



TRABAJO FIN DE GRADO  
INGENIERÍA EN INFORMÁTICA

# Desarrollo de una aplicación para la transcripción de sesiones parlamentarias

---

**Autora**

Elisabet Romero Vaquero

**Tutor**

Juan Manuel Fernández Luna



Escuela Técnica Superior de Ingenierías Informática y de  
Telecomunicación

—  
Granada, Julio de 2015







# Desarrollo de una aplicación para la transcripción de sesiones parlamentarias

---

## **Autora**

Elisabet Romero Vaquero

## **Tutor**

Juan Manuel Fernández Luna



# Desarrollo de una aplicación para la transcripción de sesiones parlamentarias

*Elisabet Romero Vaquero*

**Palabras clave:** XML, DTD, Conversión a PDF, Parlamento, transcribir

## Resumen

---

El desarrollo de este *Trabajo Fin de Grado (TFG)* tiene como objetivo principal la creación de una aplicación capaz de crear un fichero *XML* basado en un *DTD* concreto y cuyo contenido habrá sido aportado por el usuario.

La aplicación directa de este software será la transcripción de los documentos generados en la cámara del Parlamento de Andalucía y la generación de sus correspondientes ficheros *XML* y *PDF*. No obstante, el programa realizado es extensible a cualquier texto que deba ser organizado con una estructura determinada.

Una estructuración inicial en forma de *DTD* indicará al sistema la estructura gráfica a usar para la introducción de datos por parte del usuario para cada uno de los diferentes tipos de documentos a tratar.

De este modo, existirá una interfaz dinámica adaptada a cada tipo de estructura, permitiendo al usuario la creación de un documento adaptado a sus necesidades sin salirse de la estructura preestablecida.

El programa permitirá la exportación a formato *XML* de los datos introducidos, guardando la estructura base de su tipo de documento. Permitirá, además, la exportación a formato *PDF* de estos mismos datos, ateniéndose a las reglas de diseño establecidas por el Parlamento para sus diarios de sesiones y boletines. Además soportará la importación de ficheros *XML* previamente creados por el programa para su modificación y exportación a cualquiera de los formatos comentados anteriormente.

La interfaz para la edición de la información variará en función de las necesidades del usuario, permitiendo la creación, eliminación y modificación de nuevos elementos siguiendo las reglas establecidas por la estructuración *DTD* fijada para ese documento. Contará además con un sistema de eventos de teclado que, mediante conjuntos de teclas determinadas por el propio usuario, permita añadir cadenas de texto al elemento de interfaz con el que se está interactuando en ese momento, agilizando la introducción de texto con un alto grado de repeticiones.



# Development of an application for transcription of parliamentary sessions

*Elisabet Romero Vaquero*

**Keywords:** XML, DTD, PDF conversor, parliament, transcriber

## Abstract

---

This "*final grade work*" (*TFG*) main target is the design and implementation of a system capable of exporting to *XML* and *PDF* files the text structures (templates) designed and introduced by the user, following always the predefined structuring rules. The aim of these functionalities is, mainly, the official bulletin and session diaries of the Andalucía's Parliament, where all this information is currently stored on handwritten files. This program will allow them to organize much better all their document files, saving them future researches and allowing the easy creation of backups, location translation, or any other task related with this data.

More concretely, the system will collect the written information about the session diaries and official bulletins of the Andalucía's Parliament. However, the developed program is extensible to any document type that should be arranged in a specific structure, allowing *DTD* files as input in order to structure the files to generate and files in *CSS* and *XSL* format for the design and creation of *HTML* and *PDF* final documents with the given data.

Each document type to choose by the user will have an initial structure and rules for the former in form of a *DTD* file that is determining the user interface shown in each case. In the same way, it will use *CSS* and *XML* files to give a specific format to each document type, allowing the user to export to a desired document format (*PDF* and/or *HTML*), using the same format for every document type. Finally, each document type will have a final user modifiable configuration that can change the way that the interface shows it elements.

The program will allow exporting to *XML* files the inputted data, saving the base structure of it document type. It will allow, additionally, the export to a *PDF* format these former data, following always the already established design rule by the parliament for their bulletins and sessions diaries. Additionally, it will support the import of *XML* formatted files. This files are previously created by the program for it modification and export to any of the previously mentioned formats.



The user interface for the information edition will vary in order to supply each user needs, allowing the creation, modification and deletion of new elements following the, previously mentioned, *DTD* structure rules for the former document.

The system also have a fast way to show, add, modify and delete keyboard events that, through determined user key sets (what is known as "*shortcuts*") allows the addition of text strings to the interface element which is interacting with in that certain moment, speeding the highest rated repeat grade text input.

The program is being developed under the Java programming language, what allows the usage of the *Java Virtual Machine*. This, makes really easy the multiplatform usage of the developed program, making possible to use it on any operative system supporting the java virtual machine above or equals it 1.7 version (almost every desktop operative system) with a simple copy and paste (no additional compilation is needed due how the *JVM* works) of the former and its libraries.



---

Yo, **Elisabet Romero Vaquero**, alumna de la titulación *Grado en Ingeniería Informática* de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75570827, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

**Fdo:** *Elisabet Romero Vaquero*

Granada a 6 de Julio de 2015.





---

D. **Juan Manuel Fernández Luna**, Profesor área de Ciencia de la Computación e Inteligencia Artificial del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

**Informa:**

Que el presente trabajo, titulado *Desarrollo de una aplicación para la transcripción de sesiones parlamentarias*, ha sido realizado bajo su supervisión por **Elisabet Romero Vauquero**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 6 de Julio de 2015.

**El tutor:**

**Juan Manuel Fernández Luna**

# Agradecimientos

A mi familia y a Iñaki por darme todo su apoyo.





# Índice

---

<b>RESUMEN.....</b>	<b>5</b>
<b>ABSTRACT .....</b>	<b>7</b>
<b>1. INTRODUCCIÓN.....</b>	<b>22</b>
1.1 Motivación .....	23
1.2 Estructura del documento .....	25
1.3 Objetivos del proyecto .....	26
<b>2. PLANIFICACIÓN .....</b>	<b>28</b>
2.1 Fases.....	29
2.1.1 Fase inicial .....	29
2.1.2 Fase de análisis .....	29
2.1.3 Fase de desarrollo .....	30
2.1.4 Fase de prueba y corrección .....	31
2.1.5 Redacción de la memoria .....	31
2.2 Presupuesto .....	33
2.2.1 Recursos.....	33
2.2.2 Costes.....	34
<b>3. ANÁLISIS .....</b>	<b>36</b>
3.1 Especificación de requisitos .....	37
3.1.1 Requisitos funcionales .....	37
3.1.2 Restricciones semánticas .....	47
3.1.3 Requisitos de rendimiento.....	48
3.2 Modelos de caso de uso .....	49
3.2.1 Diagrama de casos de uso .....	49
3.3 Modelos de comportamiento.....	51
<b>4. DISEÑO.....</b>	<b>59</b>
4.1 Diseño de clases .....	60
4.2 Diseño de interfaz .....	65
4.2.1 Área principal.....	65
4.2.2 Módulos .....	66
<b>5. IMPLEMENTACIÓN.....</b>	<b>71</b>
5.1 Herramientas utilizadas.....	72
5.1.1 IDE - Netbeans .....	72
5.1.2 Lenguajes utilizados .....	73
5.1.3 Bibliotecas usadas .....	76

5.2	Fases del desarrollo .....	78
5.2.1	<i>Parseo de DTD</i> .....	78
5.2.2	<i>Metodología para la creación de la interfaz</i> .....	81
5.2.3	<i>Creación de XML</i> .....	82
5.2.4	<i>Sistema de configuración</i> .....	85
5.2.5	<i>Importación y exportación de XML</i> .....	85
5.2.6	<i>Modificación de la interfaz</i> .....	87
5.2.7	<i>Creación de ficheros de formato y estilo</i> .....	87
5.2.8	<i>Exportación de HTML / PDF</i> .....	89
5.2.9	<i>Atajos de teclado</i> .....	90
5.2.10	<i>Diseño final de la interfaz</i> .....	91
<b>6.</b>	<b>CONCLUSIONES .....</b>	<b>110</b>
6.1	Objetivos alcanzados.....	111
6.2	Lecciones aprendidas .....	114
6.3	Vías futuras .....	115
<b>7.</b>	<b>BIBLIOGRAFÍA .....</b>	<b>116</b>
7.1	Enlaces de interés.....	117
<b>8.</b>	<b>ANEXOS.....</b>	<b>118</b>
8.1	Glosario de términos .....	119
8.1.1	<i>Términos</i> .....	119
8.1.2	<i>Acrónimos</i> .....	119
8.2	Manual de usuario .....	121
8.2.1	<i>Antes de empezar</i> .....	121
8.2.2	<i>Creando un nuevo documento</i> .....	121
8.2.3	<i>Importando un documento XML</i> .....	122
8.2.4	<i>Editando el documento en el programa</i> .....	122
8.2.5	<i>Seleccionando ruta de guardado</i> .....	123
8.2.6	<i>Editar configuración del proyecto</i> .....	123
8.2.7	<i>Exportando un documento a XML</i> .....	125
8.2.8	<i>Exportando un documento a PDF</i> .....	126
8.2.9	<i>Sistema de atajos de teclado</i> .....	127

# Índice de figuras

---

## IMÁGENES

Imagen 1 - Diagrama de Gannt .....	32
Imagen 2 - Diagrama de caso de uso.....	50
Imagen 3 - Diagrama de secuencia (Iniciar aplicación) .....	51
Imagen 4 - Diagrama de secuencia (Cerrar programa) .....	51
Imagen 5 - Diagrama de secuencia (Nuevo XML) .....	52
Imagen 6 - Diagrama de secuencia (Exportar XML).....	52
Imagen 7 - Diagrama de secuencia (Importar XML) .....	53
Imagen 8 - Diagrama de secuencia (Crear DTD).....	54
Imagen 9 - Diagrama de secuencia (Añadir elemento XML) .....	55
Imagen 10 - Diagrama de secuencia (Eliminar elemento XML) .....	55
Imagen 11 - Diagrama de secuencia (Crear atajo) .....	56
Imagen 12 - Diagrama de secuencia (Eliminar atajo) .....	56
Imagen 13 - Diagrama de secuencia (Editar atajo) .....	57
Imagen 14 - Exportar atajos).....	57
Imagen 15 - Diagrama de secuencia (Importar atajos).....	58
Imagen 16 - Diagrama de clases de trato de ficheros y configuración.....	60
Imagen 17 - Diagrama de clases de árboles de datos .....	62
Imagen 18 - Diagrama de clases de uso de interfaz .....	63
Imagen 19 - Diseño interfaz – Área principal.....	65
Imagen 20 - Diseño interfaz – Selección de nombre XML.....	66
Imagen 21 - Diseño interfaz –Importar XML sin DTD .....	66
Imagen 22 - Diseño interfaz –Selección DTD existente .....	67
Imagen 23 - Diseño interfaz – Creación de conjunto DTD.....	68
Imagen 24 - Diseño interfaz –Configuración DTD.....	69
Imagen 25 - Diseño interfaz –Mensaje de aviso .....	69
Imagen 26 - Diseño interfaz –Modificación de atajos .....	70
Imagen 27 - Flujo de lectura de un DTD.....	78
Imagen 28 - Lectura de árbol usando tecnología DOM .....	85
Imagen 29 - Exportación de un árbol usando tecnología DOM.....	86
Imagen 30 - Encabezado <i>BOPA</i> .....	87
Imagen 31 - Pie de página <i>BOPA</i> .....	88
Imagen 32 - Encabezado Diario de Sesión.....	88

Imagen 33 -	Código de transformación de XML a PDF .....	90
Imagen 34 -	Interfaz – Área principal .....	91
Imagen 35 -	Interfaz – Menú desplegado .....	92
Imagen 36 -	Interfaz – Tabla de atajos .....	93
Imagen 37 -	Interfaz – Botones inferiores de tabla de atajos .....	94
Imagen 38 -	Interfaz – Fila de tabla de atajos .....	94
Imagen 39 -	Interfaz – Zona XML .....	95
Imagen 40 -	Interfaz – Elemento XML con hijos .....	96
Imagen 41 -	Interfaz – Elemento con botones.....	96
Imagen 42 -	Interfaz –Elemento con botón de eliminar .....	96
Imagen 43 -	Interfaz – Elemento tipo lista .....	97
Imagen 44 -	Interfaz – Elemento tipo línea .....	97
Imagen 45 -	Interfaz – Elemento tipo imagen / tabla .....	97
Imagen 46 -	Interfaz – Elemento tipo texto.....	98
Imagen 47 -	Interfaz- Atributos.....	98
Imagen 48 -	Interfaz – XML importado sin DTD .....	99
Imagen 49 -	Interfaz – Selección de nombre XML.....	100
Imagen 50 -	Interfaz – Selección DTD .....	101
Imagen 51 -	Interfaz – Creación de conjunto DTD.....	102
Imagen 52 -	Interfaz – Configuración DTD .....	104
Imagen 53 -	Interfaz – Selección de fichero .....	106
Imagen 54 -	Interfaz – Guardado de fichero .....	107
Imagen 55 -	Interfaz – Modificación de atajo .....	108
Imagen 56 -	Interfaz – Mensaje de aviso 1 .....	109
Imagen 57 -	Interfaz – Mensaje de aviso 2 .....	109
Imagen 58 -	Manual de usuario – Botón - Nuevo... ..	121
Imagen 59 -	Manual de usuario – Archivo – Nuevo... ..	121
Imagen 60 -	Manual de usuario – Importar XML.....	122
Imagen 61 -	Manual de usuario – Botón de añadir elemento .....	122
Imagen 62 -	Manual de usuario – Botón de eliminar elemento .....	122
Imagen 63 -	Manual de usuario – Selección de ruta de guardado.....	123
Imagen 64 -	Manual de usuario – Editar configuración de proyecto .....	123
Imagen 65 -	Manual de usuario – Selección de proyecto a modificar .....	124
Imagen 66 -	Manual de usuario – Modificación del proyecto .....	124
Imagen 67 -	Manual de usuario – Botón de guardar XML .....	125
Imagen 68 -	Manual de usuario – Archivo – Exportar a XML .....	125
Imagen 69 -	Manual de usuario – Botón de guardar PDF.....	126

Imagen 70 -	Manual de usuario – Archivo – Exportar a PDF.....	126
Imagen 71 -	Manual de usuario – Añadir atajo .....	127
Imagen 72 -	Manual de usuario – Editar un atajo .....	127
Imagen 73 -	Manual de usuario – Eliminar un atajo .....	128
Imagen 74 -	Manual de usuario – Importar atajos.....	128
Imagen 75 -	Manual de usuario – Exportar atajos.....	128

## TABLAS

Tabla 1 -	Estimación de costes .....	35
Tabla 2 -	Comparación JavaFX – Java Swing .....	81



# 1. Introducción

---

---

# 1.1 Motivación

---

El *Parlamento de Andalucía* es uno de los tres órganos que forman la *Junta de Andalucía* (junto con el *Consejo de Gobierno* y la *Presidencia de la Junta*). Este órgano es el encargado del poder legislativo.

Es, por tanto, el órgano facultado para la creación de leyes reguladoras del funcionamiento de la sociedad, la administración de estas y de la concretación del desarrollo de las políticas de carácter público.

Para dejar constancia de estas leyes, el *Parlamento de Andalucía* hace uso de los *boletines oficiales del Parlamento de Andalucía (BOPA)* y los *diarios de sesiones del Parlamento de Andalucía (DSPA)*, documentos oficiales escritos que sirven para llevar un seguimiento del trabajo realizado en el Parlamento.

Más concretamente, los **BOPA** son los diarios dedicados a la publicación de leyes, disposiciones y actos de inserción obligatoria. Y los **DSPA** son los documentos que contienen las reseñas de las intervenciones y discursos acontecidos en las cámaras parlamentarias.

Actualmente, en el *Parlamento de Andalucía*, la transcripción y creación de estos documentos requiere de un proceso dividido en varias fases, detalladas a continuación:

- Grabación de la sesión, pleno o comisión en formato mp3 para su posterior transcripción.
- Transcripción de la conversación grabada a un fichero de texto.

Para ello se usa el programa **MS Word Plantilla**, con el cual se genera un fichero de texto sin apenas diseño, conteniendo la información transcrita.

En el caso de los documentos **BOPA** se genera el fichero de texto haciendo uso de un guión preestablecido con los diferentes apartados que conforman el documento, de modo que se introduce el texto en su apartado correspondiente.

- Creación del documento *PDF* a partir del documento de texto anterior.

En este paso se utiliza el programa **Adobe Indesign**, en el cual ya tienen definidas las macros necesarias para dar formato y estilo al texto introducido, produciendo el documento *PDF* que será mostrado al público.

- Creación del documento *XML* a partir del fichero *PDF* anterior.

Los archivos *PDF* se almacenan en una librería digital, de modo que se puede acceder a cualquiera de ellos de forma rápida y fácil. No obstante, la búsqueda de un bloque concreto de información entre todos los documentos se complica si se hace esta búsqueda sobre los documentos *PDF*. Es por ello por lo que se decidió crear documentos *XML* a partir de estos documentos *PDF* para facilitar la búsqueda.



Este archivo *XML* será utilizado para realizar la búsqueda de toda o parte de la información transcrita, mostrando la información en bloques estructurados facilitados por el archivo *XML* o bien todo el documento *PDF* en el cual se encuentra esa información.

Actualmente, se hace uso del programa creado por el grupo de investigación *TIC-103 "Tratamiento de la Incertidumbre en Inteligencia Artificial"* compuesto por:

- *Luis M. de Campos*
- *Juan M. Fernández-Luna*
- *Juan F. Huete*

Este programa permite el traspaso de información de un documento *PDF* a su correspondiente *XML*, creando un sistema de estructuración acorde a los documentos tratados.

No obstante, a día de hoy el programa no es 100% exacto, de modo que algunos datos contenidos en el *PDF* quedan sin ser bien estructurados, como puede ser el caso de las tablas, por lo que se tiene una pérdida de información con respecto a la contenida en el formato *PDF*.

Estudiando el proceso anterior, se encuentran bastantes defectos que, aunque no son tan importantes como el extravío de la información descrito con anterioridad, dificulta y ralentiza todo el proceso. Ya que está dividido en varias fases, se crea una pérdida de tiempo en el paso entre fases, volviendo el proceso mucho más lento. Esto además se traduce en que el proceso se vuelve tedioso, ya que tareas mecánicas como la obtención de los distintos ficheros debe realizarse manualmente por parte del usuario, cuando podría ser automatizado, aumentando rendimiento y eficacia.

Analizando los pasos y problemas del proceso actual, se propone una nueva idea de proyecto, pensado para facilitar el trabajo al personal del servicio de publicaciones oficiales del *Parlamento de Andalucía* en las transcripciones de sesiones, plenos o comisiones, en la que se aúnan todas las fases desde la transcripción a la generación de ficheros *XML* y *PDF*, evitando así pasos intermedios que ralenticen el trabajo, mejorando considerablemente la eficacia.

Añadiendo a lo comentado con anterioridad, el programa creado permitirá la creación de archivos *XML* a partir de un documento *DTD* preestablecido, junto con los archivos de formato y estilo en caso de querer también exportarlo a *PDF*. Esto amplía enormemente el abanico de actuación del programa, que en nuestro caso se ha centrado sólo en la creación de estos archivos para el Parlamento de Andalucía, dándole un estilo de interfaz de usuario acorde a este, pero que puede fácilmente extenderse a cualquier otro campo o materia.

## 1.2 Estructura del documento

---

Este documento está compuesto por las siguientes partes:

- **Resumen:** Breve resumen de la funcionalidad del proyecto, así como un resumen más extendido de ésta en inglés.
- **Introducción:** Pequeña introducción al proyecto, tratando los siguientes temas:
  - Motivación que ha llevado a la creación del proyecto.
  - Estructura básica del documento.
  - Lista de objetivos a lograr durante la realización del proyecto.
- **Planificación:** Exposición de la planificación del proyecto, las distintas etapas que está compuesto y una estimación de presupuesto del mismo.
- **Análisis:** Fase en la que se especifican los requisitos del sistema y se hace un estudio de los casos de uso y flujos del sistema.
- **Diseño:** Fase en la que se crea una estructuración y diseño inicial de las clases necesarias para el programa y la interfaz de usuario.
- **Implementación:** Aspectos más relevantes de la implementación del proyecto y problemas ocurridos durante esta, junto con las soluciones aplicadas, separados en los siguientes apartados:
  - Herramientas utilizadas, donde se exponen y explican las diversas herramientas empleadas para la creación del proyecto.
  - Fases del desarrollo, donde se explican los diversos pasos seguidos en la implementación, analizando los problemas encontrados y las soluciones propuestas, así como las herramientas utilizadas en cada caso.
  - Diseño de interfaz de usuario, donde se muestra todas las pantallas y módulos de la interfaz y su utilidad y funcionamiento.
- **Conclusiones:** Resumen final del proyecto explicando las conclusiones obtenidas tras la creación de este y posibles vías futuras a las que encaminar el proyecto.
- **Bibliografía:** Lista de fuentes de información utilizadas para la realización del proyecto.
- **Anexos:** Otros documentos de valor a la hora de realizar el proyecto, adjuntados en la propia memoria, así como un glosario de términos y acrónimos.

## 1.3 Objetivos del proyecto

---

Para cubrir las necesidades que han llevado a la creación de este proyecto se necesitan una serie de objetivos. Unos serán obligatorios para el correcto funcionamiento de la aplicación, y otros serán opcionales, que facilitarán la tarea del usuario y realzarán la funcionalidad del programa. A continuación se muestra una lista de aquellos objetivos a lograr con el proyecto, los cuales volverán a ser tratados al final de la memoria, comprobando si se han cumplido y, en caso de no ser así, por qué.

<b>Objetivo 1</b>	<i>Introducción de texto en el programa</i>
<b>Tipo</b>	Obligatorio
<b>Descripción</b>	El programa debe ser capaz de mostrar una interfaz personalizada para la introducción de texto por parte del usuario y de almacenarlo para su posterior trato.

<b>Objetivo 2</b>	<i>Estructuración del texto según un DTD</i>
<b>Tipo</b>	Obligatorio
<b>Descripción</b>	El programa debe ser capaz de estructurar el texto introducido según un DTD preestablecido y seleccionado (directa o indirectamente) por el usuario.

<b>Objetivo 3</b>	<i>Guardado de texto estructurado en XML</i>
<b>Tipo</b>	Obligatorio
<b>Descripción</b>	El programa debe ser capaz de recoger el texto almacenado y exportarlo a formato XML.

<b>Objetivo 4</b>	<i>Guardado de texto en formato PDF</i>
<b>Tipo</b>	Obligatorio
<b>Descripción</b>	El programa debe ser capaz de tomar un XML generado por él mismo y darle un formato y estilo predeterminado para la creación de un archivo PDF.

<b>Objetivo 5</b>	<i>Importación de ficheros XML</i>
<b>Tipo</b>	Opcional
<b>Descripción</b>	Permitir la importación de ficheros XML para su edición, guardado y exportación dentro del mismo programa.

<b>Objetivo 6</b>	<i>Agregación de nuevos métodos de estructuración</i>
<b>Tipo</b>	Opcional
<b>Descripción</b>	Permitir que el usuario introduzca los datos necesarios para la creación de un nuevo conjunto que permita estructurar, dar formato y estilo al texto introducido.

<b>Objetivo 7</b>	<i>Atajos de teclado</i>
<b>Tipo</b>	Opcional
<b>Descripción</b>	Permitir el uso de atajos de teclado para introducir de forma rápida frases predefinidas por el propio usuario dentro del texto que se está tratando.

<b>Objetivo 8</b>	<i>Autocorrección</i>
<b>Tipo</b>	Opcional
<b>Descripción</b>	Marcar aquellas palabras mal escritas por el usuario según un diccionario preestablecido y ofrecer sugerencias de corrección para estas.

<b>Objetivo 9</b>	<i>Diferentes modos de introducción de datos</i>
<b>Tipo</b>	Opcional
<b>Descripción</b>	Permitir introducir información de diferentes formas según el tipo de esta información, teniendo una interfaz diferente para cada tipo. De este modo será diferente el método de introducir en el sistema una tabla, una frase, un párrafo, una imagen, etc.

## 2. Planificación

---

## 2.1 Fases

---

La planificación realizada para el desarrollo del proyecto se dividió en varias partes, para poder abarcar correctamente los distintos campos de acción del proyecto:

### 2.1.1 Fase inicial

Partimos de la necesidad inicial de crear un sistema que permita una fácil transcripción de texto y su conversión a un fichero *XML* y otro *PDF*, siguiendo una estructura determinada. Para esta necesidad se planteó una solución con ayuda de Juan Manuel Fernández Luna, esbozando una idea y requisitos generales que debía cumplir un programa adaptado a esta.

De común acuerdo se decidió que el lenguaje en el que se desarrollaría el proyecto sería *Java*, por su modularidad, robustez y portabilidad, permitiendo que el programa se ejecute correctamente en cualquier sistema operativo sin ningún esfuerzo adicional. Partiendo de este punto se barajaron distintas bibliotecas posibles que ofrecieran solución a los problemas planteados por el programa, dando prioridad siempre a aquellas con licencias menos restrictivas o de código libre.

### 2.1.2 Fase de análisis

En esta etapa se estudió aquellos requisitos básicos para el correcto funcionamiento de la aplicación, así como los recursos, tanto software como hardware, necesarios para cumplir estos requisitos.

#### 2.1.2.1 Especificación de requisitos

Se realizó un estudio de todos los requisitos que debería cumplir el programa, así como de aquellos requisitos opcionales que le aportarían un valor añadido al mismo.

#### 2.1.2.2 Recursos necesarios

Se analizaron las distintas bibliotecas y recursos disponibles y se seleccionaron aquellas que encajaban mejor con los requisitos del sistema.

## 2.1.3 Fase de desarrollo

Esta fase ha sido la más extensa de todas, en la que se ha dado forma al programa en sí.

