Desarrollo de Surrogates para la representación dinámica de la información digital

Eduardo Maurel Ramos

Proyecto Fin de Carrera
Director: Ayman Moghnieh
Junio 2009
Ingeniería Técnica en Informática de Sistemas
Universitat Pompeu Fabra

Agradecimientos

Después de meses de arduo trabajo ha llegado el momento de redactar estas ansiadas líneas. En primer lugar quiero dar las gracias a todas aquellas personas que me han apoyado durante todo el tiempo que ha durado el proyecto. Sobretodo a aquellas personas que he tenido cerca en los momentos difíciles y que han tenido que soportar mis angustias y preocupaciones en primera persona. Especialmente se las quiero dar a mis padres y a María, la persona sin la que no hubiera podido realizar este proyecto de ninguna manera, ya que ha tenido que levantarme en los momentos difíciles y me ha prestado su apoyo incondicional en todo momento.

También quiero dar mi eterno y sincero agradecimiento a Ayman, mi director de proyecto y guía. Ya que sin su inestimable ayuda, apoyo, confianza y gran dosis de paciencia no habría podido conseguir este logro.

Igualmente quiero agradecer a mi hermano Natxo, a su mujer Eva y también a Miguel, el tiempo que me han dedicado para llevar a cabo las pruebas de este proyecto. Sus palabras de ánimo me ayudaron muchísimo.

Por último, quiero dar las gracias también al resto de compañeros que me han ayudado a recoger requerimientos y hacer las pruebas. A los que se ofrecieron a trabajar en equipo y a los amigos que siempre se preocuparon por el estado de este proyecto no sin desearme siempre buena suerte.

A todos vosotros, mil gracias.

Resumen

El volumen de información a la que un usuario tiene acceso en Internet es enorme. La representación de la misma ha ido evolucionando a lo largo del tiempo a la vez que se han establecido reglas, recomendaciones y directrices que conforman toda una ingeniería de diseño de interfaces con el propósito de mejorar la representación e interacción entre el usuario y el ordenador. Aún así, hay que ir evolucionando en la manera de tratar esta representación de la información ya que Internet es un espacio donde afloran, día tras día, nuevas tecnologías y servicios.

De hecho, en este proyecto trataremos de mejorar la representación de la información mediante unos objetos que llamaremos surrogates o sustitutos. Como su nombre indica, estos sustitutos serán la representación de los elementos a reemplazar y cuya función podríamos calificar como de "puerta de enlace". Dichos objetos estarán formados por elementos propios de la información a la que representan con el objetivo de que el usuario identifique de una manera más rápida y sencilla esta información.

En definitiva, desarrollaremos unos surrogates que puedan ser usados en diferentes interfaces independientemente de la tecnología que se haya empleado para su desarrollo y dotándolos de una serie de características y métodos con el fin de mejorar la interacción entre el usuario y la información.

Abstract

The volume of information that a user access on the web is enormous. The representation of this information has been evolving since the web gained popularity, making way for the establishmment of rules, recommendations and directives. These are evidence of the efforts to try and satisfy the engineering of interface designs that improve the representation and interaction between the user and the information. Nevertheless, on the web the number of technologies and services is growing day after day, posing more design challenges, and because of this it's important to think more about facilitating the representation of the information it hosts.

In fact, in this project we try to improve the representation of the online information by introducing a new surrogate definition to facilitate the production and transfer of standardized or partially standardized representations of information elements. The surrogates are composed from several information ítems extracted from the same element. The resulting representation makes it easy on the user to identify and understand the information represented.

To sum up, we are going to develop new surrogate structures and use them in different interfaces in order to improve the interaction between the user and the information.

Índice

Índice de las figuras

Planificación del Proyecto

En este apartado se definirán las etapas que se han desarrollado para la ejecución del proyecto:

Período	Descripción
Octubre - Noviembre 2008	Investigación y lectura de bibliografía recomendada
12 -16 Enero 2009	Redacción de los objetivos, resumen y planificación del proyecto
17 enero - 8 Febrero 2009	Redacción de la literatura y el trabajo relacionado
9 -22 Febrero 2009	Analizar el problema planteado
23 - 28 Febrero 2009	Descripción del sistema a implementar para obtener la solución
1 Marzo - 31 Mayo 2009	Implementación de los componentes del programa
1 - 12 Junio 2009	Realización de las pruebas y extracción de los resultados
15-18 Junio 2009	Corrección de los fallos en la memoria

1- Introducción

Hasta no hace muchos años atrás, era impensable encontrarnos rodeados por una cantidad tan elevada de dispositivos electrónicos como en la actualidad. De hecho, apenas se disponían de éstos dispositivos. Pero a lo largo de los últimos años, hemos cambiado esta realidad progresivamente a un ritmo muy acelerado hasta llegar a un punto en que estamos rodeados de dispositivos que nos ofrecen información constantemente. Hemos pasado de la Web 1.0 a la Web 2.0 casi sin darnos cuenta y sabemos que todavía queda un largo camino lleno de innovación por recorrer.

La información, a su vez, se nos representa de forma interactiva a través de multitud de interfaces. Cada día consultamos información a través de un número elevado de éstas, como por ejemplo, en el ordenador de sobremesa, en ordenadores portátiles, teléfonos móviles, pantallas interactivas, entre otros elementos que formarían una lista muy extensa. Lo más sorprendente de todo, es que cada dispositivo presenta la información a su manera a través de una interfaz y dicha información se manipula de una manera diferente en cada caso.

Reflexionando, nos daremos cuenta que en el transcurso de un día, llegamos a interactuar con un número elevado de aparatos electrónicos que nos muestran información. Interactuamos con ellos sin habernos percatado de que para interactuar con cada uno de ellos correctamente hemos tenido que aprender a desenvolvernos con sus interfaces y que éstas han evolucionado muy rápido también ya que detrás ha habido todo un estudio de diseño de interfaces.

Aún así, no hay establecido un patrón para representar la información. De hecho, si nos fijamos en la información que podemos consultar en internet, comprobaremos que cada aplicación o web trata a la información de una manera distinta. Nos podemos perder en un volumen enorme de información y todo y que se están haciendo progresos significativos en cuanto a ordenación de ésta, no encontramos métodos de visualización que faciliten la búsqueda. Es por ello, que intentaremos ofrecer unos elementos con los que representar cualquier tipo de información de manera que facilite al usuario su búsqueda.

2 - Definición del proyecto

Durante los últimos años, Internet ha ido creciendo de forma exponencial por lo que el volumen de información a la que un usuario tiene acceso es enorme. En un primer momento y ante este crecimiento tan brutal se han buscado sistemas para recopilar y ordenar la información más relevante a través de directorios y motores de búsqueda sin prestar atención a las capacidades cognitivas de búsqueda del usuario. Actualmente, esto ha cambiado y la tendencia pasa por ofrecer al usuario, información relacionada con el objetivo de su búsqueda. Aún así, la forma en que se sigue presentando dicha información suele ser en forma textual y el usuario debe seguir haciendo un esfuerzo para buscar y filtrar la información que está buscando.

A grandes rasgos, lo que se pretenderá en este proyecto será diseñar un prototipo de representación gráfica que mejore los métodos de visualización de la información ya existentes en la red. Aplicaremos conceptos del minimalismo para abstraer al usuario de volúmenes enormes de información presentándole ésta de manera gráfica e interactiva de forma que la atención del usuario a estos elementos gráficos (*surrogates*) será mayor y por consiguiente, mejoren su búsqueda.

