static Checkpoint113 checkpoint113;

    /**
     * Prepara los valores necesarios para
     * realizar los test
     */
    @BeforeClass
    public static void prepareTest(){
        checkpoint113 = new Checkpoint113();
    }

    /**
     * Test del método {@link Checkpoint113#toMeasure(Document)}
     */
    @Test
    public void toMeasureTest(){
        Parser parser = new Parser();
        Document d = parser.parseUrl(URLPrueba);
        Element e = d.createElement("Url");
        e.setAttribute("url", URLPrueba);
        if (d.getLastChild() != null) {
            d.getLastChild().appendChild(e);
        }

        Measure resultado = checkpoint113.toMeasure(d);
        float palabrasCuerpo = NUM_PALABRAS_TOTAL - NUM_PALABRAS_TITULOS;
        Float valor = palabrasCuerpo*(float)100/NUM_PALABRAS_TOTAL;

        assertTrue("El porcentaje de texto del cuerpo no se ha calculado correctamente",
            valor.equals(resultado.getValor(MeasuresEnum.BODY_TEXT.getName())));
    }
}
}
```

15.3.4.6 Fichero MetricIVORYTest.java

```
package IVORY.test;

import static org.junit.Assert.assertTrue;

import org.junit.BeforeClass;
import org.junit.Test;

import utilities.measures.ComplexMeasure;
import utilities.measures.Measure;

import engine.Buffer;

import IVORY.metrics.MetricIVORY;
import IVORY.util.CheckpointsNamesEnum;
import IVORY.util.MeasuresEnum;
```

```

/**
 * Implementa los test unitarios para la clase
 * {@link MetricIVORY}
 *
 * @author Cristina Alonso Sierra
 *
 */
public class MetricIVORYTest {

    //Métrica
    private static MetricIVORY metricIVORY;

    /**
     * Prepara los valores necesarios para
     * ejecutar los test
     */
    @BeforeClass
    public static void prepareTest(){
        //crea la métrica
        metricIVORY = new MetricIVORY();
        //añadimos los checkpoints
        String[] pointstomeasure = new String[]{
            CheckpointsNamesEnum.CHECK_109.getName().concat("|"),
            CheckpointsNamesEnum.CHECK_110.getName().concat("|"),
            CheckpointsNamesEnum.CHECK_111.getName().concat("|"),
            CheckpointsNamesEnum.CHECK_112.getName().concat("|"),
            CheckpointsNamesEnum.CHECK_113.getName().concat("|")
        };
        metricIVORY.setPointstomeasure(pointstomeasure);
    }

    /**
     * Test para el método {@link MetricIVORY#calculate(Buffer, int)}
     */
    @Test
    public void calculateTest(){
        Buffer buffer = new Buffer();
        Measure linkCount = new
ComplexMeasure(CheckpointsNamesEnum.CHECK_109.getName().concat("|"));
        linkCount.putValor(MeasuresEnum.LINK_COUNT.getName(), new Float(5));
        buffer.setMeasure(linkCount.getNombreMedida().concat("0"), linkCount);

        Measure textPositioningCount = new
ComplexMeasure(CheckpointsNamesEnum.CHECK_110.getName().concat("|"));
        textPositioningCount.putValor(MeasuresEnum.TEXT_POSITION_COUNT.getName(),
new Float(0));
        buffer.setMeasure(textPositioningCount.getNombreMedida().concat("0"),
textPositioningCount);

        Measure colorCount = new
ComplexMeasure(CheckpointsNamesEnum.CHECK_111.getName().concat("|"));
        colorCount.putValor(MeasuresEnum.COLOR_COUNT.getName(), new Float(10));
        buffer.setMeasure(colorCount.getNombreMedida().concat("0"), colorCount);

        Measure tamaño = new
ComplexMeasure(CheckpointsNamesEnum.CHECK_112.getName().concat("|"));
        tamaño.putValor(MeasuresEnum.TAMANIO.getName(), new Float(5500));
        buffer.setMeasure(tamaño.getNombreMedida().concat("0"), tamaño);

        Measure bodyText = new
ComplexMeasure(CheckpointsNamesEnum.CHECK_113.getName().concat("|"));
        bodyText.putValor(MeasuresEnum.BODY_TEXT.getName(), new Float(80));
        buffer.setMeasure(bodyText.getNombreMedida().concat("0"), bodyText);

        Float valor = new Float(0.129)
+ (new
Float(0.002) * (Float)linkCount.getValor(MeasuresEnum.LINK_COUNT.getName()))
- (new
Float(0.011) * (Float)textPositioningCount.getValor(MeasuresEnum.TEXT_POSITION_COUNT.getNa
me()))
+ (new
Float(0.02) * (Float)colorCount.getValor(MeasuresEnum.COLOR_COUNT.getName()))
+ (new
Float(0.000000873) * (Float)tamaño.getValor(MeasuresEnum.TAMANIO.getName()))
+ (new
Float(0.002) * (Float)bodyText.getValor(MeasuresEnum.BODY_TEXT.getName()));
    }
}

```

```
Float resultado = (Float) metricIVORY.calculate(buffer, 1);
    assertTrue("El valor calculado por la métrica Ivory no es correcto. El
resultado es " + resultado + " en vez de " + valor, valor.equals(resultado));
    }
}
```