La fase se dividió en pequeños hitos en los que se iban resolviendo uno por uno los requisitos del programa. Estos hitos fueron:

- Creación de un sistema de interfaz básico que permita introducir la dirección de un archivo *DTD* y muestre la estructura básica de este.
- Creación de un sistema de interfaz dinámico que permita crear un *XML* en blanco a partir de un *DTD*.
- Creación de un sistema de interfaz que permitiera importar *XML* externos, con o sin *DTD*.
- Creación de un sistema de añadir, eliminar y editar atajos de teclado.
- Creación de un sistema de guardado de ficheros *XML* y configuración.
- Creación de un sistema de creación de *PDF*.
- Creación de los ficheros de estilo.
- Estilización del diseño final del programa.

Para lograr estos hitos, fue necesario el estudio y adquisición de conocimientos en las diversas herramientas a tratar a lo largo del proyecto. Más en concreto, fue necesario un estudio adicional en los siguientes apartados:

- ***JavaFX***

Dado que la interfaz más conocida y usada en *Java* (*Swing*) era insuficiente para el programa y poco flexible, se enfocó todo el desarrollo de la interfaz en la tecnología *JavaFX*, que permite mucha más personalización y animación de la interfaz de un programa, mejorando considerablemente su aspecto visual.

Ya que esta tecnología usa el modelo *MVC* (modelo, vista, controlador), no tratado con anterioridad, el estudio consistió en un acercamiento a este tipo de estructuración y la metodología típica de este.

También fue necesario un estudio acerca de las hojas de estilo en cascada (*CSS*), necesarias para dar formato a los elementos de la interfaz, y del formato de los ficheros *FXML*, basados en el lenguaje de marcas *XML*, necesarios para la estructuración de la interfaz.

- **Estructuras *XML* con *DOM***

Puesto que se utilizará la *API* de modelo de objetos del documento (*DOM*), se necesitó una serie de test para comprobar su utilidad y limitaciones para el proyecto.

Ya que *DOM* no trabaja con *DTD* hubo que crear un código por separado para parsear archivos *DTD* y obtener información útil de ellos que se pudiese integrar correctamente con los elementos creados por el *DOM*.

- **Exportando a *PDF***

Tras barajar distintas posibilidades, se escogió un sistema en dos partes para exportar un fichero *XML* a *PDF*, pasando como punto intermedio por un fichero temporal *HTML*.

La primera parte consiste en dar formato al *XML* dependiendo del *DTD* del que partan. Para ello se usa el lenguaje *XSLT*, que toma los valores del *XML* y les aplica una estructura determinada para crear un archivo *HTML*. Este archivo *HTML* incluirá un enlace a una hoja de estilos *CSS*.

La segunda parte toma el archivo *HTML* generado anteriormente y, mediante el uso de la biblioteca *iText* se crea el fichero *PDF* final.

## **2.1.4 Fase de prueba y corrección**

Enlazada con la fase anterior, en la finalización de cada hito se realizaba una prueba de funcionamiento, solucionando los errores que ocurrieran.

## **2.1.5 Redacción de la memoria**

Durante el tiempo que duraron todas las fases anteriores se hizo una recogida de información y datos que, llegando a los últimos hitos a completar del desarrollo se plasmaron y ordenaron correctamente, para conformar la memoria actual.

El proyecto comenzó en Febrero, abarcando 5 meses hasta su finalización, este periodo de tiempo ha estado dividido en las fases expuestas anteriormente según el siguiente diagrama de Gannt:



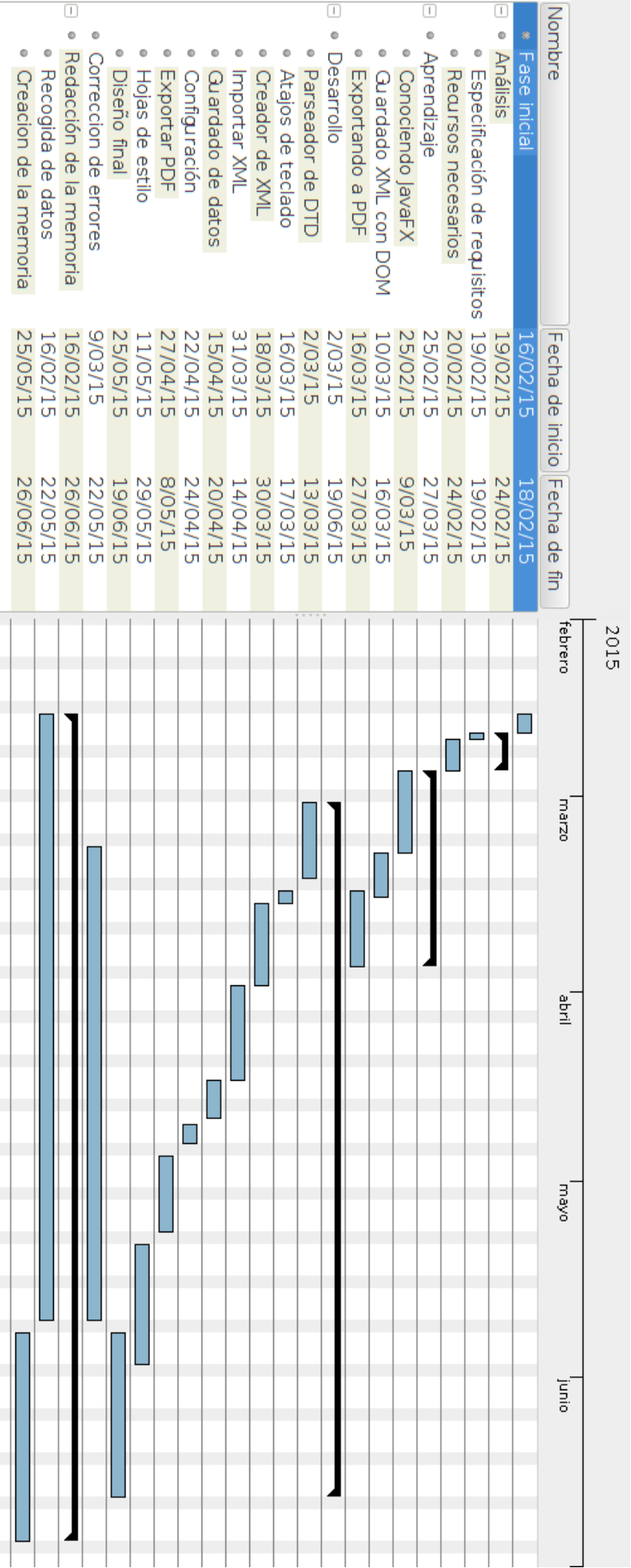


Imagen 1 - Diagrama de Gantt

## 2.2 Presupuesto

---

### 2.2.1 Recursos

En este apartado se van a listar todos los recursos inventariables de hardware y software, así como las herramientas utilizadas y los lenguajes de programación.

#### 2.2.1.1 Hardware

- Laptop donde se ha realizado el proyecto: *ASUS X5DIN, Intel(R) Core(TM)2 Duo CPU - P8800 @ 2.66GHz.*

#### 2.2.1.2 Software

- **Sistema Operativo:** *Ubuntu 14.04 LTS 64 bits*
- **IDE:** *NetBeans 8.0.2*
- **Lenguajes de programación:**
  - *Java 1.7*
- **Bibliotecas:**
  - *JavaFX - v2.2.7*
  - *iText - v2.0.8*
  - *XMLWorker - v2.9.1*
- **Lenguajes de etiquetado:**
  - *XHTML*
  - *FXML*
  - *XML*
- **Lenguaje de estilo:**
  - *XSLT*
  - *CSS3*
- **Diseño de diagrama UML:** *Plugin easyUML*
- **Procesador de texto:** *Microsoft Word*
- **Diseño de diagrama de Gannt:** *GanntProject*

## 2.2.2 Costes

### 2.2.2.1 Licencias

Un apartado a tener en cuenta a la hora de estimar costes de un proyecto son las licencias asociadas a los recursos usados, por ello se ha elaborado una lista con los recursos y sus licencias asociadas, con un enlace a la explicación detallada de cada una:

- **Java 1.7 y JavaFX 2.2.7**
  - Licencia *Oracle BCL*
  - <http://www.oracle.com/technetwork/java/javase/terms/license/index.html>
- **iText**
  - Licencia *AGPL* para productos no comerciales.
  - [http://en.wikipedia.org/wiki/Affero\\_General\\_Public\\_License](http://en.wikipedia.org/wiki/Affero_General_Public_License)

Teniendo en cuenta estas licencias, el coste por el uso de software utilizado en el proyecto es de 0€.

### 2.2.2.2 Recursos materiales

En recursos materiales se tendrá en cuenta el laptop usado para el proyecto.

Los ordenadores suelen tener un período de amortización estipulado entre 2 y 4 años. Para este caso se hará una media y se estipulará un período de 3 años de amortización.

El equipo costó 800€, por lo tanto, se amortizará 267€ al año. Teniendo en cuenta que la duración del proyecto es de 5 meses, el coste final de los recursos materiales asciende a 112€.

### 2.2.2.3 Recursos humanos

Para el cálculo de costes de personal se ha tenido en cuenta un único empleado Programador Senior.

Tomando datos obtenidos para el trabajo anterior en *InfoJobs*, este cobra entre 23.000 y 26.000€ anuales. Teniendo esto en cuenta se hace la correspondencia para el tiempo de desarrollo del proyecto.

Puesto que han sido 5 meses y un trabajador cobra unos 2.042 € mensuales, se estima un salario de 10.210 €.

#### 2.2.2.4 Otros

Dentro de este apartado se engloban costes indirectos como material de mesa, conexión a Internet, luz, gastos de transporte, etc. Su coste suele rondar el 10% del gasto del personal, por lo que se estima un coste aproximado de *1.021€*.

#### 2.2.2.5 Total

A continuación se muestra una tabla resumen con los datos de costes obtenidos en los apartados anteriores, haciendo una estimación del coste mensual del proyecto y su coste total, tanto global como para cada apartado.

Descripción	Coste mensual	Coste total
Licencias de software	0 €	0 €
Estación de trabajo	22,4 €	112 €
Programador Senior	2.042 €	10.210 €
Otros	204.2 €	1.021 €
<b>Total</b>	<b>2.268.6 €</b>	<b>11.343 €</b>

Tabla 1 - Estimación de costes

# 3. Análisis

---

## 3.1 Especificación de requisitos

---

En esta sección se detallan las distintas condiciones y necesidades que ha de satisfacer nuestro producto. Seguiremos para ello un esquema que nos permita describir los requisitos de una forma clara y concisa.

Todos los requisitos se identifican de forma inequívoca mediante un código que constará de una codificación indicando el tipo de requisito y un número de orden. Este código será utilizado como referencia cada vez que sea necesario mencionarlo a lo largo del desarrollo del proyecto.

### 3.1.1 Requisitos funcionales

Para que el programa a crear sea funcional y cumpla las necesidades del usuario, debe cumplir una serie de requisitos. Estos requisitos se pueden dividir en grupos según su función, de modo que se crean 6 grandes grupos:

- En el primero de ellos, relacionado con la aplicación, se encontrarán aquellos requisitos relacionados con el funcionamiento general de la aplicación, tales como abrir o cerrar el programa y tratar los archivos de configuración.
- En el segundo grupo se encuentran aquellos requisitos relacionados con el tratado de documentos *XML* y su interfaz asociada.
- En el tercero se encuentran aquellos requisitos que guardan relación con el tratado de información *DTD* y su interfaz asociada.
- En el cuarto grupo situamos aquellos requisitos encargados del guardado y exportación de datos a ficheros externos.
- En el quinto se sitúan los requisitos necesarios para la correcta creación de un documento *PDF*.
- Por último, en el sexto grupo se encuentran aquellos requisitos encargados de tratar los atajos de teclado y su interfaz asociada.

Siguiendo la guía anterior, agruparemos los 34 requisitos creados en estos 6 grupos de la manera que se muestra a continuación.

### 3.1.1.1 Aplicación

<b>RF1</b>	<i>Iniciar la aplicación.</i>
<b>Explicación</b>	Abre el programa y carga su interfaz básica, así como la interfaz de inicio.
<b>Datos de entrada</b>	Ninguno.
<b>Datos de salida</b>	Interfaz inicial.

<b>RF2</b>	<i>Salir de la aplicación.</i>
<b>Explicación</b>	Cierra correctamente el programa.
<b>Datos de entrada</b>	Ninguno.
<b>Datos de salida</b>	Ninguno.

<b>RF3</b>	<i>Creación de carpetas de configuración.</i>
<b>Explicación</b>	Comprueba la existencia y versión de los ficheros de configuración del programa. En caso de no existir o tener una versión anterior a la existente en el código se crean nuevos ficheros de configuración actualizados.
<b>Datos de entrada</b>	Ficheros de configuración existentes dentro del código del programa.
<b>Datos de salida</b>	Genera, en caso de ser necesario, ficheros de configuración en la carpeta propia del programa dentro del sistema de carpetas del usuario.

<b>RF4</b>	<i>Leer configuración.</i>
<b>Explicación</b>	Permite establecer los parámetros básicos del programa según su fichero de configuración asociado.
<b>Datos de entrada</b>	Necesitará la existencia de un fichero de configuración. El cual se asegura tras la ejecución de <b>RF3</b> .
<b>Datos de salida</b>	Guarda la información de la configuración en el sistema.

### 3.1.1.2 XML

<b>RF5</b>	<i>Nuevo XML.</i>
<b>Explicación</b>	Pide un nombre de <i>XML</i> a partir del cual obtiene el <i>DTD</i> del cual debe formarse el <i>XML</i> .
<b>Datos de entrada</b>	Nombre que tendrá el nuevo fichero <i>XML</i> . Contiene el nombre del <i>DTD</i> a usar, un número de hasta dos cifras y un segundo número de hasta 4 cifras.
<b>Datos de salida</b>	Nombre del <i>DTD</i> a usar.

<b>RF6</b>	<i>Mostrar la interfaz de usuario de un XML vacío.</i>
<b>Explicación</b>	Muestra una interfaz con elementos de entrada adaptados al <i>XML</i> introducido y su <i>DTD</i> asociado.
<b>Datos de entrada</b>	Árbol del <i>DTD</i> asociado que contiene el sistema.
<b>Datos de salida</b>	Panel que contiene todo el árbol de interfaz generado para el <i>XML</i> asociado al <i>DTD</i> introducido.

<b>RF7</b>	<i>Importar un XML.</i>
<b>Explicación</b>	Lee un fichero <i>XML</i> , obteniendo su árbol de información. Busca también el <i>DTD</i> asociado, mostrando un mensaje en caso de no existir que permite elegir entre ejecutar el <b>RF8.</b> , <b>RF9.</b> , <b>RF14.</b> o cancelar la acción.
<b>Datos de entrada</b>	Fichero <i>XML</i> del cual se quiere mostrar su información.
<b>Datos de salida</b>	Árbol <i>XML</i> guardado en el sistema y nombre del <i>DTD</i> asociado en caso de existir, o interfaz de mensaje de selección de siguiente acción en caso contrario.



<b>RF8</b>	<i>Mostrar la interfaz de usuario de un XML importado sin DTD.</i>
<b>Explicación</b>	Muestra el conjunto de elementos de interfaz obtenidos de la estructura del XML introducido.
<b>Datos de entrada</b>	Árbol de XML guardado en el sistema.
<b>Datos de salida</b>	Panel que contiene todo el árbol de interfaz generado para el XML introducido.

<b>RF9</b>	<i>Mostrar la interfaz de usuario de un XML importado con DTD.</i>
<b>Explicación</b>	Muestra el conjunto de elementos de interfaz obtenidos del DTD introducido con la información del XML introducido.
<b>Datos de entrada</b>	Árbol DTD contenido en el sistema y árbol del XML contenido en el sistema.
<b>Datos de salida</b>	Panel que contiene todo el árbol de interfaz generado por el DTD y el XML introducidos.

<b>RF10</b>	<i>Añadir elemento XML.</i>
<b>Explicación</b>	Añade un elemento a la interfaz de usuario y al árbol XML guardado en el sistema.
<b>Datos de entrada</b>	Elemento a introducir y posición donde colocarlo.
<b>Datos de salida</b>	Panel con la interfaz necesaria para el nuevo elemento introducido.

<b>RF11</b>	<i>Eliminar elemento XML.</i>
<b>Explicación</b>	Elimina un elemento de la interfaz de usuario y del árbol XML guardado en el sistema.
<b>Datos de entrada</b>	Elemento a eliminar.
<b>Datos de salida</b>	Ninguno.

<b>RF12</b>	<i>Modificar el XML almacenado con la información de la interfaz.</i>
<b>Explicación</b>	Toma los datos introducidos en la interfaz de usuario y los almacena en su posición correspondiente en el árbol <i>XML</i> guardado por el sistema.
<b>Datos de entrada</b>	Datos de la interfaz de usuario y el árbol <i>XML</i> guardado por el sistema.
<b>Datos de salida</b>	Árbol <i>XML</i> actualizado que se almacenará en el sistema.

### 3.1.1.3 DTD

<b>RF13</b>	<i>Almacenar el DTD asociado a un XML.</i>
<b>Explicación</b>	Guarda en la configuración el <i>DTD</i> usado actualmente.
<b>Datos de entrada</b>	Nombre del <i>DTD</i> .
<b>Datos de salida</b>	Actualiza la configuración del sistema.

<b>RF14</b>	<i>Crear DTD.</i>
<b>Explicación</b>	Lee el <i>DTD</i> guardado en la configuración y crea el árbol asociado a este, tomando como valores aquellos que tienen el fichero de configuración general y los del fichero de configuración de ese <i>DTD</i> .
<b>Datos de entrada</b>	Fichero de configuración, en caso de existir, del <i>DTD</i> y el fichero de configuración general.
<b>Datos de salida</b>	Árbol <i>DTD</i> con información sobre su configuración que se guardará en el sistema.

<b>RF15</b>	<i>Seleccionar DTD en caso de no existir en el XML seleccionado.</i>
<b>Explicación</b>	Te permite seleccionar entre la lista de nombres obtenidos en <b>RF16</b> , uno de ellos para usarlo como <i>DTD</i> para el <i>XML</i> .
<b>Datos de entrada</b>	Lista de <i>DTD</i> existentes en la carpeta de configuraciones del programa.
<b>Datos de salida</b>	Nombre del <i>DTD</i> seleccionado.

<b>RF16</b>	<i>Mostrar lista de DTD existentes.</i>
<b>Explicación</b>	Hace un recorrido de los ficheros existentes en la carpeta de <i>DTD</i> del programa y muestra una lista de los nombres de los <i>DTD</i> existentes.
<b>Datos de entrada</b>	Dirección del directorio donde se encuentran los <i>DTD</i> almacenados por el programa.
<b>Datos de salida</b>	Lista de nombres <i>DTD</i> existentes.

<b>RF17</b>	<i>Crear un nuevo DTD apto para el programa.</i>
<b>Explicación</b>	Muestra una interfaz para el guardado del conjunto de ficheros necesarios para la creación de un nuevo <i>DTD</i> . Contiene los requisitos <b>RF18.</b> , <b>RF19.</b> , <b>RF20.</b> , <b>RF21.</b> , <b>RF22.</b> y <b>RF23.</b>
<b>Datos de entrada</b>	Ninguno.
<b>Datos de salida</b>	Nombre del <i>DTD</i> generado.

<b>RF18</b>	<i>Asignar nombre a DTD.</i>
<b>Explicación</b>	Recoge los datos de la interfaz para asignarle un nombre a todos los ficheros necesarios para el guardado de un nuevo <i>DTD</i> , que estará basado en estos datos.
<b>Datos de entrada</b>	Elemento de interfaz donde el usuario introducirá el nombre del nuevo <i>DTD</i> .
<b>Datos de salida</b>	Nombre del <i>DTD</i> seleccionado.

<b>RF19</b>	<i>Mostrar interfaz de modificación de la configuración de un DTD.</i>
<b>Explicación</b>	Muestra la interfaz correspondiente al árbol <i>DTD</i> guardado por el sistema e información sobre su configuración.
<b>Datos de entrada</b>	Árbol <i>DTD</i> guardado en el sistema.
<b>Datos de salida</b>	Árbol <i>XML</i> con la información de configuración del <i>DTD</i> introducido.

### 3.1.1.4 Guardado de datos

<b>RF20</b>	<i>Guardar DTD.</i>
<b>Explicación</b>	Almacena el árbol <i>DTD</i> guardado en el sistema en un fichero externo, cuya dirección ha sido dada.
<b>Datos de entrada</b>	Árbol <i>DTD</i> guardado en el sistema y dirección del fichero en donde se almacenará.
<b>Datos de salida</b>	Fichero con extensión “ <i>.dtd</i> ” con la información contenida en el árbol de <i>DTD</i> .

<b>RF21</b>	<i>Guardar XML de configuración de DTD.</i>
<b>Explicación</b>	Almacena el árbol de configuración de <i>DTD</i> existente en la configuración del programa en un fichero cuya dirección ha sido dada.
<b>Datos de entrada</b>	Árbol de configuración del <i>DTD</i> existente en la configuración del programa y la dirección del fichero en el que se guardarán los datos.
<b>Datos de salida</b>	Fichero con extensión “ <i>.xml</i> ” que contiene la información de configuración del <i>DTD</i> .

<b>RF22</b>	<i>Guardar XSLT.</i>
<b>Explicación</b>	Copia un archivo con extensión “.xsl” en la carpeta destinada al <i>DTD</i> al que está asociado, cuya dirección se le dará a la función.
<b>Datos de entrada</b>	Fichero con extensión “.xsl” a copiar y directorio del <i>DTD</i> asociado donde se copiará.
<b>Datos de salida</b>	Fichero con extensión “.xsl” copia del fichero pasado por parámetro que se localizará en la dirección indicada.

<b>RF23</b>	<i>Guardar CSS.</i>
<b>Explicación</b>	Copia un archivo con extensión “.css” en la carpeta destinada al <i>DTD</i> al que está asociado, cuya dirección se le dará a la función.
<b>Datos de entrada</b>	Fichero con extensión “.css” a copiar y directorio del <i>DTD</i> asociado donde se copiará.
<b>Datos de salida</b>	Fichero con extensión “.css” copia del fichero pasado por parámetro que se localizará en la dirección indicada.

<b>RF24</b>	<i>Exportar XML.</i>
<b>Explicación</b>	Toma la información almacenada en el árbol de <i>XML</i> del sistema y lo guarda como fichero en la dirección indicada.
<b>Datos de entrada</b>	Árbol <i>XML</i> guardado en el sistema y dirección del fichero en el que se guardará este.
<b>Datos de salida</b>	Fichero con la información del árbol <i>XML</i> situado en la dirección indicada.

<b>RF25</b>	<i>Exportar a PDF.</i>
<b>Explicación</b>	Toma el árbol <i>XML</i> guardado en el sistema y lo convierte en un fichero con extensión “.pdf”, haciendo uso de los requisitos <b>RF24.</b> , <b>RF26.</b> y <b>RF27.</b>
<b>Datos de entrada</b>	Árbol <i>XML</i> guardado en el sistema y dirección del fichero en el que se guardará el <i>PDF</i> .
<b>Datos de salida</b>	Fichero <i>PDF</i> generado.

### 3.1.1.5 PDF

<b>RF26</b>	<i>Creación de XHTML.</i>
<b>Explicación</b>	Convierte un fichero <i>XML</i> en uno <i>XHTML</i> haciendo uso del fichero <i>XSL</i> asociado al <i>DTD</i> especificado en el <i>XML</i> .
<b>Datos de entrada</b>	Fichero que contiene un árbol <i>XML</i> , archivo con extensión “. <i>xsl</i> ” asociado a este y dirección en la que se guardará el fichero <i>XHTML</i> generado.
<b>Datos de salida</b>	Fichero con extensión “. <i>xhtml</i> ”.

<b>RF27</b>	<i>Creación del fichero PDF.</i>
<b>Explicación</b>	Convierte un fichero con extensión “. <i>xhtml</i> ” en un fichero <i>PDF</i> , haciendo uso de la hoja de estilos <i>CSS</i> asociada a este.
<b>Datos de entrada</b>	Fichero con extensión “. <i>xhtml</i> ”, archivo con extensión “. <i>css</i> ” y dirección en la que se guardará el fichero <i>PDF</i> generado.
<b>Datos de salida</b>	Fichero <i>PDF</i> generado en la localización indicada.

### 3.1.1.6 Atajos de teclado

<b>RF28</b>	<i>Crear un atajo de teclado.</i>
<b>Explicación</b>	Muestra la interfaz de creación de un atajo de teclado para la introducción de datos por parte del usuario.
<b>Datos de entrada</b>	Ninguno.
<b>Datos de salida</b>	Interfaz mostrando los datos necesarios para la creación de un nuevo atajo de teclado.

<b>RF29</b>	<i>Guardar un atajo de teclado.</i>
<b>Explicación</b>	Toma los datos de la interfaz y los almacena como un atajo en el sistema. Si este atajo ya existía con anterioridad se modifican los datos de este.
<b>Datos de entrada</b>	Interfaz con la información sobre el atajo de teclado.
<b>Datos de salida</b>	Datos del atajo guardados en el sistema.

<b>RF30</b>	<i>Editar un atajo de teclado.</i>
<b>Explicación</b>	Toma los datos de un atajo y los muestra en una interfaz para su modificación.
<b>Datos de entrada</b>	Datos contenidos en el sistema sobre el atajo a modificar.
<b>Datos de salida</b>	Interfaz mostrando los datos necesarios para la edición de un atajo de teclado.

<b>RF31</b>	<i>Eliminar un atajo de teclado.</i>
<b>Explicación</b>	Elimina los datos existentes sobre un atajo en el sistema.
<b>Datos de entrada</b>	Datos del atajo que se va a eliminar.
<b>Datos de salida</b>	Ninguno.

<b>RF32</b>	<i>Mostrar lista de atajos.</i>
<b>Explicación</b>	Muestra una lista de todos los atajos contenidos en el sistema e información acerca de ellos.
<b>Datos de entrada</b>	Datos de atajos contenidos en el sistema.
<b>Datos de salida</b>	Lista de nombres e información de los atajos guardados en el sistema.