Para llevar a cabo esta propuesta, necesitaremos hacer uso de fuentes web de noticias, en nuestro caso recogidas desde los canales RSS de Yahoo News **[11]** y compararemos nuestro prototipo de representación gráfica con el de su sitio web.

Para conseguir el objetivo de mejorar la interacción entre el usuario y el servicio de noticias dotaremos a nuestro prototipo de características propias del minimalismo con el objetivo de que el usuario pueda entender la información de una manera más eficaz.
Para comprobar que nuestro prototipo presenta beneficios realizaremos unas pruebas con un grupo de usuarios. Estos usuarios nos ayudarán a extraer datos sobre si nuestros prototipos están consiguiendo su propósito o no.

Un primer experimento estará enfocado a nivel de elementos. Es decir, entender cómo la gente entiende los *surrogates* (nuestro sistema de representar la información). Cómo identifica una noticia concreta.

El segundo experimento estará enfocado a nivel de colecciones. Los usuarios recibirán un número de elementos, y comprobaremos cómo realizarán búsquedas y cómo entenderán los espacios informativos que formaran estas colecciones.

En conclusión, nuestros elementos estarán enfocados a mejorar la interacción entre el usuario y los espacios informativos principalmente, modelando las noticias a un nuevo formato que no sean una simple muestra textual. Por lo que este proyecto estará orientado al estudio del área de *Human Interaction Interface (HII)*. Trataremos de experimentar y avanzar en el campo de la interacción entre las personas y los ordenadores haciendo que la interacción entre éstos sea más intuitiva y requiera menos esfuerzo por parte del usuario que la esté utilizando.

3 - Literatura y trabajo relacionado

3.1 - Web 2.0

Internet ha sufrido en los últimos años grandes cambios en cuanto al uso que se le ha dado. Se ha pasado de una generación de páginas web estáticas donde el usuario simplemente se limitaba a consultar la información que se le presentaba y donde las actualizaciones de éstas se producían esporádicamente por lo que la interacción era inexistente a otra generación donde se presta una total atención hacia el usuario final.

Esta segunda generación de Internet está basada en una serie de herramientas y servicios que se les presta a comunidades de usuarios para que sean éstos mismos, los usuarios finales, los que se encarguen de ofrecer, ampliar y/o modificar su contenido, o bien, **cambiar la forma de presentar la información,** o bien, ambas cosas a la vez. Por lo que los creadores de la información son a su vez, los consumidores de la misma.

Además, un aspecto clave de la web 2.0 es la **redifusión** de la información. De esta manera, con este cambio de filosofía y uso de la web, se está priorizando la mejora en cuanto a cómo se almacena y gestiona todo el volumen de la información de una manera efectiva. De hecho, para poder llevar a cabo esta compartición de información de manera eficaz entre webs se han creado formatos estándar como RSS**[5]** (Really Simple Syndication) o Atom.

Llegados a este punto, nos encontramos con la posibilidad de aprovechar las ventajas que ofrece la web 2.0 para estudiar cómo se está tratando la visualización de la información que se va transfiriendo por la red y lo que es más importante, cómo mejorarla.

Actualmente, la web 2.0 nos ofrece una serie de servicios tales como blogs, wikis, redes sociales de diferentes índoles o las folcsonomías, que fomentan la colaboración y el intercambio ágil de información entre los usuarios. De hecho, si nos centramos en el campo de los espacios informativos o las noticias, vemos como existen diferentes portales de cada medio de comunicación donde la presentación de las mismas es muy similar. Son páginas con un sinfín de titulares donde también existen numerosos menús para tratar de categorizar la información en bloques o espacios informativos.

3.2 - Visualización de la información.

Desde la perspectiva de la óptima visualización de la información, la riqueza de un entorno de información no radica solamente en una alta densidad de información, también radica en una baja demanda cognitiva en los esfuerzos necesarios para acceder a sus diferentes segmentos por parte del usuario.

El sistema de visualización, por lo tanto, debe conciliar estas dos necesidades. Ha de generar un rico entorno de información con un sistema poco exigente de navegación. Esta afirmación está fundamentada en los estudios de Miller **[1]** sobre la psicología cognitiva, ya que éste sostiene que las limitaciones de los sentidos humanos pueden afectar a nuestra capacidad de percepción y el tratamiento de la información recogida por éstos.

Además, Miller **[1]** define un umbral, conocido como el número mágico de Miller, que limita a siete el número de elementos visuales que cognitivamente una persona puede manipular con facilidad. La aplicación de esta recomendación sobre en el número de elementos a visualizar en nuestro diseño garantiza que la cognición humana no estará sobrecargada.

De hecho, la gran mayoría de los sitios web dedicados a noticias priorizan la alta densidad de la información en sus páginas iníciales y se ayudan de elementos de control para tratar de organizar la información de alguna manera. Pero en ningún caso apuestan por la segunda parte de nuestra afirmación. Para poder llegar a conciliar las dos condiciones a las que hacemos referencia anteriormente y por tanto, tratar de lograr ambos propósitos, haremos uso de dos conceptos:

- Interfaz minimalista: aplicaremos principios del minimalismo para el diseño de una propuesta de interfaz donde destacaremos la supresión al máximo de elementos de control tales como menús, botones, barras de herramientas, entre otros, de manera que éstos no distraigan la atención del usuario y éste interactúe exclusivamente con la información que se le presenta.

- Uso de **surrogates** que permitan la interacción. Estos **surrogates** deben de ser altamente expresivos sobre la información que representan y al mismo tiempo mantener los niveles cognitivos del usuario muy bajos.

3.3 - Interfaz minimalista

Para alcanzar el propósito de presentar toda la información que un usuario puede asimilar manteniendo sus sentidos cognitivos a niveles bajos, lo primero que se debe realizar es una interfaz donde desaparezcan la mayoría de las estructuras de control, ya que son éstas las que hacen que el usuario pierda la atención y reduzca considerablemente la concentración sobre el verdadero objeto de su búsqueda. En el caso de las noticias, el usuario debería estar centrado exclusivamente en interactuar con éstas con el mínimo esfuerzo y la menor distracción posible.

Lo ideal para conseguir este propósito es el de aportar rasgos propios del minimalismo y aplicarlo a una interfaz de manera que toda la atención del usuario recaiga sobre las noticias, representadas en nuestro caso como **surrogates**.

El minimalismo es una corriente artística que se podría resumir en una simple frase que hizo famosa Ludwing Mies van der Rohe **[4]**, "*menos es más"*. La idea de esta corriente es la de reducir a lo esencial un objeto o método. El término minimalista se refiere a cualquier cosa a la que se le haya extraído los complementos o elementos sobrantes y proporcionar solamente un esbozo de su estructura interna.

El minimalismo aplicado a interfaces no es simplemente el limitarse a usar colores pálidos, tipografías sencillas, sino de ir más allá y analizar cómo interactúa el usuario con la interfaz. De esta manera podremos descubrir en qué situaciones el usuario no encuentra respuesta a sus necesidades; qué utilidades deja de usar por no encontrarlas, o por no ver su auténtico potencial. Todo esto, con el objetivo de simplificarle la tarea y satisfacer sus pretensiones.