<b>RF33</b>	<i>Exportar atajos de teclado.</i>
<b>Explicación</b>	Toma los datos de atajos contenidos en el sistema y los guarda en forma de fichero <i>XML</i> en la dirección indicada.
<b>Datos de entrada</b>	Datos de atajos contenidos en el sistema y dirección en el que se guardarán.
<b>Datos de salida</b>	Fichero <i>XML</i> que contiene la información de los atajos contenidos en el sistema.

<b>RF34</b>	<i>Importar atajos de teclado.</i>
<b>Explicación</b>	Toma los datos existentes en un fichero <i>XML</i> y los guarda como atajos en el sistema.
<b>Datos de entrada</b>	Fichero <i>XML</i> con la información de atajos.
<b>Datos de salida</b>	Datos de atajos almacenados en el sistema.

### 3.1.2 Restricciones semánticas

En este apartado se muestran aquellas restricciones semánticas que afectan a los requisitos funcionales expuestos con anterioridad:

- No puede introducirse nunca una dirección de fichero inválida. Esto se chequeará con anterioridad y, en caso de ser incorrecto, se finalizará la función sin completarse.
- En caso de indicar la dirección de un fichero inexistente para guardar un dato, se creará la ruta de directorios y a continuación el fichero.
- Los ficheros introducidos deben tener una estructura correcta para el uso que se le van a dar, en caso contrario se terminará la función sin completarse.
- Los datos introducidos para el nombre del *XML* deben ser correctos, no permitiendo la continuación al siguiente paso hasta la formación correcta de este.
- Un fichero *XML* debe tener un fichero *DTD* asociado para poder exportarse como *PDF*, en caso de no ser así se finalizará la función sin completarse.



- Los datos introducidos para la creación o edición de un atajo deben estar limitados, teniendo que tener la estructura de una a tres teclas especiales (*Ctrl*, *Shift* y/o *Alt*) y una letra sin acentuar existente y que no sea un carácter especial.
- No se permitirá eliminar un elemento de la interfaz de *XML* si este tiene como cantidad en su elemento *DTD* asociado de como mínimo uno (indicado en el *DTD* con el símbolo “+” o si no tiene símbolo).
- No se podrá añadir un elemento a la interfaz de *XML* si este ya existe y tiene como cantidad en su elemento *DTD* asociado de como máximo uno (indicado en el *DTD* con el símbolo “?” o si no tiene símbolo).
- Toda la interfaz debe ser modular, de forma que sea simple y rápida la adición de nuevas ventanas.

### 3.1.3 Requisitos de rendimiento

El rendimiento de la aplicación debe ser tal que no entorpezca con la acción del usuario, exceptuando los casos de exportado e importado de datos en los que el sistema puede ralentizarse unos segundos hasta la finalización de dicha tarea.

## 3.2 Modelos de caso de uso

---

Para describir los distintos comportamientos que tendrá el sistema, usaremos el lenguaje de modelado de sistemas UML; que representa los requisitos funcionales del sistema, centrandose en que hace y no cómo lo hace.

### 3.2.1 Diagrama de casos de uso

#### 3.2.1.1 Actores

En nuestro programa solo contamos con un actor: El usuario final. Este es el encargado de llamar a las diferentes funciones del sistema mediante el uso de la interfaz a lo largo de su interacción con el programa.

#### 3.2.1.2 Diagrama

En la siguiente figura se muestra la interacción del usuario con el sistema estructurado como diagrama de caso de uso, en la que el sistema utiliza funciones internas para realizar todas las peticiones del usuario y dando como resultado normalmente, una modificación de la interfaz con la que este interactúa.

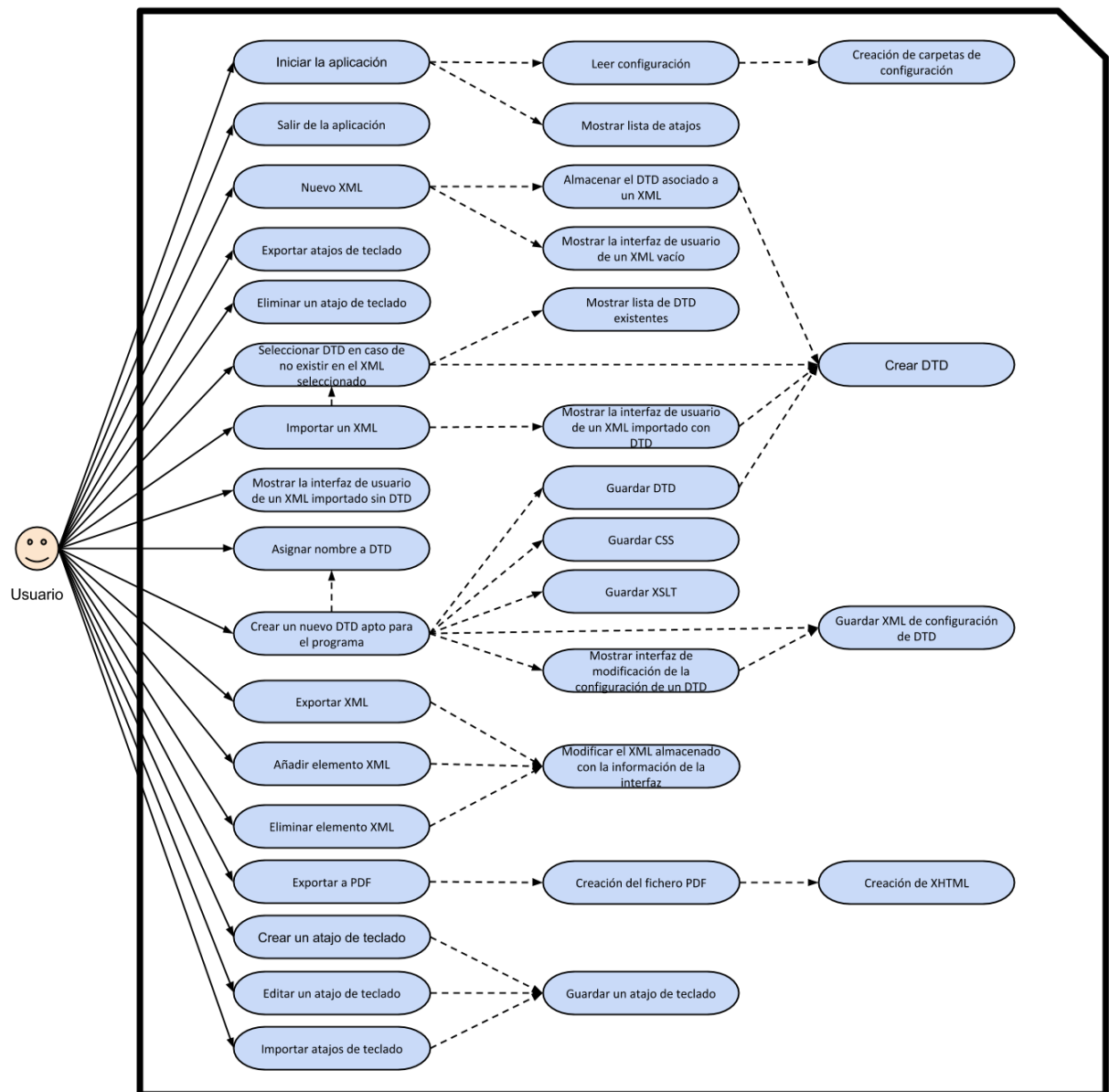


Imagen 2 - Diagrama de caso de uso

Como podemos observar en la imagen anterior, el usuario interactúa con ciertas funciones del sistema (flechas continuas) y él, internamente, interactúa con el resto (flechas con líneas discontinuas), llegando a realizar la totalidad de las funcionalidades requeridas para el programa.

## 3.3 Modelos de comportamiento

Para el correcto funcionamiento de la aplicación, esta debe seguir una serie de pasos al realizar funciones determinadas, llamadas por el usuario. En este apartado veremos con detalle estas funciones y su comportamiento a lo largo de su ejecución.

- **Iniciar aplicación:** Se ejecuta al abrir el usuario el programa, cargando los ficheros de configuración y los atajos guardados.

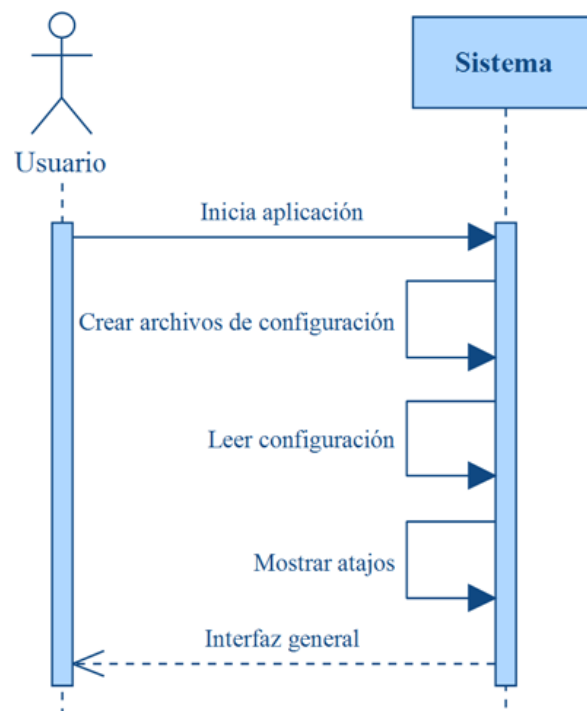


Imagen 3 - Diagrama de secuencia (Iniciar aplicación)

- **Cerrar aplicación:** Se inicia al indicar el usuario que desea cerrar el programa, cerrando la aplicación correctamente.

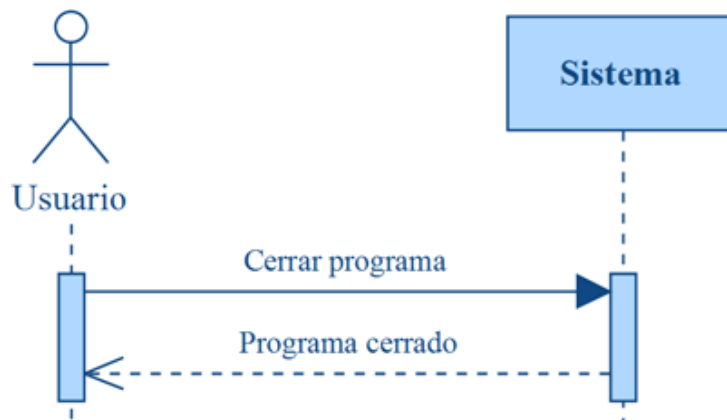


Imagen 4 - Diagrama de secuencia (Cerrar programa)

- **Nuevo XML:** Esta función pide al usuario la introducción de datos para el nombre del nuevo XML y crea una interfaz dependiendo de este.

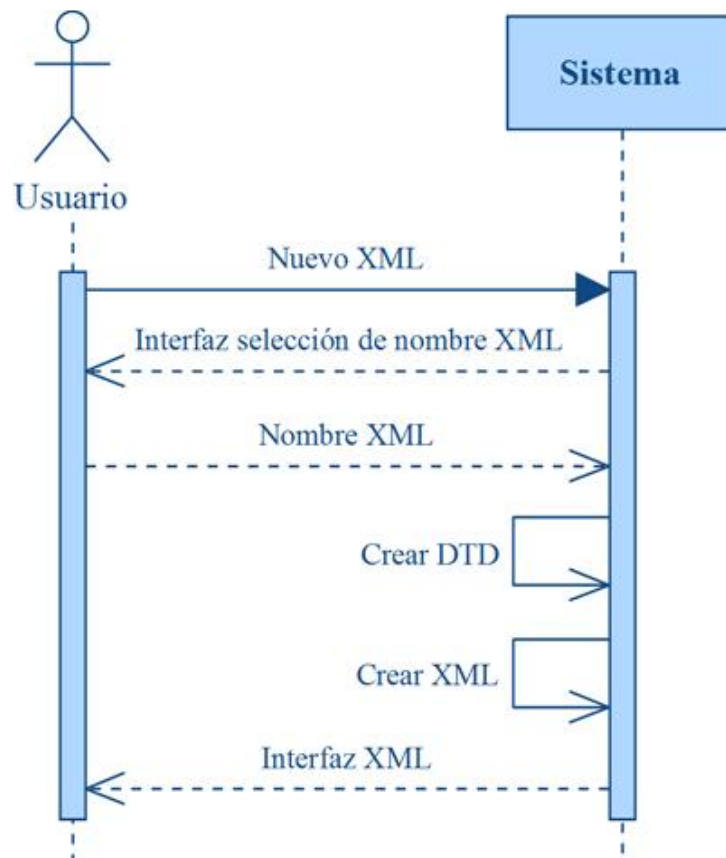


Imagen 5 - Diagrama de secuencia (Nuevo XML)

- **Exportar XML:** Esta función guarda los datos del XML actual y lo exporta a un fichero externo.

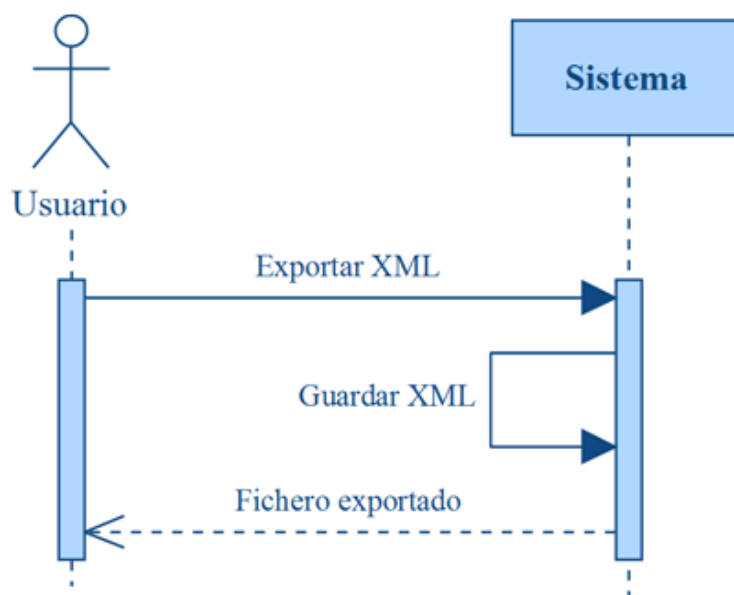


Imagen 6 - Diagrama de secuencia (Exportar XML)

- **Importar XML:** Lee los datos de un fichero externo seleccionado por el usuario e intenta cárgalo al programa. En caso de encontrarse con que el XML no tiene un DTD asociado pregunta que hacer al usuario para cargar el XML sin DTD, seleccionar un DTD de entre los existentes o crear un nuevo conjunto DTD.

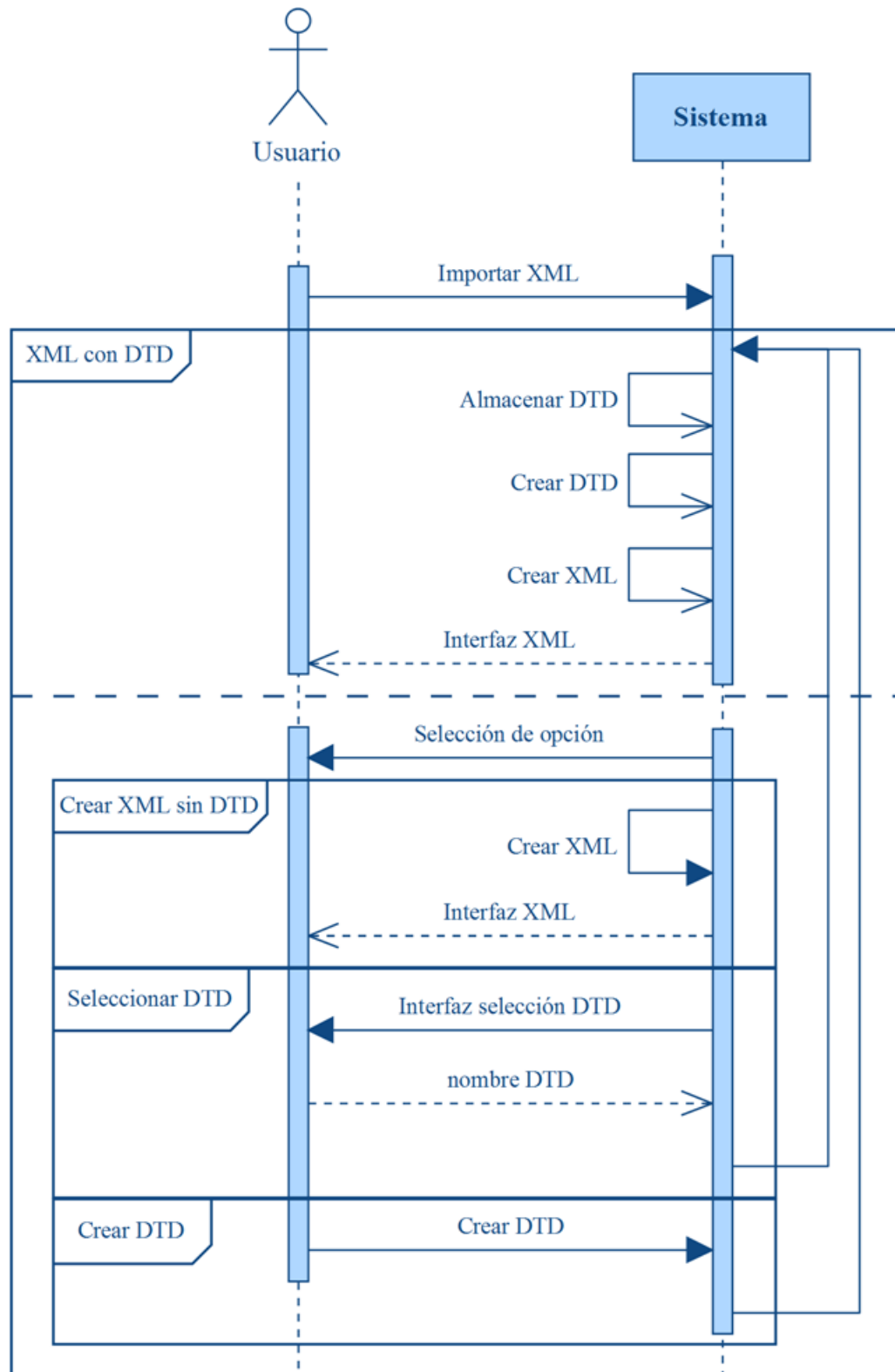


Imagen 7 - Diagrama de secuencia (Importar XML)

- **Crear DTD:** Esta función permite crear un nuevo conjunto *DTD* y añadirlo al sistema, dejando que el usuario escoja su nombre, fichero de configuración de *DTD*, archivo *XSLT* de formato y *CSS* de estilo.

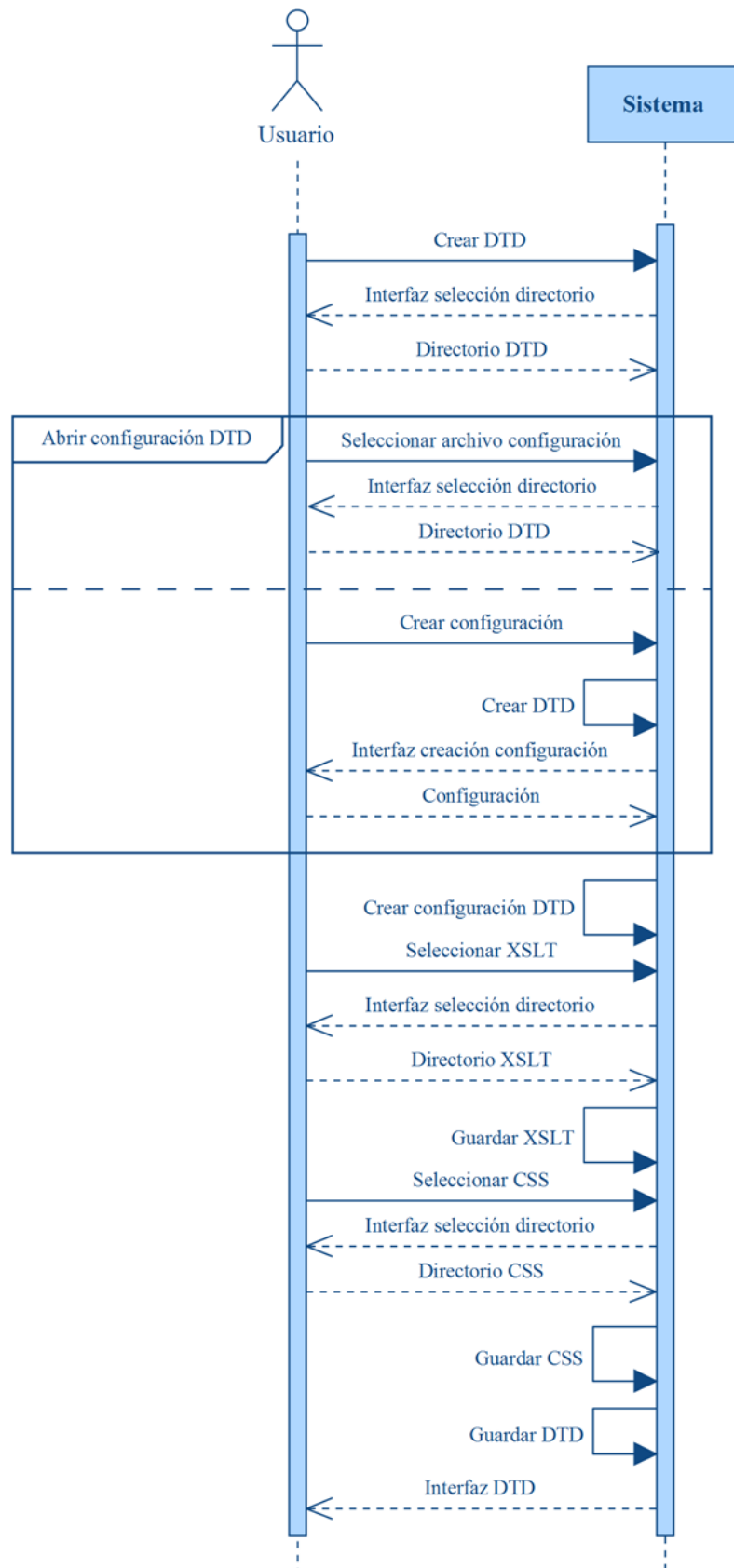


Imagen 8 - Diagrama de secuencia (Crear DTD)

- **Añadir elemento XML:** Añade al XML actual un elemento concreto, modificando la interfaz.

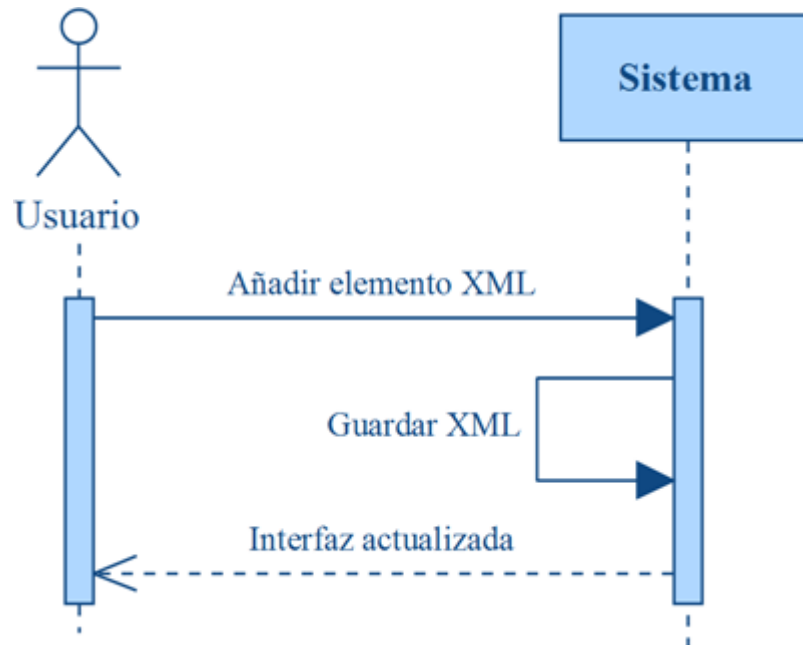


Imagen 9 - Diagrama de secuencia (Añadir elemento XML)

- **Eliminar elemento XML:** Elimina un elemento concreto del XML, modificando la interfaz actual.

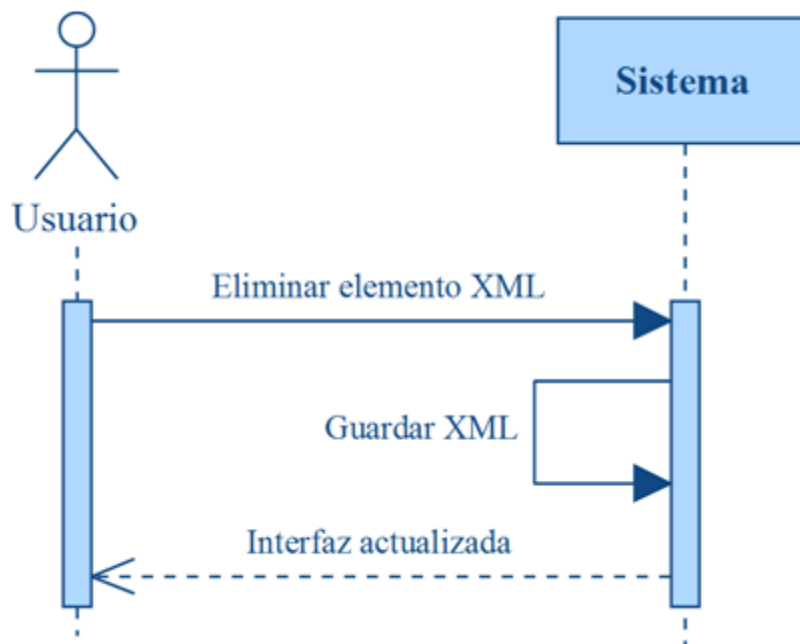


Imagen 10 - Diagrama de secuencia (Eliminar elemento XML)



- **Crear atajo:** Muestra al usuario la interfaz para crear un nuevo atajo, lo añade al sistema y actualiza la interfaz.