De nada sirve dotar a un sitio de un amplio abanico de funcionalidades o información si a la hora de la verdad, el usuario no sabe cómo utilizarlas y se limita a obviarlas o

simplemente recibe tanta información simultáneamente que no es capaz de procesarla toda a la vez.

De hecho, aplicaremos varios conceptos asociados a dicha corriente para diseñar nuestra interfaz como pueden ser la sencillez máxima, eliminando, como hemos estado comentando hasta el momento, el máximo de estructuras de control. También podemos asociar este concepto a la ley de Fitts **[3]** aplicada al diseño de interfaces, ya que ésta expresa que cuanto más grande y más cercano al puntero del ratón sea un objeto, más sencillo será pulsar sobre él. Este concepto lo asociaremos a los surrogates o representaciones de las noticias, ya que lo que pretendemos es que la interacción entre el usuario y nuestra interfaz radique casi exclusivamente con estos elementos, por lo que dichos objetos tendrán un tamaño adecuado para que no resulte costosa la interacción con éstos.

Por último, comentar que la idea de utilizar agrupaciones de surrogates en una interfaz está inspirada en la idea de la *Collage Machine* de Kerne **[2]**. La *Collage Machine* genera, a grandes rasgos, un collage de páginas webs. Descompone páginas web para generar colecciones de elementos multimedia o surrogates (imágenes, texto e hipervínculos), de manera que por su representación y contexto generan indeterminación al usuario. Esta indeterminación, más el contexto que generan estas colecciones, pueden hacer variar la interpretación de un surrogate y pueden, a su vez, hacer variar la elección de navegación del usuario. La finalidad que persigue la Collage Machine es la de ofrecer una nueva manera de navegar por las diferentes webs al usuario, motivando la elección de navegación según el contexto e ir generando nuevos collages según las elecciones de manera que los nuevos elementos que aparezcan estén más acorde a los intereses del usuario.

3.4 - Surrogates

Hasta el momento hemos estado mencionando a dichos elementos sin entrar en detalle. En este apartado describiremos a estos elementos, que serán el pilar fundamental sobre el que girará este proyecto.

3.4.1 - Definición

Los surrogates podrían definirse como representaciones gráficas de las noticias. Estos elementos estarán presentes en nuestra interfaz minimalista y como parte de ella, poseerán rasgos propios del minimalismo. El concepto fundamental inherente en estos elementos será el de la abstracción. Ya que el usuario deberá entender de entrada a estos elementos como un sustituto de una noticia en una web convencional.

3.4.2 - Características

Antes de analizar cómo estarán compuestos los surrogates, deberíamos detallar dos características que se pretenderán dar a éstos. La primera característica de un surrogate es la interoperabilidad. Los surrogates estarán definidos en el estándar XML **[6]**, por lo que serán documentos bien formados, es decir, cumplirán con todas las normas de definiciones del formato y por lo tanto podrán ser parseados con facilidad desde cualquier lenguaje que soporte XML. De esta manera, hacemos que la lectura de los surrogates pueda hacerse efectiva desde varias interfaces independientemente del lenguaje con el que están desarrolladas.

La segunda característica de un surrogate es la usabilidad. La usabilidad es un concepto proveniente del diseño centrado hacia el usuario y los surrogates pretenden tener una alta usabilidad. Esto es así, ya que los surrogates pretenden reducir el coste de aprendizaje para el usuario haciendo que éstos desarrollen una interacción más efectiva con el sistema que con las herramientas existentes. Por último, reforzar la idea sobre la usabilidad de los surrogates afirmando que éstos presentarán una robustez muy alta, ya que dicha característica propia de la usabilidad responde a que la capacidad de observación del usuario aumentará y que la recuperación de la información resultará ser más sencilla también.

3.4.3 Presentación

Los surrogates presentarán un aspecto con una geometría elemental y austera. Siguiendo la línea minimalista que presentará la interfaz y al formar parte de ésta, los surrogates no presentarán ornamentos ya que la finalidad principal de los mismos no es que sean agradables a la vista, sino que se han de entender como contenedores de información. Con el fin de provocar la interacción entre los surrogates y el usuario, lo que se pretende con esta representación gráfica es realizar una metáfora entre dichos elementos y objetos convencionales que los usuarios interactúan directamente y a diario con ellos, como pueden ser tarjetas, postales, documentos en papel, carteles, sobres, entre otros elementos. De esta manera, los usuarios deben identificar a los surrogates como elementos manipulables como si se tratará de algo físico.

Como decíamos anteriormente, los surrogates han de estar compuestos por una información concreta para que sean altamente expresivos manteniendo a su vez, las capacidades cognitivas del usuario en niveles bajos para que el usuario tome decisiones de navegación lo más relajado posible. Así que analizando los elementos que suelen constituir los artículos de noticias, para componer un surrogate extraeremos las componentes que consideramos esenciales para crear una aproximación a éste.
Como podemos ver en el gráfico siguiente, un artículo convencional está formado por diferentes componentes o elementos. Podemos ver el titular de la noticia; su fecha; una pequeña descripción; una imagen; la fuente; el artículo en sí y diferentes servicios que ofrecen los diferentes portales de noticias (noticias relacionadas, noticias más vistas, entre otros).

Figura 3.1 Formato de noticia convencional con elementos esenciales destacados.

3.4.4 Tipos de Surrogates existentes

En la figura anterior podemos ver un tipo de artículo convencional, con toda la serie de elementos que listamos anteriormente que lo forman.

Para formar un surrogate de esta noticia extraeremos los elementos destacados en la figura 3.1 y podremos elegir entre diferentes versiones de representación. En el apartado 6.6.2 se observa como a la hora de generar surrogates a través del programa *SurrogateComposer* se nos da la opción de elegir la apariencia que queremos darles. A continuación se muestra un ejemplo de cómo sería el surrogate del artículo con cada una de las opciones implementadas en el programa:

Figura 3.2. Surrogates compuestos según la opción 1 de *SurrogateComposer*

Si escogemos la opción 1 del menú de *SurrogateComposer,* el programa determinará la apariencia final según las proporciones de la imagen del artículo. Si dicha imagen es más ancha que alta se generará el surrogate con la apariencia de la izquierda según la figura 3.2. En caso que la imagen tenga proporciones contrarias, el surrogate se generará como el ejemplo central de la figura. Por último, si el artículo no dispone de una imagen asociada el surrogate solamente presentará tres componentes: el título, la descripción y la fecha, como podemos observar en el ejemplo de la derecha de la figura.

Figura 3.3. Surrogate compuesto según la opción 2 de *SurrogateComposer*

Si escogemos la opción 2 del menú de *SurrogateComposer,* el programa determinará la apariencia final según la figura 3.3. Como se puede observar, esta opción dará un mayor espacio a la imagen, quedando los componentes textuales a un lado de ésta. Si el artículo no dispone de una imagen asociada, el surrogate se generará de igual modo que en el caso de los surrogates sin imagen de la opción 1 como se muestra en la figura 3.2.