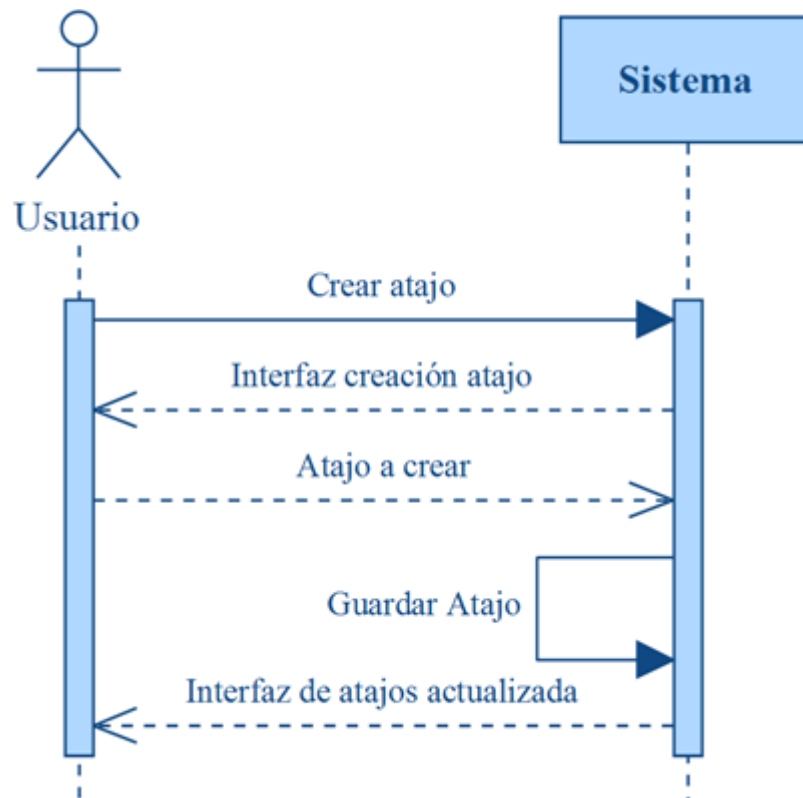


Imagen 11 - Diagrama de secuencia (Crear atajo)

- **Eliminar atajo:** Elimina un atajo concreto del sistema y actualiza la interfaz.

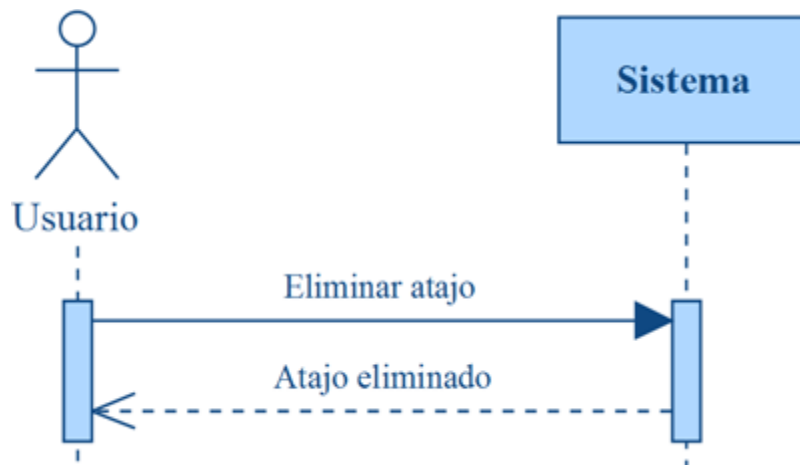


Imagen 12 - Diagrama de secuencia (Eliminar atajo)

- **Editar atajo:** Muestra al usuario la interfaz para editar un atajo determinado, lo modifica en el sistema y actualiza la interfaz.

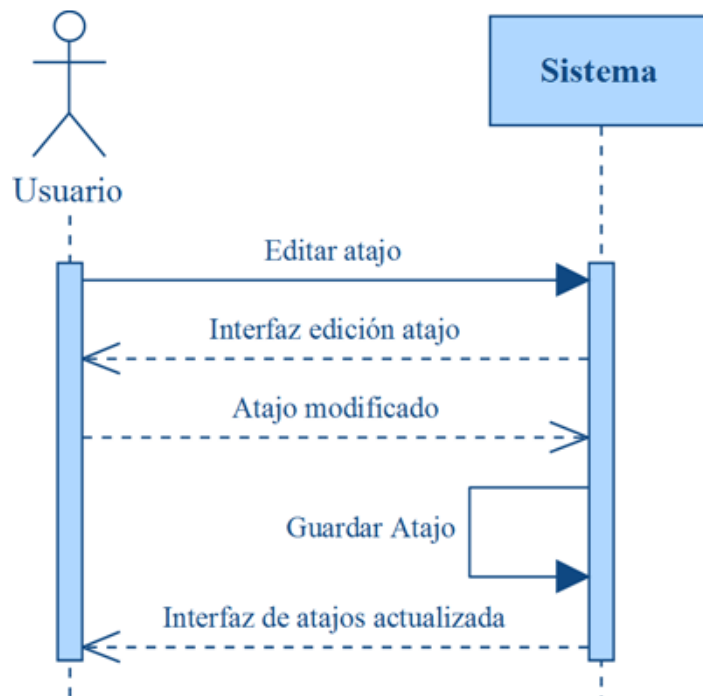


Imagen 13 - Diagrama de secuencia (Editar atajo)

- **Exportar atajos:** Guarda los atajos del sistema en un fichero indicado por el usuario.

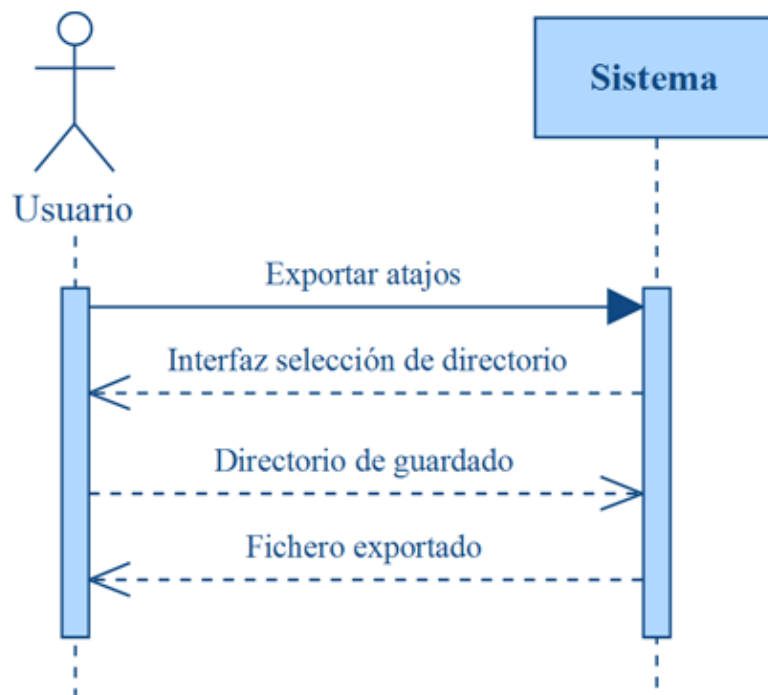


Imagen 14 - Exportar atajos)

- **Importar atajos:** Carga un fichero de atajos seleccionado por el usuario, los añade al sistema y actualiza la interfaz.

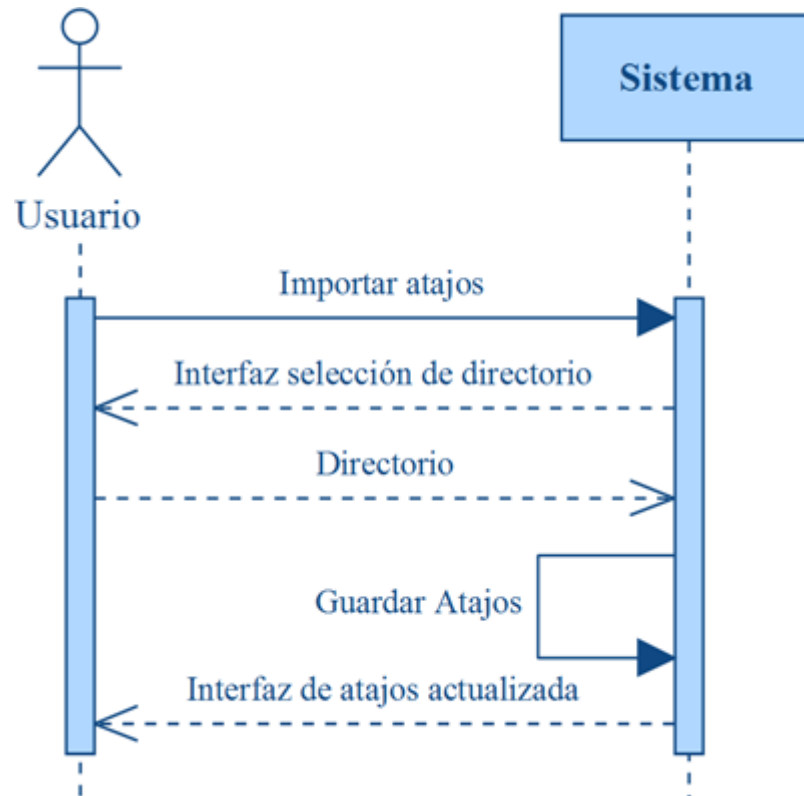


Imagen 15 - Diagrama de secuencia (Importar atajos)

# 4. Diseño

---

---

## 4.1 Diseño de clases

Para la creación del programa se ha necesitado un total de 28 clases divididas en tres grupos según su funcionalidad. A continuación se explica cada grupo.

### 4.1.1.1 Manejo de ficheros y configuración

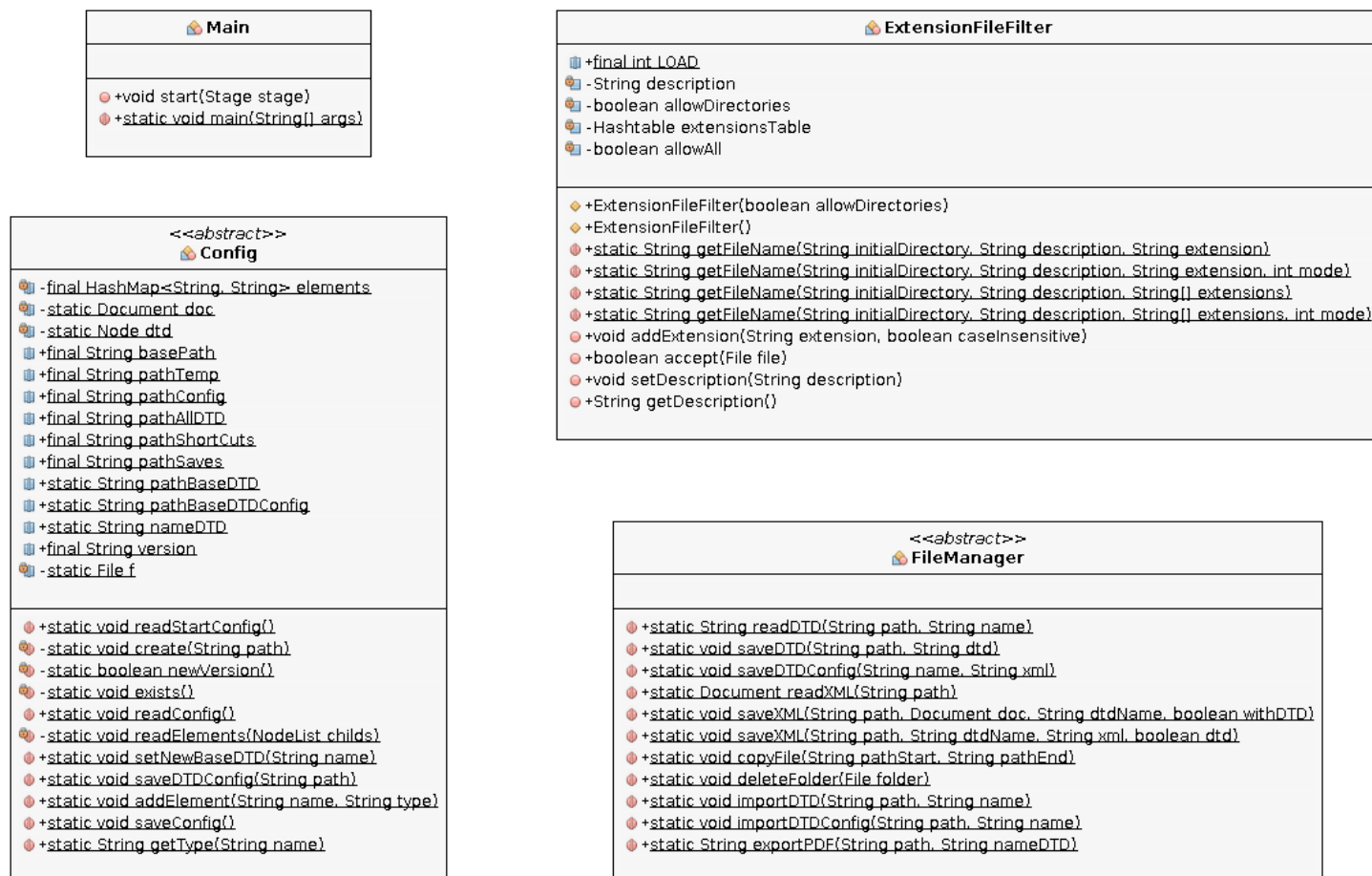


Imagen 16 - Diagrama de clases de trato de ficheros y configuración

- **Main:** Ejecuta el programa.
- **Config:** Encargada de la lectura, escritura y modificación de la configuración del sistema y sus ficheros.
- **ExtensionFileFilter:** Muestra una interfaz para la apertura y guardado de ficheros.
- **FileManager:** Se encarga del guardado y lectura de los archivos tratados por el programa.

#### 4.1.1.2 Árboles de datos

Este conjunto de clases se encarga de la creación y modificación de cada uno de los árboles creados en el sistema.

##### 4.1.1.2.1 Árbol XML

- **UIXMLTree** : Guarda y crea el árbol de interfaz *XML*.
- **XMLTree**: Guarda, crea y modifica el árbol *XML*.
- **Int**: Almacena un entero mutable.

##### 4.1.1.2.2 Árbol de interfaz de usuario XML

- **UIItem**: Contenedor que almacena un elemento de *XML* y la interfaz necesaria para indicar su contenido o una lista de hijos.
- **UIChild**: Contenedor de interfaz que almacena una lista del mismo tipo de *UIItem*.
- **UIOr**: Contenedor de interfaz que almacena una lista de módulos.
- **Module**: Módulo de interfaz para mostrar elementos de un *XML*.
- **XMLElementPane**: Elemento de interfaz para introducir la información de un elemento *XML*.

##### 4.1.1.2.3 Árbol DTD

- **DTDTree**: Crea y almacena el árbol correspondiente a un *DTD* leído.
- **UIDTDTree**: Crea y almacena el árbol de interfaz con la que se tratará la configuración de los elementos de un *DTD*.

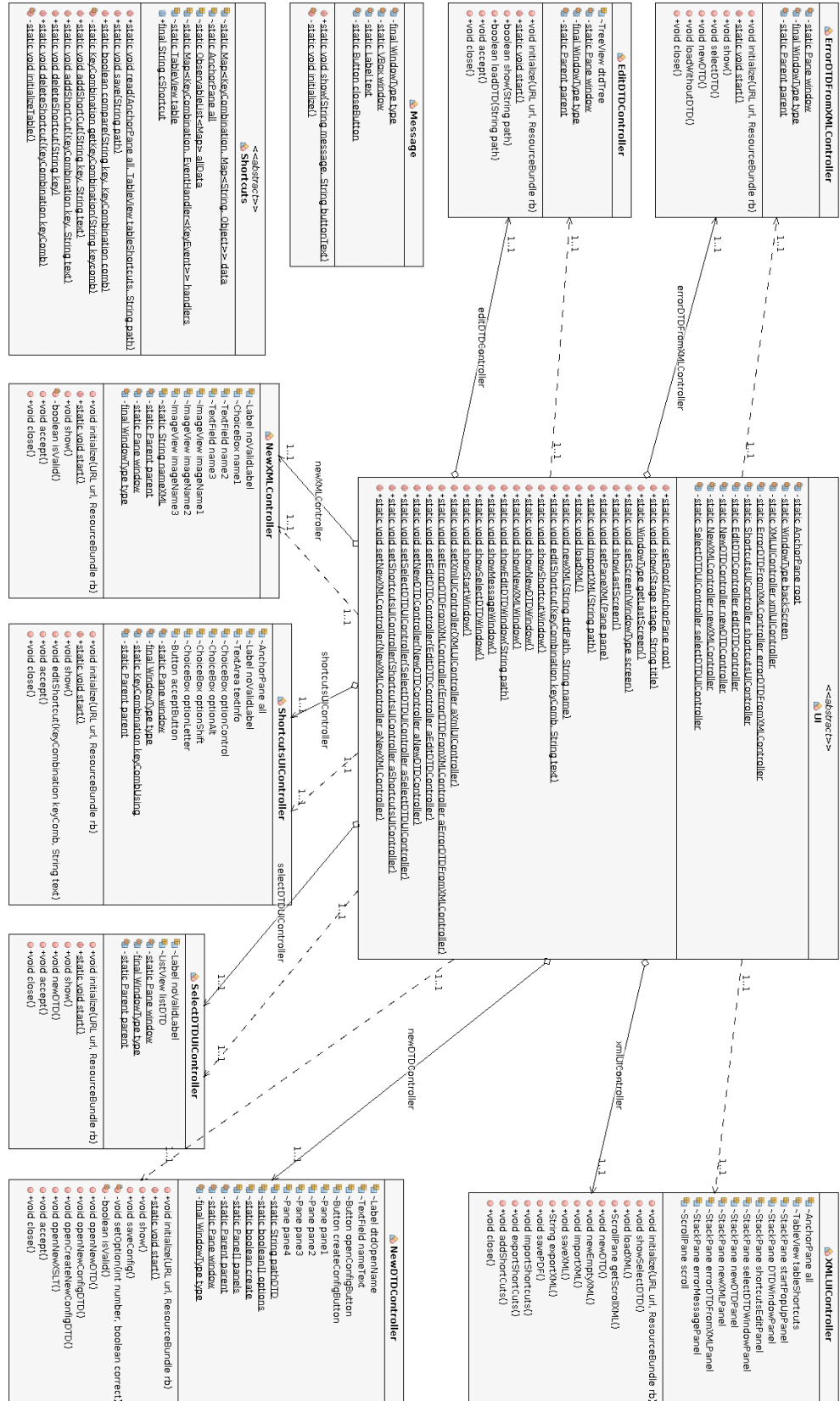
##### 4.1.1.2.4 Árbol de interfaz de usuario DTD

- **DTDChild**: Contenedor que almacena un *DTDItem* e información sobre su cantidad.
- **DTDElementPane**: Elemento de interfaz que contiene el nombre del elemento DTD asociado y un componente para modificar su configuración.
- **DTDItem**: Elemento que contiene una lista de hijos y atributos asociados a un elemento *DTD*.
- **DTDOOr**: Contenedor que almacena una lista de *DTDChild* e información sobre su cantidad.



### 4.1.1.3 Interfaz

Estas clases se encargan de la interacción con los distintos módulos de interfaz del sistema.





- **UI:** Clase que gestiona la interfaz.
- **Shortcuts:** Clase que guarda los atajos actuales y permite modificarlos.
- **SwitchButton:** Elemento de interfaz que representa un botón con dos posibilidades.
- **Módulos**
  - **Controller:** Clase abstracta de la cual heredan todos los módulos. Encargada del manejo de la ventana.
  - **SelectDTDUIController:** Controlador de la interfaz de seleccionar un *DTD* entre los existentes.
  - **NewXMLController:** Controlador de la interfaz de añadir un nuevo *XML*.
  - **NewDTDController:** Controlador de la interfaz de añadir un nuevo *DTD*.
  - **ErrorDTDFromXMLController:** Controlador de la interfaz de mensaje de error al abrir un *XML* sin *DTD*.
  - **EditDTDController:** Controlador de la interfaz de editar la configuración de un *DTD*.
  - **ShortcutsUIController:** Controlador de la interfaz de añadir o editar atajos.
  - **XMLUIController:** Controlador de la interfaz del programa.
  - **Message:** Muestra una ventana de información con un mensaje.

## 4.2 Diseño de interfaz

En este apartado se encuentran los bocetos y esquemas de la interfaz de usuario que tendrá el programa, tanto del área principal como de cada uno de los módulos.

### 4.2.1 Área principal

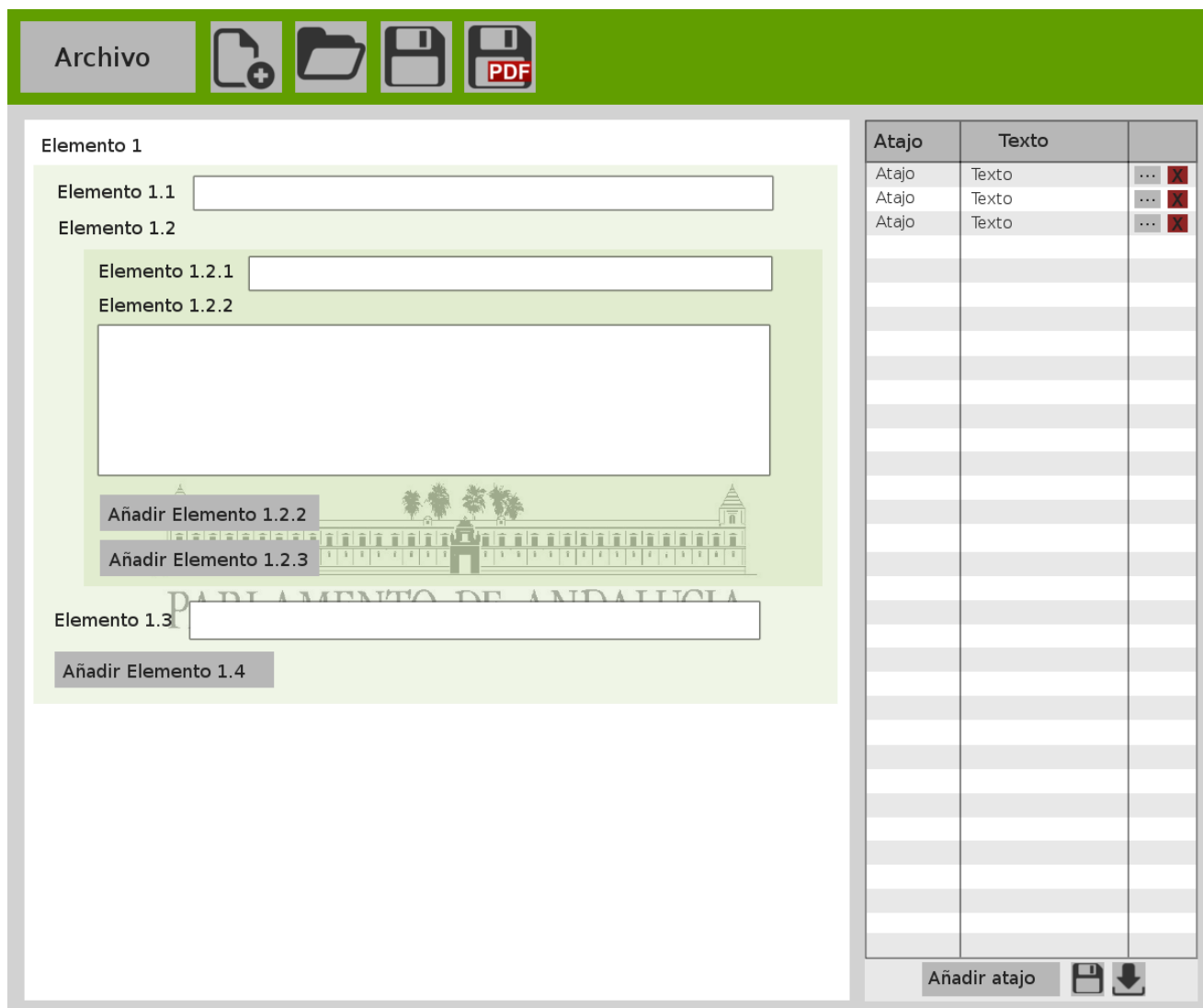


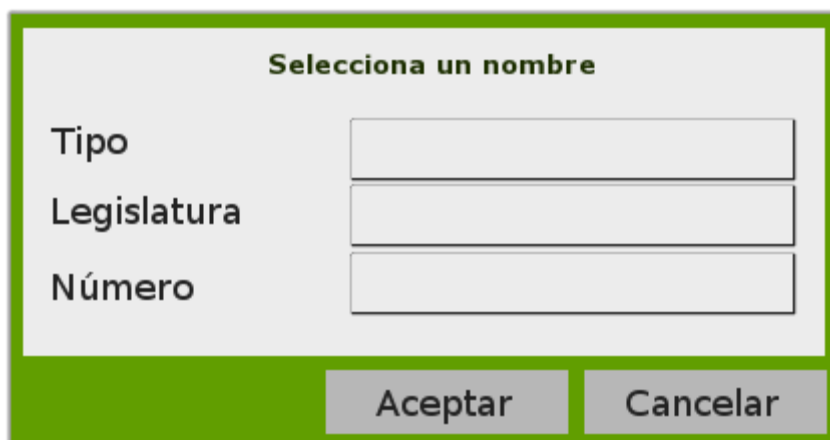
Imagen 19 - Diseño interfaz – Área principal

En esta imagen observamos el área principal del programa que se divide en:

- **Menú:** (*Zona superior*) Donde se encuentran los botones para realizar las distintas acciones.
- **Zona de atajos:** (*Zona derecha*) Tabla con los atajos disponibles en el sistema y botones para interactuar con estos.
- **Zona de XML:** (*Zona izquierda*) Muestra la interfaz con los elementos del XML actual.