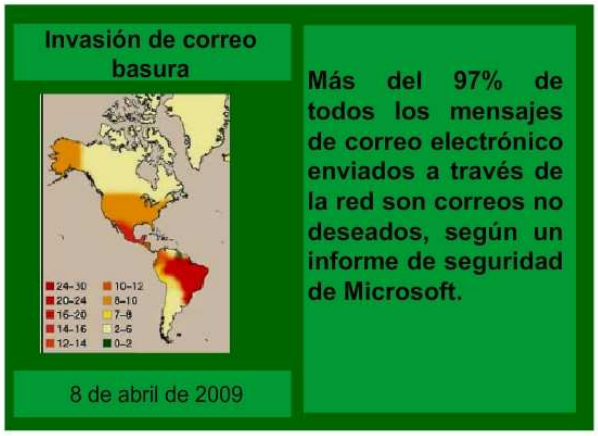

Figura 3.4. Surrogate compuesto según la opción 3 de *SurrogateComposer*

En caso de escoger la opción 3 del menú de *SurrogateComposer,* el programa determinará la apariencia final según la figura 3.4. Esta opción dará una mayor importancia a la descripción textual del artículo. Si el artículo no dispone de una imagen asociada, el surrogate se generará de igual modo que en el caso de los surrogates sin imagen de la opción 1 como se muestra en la figura 3.2.

3.4.5 Beneficios esperados

Los beneficios a los que esperamos llegar con el uso de los surrogates tras este proyecto son los siguientes:

1. Que los surrogates faciliten la representación de una noticia en una interfaz gráfica y reduzcan el esfuerzo del usuario por entender el artículo.

2. Conseguir que los usuarios puedan explorar noticias en una interfaz que disponga de surrogates de manera más eficaz, acorde a sus capacidades cognitivas y prestando total atención a aquello que realmente le interesa: las noticias.

3. Reducir por tanto, la curva de aprendizaje para el uso de la interfaz al no disponer prácticamente de elementos de control para su uso provocando de entrada la interacción directa entre el usuario y las noticias.

4. Conseguir una interoperabilidad alta a la hora de utilizar los surrogates. Desarrollar una serie de clases y métodos que se puedan aplicar a varios

proyectos independientemente del lenguaje con el que se intente implementar siempre y cuando éste soporte el formato XML.

5. Universalizar la identificación de los surrogates en interfaces distintas para que el usuario sepa interactuar con ellos de una manera familiar.

6. Poder usar los surrogates en otros servicios o aplicaciones. En este proyecto hemos enfocado a los surrogates como representaciones de artículos de noticias. Los surrogates también se pueden aplicar a otros campos o aplicaciones. Por ejemplo, los surrogates pueden representar hoteles sobre un mapa donde el título del artículo sea el nombre del hotel; la fecha sea su dirección y la imagen y la descripción sean las propias del hotel.

3.5 RSS

3.5.1 Conceptos previos

Para poder entender con claridad qué es RSS debemos introducir varios conceptos previos.

El primero de ellos es el de **Fuente web**. Una fuente web (o también canal web o *web feed*) es un medio de redifusión de contenido web cuyo objetivo es el de suministrar información actualizada frecuentemente a los suscriptores del RSS. Actualmente existen dos formatos de fuente web principales: el propio RSS y Atom.

La **redifusión web** (o sindicación web) consiste en el reenvío de contenidos desde una fuente original (sitio web de origen) hasta otro sitio web de destino (receptor) que a su vez se convierte en emisor puesto que pone a disposición de sus usuarios los contenidos a los que en un principio sólo podían tener acceso los usuarios del sitio web de origen.

Un **agregador** o **agregador de noticias** es un software que permite suscribirse a fuentes de noticias en formatos RSS, Atom y otros derivados de XML/RDF. El agregador reúne las noticias o historias publicadas en los sitios con redifusión web elegidos, y muestra las novedades o modificaciones que se han producido en esas fuentes web. Es decir, avisa de qué webs han incorporado contenido nuevo desde nuestra última lectura y cuál

es ese contenido. Un agregador es muy similar en sus presentaciones a los anteriores lectores de noticias (client newsreaders/NNTP), pero la tecnología XML y el web semántico los ha hecho más populares. Hoy en día, una enorme cantidad de blogs y sitios web ofrecen sus actualizaciones, que pueden ser fácilmente reunidas y administradas en un solo punto, como es el caso del servicio My Yahoo!, Google Reader, Netvibes y otros agregadores de escritorio.

3.5.2 Definición

Una vez aclarados estos conceptos estamos en disposición de definir qué es RSS.
RSS es una familia de formatos de fuentes web codificados en XML conforme especificaciones publicadas por el World Wide Web Consortium (W3C **[8]**) que se utilizan para suministrar a suscriptores información actualizada frecuentemente.
Este formato también permite distribuir contenido sin necesidad de un navegador ya que existe software que puede leer estos contenidos RSS. Dicho tipo de software es conocido como **agregador** o lectores de fuentes web.

3.5.3 Beneficios que aporta

- Centralización de información en un solo lugar de diferentes fuentes.
- Ahorro en el coste de tiempo para el usuario ya que elimina la tarea de visitar diferentes portales de noticias.
- Mediante las fuentes web se pueden obtener las últimas noticias en cuanto éstas sean actualizadas.

4 - Ánalisis del problema

Como comentábamos al inicio del apartado anterior, actualmente disponemos de mucha información en muchos tipos de interfaces. De hecho, la misma información puede estar presentada de muchas maneras distintas y de hecho, no todas son lo suficientemente útiles y expresivas, por lo que el usuario puede perderse información por una mala representación.

A su vez, han aparecido una serie de *mashups* **[7]**, revolucionarias aplicaciones web que integran en un nuevo sitio los contenidos y funcionalidades de diferentes sitios web con el objetivo de ofrecer información de una manera más completa e innovadora. Por ejemplo, muchos de estos mashups usan GoogleMaps como soporte para añadir información de otros sitios en los mapas. Aún así, existe un problema y se trata de que no se dispone de un apoyo claro a la visualización de la información. Es decir, con estos mashups se ha logrado integrar información de diferentes sitios web, pero no se ha dado un soporte claro para mostrar la información. La solución que proponemos en este proyecto es la de dar apoyo al diseño de representaciones dinámicas de los elementos informativos que sirven para visualizar la información de la web de la manera que elije el diseñador mediante surrogates.

Así que una vez aproximada la solución al problema que se plantea, orientaremos el desarrollo de los surrogates hacia las noticias, ya que son un recurso que se está sindicando fiablemente desde hace tiempo y de esta manera, ofreceremos una manera de dar apoyo para representar la información en diferentes mashups.

4.1 Sitios más visitados

Actualmente existen en Internet multitud de portales que ofrecen información de todo tipo. Particularmente hay multitud de sitios web cuyo objetivo es el de informar al usuario ofreciendo noticias diariamente. Mayoritariamente se trata de ediciones digitales de los distintos diarios o portales dedicados a las noticias, como por ejemplo, Yahoo News.

A continuación tomaremos y analizaremos las estadísticas que nos ofrecerán diferentes benchmarks de Internet para escoger los portales web más visitados; conocer su repercusión y en definitiva, estudiar cómo están representando la información para tratar de mejorar dicha tarea. Un benchmark es una técnica utilizada para medir el rendimiento de un sistema.

Visitando OJD Interactiva **[9]**, un benckmark de ámbito nacional, podemos ver como el medio más visitado con diferencia es El Mundo.

	TITULO	URL	CLASIFICACION	U.UNICOS	VAR.%	VISITAS	D.MEDIA	PAGINAS
1	EL MUNDO	WEB	Noticias e Información	19.416.694	75,50	60.189.602	05:25	320.218.674
2	20MINUTOS.ES	WEB	Noticias e Información	6.803.325	-2,13	17.634.626	06:37	90.420.781
3	PAGINAS AMARILLAS	WEB	Noticias e Información	6.390.567	-0,14	11.273.234	04:30	81.100.152
4	ABC	WEB	Noticias e Información	4.333.739	-7,85	9.986.093	04:33	40.793.225
5	CONSUMER EROSKI	WEB	Noticias e Información	2.664.400	0,53	3.463.390	03:24	10.324.694
6	LAVANGUARDIA.ES	WEB	Noticias e Información	2.476.965	4,95	5.785.176	05:49	24.510.289
7	EUROPAPRESS	WEB	Noticias e Información	1.916.974	-0,24	2.821.119	05:09	11.939.277
8	LIBERTAD DIGITAL	WEB	Noticias e Información	1.855.588	3,49	7.430.005	04:46	25.949.875
9	EL CONFIDENCIAL.COM	WEB	Noticias e Información	1.824.983	2,25	7.320.525	06:08	30.253.228
10	EL PERIÓDICO DE CATALUNYA	WEB	Noticias e Información	1.814.194	-2,62	5.004.225	05:55	21.709.337

Figura 4.1. Top10 en el mes de Febrero de los medios auditados por OJD Interactiva en la categoría Noticias e Información

Como muestra el gráfico anterior y mirando las estadísticas sobre visitas y usuarios únicos podemos llegar a la conclusión que la repercusión de estos medios es enorme, ya que prácticamente 20 millones de usuarios visitaron *www.elmundo.es*.

Visitando un benchmark de Internet a nivel mundial como puede ser Alexa **[10]**, Yahoo News es el portal de noticias más visitado, seguido por BBC y CNN.

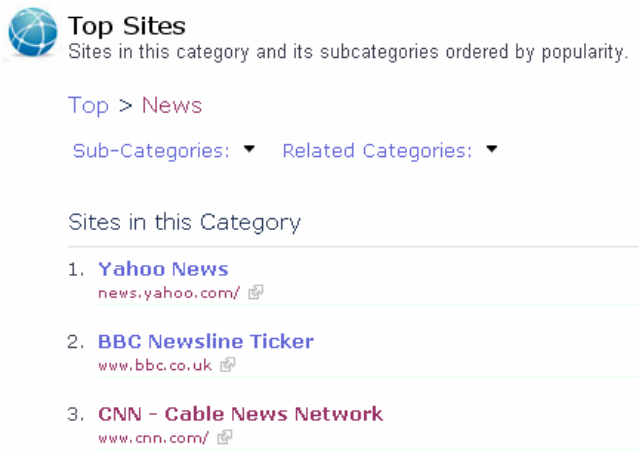

Figura 4.2 - Top 3 a nivel mundial de sitios de noticias por Alexa

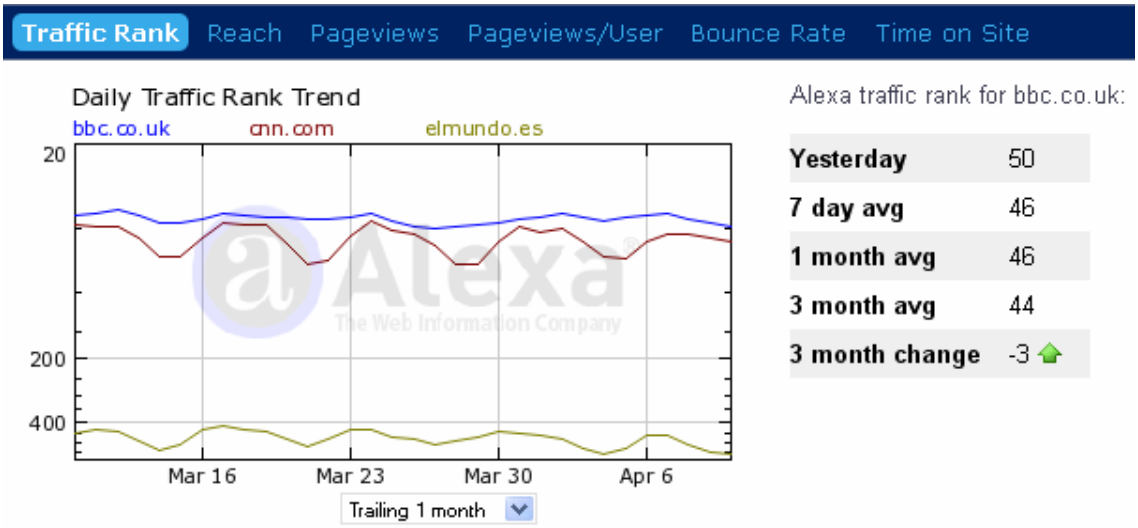

Figura 4.3 - Gráfico sobre el tráfico de los sitios de noticias más importantes por Alexa

Como podemos ver en la figura 4.3, el portal de *El Mundo*, que fue el portal de ámbito nacional más visitado, está a mucha distancia de portales como el de la BBC o de CNN y a su vez, ocurre lo mismo entre éstos y Yahoo.

Por lo tanto, y antes de pasar al siguiente punto, es importante decir que nuestro proyecto estará destinado a un conjunto de usuarios muy grande, de millones de usuarios, por lo que no es posible definir un tipo específico de usuario que usará nuestra interfaz.

4.2 Cómo se trata la visualización de la información

Después de haber analizado la repercusión de los sitios webs dedicados a noticias más visitados, pasaremos a analizar el tratamiento que les da Yahoo, por ser el sitio más visitado y El Mundo, por ser el primer medio nacional, a sus noticias.

Figura 4.4 - Portadas de Yahoo News (izquierda) y El Mundo (derecha)

Las portadas de estos portales, y la mayoría de los restantes, presentan prácticamente los mismos elementos, que son los siguientes:

1. Barras de búsqueda
2. Menús con las categorías
3. Noticias principales
4. Noticias
5. Espacios reservados a la publicidad
6. Servicios que ofrece el portal

Cuando accedemos a una noticia en concreto en Yahoo News, vemos como se mantienen todos los elementos analizados anteriormente. Lo mismo pasa en el resto de portales, por lo que podemos decir que los portales dedicados a las noticias, a parte del

contenido de la noticia en sí, ofrecen toda una serie de elementos que por su posicionamiento, hacen que los niveles cognitivos del usuario se desborden, restando atención a la notica:

Figura 4.5 - Artículo de Yahoo News. Se mantienen los elementos que no son propios del mismo.

Después de analizar las noticias de los diferentes portales se puede decir que éstas tienen un patrón bastante estandarizado, ya que salvo pequeñas variaciones, se componen por los mismos elementos.

Cabe destacar, por último, que cada uno de los seis elementos que hemos enumerado se podría entender como un surrogate, donde cada uno de ellos se nutre de componentes para representar lo que desean. Por ejemplo, las barras de búsquedas se pueden entender como surrogates formados por tres elementos: un texto; la propia barra para introducir la información y el botón para ejecutar la búsqueda. Los menús se pueden entender como surrogates formados de botones. Los servicios que ofrece el portal pueden estar compuestos por un título de servicio; imágenes; enlaces u otros elementos. Y finalmente, las noticias, como hemos estado analizando hasta el momento, están formadas por un titular, un elemento multimedia, una descripción, la fecha, la fuente y el texto de la propia noticia.