## 4.2.2 Módulos

### 4.2.2.1 Selección de nombre para un nuevo XML



Selecciona un nombre

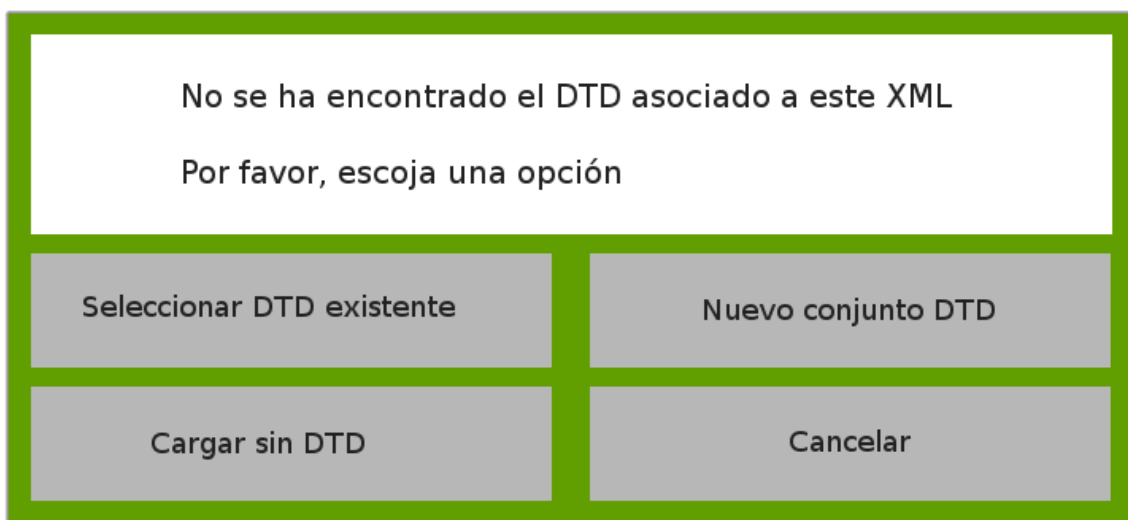
Tipo	<input type="text"/>
Legislatura	<input type="text"/>
Número	<input type="text"/>

Aceptar Cancelar

Imagen 20 - Diseño interfaz – Selección de nombre XML

En esta imagen se ve el módulo para la selección de un nombre XML, separado en las diferentes partes del nombre: Tipo, legislatura y número.

### 4.2.2.2 Selección de opción al importar XML sin DTD



No se ha encontrado el DTD asociado a este XML

Por favor, escoja una opción

Seleccionar DTD existente	Nuevo conjunto DTD
Cargar sin DTD	Cancelar

Imagen 21 - Diseño interfaz –Importar XML sin DTD

En esta imagen se muestra el módulo para seleccionar una opción al intentar importar un XML sin ningún documento DTD asociado o correcto.

#### 4.2.2.3 Selección de *DTD* existente



DTD1
DTD2
DTD3

**Nuevo** **Aceptar** **Cancelar**

Imagen 22 - **Diseño interfaz –Selección DTD existente**

Este módulo muestra una lista de los *DTD* existentes en las carpetas del sistema para poder seleccionar uno de ellos.

#### 4.2.2.4 Creación de nuevo conjunto *DTD*

El diagrama muestra una interfaz de usuario con un fondo verde claro y un borde verde oscuro. Se presentan cinco pasos numerados en círculos grises:

- 1** Archivo DTD: Incluye un botón "Abrir".
- 2** Nombre del conjunto: Incluye un campo de texto vacío.
- 3** Archivo de configuración DTD: Incluye botones "Abrir" y "Crear".
- 4** Archivo XSLT: Incluye un botón "Abrir".
- 5** Archivo CSS: Incluye un botón "Abrir".

En la parte inferior derecha, hay dos botones: "Aceptar" y "Cancelar".

Imagen 23 - **Diseño interfaz – Creación de conjunto DTD**

Este módulo permite seleccionar o crear los distintos archivos necesarios para la creación de un nuevo conjunto *DTD*.

#### 4.2.2.5 Edición de la configuración de un *DTD*



The image shows a dialog box titled 'Edición de la configuración de un DTD'. It contains a tree structure of elements and their configurations. The tree is as follows:

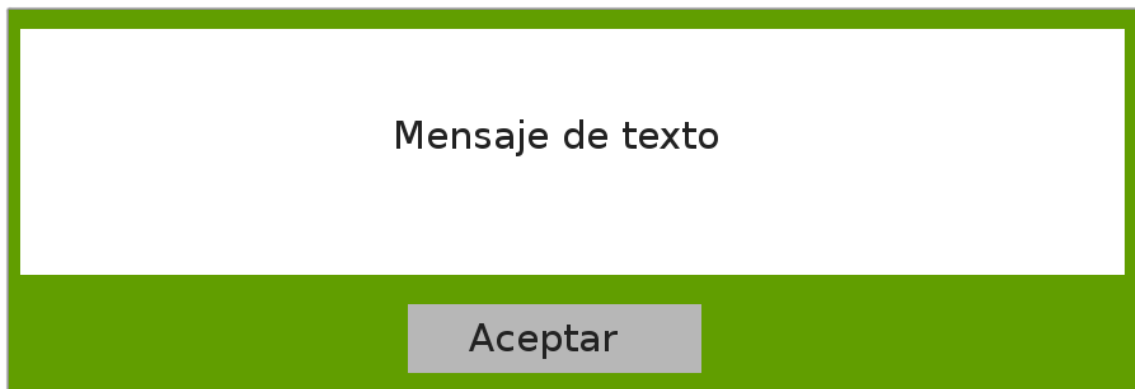
- Elemento 1
  - Elemento 1.1: Línea
  - Elemento 1.2
    - Elemento 1.2.1: Línea
    - Elemento 1.2.2: Texto
    - Elemento 1.2.3: Tabla
  - Elemento 1.3: Imagen

At the bottom of the dialog are two buttons: 'Aceptar' and 'Cancelar'.

Imagen 24 - Diseño interfaz –Configuración DTD

En la imagen se observa el módulo para la edición de la configuración de un *DTD*, con un árbol que contiene los elementos del *DTD* y su configuración actual.

#### 4.2.2.1 Mensaje de aviso



The image shows a dialog box titled 'Mensaje de texto'. It has a large white rectangular area for the message text. At the bottom of the dialog is a single button labeled 'Aceptar'.

Imagen 25 - Diseño interfaz –Mensaje de aviso

Este módulo genérico permite al sistema comunicar un mensaje al usuario.

#### 4.2.2.2 Edición / Adición de atajos

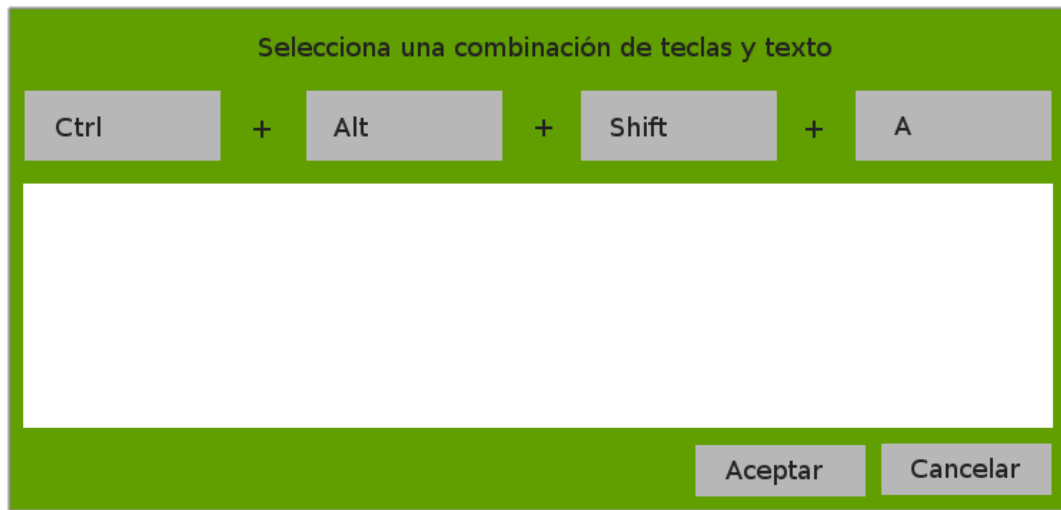


Imagen 26 - **Diseño interfaz –Modificación de atajos**

Esta imagen representa el módulo para editar o añadir atajos de teclado al sistema.

# **5. Implementación**

---



## 5.1 Herramientas utilizadas

---

A lo largo del desarrollo del proyecto se ha hecho uso de diversas tecnologías, lenguajes de programación, bibliotecas, etc. En este capítulo se habla con detalle acerca de cada una de estas herramientas y de su aplicación que han tenido dentro del proyecto.

### 5.1.1 IDE - Netbeans

Se comenzó su desarrollo en 1996 con el nombre de *Xelfi* en la facultad de matemáticas y física “*Charles University*” en Praga. Posteriormente fue comprado por *SunMicrosystems* en 1999 ya bajo el nombre actualmente conocido de **NetBeans IDE** y terminó siendo adquirido por *Oracle* en el año 2010. Es una aplicación que sigue en continuo desarrollo con una gran comunidad a sus espaldas.

Esta aplicación es completamente gratuita y sin restricciones de uso, por lo que evita un aumento del presupuesto al no tener que realizar ningún gasto por su uso. Además, al estar desarrollada completamente en *Java* permite su uso en todos los sistemas operativos siempre que soporten la máquina virtual de *Java*.

**NetBeans** está pensado mayormente para la programación en *Java*, aunque también soporta otros lenguajes como pueden ser *C*, *C++* o *HTML5*.

Cuenta con un módulo para la interacción visual con Java Swing, una biblioteca gráfica para *Java* que incluye elementos de interfaz gráfica. Gracias a este módulo el programador puede situar los elementos de la interfaz manualmente, comprobando en el momento cuál va a ser el resultado final de esta, facilitando enormemente su tarea. No obstante, como veremos más adelante en la fase de desarrollo que trata con la interfaz, no haremos uso de esta biblioteca.

Actualmente (a julio de 2015), su versión estable más reciente es la 8.0.2 en todas los sistemas operativos, lanzada el 28 de noviembre de 2014. Esta es la versión del *IDE* que ha sido usada para el desarrollo completo del proyecto.

## 5.1.2 Lenguajes utilizados

En este apartado se hablará acerca de los lenguajes usados para la creación del programa en el que se basa el proyecto, tanto aquellos lenguajes de programación, como de etiquetado y estilo.

### 5.1.2.1 Java

*Java* es un lenguaje de programación compilado dirigido a objetos. Originalmente desarrollado por *James Gosling*, perteneciente a la compañía *SunMicrosystems* y publicado en 1995. Y fue adquirido por *Oracle* en 2010.

Su sintaxis deriva en gran medida de los lenguajes *C* y *C++*, aunque cuenta con menos utilidades de bajo nivel, centrándose más en aquellas utilidades orientadas a objetos.

Fue pensado especialmente para permitir que un único código pueda ejecutarse en cualquier dispositivo y sistema operativo sin tener que hacer ningún cambio adicional ni tener que ser recompilado. Para ello cuenta con una máquina virtual conocida como *JVM* ("*Java Virtual Machine*"). Todas las aplicaciones *Java* se compilan en un conjunto de instrucciones máquina que reconoce *JVM*, de modo que separa la arquitectura usada para ejecutar los programas del propio programa, creando una capa de abstracción, algo realmente útil para nuestro proyecto.

El programa en el que se basa el proyecto ha sido desarrollado usando como base este lenguaje de programación, escogido por su portabilidad, modularidad y robustez.

### 5.1.2.2 CSS

*CSS* es un lenguaje de hojas de estilo propuesto por *Håkon Wium Lie* en 1994 y fue incluido en *World Wide Web Consortium (W3C)* en 1996.

Es usado para dar formato y estilo a estructuras *HTML*, *XHTML* o *XML*, permitiendo variar cada elemento por separado con su *ID* único o agrupándolos en clases.

En nuestro proyecto, haremos uso de este lenguaje para dar formato a los ficheros creados de *XHTML*, para acomodarse a los requerimientos de estilo necesarios para la creación de los *PDF*.

### 5.1.2.3 XML

*XML* es un lenguaje de marcas desarrollado en 1996 por *World Wide Web Consortium (W3C)*. Deriva del lenguaje *SGML* y permite definir una estructura para documentos de gran volumen.

Es una tecnología sencilla muy usada como base por otras que la complementan y detallan para los distintos campos en la que es usada. Como ejemplo, en nuestro proyecto usamos *FXML*, una tecnología basada en *XML* para la definición de la interfaz de usuario con *JavaFX*. Esto permite establecer un estándar para la creación de estructuras de datos con una gran compatibilidad.

Sus usos son variados, proponiéndose como un estándar para el intercambio de información estructurada, pudiendo usarse en bases de datos, hojas de cálculo, editores de texto, configuraciones de sistemas, etc.

En nuestro proyecto emplearemos este lenguaje en diferentes entornos, usándolo como estructura para los datos que leeremos y almacenaremos en el programa, como definición de la disposición de elementos dentro de la interfaz de usuario, como sistema para estructurar y guardar la información acerca de la configuración del sistema, entre otras.

Acompañando a las estructuras creadas por este lenguaje contamos con unos documentos que contienen una descripción de la estructura y sintaxis obligada para ellos. Conocemos estos documentos como *DTD*.

Usaremos estos documentos para establecer una serie de normas a la hora de la creación y modificación de los ficheros *XML*, asociándolos a estos ficheros. De esta forma, nos aseguraremos de que son correctos validándolos con su documento asociado.

Más concretamente, los *DTD* contienen información acerca de la estructura de los elementos de un *XML*, los datos e hijos que debe contener cada uno, con sus respectivas cantidades posibles, así como los atributos que debe o puede tener cada elemento.

Pueden estar incluidos dentro del propio fichero *XML* pero, en nuestro caso, se almacenarán en ficheros separados, ya que haremos uso de un mismo *DTD* para una gran cantidad de ficheros *XML*.

#### 5.1.2.4 XHTML

Este lenguaje es el mismo que *HTML* pero expresado con la estructura propia de un *XML*, por tanto pasaremos a explicar el lenguaje *HTML*.

*HTML* comenzó su desarrollo en 1991, donde *Tim Berners-Lee* describió los primeros elementos que formarían parte de este lenguaje y en 1993 fue reconocido como una ampliación del lenguaje *SGML*.

Es un lenguaje de marcado para la creación de páginas web, que hace uso de etiquetas para la distribución de elementos dentro de la página y de referencias para la introducción de elementos como imágenes, vídeos, etc.

Para el formato de la página creada, *HTML* hace una referencia a un archivo *CSS* el cual contiene toda la información de estilo.

En nuestro proyecto, este lenguaje será utilizado como puente para la creación de un fichero *PDF* a partir de uno *XML*, de modo que los ficheros *HTML* generados por el programa no estarán pensados para ser visualizados en una página web, sino contener los elementos y estructuración necesarios para una correcta visualización en el archivo *PDF* generado.

#### 5.1.2.5 XSLT

*XLST* o transformaciones *XSL* son hojas de estilo que marcan unas reglas y transformaciones a realizar sobre el documento *XML* al que se les aplique.

Influenciadas por los lenguajes funcionales, es el predecesor de *DSSSL*, que hacía lo mismo que *XSLT* hace para *XML* pero para el lenguaje *SGML*. Es parte del lenguaje *XSL* de *World Wide Web Consortium (W3C)* y fue desarrollado en los años 1998 y 1999.

La unión de un archivo *XML* y un *XSLT* asociado permite la separación del contenido y la presentación, aumentando la productividad.

Al aplicar un *XSLT* sobre un archivo *XML* se obtiene un archivo *HTML* o *XHTML* que se atiene a las normas expuestas por el fichero *XSLT*.

En nuestro proyecto, usaremos *XSLT* para crear un archivo *XHTML* a partir de un *XML*, con los elementos y estructuración necesarios para crear a continuación el archivo *PDF* resultado.

## 5.1.3 Bibliotecas usadas

En este apartado se hablará acerca de las bibliotecas que ha sido necesario importar al proyecto para el correcto funcionamiento de este.

### 5.1.3.1 iText

*iText* es una biblioteca de código abierto que permite la creación y manipulación de archivos *PDF* y *HTML*.

Comenzó a ser desarrollada en el año 2000 por *Bruno Lowagie* y *Paulo Soares* entre otros y fue distribuida bajo la *Affero General Public License* (AGPL).

En nuestro proyecto, esta biblioteca nos permitirá la conversión de un fichero *XHTML*, junto con su hoja de estilos *CSS*, a un fichero *PDF*.

### 5.1.3.2 JavaFX

*JavaFX* es una biblioteca desarrollada por *SunMicrosystem* en 2009 y después fue adquirida por *Oracle*.

Esta biblioteca puede ser ejecutada en gran variedad de dispositivos, pretendiendo reemplazar a *Java Swing* como la biblioteca estándar para la creación de interfaces de usuario. Ambas están incluidas actualmente dentro de la plataforma *Java*, de modo que no necesitan una instalación externa de la biblioteca para ser utilizadas.

*JavaFX* utiliza el modelo vista controlador (*MVC*) para la creación de los diferentes módulos o pantallas de la interfaz, permitiendo una mayor modularización de esta, facilitando el desarrollo. Para ello hace uso de varios ficheros:

- **Clase controlador:** Una clase *Java* que hace uso de la interfaz *Initializable* y que se encarga de crear la interfaz y tratar con ella.
- **Archivo *FXML*:** Contiene todos los elementos de la interfaz y su disposición dentro de ella. *JavaFX* cuenta con un editor propio contenido en la propia plataforma *Java* para la edición de un modo visual de la interfaz, que nos permite cambiar de forma fácil la interfaz, además de poder ejecutarla en el momento con datos de prueba para comprobar su correcto funcionamiento y modificación, tanto de cambio de tamaño, visualización de árboles, etc, algo no posible con el módulo de visualización de *Java Swing* existente en *NetBeans*.

*JavaFX* contiene un gran número de elementos, tanto estáticos como dinámicos, gráficos, objetos 3D, etc. Permitiendo crear una interfaz atractiva mucho más fácilmente que con *Java Swing*.

Además, permite establecer un estilo para cada uno de los elementos de la interfaz haciendo uso de una hoja de estilos *CSS*, permitiendo una gran personalización de la interfaz de un modo fácil y cómodo.

## 5.2 Fases del desarrollo

---

En este apartado se detallarán cada una de las distintas fases del desarrollo del programa, analizando los problemas hallados en cada una de ellas y la solución propuesta.

### 5.2.1 Parseo de DTD

Para comenzar se necesitaba obtener la información contenida en un *DTD* para poder crear un *XML* a partir de este, cumpliendo sus reglas y restricciones.

No existe ninguna biblioteca gratuita actualmente que realice una lectura de un documento *DTD* y obtenga su información, por lo que se tuvo que hacer manualmente.

En primer lugar se obtuvo la cadena de caracteres que contenía la información del DTD. Para ello se usó un flujo de lectura para leer el fichero, tal y como se ve a continuación:

```
BufferedReader entrada = new BufferedReader (new FileReader(path));
String line, dtd = "";

while(true) {
    line = entrada.readLine();
    if(line == null) break;
    dtd = dtd.concat(line.trim() + " ");
}
```

Imagen 27 - Flujo de lectura de un DTD

Una vez obtenida la información, se hace uso de un algoritmo de procesamiento de esta para obtener un árbol estructurado de la información. El proceso se explica a continuación.

#### 5.2.1.1 Algoritmo de lectura

En primer lugar se hace un recorrido de la cadena de texto para eliminar los comentarios (Subcadenas de texto que comienzan por los caracteres “<!--” y terminan por los caracteres “-->”). Se eliminan también los caracteres de salto de línea.

Una vez quitada la información innecesaria, se procede a buscar el carácter de apertura “<”, y se determina cual es la palabra que le acompaña. Esta palabra determinará si nos vamos a encontrar con un elemento o un atributo.

En caso de encontrarnos ante un elemento, leemos y almacenamos su nombre. A continuación miramos si contiene hijos o no. En caso de no tener hijos (*#PCDATA*), lo colocamos como nodo hoja en nuestro árbol. En caso de tener hijos, pasamos a leer y crear estos hijos junto con la cantidad asociada para el elemento padre.

Repetiremos este proceso hasta terminar con todos los elementos, obteniendo entonces un conjunto de elementos, en el que cada uno tiene un nombre y, o bien una lista de parejas con el nombre de cada hijo y su cantidad, o bien indicación de que es un nodo hoja.

En caso de tratarse de un atributo, leeremos a continuación de qué elemento es atributo. En caso de existir se añade el nombre del atributo que tiene que tener.

Este bucle se repetirá hasta completarse la lectura de todos los datos del archivo *DTD*, obteniéndose todos los elementos, sus atributos y las relaciones padre-hijo-cantidad entre ellos.

### 5.2.1.2 Datos obtenidos

Como hemos visto, el proceso anterior devuelve una lista de elementos con su nombre y sus hijos y cantidades asociadas o una indicación de que es un nodo hoja. Este será nuestro árbol abstracto del *DTD*, en la que cada hijo puede ser de uno de los siguientes tipos:

- Pareja de valores, donde el primer valor es la referencia a el elemento hijo y el segundo valor su cantidad asociada a la relación entre este y su padre.
- Pareja de valores donde el primer elemento es una disyunción de elementos (una disyunción contiene dos o más parejas del tipo explicado anteriormente) y el segundo valor su cantidad asociada con su padre.

La cantidad asociada es única para la relación entre el nodo padre y su hijo, de este modo, si el nodo hijo aparece colgando de un nodo padre distinto puede tener una cantidad asociada distinta.

La cantidad viene definida en el *DTD* y puede ser cualquiera de las siguientes:

- Obligatoriamente sólo un elemento. Cantidad por defecto cuando no se especifica ninguna en el *DTD*.
- uno o ningún elemento. Cantidad representada con un signo de interrogación. No obliga a tener un elemento, pero limita a uno la cantidad máxima permitida.
- Uno o más elementos. Cantidad representada por un signo de suma. Obliga a haber al menos un elemento de ese tipo, pero no limita cantidad máxima.
- Cero o más elementos. Cantidad representada por un signo asterisco. No obliga a haber elementos de ese tipo ni tampoco limita la cantidad máxima.



### 5.2.1.3 Configuración de DTD

Una vez obtenido el árbol de *DTD*, crearemos otro igual en el que además de lo anterior, cada nodo hoja contendrá información acerca del tipo de información que va a contener. Este nuevo árbol se guardará en formato *XML* como la configuración de interfaz de usuario para ese documento *DTD*.

Este árbol indicará qué debe introducir el usuario como información para cada elemento, de modo que cada tipo indicará una interfaz de usuario diferente según el elemento a tratar.

Los diferentes tipos de información son:

- **Campo de texto:** Un elemento de tipo *TextArea* para que el usuario pueda introducir más de una línea de texto, como en un párrafo.
- **Línea de texto:** empleada para elementos con poca información, en la interfaz se usa un elemento de tipo *TextField*, y sirve para introducir una frase, un número, etc.
- **Selección de fichero:** En la interfaz se muestra como una línea de texto, igual que el caso anterior, acompañada de un botón que abre el módulo para seleccionar un directorio. Esto permite definir un directorio a partir del cual se cargará la imagen o tabla.
- **Lista de elementos:** En la interfaz está representada como un *ListSpinner*, se trata de una lista que contiene valores preestablecidos, para que el usuario seleccione entre uno de ellos. Esto se usa para elementos que pueden contener un número limitado de valores invariables.
- **Si/No:** Interfaz representada por un botón con dos posibles posiciones: Encendido que representa el valor “Si” o apagado que representa el valor “No”.

Se puso especial cuidado en que el código fuera estructurado de forma modular, por lo que la creación de nuevos tipos de información por parte del programador es rápida y sencilla, facilitando la tarea en caso de querer modificar este aspecto en un futuro.

## 5.2.2 Metodología para la creación de la interfaz

Cuando se comienza a diseñar la interfaz de usuario con la que un programa debe relacionarse con un usuario deben estudiarse las diferentes bibliotecas creadas para ello, analizando cuál se adapta mejor a nuestro entorno.

Para nuestro programa, se necesitaba un entorno dinámico y altamente personalizable, con capacidad para redimensionarse y permitir una modularización de las distintas pantallas o ventanas internas del programa. Esto se debe a que el programa deberá soportar una interfaz cambiante según la estructura determinada y las necesidades del usuario restringidas a la estructura anterior.

Para un desarrollo futuro del programa, también se necesitaba que pudiera separarse cada parte de la interfaz de usuario, modularizándola, obteniendo un sistema en el que cambiar y añadir nuevas ventanas fuese fácil y sencillo.

Tras hacer una selección, se decidió estudiar más a fondo dos bibliotecas conocidas: *Java Swing* y *JavaFX*. A continuación se relatan sus características más relevantes para su incorporación al proyecto:

	<i>Java Swing</i>	<i>JavaFX</i>
Documentación	Muy buena	Buena
Posibilidad de personalización	Muy escasa	Completa, haciendo uso de una o varias hoja de estilo CSS
Modularización	Ninguna en especial	Se basa en la separación modular con el modelo vista controlador
Distribución de los elementos	Un panel tiene distintas disposiciones posibles	Muchos paneles distintos
Elementos de interfaz distintos	Aceptables para cubrir las necesidades básicas	Alto, además de elementos básicos cuenta con gráficas, modelos 3D
Permite redimensionar	Sí, costosamente	Sí, fácilmente
Editor visual	Sí, incluido en <i>NetBeans</i>	Sí, programa externo a <i>NetBeans</i> que permite además la visualización con datos de prueba

Tabla 2 - Comparación JavaFX – Java Swing

Tras analizar ambas bibliotecas se optó por *JavaFX*, por sus amplias posibilidades de personalización de los elementos y su modelo *MVC*, capaz de separar cada ventana de la interfaz, de modo que se distribuyó toda la interfaz de usuario en módulos que contenían una clase controladora y un *FXML* con la estructura del módulo.

Por modularidad, se optó por la creación de una clase abstracta que contuviera la información necesaria para el manejo de la ventana de un módulo. De modo que todos los módulos heredarán de esta clase para poder crear una capa de abstracción entre la información contenida en el módulo y el trato de la ventana en sí.

Para una cohesión e igualdad de estilos entre módulos, toda la interfaz hace uso de un solo archivo *CSS* para su definición de estilo, de modo que los módulos usan los mismos valores de estilo, dando unidad al programa.