5 - Descripción de la solución

Tras haber analizado en el punto anterior el tratamiento que se le está dando a la visualización de las noticias, estamos en disposición de tratar nuestra propuesta para cumplir el objetivo de mejorar la exploración por parte de los usuarios.

5.1 Descripción del sistema

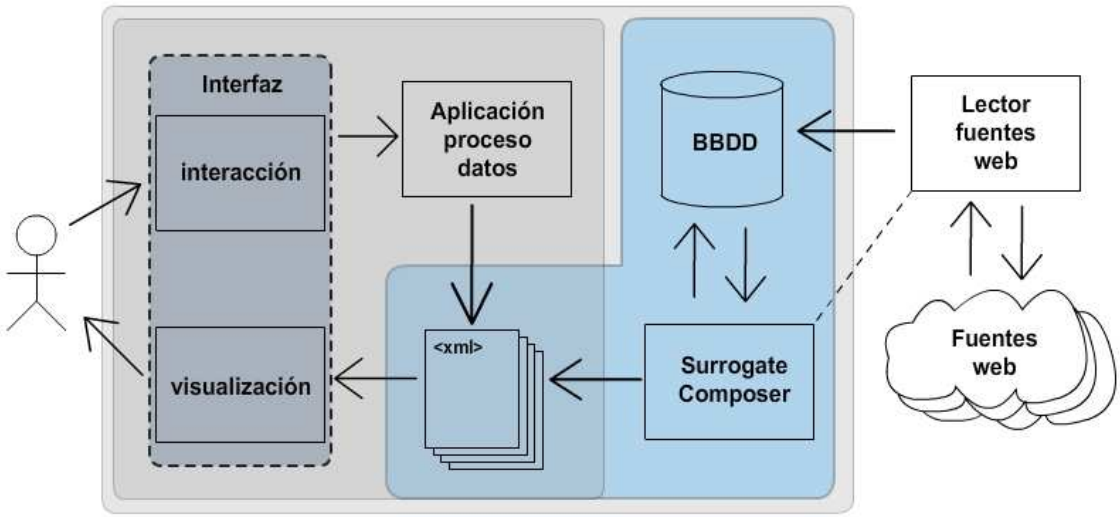

Figura 5.1 - Diagrama del sistema completo

El sistema está formado por tres partes. La primera de ellas es la composición de un agregador o lector de fuentes web que recogerá dichas fuentes y las podrá almacenar o bien en una base de datos, o bien en ficheros de formato XML. Para que el proceso sea más rápido y dinámico, implementaremos la segunda opción. Una segunda parte la formará una aplicación java llamada *SurrogateComposer* que será la encargada de componer nuevos archivos de formato XML e imágenes con formato JPG de los artículos a partir de los datos almacenados anteriormente. Finalmente, la tercera parte del sistema estará formada por una interfaz desarrollada en Flex donde se implementarán los métodos o procedimientos para la visualización de los surrogates.

Nuestro proyecto se centrará en las dos últimas partes. Las enfocadas a la composición de surrogates y su visualización. Para que la tercera parte, la dedicada a la visualización, funcione correctamente, necesitará que se haya ejecutado con anterioridad la segunda parte del sistema, la de composición de surrogates, ya que de lo contrario, no existirían los surrogates y no se podrían mostrar en la interfaz. Cuando esta segunda parte se haya ejecutado, la interacción entre el usuario y la interfaz podrá realizarse sin problemas.

5.2 Composición

5.2.1 Estructura surrogate

Un surrogate tendrá el formato propio de un archivo XML ya que su manipulación ha de ser independiente de la plataforma en la que se incluya. Por ello ha de tener una estructura bien etiquetada para su correcta manipulación.

```
<element id="1" url="" source="">
      <text>
      </text>
      <media type="image" url="">
      </media>
      <surrogate>
            <GShape id="0" type="main" n="4">

            </GShape>
            <GShape id="1" type="txt" n="0">

            </GShape>
            <GShape id="2" type="txt" n="0">

            </GShape>
            <GShape id="3" type="img" n="0">

            </GShape>
            <GShape id="4" type="txt" n="0">

            </GShape>
      </surrogate>
</element>
```

Figura 5.2 - Estructura básica de un surrogate

Como podemos observar en la figura 5.2, un surrogate estará etiquetado con *element*, que contendrá toda la información de la noticia. Cada elemento tendrá una id única y se especificará con los atributos *url* y *source* la fuente de donde se ha sido recogida la información. A continuación, cada element o noticia tendrá tres partes:

1. **Texto** - Esta primera parte estará definida entre las etiquetas *text*. Dentro de esta parte encontraremos el titulo y el texto de la noticia. El título se especificará en la etiqueta *text_title* y el texto de la noticia en *text_content*.
2. **Media** - Esta segunda parte vendrá definida entre las etiquetas *media*. Dicha etiqueta tendrá dos atributos: *type* y *url*. En *type* se especificará el tipo de recurso multimedia que tendrá asociado una noticia. En la figura 5.2 se puede ver como se especifica que el recurso será una imagen. Por otra parte, el atributo *url* especificará el directorio o ubicación donde estará almacenada la imagen para poder acceder a ella.

3. **Surrogate** - La tercera y última parte de una noticia vendrá definida entre las etiquetas *surrogate*. Esta parte es más compleja ya que será aquí donde se descompondrán los elementos de una noticia para su posterior manipulación. Para el caso de las noticias hemos especificado en apartados anteriores que nos interesaran 4 componentes. La correspondiente al titular de la noticia; la del texto de la noticia; la de la fecha de publicación y el recurso multimedia asociado. Estos cuatro elementos estarán bien definidos en cuatro *GShape*. Pero también necesitaremos especificar un quinto elemento o *GShape.* De hecho, en el xml será el primero que definiremos al ser éste el principal y de ahí que su tipo sea *main*. En su atributo *n* especificaremos el número de *GShapes* o elementos que tendrá el surrogate. Como se puede observar en la figura 5.2, se especifica que un surrogate estará formado por cuatro *GShapes*.

   Pueden haber tres tipos de *GShape* definidos por el atributo *type* de cada uno: el *main*, el *txt* y el *img*. Los GShapes cuyo atributo *type* sea *txt* serán los correspondientes a los que su información sea texto (titular y noticia) mientras que los que *type* sea *img* será el que haga referencia al recurso multimedia, en este caso la imagen.

   En cada GShape se especificarán una serie de etiquetas con atributos. En los GShape de tipo *txt* figurarán las siguientes etiquetas con sus correspondientes atributos:

```
<GShape id="1" type="txt" n="0">
      <position x="3" y="3" width="219" height="27"/>

      <bkcolor r="FF" g="FF" b="00"/>

      <area x="5" y="5" width="209" height="17"/>

      <font size="14" font="Dialog" bold="0" italic="0" underline="0"
      justify="center" color="0" transparency="0"/>

      <text lmargin="0" rmargin="0" tmargin="0" bmargin="0">TITULO</text>
</GShape>
```

Figura 5.3 - Estructura y atributos de un GShape de tipo texto

- position: contendrá los atributos sobre la posición y el tamaño del surrogate.
- bkcolor: definirá las componentes RGB del color de fondo del GShape.
- area: contendrá los atributos sobre la posición relativa del GShape en el surrogate y su tamaño.