Para acceder a los módulos se usa una clase común que contiene una instancia de cada uno de ellos, de modo que se crea una capa de abstracción encapsulando todos los módulos de la interfaz y accediendo a ellos a través de una única clase que los gestiona.

### 5.2.3 Creación de XML

Teniendo los árboles de *DTD* y configuración de la interfaz de este, se procede a crear dos árboles más: uno utilizando la tecnología *DOM* para almacenar el *XML* con la información introducida por el usuario, y otro árbol que contiene la estructura mostrada en la interfaz para la modificación del árbol anterior.

Ambos árboles se crean en paralelo, referenciando unívocamente cada elemento de uno de ellos a su pareja en el otro árbol.

Los árboles se crean siguiendo la estructura del árbol *DTD* y usando la configuración para asignar el elemento de interfaz de introducción de datos a los nodos hoja, según el tipo al que pertenezca el elemento, tal y como se explica con más detalle en el **Apartado 5.2.1.3**.

### 5.2.3.1 Árbol *XML*

Para la creación de este árbol se lee desde el nodo raíz del árbol *DTD* hasta todos los nodos hoja.

Se crea el primer nodo, con el nombre del nodo raíz del *DTD*.

A continuación se leen sus hijos uno por uno y se comprueba su cantidad. En caso de ser una cantidad que necesita al menos un elemento (cantidad uno o cantidad “+”) se crea un nodo conteniendo a ese elemento hijo.

Se procede de la misma manera para todos los nodos con hijos, creando sólo aquellos nodos necesarios obligatoriamente.

Para aquellos elementos sin hijos se crea un nodo hoja sin ningún contenido excepto del nombre del propio elemento.

Este proceso continúa hasta leer completamente el árbol *DTD* y tras crear todos los nodos y sus hijos.

Este algoritmo de creación de árboles cuenta con un grave problema: La recursividad. En caso de que un elemento se tenga de hijo a sí mismo con una cantidad mínima de uno (es decir, se tiene que generar un elemento obligatoriamente), el algoritmo entraría en un bucle infinito y nunca acabaría.

Este fallo no debe ocurrir nunca, ya que en este caso significaría que el documento *DTD* aportado es incorrecto e inválido.

### 5.2.3.2 Árbol interfaz *XML*

Para crear el árbol de interfaz utilizamos la misma lectura del archivo *DTD* que en el paso anterior, creando un árbol paralelamente al visto antes.

En primer lugar se crea el nodo raíz, con el nodo raíz del árbol *DTD*. A continuación se crea y asocia al nodo raíz un contenedor de hijos, el cual contendrá los siguientes elementos:

- Una lista de atributos del elemento (en caso de tenerlos).
- Una lista de hijos y/o contenedores disyuntivos (explicados más adelante) que tengan como cantidad mínima 1.
- Zona de botones para aquellos hijos o contenedores que tengan de cantidad posible diferente a sólo uno. Estos botones permitirán añadir estos elementos.

Un contenedor disyuntivo son un conjunto de hijos indicados en el *DTD* como que puede aparecer sólo uno de ellos cada vez. Este contenedor contendrá los siguientes elementos:

- Una lista de hijos y/o contenedores disyuntivos que tengan como cantidad mínima 1.
- Zona de botones para aquellos hijos o contenedores que tengan de cantidad posible diferente a sólo uno. Estos botones permitirán añadir estos elementos.

Un hijo contiene información sobre un elemento específico del *DTD*, conteniendo la siguiente información:

- Cantidad de elementos de ese tipo que puede haber según su cantidad con relación a su padre.
- Cantidad actual. Esto se utilizará para saber si se pueden añadir o eliminar elementos de este tipo comparando este valor con la cantidad permitida.
- Una zona de interfaz donde añadir cada nodo hijo añadido.
- Una zona de botones para añadir o eliminar nodos hijos del mismo tipo.

Un nodo hijo contiene la información de interfaz necesaria para mostrarse. En caso de ser un nodo con más hijos tendrá los mismos elementos explicados antes para el nodo raíz. En cambio, si no tiene hijos, el nodo hijo contendrá los siguientes elementos:

- Interfaz para añadir datos sobre ese elemento que irá definida por la configuración establecida en el fichero de configuración asociado al *DTD* leído. Esta interfaz mostrará un área de texto, una lista de valores, etc. Según lo explicado en el **Apartado 5.3.1.3**.
- Un botón de eliminar elemento en caso de que la cantidad lo permita.

Un atributo contendrá la información de interfaz necesaria para modificar su valor, además de su propio nombre.

Este árbol será el mostrado en la zona de interfaz de usuario dedicada al *XML* actual y se modificará con los botones de añadir o eliminar vistos anteriormente.

De esta forma, al pulsar un botón de añadir un hijo o contenedor este se creará en su posición del árbol correspondiente y creará también en la posición adecuada un elemento en el árbol *XML*.

De igual manera, al pulsar para eliminar un nodo hijo, este se desancla del árbol de interfaz y elimina el nodo asociado en el árbol *XML*.

## 5.2.4 Sistema de configuración

Puesto que se debe trabajar con un conjunto de ficheros base para el correcto funcionamiento del programa, se ha creado un archivo y una clase de configuración para manejarlos.

En primer lugar se almacena la versión actual del programa. Cada vez que se inicia el programa se compara la versión del fichero de configuración con la interna del programa. En caso de ser distintas versiones, se sustituyen todos los ficheros externos por la versión actualizada existente dentro del comprimido del programa. De esta forma se asegura el buen funcionamiento de este.

El fichero de configuración guarda además la configuración de interfaz común para todos los *DTD*, de modo que si un elemento no tiene su configuración especificada en su archivo de configuración asociado, se acude a esta lista para comprobar la configuración a tener o usar una por defecto si tampoco existe aquí.

Además, en el fichero de configuración se almacena en el mismo formato XML una lista de elementos que van a tener unos valores limitados y una lista asociada a este elemento con esos posibles valores (cadenas de texto). Esto será común para todos los *DTD* creados, aunque no es obligatorio su uso.

## 5.2.5 Importación y exportación de XML

Para importar un archivo de tipo *XML* externo al programa primero se comprueba, ayudados por la *API DOM*, si este *XML* tiene un *DTD* asociado, y si este existe.

En caso no existir un documento *DTD* asociado y el sistema permite cargar el fichero *XML* eliminando la posibilidad de eliminar o añadir nuevos elementos. Para ello crea el árbol *XML* haciendo uso de *DOM* y crea a partir de este el árbol de interfaz asociado con la información existente en él.

Para leer el árbol DOM del documento XML se hace uso de la siguiente función:

```
/**
 * Obtiene un árbol tipo DOM que almacena un XML del fichero indicado.
 * @param path Ruta del fichero XML a leer.
 * @return Documento con la información del XML leído.
 */
public static Document readXML(String path) {
    if(path == null) return null;
    Document doc = null;
    try {
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        dbf.setValidating(false);
        dbf.setFeature("http://apache.org/xml/features/nonvalidating/load-external-dtd", false);

        doc = dbf.newDocumentBuilder().parse(path);
    } catch (Exception ex) {
        Logger.getLogger(Config.class.getName()).log(Level.SEVERE, null, ex);
    }
    return doc;
}
```

Imagen 28 - Lectura de árbol usando tecnología DOM

### 5.2.5.1 Creación del árbol XML

En caso de existir un documento *DTD* asociado, el sistema carga en primer lugar este como árbol usando el procedimiento explicado con anterioridad. A continuación, crea el árbol *XML* usando *DOM* y el árbol de interfaz valiéndose de la estructura *DTD* y los datos y elementos existentes en el árbol *XML* recién creado.

### 5.2.5.2 Creación del árbol de interfaz XML

Para crear este árbol de interfaz seguiremos los mismos pasos que en la creación de un árbol de interfaz vacío, visto en el **Apartado 5.3.3.2.**, con la única diferencia de que, a la hora de añadir elementos, no se mira la cantidad asociada al elemento en el documento *DTD*, si no que se toman tantos elementos como existan en el *XML* cargado, junto con la información contenida en este.

Al crear los nodos hijos con sus interfaces se incluirán en los campos de texto de cada uno los valores tomados del *XML*.

### 5.2.5.3 Exportación

Para la exportación, se vuelve a hacer uso de *DOM* para el guardado el árbol *XML* creado con este estándar en un fichero externo. Para ello tan solo debe indicarse el directorio en el que se creará o modificará el fichero y la biblioteca se encargará de crear el archivo con la información contenida en el árbol *XML*.

Se indicarán también otros valores, como guardar el XML con indentaciones y el tamaño de estas. Para el guardado de un XML se usa pues el siguiente código:

```
try {
    Transformer transformer = TransformerFactory.newInstance().newTransformer();
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");
    transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
    transformer.transform(new DOMSource(doc), new StreamResult(new File(pathConfig)));
    Logger.getLogger(Config.class.getName()).log(Level.INFO, "Guardando");
} catch (Exception ex) {
    Logger.getLogger(Config.class.getName()).log(Level.SEVERE, null, ex);
}
```

Imagen 29 - Exportación de un árbol usando tecnología DOM

## 5.2.6 Modificación de la interfaz

Cuando la estructura *DTD* asociada al *XML* actualmente visualizado lo permite, la interfaz debe permitir la agregación o eliminación de elementos ateniéndose a la estructura determinada.

Para ello el sistema se vale de botones que permiten añadir un elemento vacío a la estructura, que además añaden un elemento vacío al árbol *XML* y al de interfaz. Este botón se ocultará en caso de no poder añadir ese elemento y se volverán a mostrar en caso de poder volver a añadirlo.

Cada elemento además contará con un botón de eliminación, que borra ese elemento tanto del árbol *XML* como del de interfaz. Este botón estará oculto cuando no sea posible eliminar más elementos de ese tipo y será visible cuando sí lo sea.

Esto crea una interfaz dinámica que se adapta a las necesidades del usuario, restringiendo su modificación a lo permitido por la estructura predeterminada para el documento *XML* tratado.

Todo esto será tratado en profundidad en el **Apartado 5.2.10**.

## 5.2.7 Creación de ficheros de formato y estilo

Para la personalización de la interfaz completa del programa se hizo uso de un único fichero *CSS*, siendo común para todos los módulos, permitiendo una unificación del estilo del programa.

Para dar formato a los archivos *XML* creados, cada conjunto *DTD* cuenta con un archivo de formato *XSLT*, que contiene la estructuración que se debe hacer de los datos para crear correctamente el *PDF*.

Al igual que con los archivos de formato, cada conjunto *DTD* cuenta con una hoja de estilo *CSS* específica que da las reglas de estilo necesarias al archivo generado con el fichero *XSLT* anterior.

### 5.2.7.1 Formato de documentos BOPA

Para crear los documentos *BOPA* se tomaron como referencia los últimos *PDF* de un *BOPA* de la novena legislatura.

En primer lugar se creó el encabezado que llevarán todas las páginas del documento excepto la primera. Este encabezado contendrá la fecha de generación del *BOPA*, el número de legislatura al que pertenece y el número de *BOPA* que es.

BOLETÍN OFICIAL DEL PARLAMENTO DE ANDALUCÍA		
Núm. 401	IX LEGISLATURA	26 de febrero de 2014

Imagen 30 - Encabezado *BOPA*



A continuación se crea el pie de página, que marcará el número de página actual en todas las páginas del documento excepto la primera.

Pág. 2
--------

Imagen 31 - Pie de página *BOPA*

Una vez creados estos elementos, se pasa a crear el comienzo del documento. Este mostrará una imagen con el nombre *BOPA*, junto con el **Sumario**.

En el **Sumario** se muestra una lista de los títulos de todas las iniciativas que se van a mostrar más detalladamente a lo largo del documento. Para obtener esta lista recorreremos el *XML* extrayendo los títulos de las iniciativas.

Por último, se muestra cada iniciativa por separado con todos sus elementos.

### 5.2.7.2 Formato de Diarios de Sesiones

Los diarios de sesiones tienen el encabezado y pie de página iguales a los *BOPA*, tan solo cambiando el nombre en los encabezados:

DIARIO DE SESIONES DEL PARLAMENTO DE ANDALUCÍA		
Núm. 79	IX LEGISLATURA	15 de mayo de 2014

Imagen 32 - Encabezado Diario de Sesión

No obstante, el comienzo del documento varía bastante.

Al comenzar, tras la imagen de portada se nos muestra la **Orden del día** y a continuación, el **Sumario**. En estos apartados, al contrario que en los *BOPA*, se nos muestran algunos (no todos) títulos de iniciativas, correspondientes a los que se han tratado en esa sesión. No todas las iniciativas que se muestren en documento estarán expuestas en estos apartados.

Para saber cuáles iniciativas se mostrarán en el Orden del día y el Sumario debemos leer el valor que tiene el atributo “**Aparece\_en\_orden\_del\_día**” y “**Aparece\_en\_sumario**” que tiene el elemento iniciativa. Tomando estos datos mostraremos aquellas iniciativas que aparezcan, agrupándolas en grupos según el elemento “**tipo\_iniciativa**”. De esta forma todas las iniciativas del mismo tipo aparecerán juntas bajo el nombre del tipo al que pertenecen.

Por último, al igual que en *BOPA*, se muestran todas las iniciativas, estén o no en el Orden del día o el Sumario, con todos sus detalles, obtenidos del *XML*.

## 5.2.8 Exportación de HTML / PDF

Existen diversos métodos para la exportación de un fichero *XML* a *PDF*.

El primer proceso a pensar es la conversión automática de *XML* a *PDF*. Por desgracia, este proceso no existe para Java con una forma fácil y modular para integrarse en el código, además de que aquellas bibliotecas que soportan la conversión de *XML* a *PDF* no dan facilidades a la hora de dar estilo al *PDF*, algo necesario para nuestra aplicación.

Se buscó entonces un pequeño desvío por el cual primero se crea un archivo temporal en formato *HTML* y, a continuación, este se convierte a *PDF*. Esta forma permite el sistema de estructuración ya conocida de *HTML* y, además, permite el uso de hojas de estilo *CSS* para la personalización del *PDF*. Por otro lado, la conversión de *XML* a *HTML* y de *HTML* a *PDF* están ampliamente documentados en la web como procesos separados, lo que facilita la implementación del proceso.

Otro punto a favor de este proceso propuesto es la creación de un conjunto de datos visibles en un navegador. Esto abre una vía de trabajo futura con el programa para que, en caso de necesitarlo, se genere, además del fichero *XML* y *PDF*, un archivo *HTML* sin necesitar grandes cambios en el código.

El proceso de creación de *PDF* se divide entonces en dos apartados, explicados a continuación.

### 5.2.8.1 Conversión de XML a XHTML

En primer lugar se toma la información en formato *XML* y se le da estructura y formato con el fichero *XSLT*. Este se encarga también de crear los elementos correspondientes a los que serán el encabezado y pie de página del *PDF*, así como del sumario y diferentes estructuras en las que está dividido el documento. En este momento se le asignará una clase o identificador a cada elemento, haciéndolos corresponder con una clase o identificador especificado en el archivo *CSS* creado.

Este paso creará un archivo *XHTML* funcional que será eliminado más tarde.

### 5.2.8.2 Conversión de XHTML a PDF

En segundo lugar se toma el fichero *XHTML* temporal generado con anterioridad y se emplea la biblioteca *iText* para la creación, junto con su fichero *CSS*, de un fichero *PDF*.

Este fichero tomará elementos específicos del *XHTML* como el elemento denominado “*header*” o el denominado “*footer*” para su uso como encabezado y pie de página respectivamente. Elementos no visibles en el archivo *XHTML*.

El código usado para la conversión de XML a PDF es el mostrado a continuación:

```
// Creando fichero HTML temporal a partir del XML y el XSLT
try {
    TransformerFactory tFactory = TransformerFactory.newInstance();
    Transformer transformer = tFactory.newTransformer(new StreamSource(xslPath));
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");
    transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
    transformer.transform(new StreamSource(xmlPath), new StreamResult(new FileOutputStream(htmlPath)));
} catch (Exception ex) {
    Logger.getLogger(FileManager.class.getName()).log(Level.SEVERE, "Excepcion al generar el XML. ", ex);
    return "No se ha podido crear el XML para generar el PDF.";
}

// Creando el PDF del HTML anterior
try (OutputStream os = new FileOutputStream(pdfPath)) {
    ITextRenderer renderer = new ITextRenderer();

    renderer.setDocument(new File(htmlPath));
    renderer.layout();
    renderer.createPDF(os);
    Logger.getLogger(FileManager.class.getName()).log(Level.INFO, "Creado PDF");
} catch (Exception ex) {
    Logger.getLogger(FileManager.class.getName()).log(Level.SEVERE, "Excepcion al generar el PDF. ", ex);
    return "No se ha podido generar el PDF.\n Pregunta a un administrador.";
}
```

Imagen 33 - Código de transformación de XML a PDF

## 5.2.9 Atajos de teclado

Para facilitar la transcripción de texto por parte del usuario, se ha añadido al programa un sistema de atajos de teclado para introducir texto de forma rápida.

El sistema de atajos permite al usuario determinar un conjunto de teclas con el cual se pegará en el elemento remarcado el texto que se ha asociado a este atajo.

El conjunto de atajos permite la combinación de las teclas **Control**, **Alt** y **Shift** (mínimo una de ellas) junto con una letra del alfabeto sin tildar.

Este conjunto de atajos y texto se almacenará en el sistema y permitirá ser exportado a un fichero externo escogido por el usuario para permitir el traspaso de atajos entre distintos dispositivos.

La lista de atajos es visible continuamente, de modo que se puede consultar y modificar fácilmente.



### 5.2.10.1.1 Menú

Esta sección muestra un acceso rápido a través de botones y menús de la amplia mayoría de acciones disponibles para el usuario.

Dentro del menú “*Archivo*” nos encontramos con los siguientes elementos:

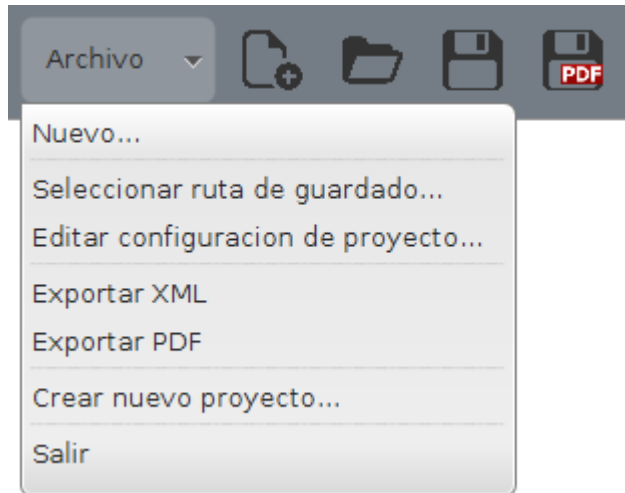


Imagen 35 - Interfaz – Menú desplegado

- **Nuevo...:** Este elemento abre el módulo de seleccionar un nombre para un nuevo *XML* (Módulo 5.2.10.2.2), permitiendo al usuario crear un nuevo documento vacío.
- **Seleccionar ruta de guardado:** Abre una interfaz de selección de directorio para seleccionar la ruta a partir de la cual se guardarán los archivos *XML* y *PDF* generados.
- **Editar configuración de proyecto:** Abre el módulo para la selección de *DTD* (Módulo 5.2.10.2.3). Tras seleccionar un *DTD* abre el módulo de configuración para ese *DTD* (Módulo 5.2.10.2.5)
- **Exportar XML:** Este elemento guarda los datos de la interfaz *XML* actual y los exporta en formato *XML*. Al finalizar muestra un mensaje de confirmación con la ruta de guardado del fichero, haciendo uso del módulo para mostrar mensajes de aviso (Módulo 5.2.10.2.9.).
- **Exportar PDF:** Este elemento guarda los datos de la interfaz *XML* actual y los exporta en formato *XML* y *PDF*. Al finalizar, muestra un mensaje de confirmación con la ruta de guardado del fichero *PDF*, haciendo uso del módulo para mensajes de aviso (Módulo 5.2.10.2.9.).
- **Crear nuevo proyecto...:** Abre el módulo de creación de un nuevo conjunto *DTD* (Módulo 5.2.10.2.4.), el cual permite al usuario añadir un nuevo *DTD* al sistema para poder ser usado.
- **Salir:** Permite al usuario cerrar el programa correctamente.



Esta tabla puede modificarse haciendo uso de los botones existentes en la barra inferior de esta zona:

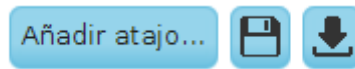


Imagen 37 - Interfaz – Botones inferiores de tabla de atajos

- **Añadir atajo...**: Este botón abre el módulo de edición/adición de atajos (**Módulo 5.2.10.2.8.**), permitiendo añadir elementos a la tabla.
- **Guardar**: Este botón con el icono de guardado abre la interfaz de guardado de ficheros (**Módulo 5.2.10.2.7.**) para guardar en un fichero con formato *XML* los atajos almacenados.
- **Importar**: Este botón con el icono de importar abre la interfaz de selección de ficheros (**Módulo 5.2.10.2.6.**) para permitir cargar atajos desde un fichero.

Dentro de la tabla, cada fila representa un evento de teclado que ejecuta la acción de añadir texto en el elemento de interfaz con focus en caso de poderse. Cada fila, por tanto, muestra la siguiente información:

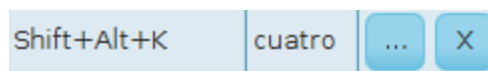


Imagen 38 - Interfaz – Fila de tabla de atajos

- **Atajo**: Texto que representa la combinación de teclas que activa el evento de teclado.
- **Texto**: Texto que se añadirá al ejecutar la combinación de teclas de ese atajo.
- **Editar**: este botón con puntos suspensivos abre el módulo de edición/adición de atajos (**Módulo 5.2.10.2.8.**), permitiendo modificar tanto el conjunto de teclas que se usarán para el evento como el texto que mostrará.
- **Eliminar**: Este botón con el símbolo de la cruz eliminará el evento de teclado indicado.

### 5.2.10.1.3 Zona XML

The image shows a web interface titled "BOPA" with a hierarchical form structure. The form is organized into several nested boxes, each representing an XML element. The top-level box is labeled "BOPA". Inside it, there are three main sections: "legislatura", "numero\_bopa", and "fecha". The "legislatura" section contains a single text input field. The "numero\_bopa" section contains a single text input field. The "fecha" section is further divided into three sub-sections: "dia", "mes", and "anio", each with its own text input field. Below the "fecha" section is a "desarrollo" section, which contains a "seccion" section. The "seccion" section has a "titulo" section with a text input field and a "subseccion" section. The "subseccion" section contains a button labeled "Añadir titulo". A faint watermark of the "PARLAMENTO DE ANDALUCÍA" is visible in the background of the form.

Imagen 39 - **Interfaz – Zona XML**

Zona principal del programa que muestra una interfaz acorde con el *DTD* seleccionado y las necesidades del usuario.

Cada elemento del *XML* mostrado puede o bien tener hijos o bien permitir al usuario añadirle texto como dato a este:

- En el caso de que un elemento tenga hijos, los mostrará a continuación de su nombre, dentro de un recuadro que aglomere a todos sus hijos y sean fácilmente separables del resto, aportando claridad visual y facilitando la labor del usuario.



fecha

dia

mes

año

PARLAMENTO DE ANDALUCÍA

Imagen 40 - Interfaz – Elemento XML con hijos

Para permitir toda la versatilidad de los archivos *DTD* usados, se toman en cuenta las cantidades de cada elemento según el *DTD*, en caso de permitir añadir un nuevo elemento de ese tipo, un botón aparecerá debajo de todos los elementos ya creados.

subseccion

Añadir título

Añadir iniciativa

Añadir subseccion

Imagen 41 - Interfaz – Elemento con botones

Además, en caso de existir uno o más elementos del mismo tipo y, si su cantidad según su *DTD* lo permite, se podrán eliminar los elementos pulsando en el botón junto a su nombre que aparecerá en caso de poder eliminarse.

título

X

Imagen 42 - Interfaz – Elemento con botón de eliminar

- En el caso de no tener hijos, el elemento mostrará un elemento de interfaz que permita al usuario introducir datos. El tipo de elemento de interfaz que se use para cada elemento vendrá definido en la configuración del *DTD* que se esté usando. En caso de no estar indicado en esta o, directamente si no existiese tal configuración, se consultaría en la configuración general del sistema. En caso de no encontrarse tampoco se establece por defecto el elemento de área de texto. Los elementos de interfaz existentes son:
  - **Lista:** Permite seleccionar entre una lista de valores predeterminados en la configuración del programa para ese elemento.



Imagen 43 - Interfaz – Elemento tipo lista

- **Línea de texto:** Permite añadir texto en un campo reducido de texto. Esto permite añadir muchos elementos de este tipo sin sobrecargar la interfaz.



Imagen 44 - Interfaz – Elemento tipo línea

- **Selección de fichero:** Contiene un campo de texto reducido como en el caso anterior y cuenta con un botón que abre el módulo de selección de fichero (**Módulo 5.2.10.2.6.**) para seleccionar la ruta del fichero a guardar para ese elemento.

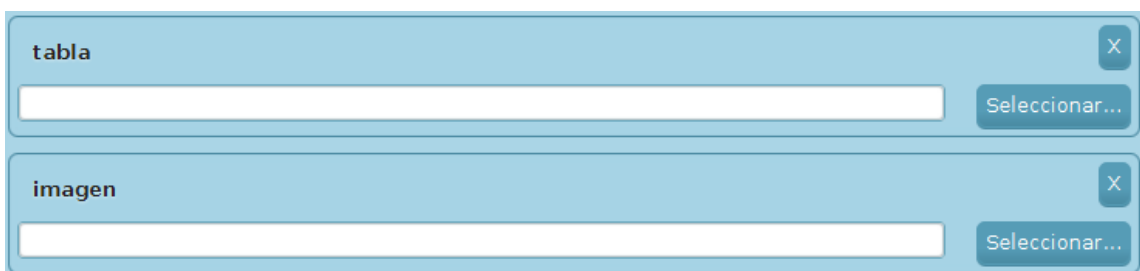


Imagen 45 - Interfaz – Elemento tipo imagen / tabla

- **Área de texto:** Elemento de interfaz por defecto, muestra un área mayor para la introducción de texto.