- font: definirá el formato del texto (tamaño, tipo de fuente, negrita, cursiva, subrayado, justificación, color y transparencia).
- text: en los atributos de dicha etiqueta se definirán los márgenes y a su vez, es aquí donde se contendrá el contenido del GShape.

En el GShape de tipo img figurarán las propiedades siguientes:

```
<GShape id="3" type="img" n="0">
      <position x="3" y="33" width="106" height="112"/>

      <bkcolor r="ff" g="00" b="00"/>

      <area x="5" y="5" width="500" height="700"/>

      <img lmargin="0" rmargin="0" tmargin="0" bmargin="0">images//imagen.jpg</img>

</GShape>
```

Figura 5.4 - Estructura y atributos de un GShape de tipo imagen

Las etiquetas *position*, *bkcolor* y *area* de este tipo de GShape tendrán los mismos atributos y funcionalidad que la de los GShape de tipo texto. Este tipo de GShape, a diferencia de los dedicados a manipular texto, no dispondrán de las etiquetas *font* y *text* mientras que tendrán la etiqueta img. En esta etiqueta se definirán los atributos con los márgenes que deberá guardar la imagen y como contenido figurará el directorio donde se almacenará la imagen asociada a la noticia.

Después de definir de manera detallada la estructura de un archivo xml de cada noticia cabe destacar que los surrogates de tipo 3, los que carecen de imagen, no dispondrán del GShape dedicado a ésta, por lo que en el atributo *n* del GShape de tipo main figurará un tres y no un cuatro, ya que el surrogate solamente estará formado por tres componentes.

Para acabar con este punto es importante reiterar que la estructura de estos xml tiene las tres partes (*text*, *media* y *surrogate*) para ofrecer interoperabilidad a la hora de manipularlos. Es decir, *text* y *media* se definen en el xml para poder acceder y usar la información de manera más general, mientras que toda la estructura de *surrogate* está diseñada para poder acceder y manipular todo tipo de variable más detalladamente. Desde recoger sólo los GShapes que nos interesan hasta, por ejemplo, modificar el valor de la propiedad cursiva de un GShape en concreto.

5.2.2 Asociar propiedades

A los surrogates se les puede asociar una serie de propiedades. A continuación mostramos tres de ellas:

Transparencia - Fecha: se pueden aplicar diferentes grados de transparencia a un ***surrogate*** con el propósito de que el usuario identifique que cuanto más transparente sea éste, la noticia más antigua será. Por el contrario, cuanto más opaco sea un ***surrogate***, significara que la noticia a la que representa, será más actual.

Color - Tema: se puede aplicar un color determinado para cada tema o categoría de noticias con el objetivo de que el usuario, una vez ha asociado dicho color al tema, sea capaz de identificar o filtrar por categoría las noticias más rápidamente.

Tamaño - Importancia: se pueden aplicar diferentes tamaños a los surrogates dependiendo de la importancia que tengan. De esta manera, una noticia importante puede ser más grande que una que no sea tan relevante.

El propósito de asociar diferentes características a los surrogates es dar una respuesta afirmativa a las siguientes preguntas:

- ¿Ayudan los surrogates a la identificación?
- ¿Ayudan los surrogates a la búsqueda?
- ¿Ayudan los surrogates al recuerdo?

5.3 - Escenarios y casos de uso

En este apartado describiremos tres escenarios y sus casos de uso sobre los que nos basaremos para diseñar nuestros surrogates. Hay que aclarar que utilizaremos estos tres escenarios como ejemplos sobre los que se pueden aplicar los surrogates, pero éstos se podrán aplicar a otros muchos proyectos. De hecho, como ya hemos comentado a lo largo de este proyecto, los surrogates que acabaremos implementando estarán enfocados a representar noticias que se puedan consultar en Yahoo News o cualquier portal de noticias, pero hemos preferido detallar tres escenarios donde poder aplicar el concepto de surrogate. La información sobre cada escenario se ha recabado tras realizar unas entrevistas semiestructuradas con los responsables de cada proyecto en los que se utilizarán los surrogates. Así pues, el objetivo de estas entrevistas es el de recoger una serie de requerimientos para diseñar y desarrollar nuestros surrogates.

5.3.1 Surrogates en la visualización interactiva sobre Google Maps

5.3.1.1 Escenario 1

Este escenario constará de un mashup que utilizará un mapa de España de GoogleMaps y por otra parte, diferentes noticias sindicadas por una fuente web que han ocurrido en las ciudades del mapa. Cuando el usuario pulse sobre ciertas ciudades del mapa de España, recibirá las noticias de dicha localidad. Dicho mashup dispondrá de varias opciones para poder escoger el tipo de noticias que mostrará cuando el usuario pulse sobre los puntos del mapa. Por ejemplo, el mashup ofrecerá la opción de escoger sólo noticias económicas, de política, deportivas, entre otras categorías.

5.3.1.2 Casos de uso del escenario 1

Caso de uso 1: El usuario quiere visualizar noticias que han sucedido en diferentes ciudades del territorio español.

El contexto en el cual se desarrolla la acción sería la casa de dicho usuario, en la habitación en la que se encuentre su ordenador personal. La motivación de esta persona sería la necesidad de conocer las diferentes noticias de ámbito nacional con el objetivo de estar bien informado y poder intercambiar impresiones con sus familiares o amigos. Para cumplir con su propósito, el usuario iniciará la interacción con el mashup y pulsará sobre los diferentes puntos del mapa para ir disponiendo de la información de cada ciudad o región.

Caso de uso 2: El usuario quiere visualizar noticias que han sucedido en el estado español de ámbito deportivo. El contexto en el cual se desarrolla la acción sería el mismo que en el primer caso de uso. La motivación o rol que desempeña esta persona sería la necesidad de conocer las diferentes noticias de ámbito nacional pero solamente de ámbito deportivo con el objetivo de estar bien informado sobre dicha temática y poder intercambiar impresiones con sus familiares o amigos. Para lograr su objetivo, el usuario pulsará sobre puntos del mapa para disponer de la información deportiva de dicha región.

5.3.1.3 Escenario 2

Este escenario constará de un mashup que utilizará un mapa de España de GoogleMaps e información sobre los resultados deportivos de la jornada de la liga de fútbol de primera división española. La aplicación mostrará un mapa de España sobre el que se colocarán 10 surrogates que representarán cada partido de la jornada de liga. Cada surrogate estará colocado en la coordenada específica a la población donde se está disputando el partido y el mashup dispondrá de una leyenda donde se especificará qué significado tiene el color de cada surrogate.

5.3.1.4 Casos de uso del escenario 2

Caso de uso 1: Este caso de uso será el correspondiente a un usuario que es aficionado a su equipo de fútbol. El contexto en el cual se desarrolla este caso de uso sería en la casa de dicho usuario, en la habitación en la que se encuentre su ordenador personal. La motivación o rol que desempeña esta persona en este caso de uso sería la necesidad de conocer toda la información de su equipo de fútbol preferido para poder socializar con sus amigos y familiares. Por tanto, su meta o principal interés se podría definir como la necesidad de saciar este interés de información sobre su equipo. Para cumplir dicho objetivo, iniciará la interacción con el mashup y centrará la atención sobre el surrogate donde esté incluido su equipo y pulsando sobre éste para ampliar la información del partido.