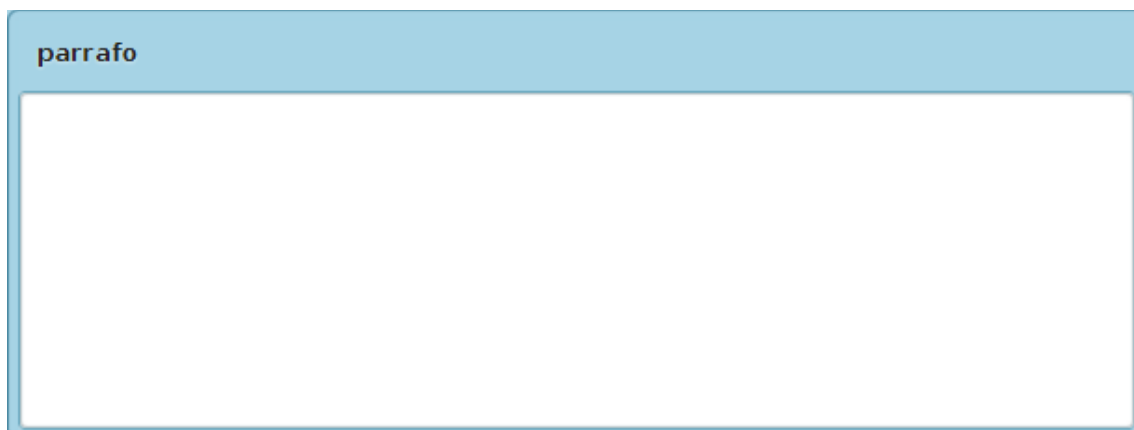
Una interfaz de usuario para un elemento de texto. En la parte superior, hay una barra de título azul con el texto "parrafo" en blanco. Debajo de esta barra, hay un área rectangular blanca con un borde azul, destinada a la introducción de texto.

Imagen 46 - Interfaz – Elemento tipo texto

En caso de que el elemento cuente con atributos, estos se mostrarán al comienzo del elemento, justo debajo de su nombre, en un color ligeramente diferenciado de los elementos. Estos atributos pueden tener cualquiera de las interfaces para la introducción de datos mostradas anteriormente o, además, la interfaz de botón “Si/No”, la cual muestra un botón que permite elegir entre ambas posibilidades.

Una interfaz de usuario para atributos. Se muestran dos atributos: "Aparece\_en\_orden\_del\_dia" y "Aparece\_en\_sumario". Cada atributo tiene un botón de selección. El botón para "Aparece\_en\_orden\_del\_dia" está etiquetado "Si" y es verde. El botón para "Aparece\_en\_sumario" está etiquetado "No" y es gris.

Imagen 47 - Interfaz- Atributos

### 5.2.10.2 Módulos

Además de la zona principal, el programa cuenta con módulos independientes que pueden ser llamados en cualquier momento de la ejecución del programa.

Estos módulos retiran el focus del área principal, centrándolo en ellos mismos, requiriendo que se termine de interactuar con ellos antes de poder hacer focus en otra área. Para indicar esto, todos los módulos ensombrecen visualmente a aquellas partes de la interfaz por debajo de ellos, colocándose ellos en la capa superior.

#### 5.2.10.2.1 Selección de opción para *XML* importado sin *DTD*

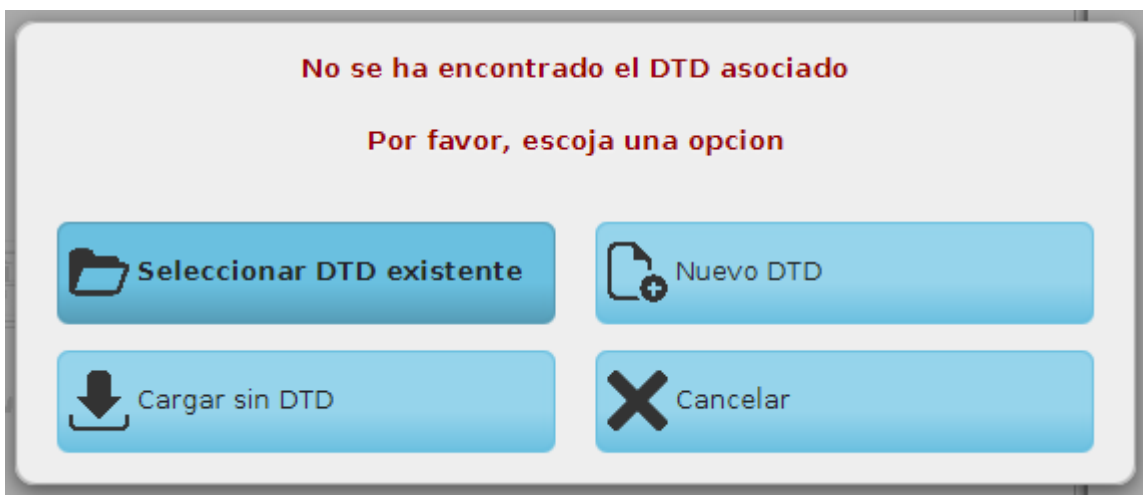


Imagen 48 - Interfaz – XML importado sin DTD

Este módulo de error permite elegir al usuario una opción a realizar al no encontrar un *DTD* asociado al *XML* que ha intentado importar. Se deberá escoger entre una de las siguientes opciones:

- **Seleccionar *DTD* existente:** Muestra el módulo de selección de *DTD* (Módulo 5.2.10.2.3.).
- **Nuevo *DTD*:** Muestra el módulo de creación de un nuevo conjunto *DTD* (Módulo 5.2.10.2.4.).
- **Cargar sin *DTD*:** Pide al sistema que se muestre el *XML* importado sin tener en cuenta ningún *DTD*. Esta opción bloqueará el poder eliminar o añadir elementos al *XML*, tan solo permitirá la edición de los campos ya existentes.
- **Cancelar:** Vuelve al módulo anterior.

Tras selección de una opción el módulo desaparece de la vista, mostrando el área principal o el módulo anterior en el caso de que se haya cancelado.

### 5.2.10.2.2 Selección de nombre de XML

La imagen muestra una ventana de diálogo con el título "Seleccione nombre para su XML". Dentro de la ventana, hay tres campos de entrada: "Tipo" con un menú desplegable que muestra "BOPA", "Legislatura" con un campo de texto vacío, y "Numero" con un campo de texto vacío. En la parte inferior derecha de la ventana, hay dos botones: "Aceptar" y "Cancelar".

Imagen 49 - Interfaz – Selección de nombre XML

Este módulo permite la selección de un nombre válido según el estándar establecido para el sistema. Este nombre se usará para determinar el *DTD* a usar y para nombrar al fichero *XML* o *PDF* exportado con su información.

Este módulo cuenta con las siguientes partes:

- **Selección de tipo:** Lista con los nombres de los *DTD* existentes en el sistema. Es obligatoria la selección de un valor de esta lista, y este indicará el *DTD* del que hará uso el nuevo *XML* creado.
- **Introducción de número de legislatura:** Este campo de texto permitirá al usuario introducir un número de hasta dos cifras indicando el número de legislatura correspondiente al *XML* a crear.
- **Introducción de número de XML:** Este campo de texto permitirá al usuario indicar el número de sesión actual, un valor de hasta 4 cifras.
- **Aceptar:** Este botón comprueba que los datos introducidos en cada apartado anterior son correctos. En caso de no ser así, se muestra un mensaje en la parte superior del módulo indicando el campo que ha dado error. En caso de ser todos los datos correctos, toma los datos de los campos, uniéndolos en un nombre y manda crear un interfaz de *XML* vacío tomando como base el *DTD* seleccionado.
- **Cancelar:** Cierra el módulo volviendo al módulo anterior a éste.

Tras la aceptación o cancelación anterior el módulo desaparece de la vista, mostrando el área principal o el módulo anterior en el caso de que se haya cancelado.

### 5.2.10.2.3 Selección de *DTD* entre los existentes

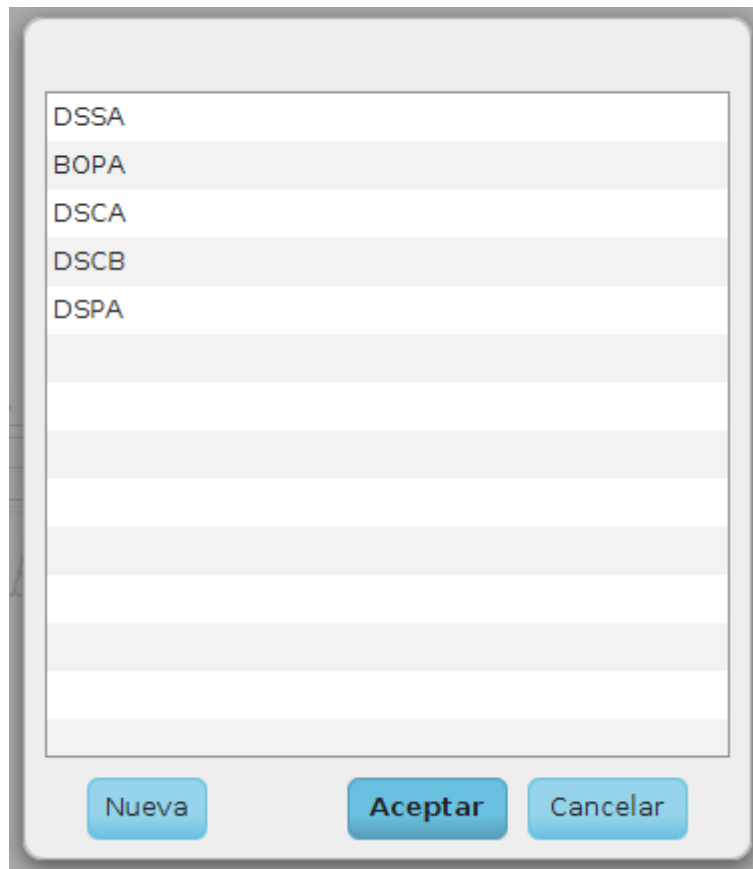


Imagen 50 - Interfaz – Selección DTD

Este módulo muestra una lista de los nombres de los conjuntos de *DTD* almacenados en el sistema, permitiendo seleccionar uno de ellos o crear un nuevo conjunto. Para todo esto cuenta con los siguientes elementos:

- **Lista de *DTD*:** Muestra una lista seleccionable con los nombres de los conjuntos *DTD* existentes en el sistema. Tan solo permite la selección de un elemento a la vez, al seleccionar uno nuevo el anterior es deseleccionado.
- **Botón nuevo:** Abre el módulo de creación de un nuevo conjunto *DTD* (Módulo 5.2.10.2.4.)
- **Botón aceptar:** Comprueba que tienes seleccionado un nombre de la lista anterior. En caso de no tener ningún nombre seleccionado muestra un texto de error en la parte superior del módulo. En caso de tener un elemento seleccionado pasa a la siguiente acción guardando ese nombre.
- **Botón cancelar:** Vuelve al módulo anterior sin realizar cambios.

Tras selección de una opción el módulo desaparece de la vista, mostrando el área principal o el módulo anterior en el caso de que se haya cancelado.

#### 5.2.10.2.4 Creación de nuevo conjunto *DTD*

La interfaz de creación de conjunto DTD se presenta como un formulario vertical con cinco pasos numerados en círculos azules. El paso 1, 'Archivo DTD', incluye un botón 'Abrir'. El paso 2, 'Escoge un nombre', incluye un campo de texto. El paso 3, 'Archivo de configuración del DTD', incluye botones 'Abrir' y 'Crear'. El paso 4, 'Archivo XSLT', incluye un botón 'Abrir'. El paso 5, 'Archivo CSS', incluye un botón 'Abrir'. En la parte inferior del formulario hay dos botones: 'Aceptar' y 'Cancelar'.

1 Archivo DTD  
Abrir

2 Escoge un nombre

3 Archivo de configuración del DTD  
Abrir Crear

4 Archivo XSLT  
Abrir

5 Archivo CSS  
Abrir

Aceptar Cancelar

Imagen 51 - Interfaz – Creación de conjunto DTD

Este módulo permite añadir al sistema el conjunto de archivos necesarios para la creación de un conjunto *DTD*.

Para poder crear este conjunto es necesario introducir todos los datos pedidos.

Cuando un dato haya sido introducido o no sea necesario se mostrará al usuario visualmente coloreando el círculo que contiene el número de opción al que pertenece el dato de tonos verdes.

En caso de que una opción permanezca aún sin introducir y sea necesaria, mostrará el mismo círculo con tonalidades azules.

Las opciones necesarias son las siguientes:

- **Archivo *DTD*:** Permitirá abrir el módulo de selección de fichero (**Módulo 5.2.10.2.6.**) para determinar la ruta del fichero *DTD* a usar.
- **Nombre:** Al abrir el archivo *DTD* tomará el valor del nombre de este, pudiendo ser modificado al gusto del usuario. Este será el nombre que usará el conjunto *DTD* y con el cual se formarán los nombres de sus ficheros asociados.
- **Archivo de configuración:** Permite elegir entre dos opciones para determinar el fichero de configuración que se usará para los elementos de interfaz creados por según el *DTD*:
  - **Seleccionar fichero *XML*:** Abre el módulo de selección de fichero (**Módulo 5.2.10.2.6.**) para obtener la ruta del fichero *XML* que almacena la configuración a usar.
  - **Crear fichero de configuración:** Abre el módulo de edición de configuración de *DTD* (**Módulo 5.2.10.2.5.**) para crear un archivo de configuración para los elementos del *DTD*.
- **Archivo *XSL*:** Abre el módulo de selección de fichero (**Módulo 5.2.10.2.6.**) para obtener la ruta del fichero *XSL* que se usará para dar formato a los ficheros PDF exportados de los *XML* que usen este conjunto *DTD*.
- **Archivo *CSS*:** Abre el módulo de selección de fichero (**Módulo 5.2.10.2.6.**) para obtener la ruta del fichero *CSS* que se usará para dar formato a los ficheros PDF exportados de los *XML* que usen este conjunto *DTD*.

Además de esta zona de opciones, el módulo cuenta con dos botones:

- **Aceptar:** Comprueba que todas las opciones estén completas y, en caso de ser así, crea o copia todos los archivos necesarios para la creación del nuevo conjunto de *DTD*. En caso de que alguna opción no esté completa marcará el círculo del número correspondiente con tonalidades rojas.
- **Cancelar:** Vuelve al módulo anterior sin hacer cambios.

Tras selección de una opción el módulo desaparece de la vista, mostrando el área principal o el módulo anterior en el caso de que se haya cancelado.



### 5.2.10.2.5 Edición de la configuración de DTD

The screenshot shows a configuration window for a DTD. It features a tree view on the left with expandable nodes (indicated by downward arrows). The nodes and their associated data types (shown in dropdown menus on the right) are as follows:

- bopa** (expanded)
  - legislatura**: Texto
  - numero\_bopa**: Texto
- fecha** (expanded)
  - dia**: Línea
  - mes**: Línea
  - anio**: Línea
- desarrollo** (expanded)
  - seccion** (expanded)
    - titulo**: Línea
  - subseccion** (expanded)
    - titulo**: (Ya definido)
  - iniciativa** (expanded)
    - numero\_expediente**: Texto
    - extracto**: Texto

At the bottom of the window are two buttons: "Aceptar" and "Cancelar".

Imagen 52 - Interfaz – Configuración DTD

Este módulo muestra un árbol sacado del árbol *DTD* que contiene el sistema, mostrando la estructura de hijos de cada nodo.

Cada elemento del árbol tiene tres opciones:

- Si contiene hijos y no está definido con anterioridad se muestra su nombre y colgando de éste sus elementos hijos. ç
- Si ya ha sido definido en un lugar superior del árbol, se mostrará el nombre del elemento junto con un texto informando de ello al usuario.

- Si no tiene hijos y no ha sido definido con anterioridad se muestra su nombre y un selector con los diferentes tipos de elementos que pueden ser. A saber:
  - **Línea:** Al mostrarse este elemento en la interfaz de *XML* tendrá un campo de texto reducido a modo de línea para introducir sus datos.
  - **Imagen:** Al mostrarse este elemento en la interfaz de *XML* tendrá, al igual que el anterior, un campo de texto de una línea y, además, un botón que abre el módulo de selección de fichero (**Módulo 5.2.10.2.6.**) para la selección de la ruta que se almacenará en este elemento.
  - **Texto:** Este elemento en la interfaz *XML* contendrá un área de texto extensa en la que poder introducir texto.

Además de esta zona de edición de la configuración, el módulo cuenta con dos botones:

- **Aceptar:** Crea el árbol *XML* asociado a la configuración indicada en la zona anterior.
- **Cancelar:** Vuelve al módulo anterior sin hacer cambios.

Tras selección de una opción el módulo desaparece de la vista, mostrando el área principal o el módulo anterior en el caso de que se haya cancelado.

### 5.2.10.2.6 Selección de fichero



Imagen 53 - Interfaz – Selección de fichero

Este módulo permite seleccionar un fichero existente.

Cada vez que es abierto muestra el directorio base del usuario. Además, al iniciar el módulo, el sistema le indica qué tipos de ficheros son los que se pueden ser seleccionados, de forma que en la lista sólo se mostrarán los directorios y los ficheros con las extensiones indicadas.

Tras selección de fichero o la cancelación el módulo desaparece de la vista, mostrando el área principal o el módulo anterior en el caso de que se haya cancelado.

### 5.2.10.2.7 Guardado de fichero

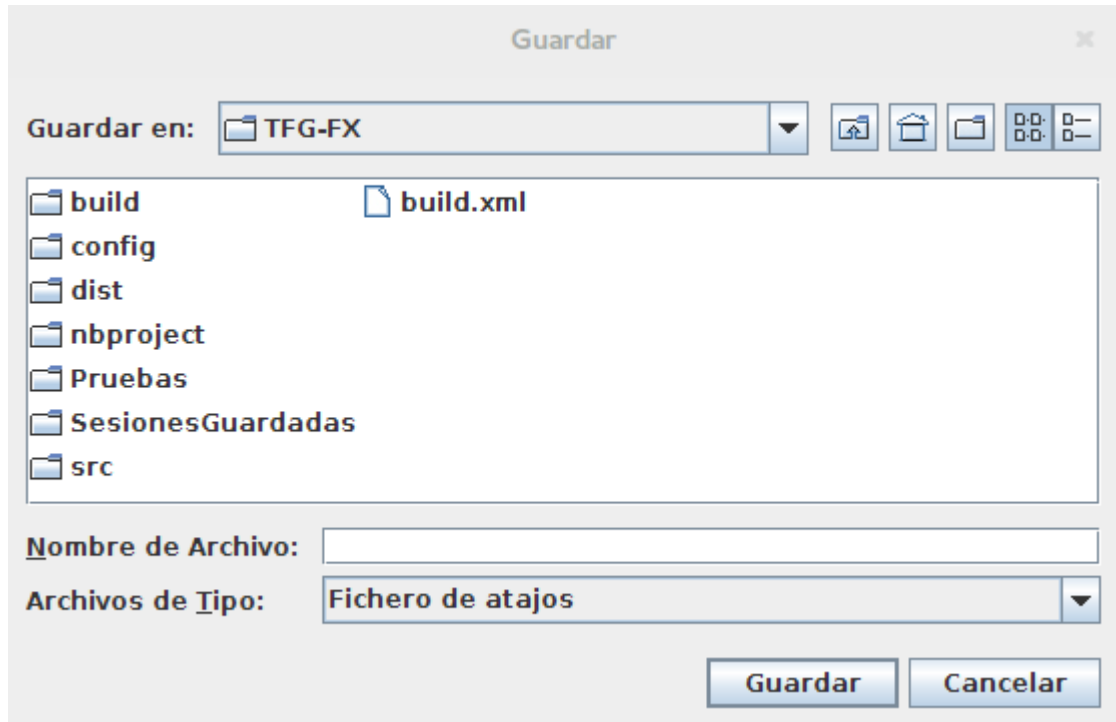


Imagen 54 - Interfaz – Guardado de fichero

Este módulo permite seleccionar un directorio donde almacenar el fichero a crear y su nombre.

Cada vez que es abierto muestra el directorio base del usuario. Además, al iniciar el módulo, el sistema le indica qué tipos de ficheros son los que se pueden ser seleccionados, de forma que en la lista sólo se mostrarán los directorios y los ficheros con las extensiones indicadas.

Tras selección de ruta y nombre o la cancelación el módulo desaparece de la vista, mostrando el área principal o el módulo anterior en el caso de que se haya cancelado.

### 5.2.10.2.8 Edición/Adición de atajo

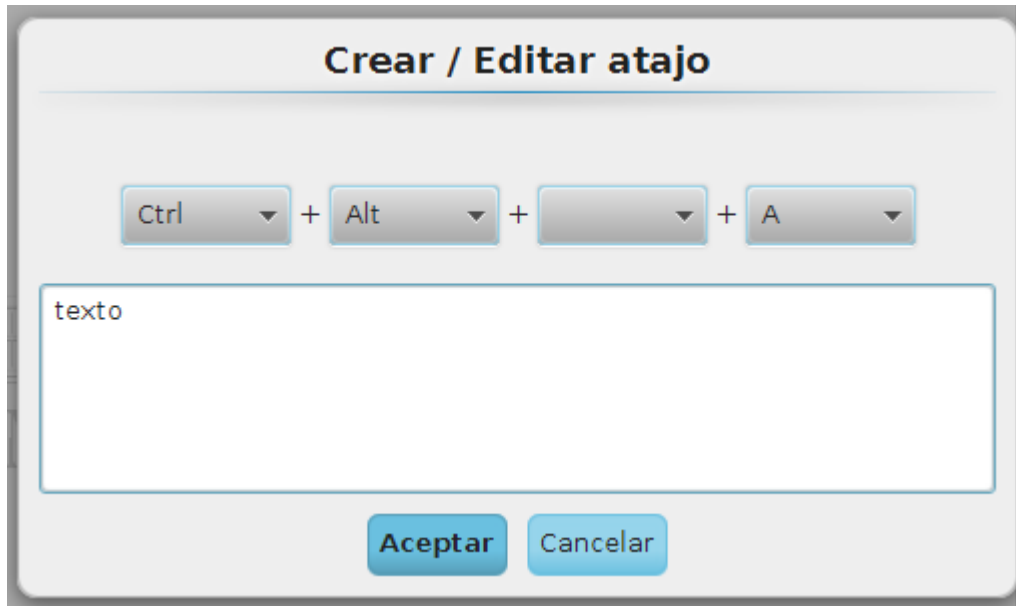


Imagen 55 - Interfaz – Modificación de atajo

Este módulo permite la edición o adición de atajos al sistema, para ello se divide en dos partes:

- **Definición del atajo:** Permite al usuario seleccionar la combinación de teclas que activarán el atajo. Permite cualquier combinación de las teclas Control, Alt y Shift (debe haber al menos una de esas teclas seleccionada) junto con una letra del abecedario no acentuada.
- **Introducción de texto:** Área de texto en la que el usuario podrá introducir el texto que desea que se escriba al pulsar la combinación de teclas escogida anteriormente.

Además de estas dos zonas cuenta con dos botones:

- **Aceptar:** Comprueba que se ha seleccionado al menos una tecla especial y crea o modifica el atajo.
- **Cancelar:** Cierra el módulo sin hacer ningún cambio.

Tras la aceptación o cancelación el módulo desaparece de la vista, mostrando el área principal o el módulo anterior en el caso de que se haya cancelado.

#### 5.2.10.2.9 Mensaje de aviso

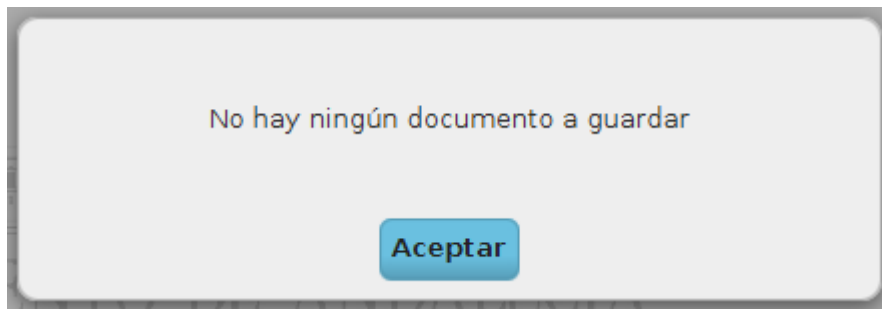


Imagen 56 - Interfaz – Mensaje de aviso 1

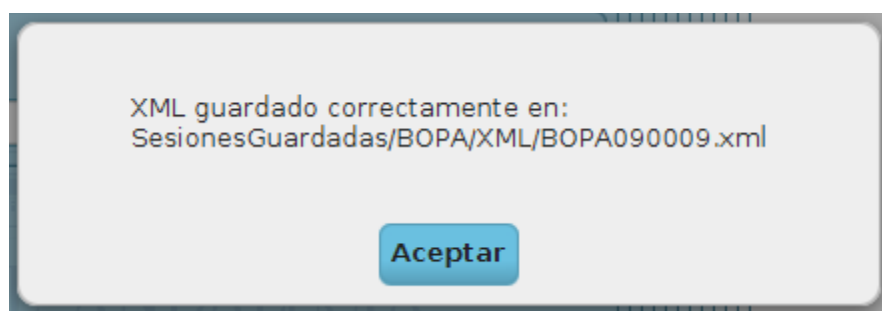


Imagen 57 - Interfaz – Mensaje de aviso 2

Este módulo muestra un mensaje variante de información para el usuario y un botón de aceptar para cerrar el módulo. Tras aceptar, el módulo desaparece de la vista volviendo al área principal o el módulo anterior.

# **6. Conclusiones**

---

## 6.1 Objetivos alcanzados

---

En este apartado retomaremos los objetivos planteados al comienzo de la memoria para analizar cuales se han cumplido, cuales no y por qué.

<b>Objetivo 1</b>	<i>Introducción de texto en el programa</i>
<b>Tipo</b>	Obligatorio
<b>Conseguido</b>	Si
<b>Detalles</b>	El programa permite introducir texto, estructurándolo según una interfaz creada según la estructuración predefinida.

<b>Objetivo 2</b>	<i>Estructuración del texto según un DTD</i>
<b>Tipo</b>	Obligatorio
<b>Conseguido</b>	Si
<b>Descripción</b>	El programa toma el <i>DTD</i> seleccionado y muestra una interfaz de usuario acorde a él, de este modo estructura el texto introducido.

<b>Objetivo 3</b>	<i>Guardado de texto estructurado en XML</i>
<b>Tipo</b>	Obligatorio
<b>Conseguido</b>	Si
<b>Descripción</b>	Dado que la interfaz estructura automáticamente el texto introducido, el paso de este a una estructura <i>XML</i> es inmediato.

<b>Objetivo 4</b>	<i>Guardado de texto en formato PDF</i>
<b>Tipo</b>	Obligatorio
<b>Conseguido</b>	Si
<b>Descripción</b>	Una vez estructurados los datos en formato <i>XML</i> , los ficheros predefinidos de formato y estilo, junto con las librerías y funciones creadas permiten la conversión a formato <i>PDF</i> .



<b>Objetivo 5</b>	<i>Importación de ficheros XML</i>
<b>Tipo</b>	Opcional
<b>Conseguido</b>	Si
<b>Descripción</b>	El sistema permite la carga de ficheros <i>XML</i> externos, permitiendo el mismo tipo de trato con estos que con las estructuras <i>XML</i> creadas en el programa. No obstante, bloquea la agregación o eliminación de elementos para aquellos <i>XML</i> importados sin un <i>DTD</i> asociado o con un <i>DTD</i> inválido.

<b>Objetivo 6</b>	<i>Agregación de nuevos métodos de estructuración</i>
<b>Tipo</b>	Opcional
<b>Conseguido</b>	Si
<b>Descripción</b>	Aunque sea algo que no se vaya a realizar continuamente, el programa permite añadir nuevos conjuntos de datos y ficheros para la estructuración y formato de archivos <i>XML</i> , esto permite no limitar el campo de acción del programa a los sistemas de estructuración actuales del <i>Parlamento de Andalucía</i> .

<b>Objetivo 7</b>	<i>Atajos de teclado</i>
<b>Tipo</b>	Opcional
<b>Conseguido</b>	Si
<b>Descripción</b>	Para facilitar la tarea del usuario, se ha creado un sistema de agregación, modificación y eliminación de atajos de teclado con un texto asociado a cada uno de ellos, permitiendo la rápida introducción de frases o palabras comúnmente usadas por el usuario.

<b>Objetivo 8</b>	<i>Autocorrección</i>
<b>Tipo</b>	Opcional
<b>Conseguido</b>	No
<b>Descripción</b>	Aunque esta funcionalidad facilitaría la tarea de corrección por parte del usuario, no ha podido ser implementada, no obstante, queda anotada en funcionalidades futuras del programa.

<b>Objetivo 9</b>	<i>Diferentes modos de introducción de datos</i>
<b>Tipo</b>	Opcional
<b>Conseguido</b>	Si
<b>Descripción</b>	<p>Cada DTD tiene asociado un fichero de configuración que indica de que tipo es cada elemento de este. Esto permite un tratado diferente para cada tipo de información a introducir. El código encargado de esto ha sido además modularizado para una fácil y rápida introducción de nuevos tipos de información.</p> <p>Aun así, queda como funcionalidad futura el permitir la creación y edición dentro del programa de tablas de contenido.</p>

## 6.2 Lecciones aprendidas

---

La elaboración de este proyecto me ha permitido aprender y mejorar en muchos campos del desarrollo software. Aunque ya conocía y manejaba el lenguaje de programación principal usado en el proyecto (Java), he podido mejorar y conocerlo más a fondo. Además he aprendido a manejar otros lenguajes de etiquetado que no había tratado con anterioridad, como es XML y XSLT, aumentando el número de lenguajes conocidos.

Este proyecto me ha permitido también emplear y mejorar mis habilidades de diseño gráfico, ya que ha hecho falta un estudio acerca de cómo diseñar la interfaz de usuario, cómo distribuir los elementos para ser lo más amigables posibles al usuario, etc.

Además, la complejidad y tamaño del proyecto a abarcar me han permitido desarrollar mi organización del tiempo, habilidades para crear una documentación clara, estudio de diferentes opciones a la hora de desarrollar el software, etc.

Una vez finalizado este proyecto creo haber logrado la experiencia necesaria para afrontar otro proyecto de iguales o incluso mayores magnitudes con un resultado mucho más afinado y depurado que antes de empezar con este.

## 6.3 Vías futuras

---

El programa contiene toda la funcionalidad requerida, pero aun así quedan aspectos por tratar dentro de él para seguir facilitando la tarea del usuario al tratar con esta aplicación. Varias funcionalidades que podrían ser añadidas en un futuro para la mejora del programa serían:

- Añadir el módulo para introducción y edición en el propio programa de tablas de contenido. Actualmente solo acepta un directorio de fichero con la tabla ya creada, en forma de imagen o PDF.
- Añadir la autocorrección mientras el usuario introduce el texto. El parseado y comprobación de las palabras que va introduciendo el usuario para saber si existen dentro de un diccionario preestablecido y su aviso y sugerencias en caso de no estarlo es algo que, aunque estaba planteado como objetivo, no ha podido ser implementado.
- Añadir expresiones regulares para la comprobación de los valores introducido según el tipo de elemento que sea. Esto permite limitar los fallos por parte del usuario a la hora de introducción de datos.
- Ya que una fase del proceso actual de transcripción de datos es la reproducción de audio para la transcripción de este a texto, el programa podría permitir la importación de estos ficheros para su reproducción en el propio programa.
- Permitir la modificación dentro del propio programa de la configuración de un DTD ya creado.

Aunque es un programa creado expresamente para el tratado de los diarios de sesiones y boletines del Parlamento de Andalucía, puede ser extendido con ligeros cambios a un gran abanico de campos de acción, por lo que el número de vías futuras en caso de querer ser usado en algún otro campo se amplía considerablemente. Teniendo en cuenta esto, quedan por tratar distintos aspectos de la aplicación como pueden ser:

- Creación de una interfaz de usuario genérica, no relacionada con el Parlamento de Andalucía.
- Permitir la creación de archivos DTD en el propio programa, que pueden ser usados para la estructuración de XML.
- Permitir una visualización previa del archivo XML o PDF a generar con los datos actuales.

# 7. Bibliografía

---

## 7.1 Enlaces de interés

---

- **Documentación sobre documentos oficiales del Parlamento de Andalucía:**
  - <http://www.parlamentodeandalucia.es/webdinamica/portal-web-parlamento/recursosdeinformacion/bopa.do>
  - <http://www.parlamentodeandalucia.es/webdinamica/portal-web-parlamento/recursosdeinformacion/diariosdesesiones.do>
- **Documentación sobre *Java*:**
  - <http://docs.oracle.com/javase/7/docs/>
- **Documentación sobre *JavaFX*:**
  - [http://docs.oracle.com/javase/8/javafx/get-started-tutorial/get\\_start\\_apps.htm#JFXST804](http://docs.oracle.com/javase/8/javafx/get-started-tutorial/get_start_apps.htm#JFXST804)
  - <http://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html>
  - [http://docs.oracle.com/javase/8/javafx/api/javafx/fxml/doc-files/introduction\\_to\\_fxml.html](http://docs.oracle.com/javase/8/javafx/api/javafx/fxml/doc-files/introduction_to_fxml.html)
- **Documentación sobre *CSS*:**
  - <http://www.w3schools.com/css/>
- **Documentación sobre *XSLT*:**
  - <http://www.w3schools.com/xsl/>
- **Documentación sobre *HTML*:**
  - <http://www.w3schools.com/html/>
- **Documentación sobre *DTD*:**
  - <http://www.w3schools.com/dtd/>
  - [http://docs.oracle.com/cd/A87860\\_01/doc/appdev.817/a86030/adx18cl5.htm](http://docs.oracle.com/cd/A87860_01/doc/appdev.817/a86030/adx18cl5.htm)
- **Documentación sobre *Java DOM*:**
  - [http://docs.oracle.com/cd/B10501\\_01/appdev.920/a96621/adx04paj.htm](http://docs.oracle.com/cd/B10501_01/appdev.920/a96621/adx04paj.htm)
  - <https://docs.oracle.com/javase/tutorial/jaxp/dom/readingXML.html>
- **Documentación sobre *iText*:**
  - <http://itextpdf.com/api>

# 8. Anexos

---

---

# 8.1 Glosario de términos

---

## 8.1.1 Términos

A continuación se listan los términos específicos del dominio del problema con una breve definición de cada uno.

- **Interfaz de usuario (UI):** Es el medio con que el usuario puede comunicarse con nuestro programa.
- **Evento de teclado:** Eventos que se activan al usuario presionar una o varias teclas de su teclado.
- **Java:** Lenguaje de programación de propósito general, concurrente y orientado a objetos.
- **JavaFX:** Tecnología para la creación de aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio.

## 8.1.2 Acrónimos

A continuación se listan los acrónimos específicos del dominio del problema con su correspondiente significado.

- **XML:** *eXtensible Markup Language* (“lenguaje de marcas extensible”), es un lenguaje de marcas utilizado para almacenar datos en forma legible.
- **DTD:** *Document Type Definition* (“definición de tipo de documento”), es una descripción de estructura y sintaxis de un documento *XML*.
- **PDF:** *Portable Document Format* (“formato de documento portátil”), es un formato de almacenamiento para documentos digitales
- **HTML:** *HyperText Markup Language* (“lenguaje de marcas de hipertexto”), es un lenguaje de marcado para la elaboración de páginas web.
- **XHTML:** *eXtensible HyperText Markup Language* (“lenguaje de marcas de hipertexto extendido”), es *HTML* expresado como *XML* válido.
- **CSS:** *Cascading Style Sheets* (“Hoja de estilo en cascada”), es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en *HTML* o *XML* (y por extensión en *XHTML*).
- **XSL:** *Extensible Stylesheet Language* (“lenguaje extensible de hojas de estilo”), es una familia de lenguajes basados en el estándar *XML* que permite describir cómo la información contenida en un documento *XML* cualquiera debe ser transformada o formateada para su presentación en un medio.



- **XSLT:** *XSL Transformations* (“Transformaciones XSL”), es un estándar que realiza la transformación del documento utilizando una o varias reglas de plantilla.
- **MVC:** *Modelo Vista Controlador* es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones en distintos componentes.
- **API:** *Application Programming Interface* (“Interfaz de programación de aplicaciones”), es el conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
- **DOM:** *Document Object Model* (“Modelo de Objetos del Documento” o “Modelo en Objetos para la Representación de Documentos”), es una *API* que proporciona un conjunto estándar de objetos para representar documentos *HTML* y *XML*.
- **IDE:** *Integrated development environment* (“entorno de desarrollo integrado”), es una aplicación de software que proporciona una serie de servicios integrales para facilitar al programador el desarrollo de software.

## 8.2 Manual de usuario

---

### 8.2.1 Antes de empezar

Este programa funciona correctamente en los sistemas operativos *Windows 7* en adelante y *Linux*.

Se necesita tener instalado *Java* en su versión más reciente para su correcta ejecución. Puede descargar la versión más reciente en desde el siguiente enlace:

- <https://www.java.com/es/download/>

### 8.2.2 Creando un nuevo documento

Al iniciar la aplicación esta aparece vacía. Para abrir un nuevo documento en blanco existen dos maneras, ambas desde el menú superior:

- Pulsando el botón de **Nuevo** del menú.



Imagen 58 - Manual de usuario – Botón - Nuevo...

- Accediendo al menú **Archivo** y seleccionando la opción **Nuevo...**

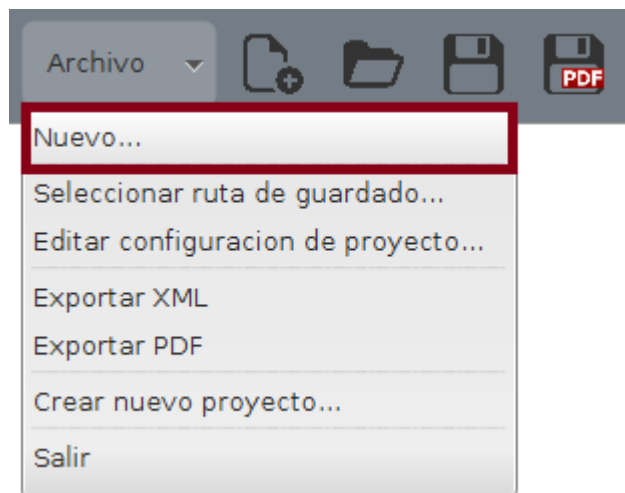


Imagen 59 - Manual de usuario – Archivo – Nuevo...

A continuación se abrirá una ventana para seleccionar el nombre del nuevo documento. Este nombre constará de 3 partes:

- **Tipo de documento:** Indicará la estructura que tendrá el documento y formará las primeras letras del nombre.
- **Legislatura:** Número de hasta 2 cifras que indica la legislatura actual.
- **Número:** Número de hasta 4 cifras indicando el número de serie para esa legislatura del documento.

### 8.2.3 Importando un documento *XML*

Para importar un documento *XML* al programa y poder editarlo existen se pulsará el botón **Abrir** del menú:



Imagen 60 - Manual de usuario – Importar XML

Tras esto se abrirá una ventana de selección de fichero para indicar el fichero que se quiere abrir.

### 8.2.4 Editando el documento en el programa

Una vez cargado el documento, aparecerá en el área principal del programa. Y ya se puede introducir el texto en los campos deseados.

Para añadir nuevos elementos al documento, se hará uso de los botones “**Añadir XX**”, donde “**XX**” indica el nombre del elemento a añadir.

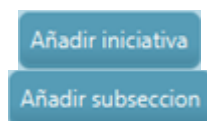


Imagen 61 - Manual de usuario – Botón de añadir elemento

También se podrán eliminar elementos no deseados pulsando el botón con la cruz situado a la derecha del nombre del elemento a eliminar.



Imagen 62 - Manual de usuario – Botón de eliminar elemento

- **Nota:** Sólo se podrá añadir o eliminar elementos del documento cuando lo permita la estructura con la que se ha creado este.

## 8.2.5 Seleccionando ruta de guardado

Para seleccionar la ruta en la que se almacenarán los nuevos XML y PDF exportados selecciona en el menú “**Archivo**” la opción “**Seleccionar ruta de guardado**”:

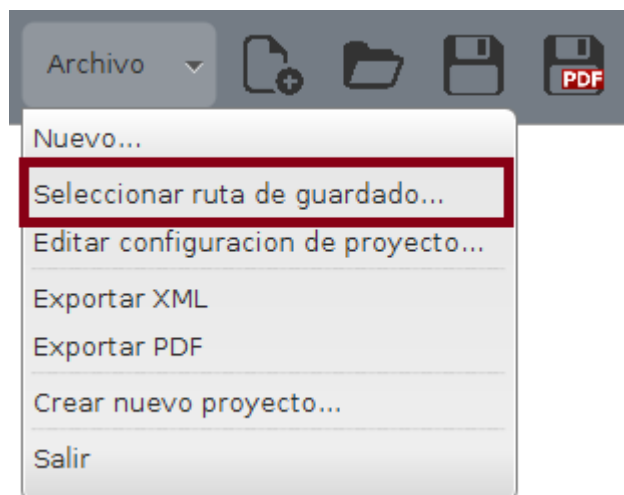


Imagen 63 - Manual de usuario – Selección de ruta de guardado

A continuación se selecciona el directorio deseado.

## 8.2.6 Editar configuración del proyecto

Para cambiar la interfaz con la que se muestran los elementos de un determinado tipo, seleccionamos en el menú “**Archivo**” la opción “**Editar la configuración de proyecto**”:

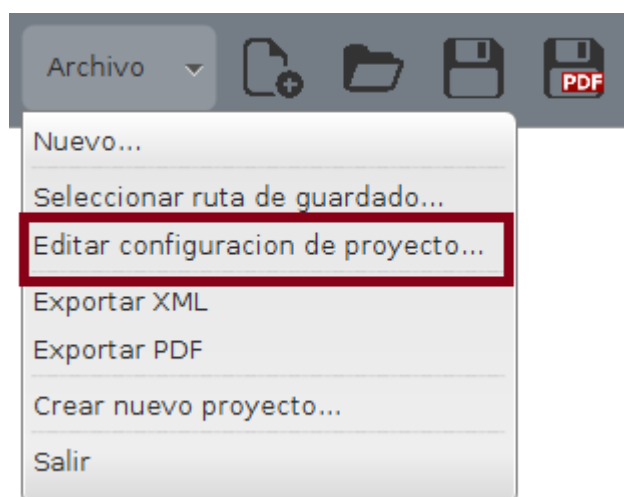


Imagen 64 - Manual de usuario – Editar configuración de proyecto

A continuación se mostrará una lista de los proyectos existentes actualmente. Selecciona el que desees modificar y pulsa **Aceptar**.

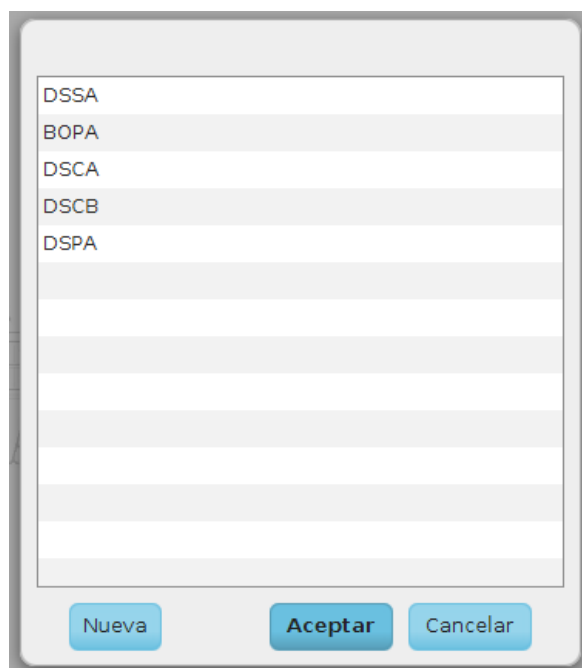


Imagen 65 - Manual de usuario – Selección de proyecto a modificar

Se abrirá a continuación una ventana en donde se mostrarán todos los elementos ordenados a modo de árbol.

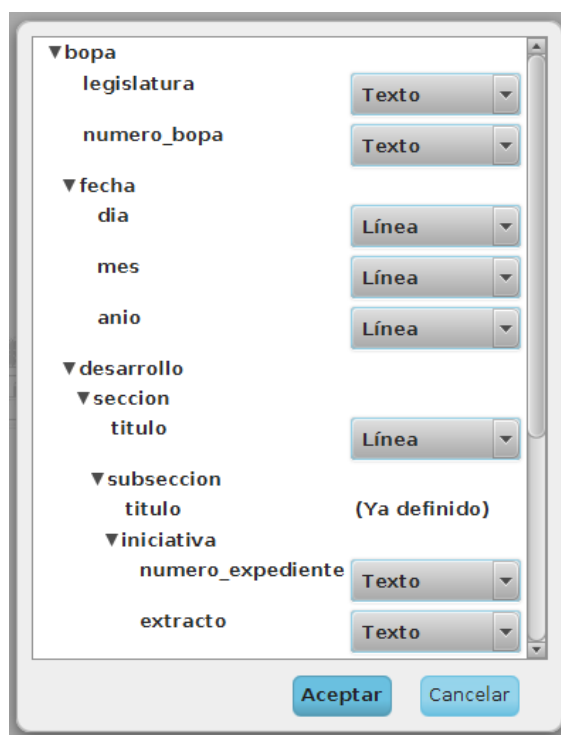


Imagen 66 - Manual de usuario – Modificación del proyecto

Modificar los valores de los elementos deseados seleccionando el tipo de interfaz que se quiere que tenga dentro de la lista desplegable junto a cada elemento. Una vez finalizado, pulsar **Aceptar**.

- **Nota:** Los cambios se aplicarán solo a los documentos creados tras la modificación.

## 8.2.7 Exportando un documento a XML

Una vez editado al gusto el documento en el programa se puede guardar en formato *XML* de las siguientes formas:

- Pulsando el botón de **Guardar** del menú.



Imagen 67 - Manual de usuario – Botón de guardar XML

- Accediendo al menú **Archivo** y seleccionando la opción **Exportar a XML...**

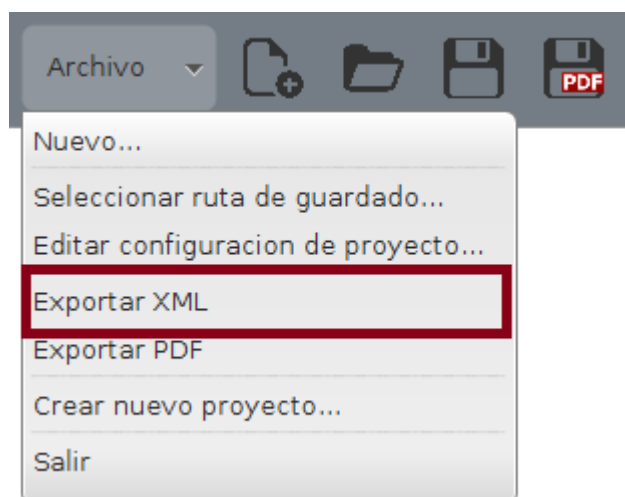


Imagen 68 - Manual de usuario – Archivo – Exportar a XML

A continuación saldrá un mensaje de confirmación donde se indica donde se ha guardado el documento generado.

- **Nota:** Si el fichero ya existe se sustituirá.

## 8.2.8 Exportando un documento a *PDF*

Una vez editado al gusto el documento en el programa se puede guardar en formato PDF de las siguientes formas:

- Pulsando el botón de **Guardar *PDF*** del menú.



Imagen 69 - Manual de usuario – Botón de guardar PDF

- Accediendo al menú **Archivo** y seleccionando la opción **Exportar a *PDF*...**

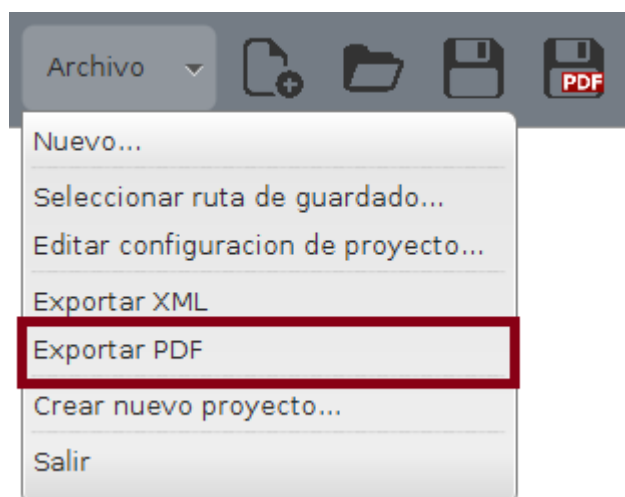


Imagen 70 - Manual de usuario – Archivo – Exportar a PDF

A continuación saldrá un mensaje de confirmación donde se indica donde se ha guardado el documento generado.

- **Nota:** La generación de *PDF* genera también un fichero *XML* del mismo documento.
- **Nota 2:** Si los ficheros ya existen se sustituirán.

## 8.2.9 Sistema de atajos de teclado

Para facilitar la tarea de transcripción del usuario se hace utiliza en el programa una tabla de atajos de teclado, donde se muestra una lista de aquellos atajos disponibles junto con el texto que producen al ejecutarlos.

### 8.2.9.1 Añadir un nuevo atajo de teclado

Para añadir un atajo se tiene que pulsar el botón de Añadir atajo... Existente debajo de la tabla de atajos. Este botón abrirá una ventana de interacción para seleccionar la combinación de teclas que tendrá el atajo y el texto a generar.

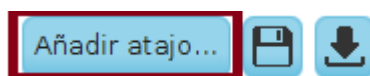


Imagen 71 - Manual de usuario – Añadir atajo

Tras aceptar, el nuevo atajo será generado y añadido a la tabla de atajos.

### 8.2.9.2 Editar un atajo de teclado existente

Para editar un atajo de teclado debe pulsarse el botón “...” que se encuentra en parte derecha de la fila a la que pertenece el atajo.

Este botón abrirá una ventana con los datos actuales del atajo para modificarlo.

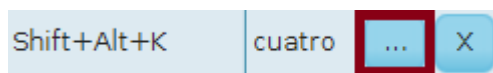


Imagen 72 - Manual de usuario – Editar un atajo

Tras aceptar, se modificará el atajo en la tabla.



### 8.2.9.3 Eliminar un atajo de teclado existente

Para eliminar un atajo debe pulsarse el botón “X” situado en la misma fila del atajo a eliminar.

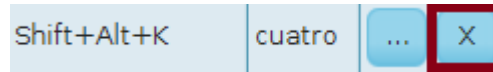


Imagen 73 - Manual de usuario – Eliminar un atajo

Tras pulsar el botón el atajo se borrará de la tabla.

### 8.2.9.4 Importar atajos

Si se quiere cargar una lista de atajos, se debe pulsar el botón “Importar” situado en la parte inferior de la tabla de atajos.

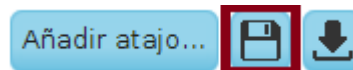


Imagen 74 - Manual de usuario – Importar atajos

Este botón abrirá una ventana de selección de fichero, para que se seleccione el archivo que contiene los atajos. Este fichero debe tener extensión “.xml”.

Una vez aceptado se sustituirán los atajos de la tabla por los leídos del fichero.

### 8.2.9.5 Exportar atajos

Para guardar una lista de los atajos actuales en un fichero externo debe pulsarse el botón de “Guardar” situado en la parte inferior de la tabla de atajos.

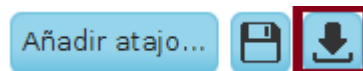


Imagen 75 - Manual de usuario – Exportar atajos

Tras pulsar el botón aparecerá una ventana de selección de directorio donde el usuario indicará el nombre y la ruta del fichero a crear.

- **Nota:** Si el fichero ya existe se sustituirá.