Caso de uso 2: Este caso de uso será el correspondiente a un usuario que es aficionado a las estadísticas que se producen en un estadio determinado. El usuario querrá saber varios datos sobre los resultados de las jornadas y temporadas pasadas. Por ello, activará la opción del mashup sobre estadísticas y pulsará sobre el estadio del que le interese conocer para ampliar dicha información.

5.3.1.5 Requerimientos de los surrogates

	Escenario 1	Escenario 2
¿A qué tipo de elemento va sustituir un surrogate?	noticias	resultados partidos fútbol
¿Cuántos surrogates van a colocarse en la interfaz?	1	10
¿Irán apareciendo más surrogates de los que se coloquen inicialmente?	si	no
¿Sustituyendo a los ya presentes o añadiéndose nuevos?	sustituyéndolos	-
¿Hasta llegar a un número determinado?	uno por otro, manteniendo 1 en pantalla	-
¿Dependerá de la interacción con el usuario?	si, si clica en otro punto del mapa	-
¿Desaparecerán los surrogates?	si, al ser sustituidos	no
¿Automáticamente o mediante interacción?	interacción:por click	-
¿Por tiempo o por número?	-	-
¿Los surrogates se presentaran de forma estática o se desplazarán por la pantalla?	estática	estática
En caso que se desplacen, ¿se moverán ellos solos automáticamente o se desplazarán a raíz de una interacción por parte del usuario?	-	-
¿Se aplicaría dinamismo a los surrogates o serán elementos más bien estáticos?	estáticos	dinamismo
¿Qué tipo de dinamismo? (elementos que lo componen, movimiento, forma, tamaño, color, transparencia,…)	color	color y transparencia(fecha)
¿Considera oportuno aplicar transparencia de los surrogates?	si	si
¿La transparencia vendría dada de entrada o cambiaría en tiempo de ejecución o mediante interacción del usuario?	de entrada	tiempo ejecución
¿Cuántos elementos tendrá un surrogate?	4 (una imagen, un titulo,resumen y link)	4 (dos imágenes, resultado y minuto)
¿El color de los surrogates podría significar algo en especial? ¿Qué expresaría con el color de un surrogate?	tipo noticia	por determinar
¿Los surrogates tendrán tamaño fijo o éste cambiará?	fijo	por determinar
¿Automáticamente o por interacción?	-	interacción
¿Según el número de surrogates en pantalla o con independencia a éstos?	-	independencia

Figura 5.5 - Resultados entrevista sobre escenario 1 y escenario 2

Los surrogates que utilizarán los proyectos sobre GoogleMaps estarán fijos en el mapa, por lo que no requerirán métodos de desplazamiento. Solamente de posicionamiento en unas coordenadas de dos dimensiones (x, y). En ambos casos requerirán también de una serie de métodos para el tratamiento del color y la transparencia, ya que serán propiedades que tendrán un significado concreto para cada caso.

Los surrogates del segundo escenario requieren estar formados por cuatro elementos: un par de *gshapes* de tipo imagen (una para cada escudo de cada equipo), y otro par de *gshapes* de tipo texto para contemplar el resultado y el minuto de juego. La información de los surrogates cambiará en tiempo real, por lo que también deberán disponer de métodos para el cambio del contenido. Por último, los surrogates de este proyecto podrán ser ocultados según las preferencias del usuario.

5.3.1.6 Aportación de los surrogates a los escenarios

Los surrogates que se implementen en estos escenarios pueden aportar varios beneficios. A nivel técnico aportan un soporte gráfico muy sólido para representar la información de forma concisa y clara pero a la vez, siendo ésta muy representativa. Con los pocos elementos que se representan en los surrogates, los usuarios tendrán suficiente información para decidir descartarlos o interesarse más por ellos. De esta manera, los surrogates aportarán una motivación para interactuar con ellos, es decir, con la información. De esta manera, podríamos decir que el surrogate ejercerá como puerta de enlace a la información completa. Así pues, los surrogates ejercen de filtro para reducir la cantidad de información en pantalla y en consecuencia, manteniendo las capacidades cognitivas del usuario estables.

5.3.2 Surrogates en la visualización de noticias en una tabla dinámica

5.3.2.1 Escenario 3

Este escenario constará de una tabla de imágenes dinámica que mostrará los resultados de una búsqueda realizada en un buscador y será de carácter informativa, donde se ubicarán diferentes noticias en base a las palabras clave de dicha búsqueda. El usuario podrá ir consultando las noticias en la tabla pulsando sobre ellas y dispondrá de una paginación en caso de haber más noticias que las que se puedan mostrar. El usuario podrá realizar nuevas búsquedas en todo momento.

5.3.2.2 Casos de uso del escenario 3

Caso de uso 1: En este caso de uso, el usuario tiene interés por buscar información sobre las noticia de Iraq. El contexto en el que se desarrolla el caso de uso sería en casa del usuario, ya que tiene tiempo libre para informarse sobre las noticias de Iraq para seguir la evolución del conflicto de dicho país. Su objetivo será el de saciar el interés sobre el tema y para conseguirlo, introducirá palabras clave en la barra de búsqueda para que se le muestre una tabla de surrogates con las noticias de Iraq.

Caso de uso 2: Un usuario quiere consultar las noticias que han sucedido en la jornada deportiva en España. Dicho usuario dispone de tiempo libre para recabar toda la información que le interesa al estar éste en el ordenador de su casa cómodamente. Para

cumplir su objetivo, el usuario introducirá las palabras clave en la barra de búsqueda. En este caso, podría hacer combinaciones de palabras como por ejemplo “futbol+españa” ó “nadal+roland+garros”.

5.3.2.3 Requerimientos de los surrogates

	Escenario 3
¿A qué tipo de elemento va sustituir un surrogate?	noticias
¿Cuántos surrogates van a colocarse en la interfaz?	6 a 24
¿Irán apareciendo más surrogates de los que se coloquen inicialmente?	no
¿Sustituyendo a los ya presentes o añadiéndose nuevos?	-
¿Hasta llegar a un número determinado?	-
¿Dependerá de la interacción con el usuario?	-
¿Desaparecerán los surrogates?	no
¿Automáticamente o mediante interacción?	-
¿Por tiempo o por número?	-
¿Los surrogates se presentaran de forma estática o se desplazarán por la pantalla?	estática
En caso que se desplacen, ¿se moverán ellos solos automáticamente o se desplazarán a raiz de una interacción por parte del usuario?	-
¿Se aplicaria dinamismo a los surrogates o serán elementos más bien estáticos?	dinamismo
¿Qué tipo de dinamismo? (elementos que lo componen, movimiento, forma, tamaño, color, transparencia,…)	color y transparencia
¿Considera oportuno aplicar transparencia de los surrogates?	si
¿La transparencia vendria dada de entrada o cambiaria en tiempo de ejecución o mediante interacción del usuario?	de entrada
¿Cuántos elementos tendrá un surrogate?	4 (imagen, titulo, texto y fecha)
¿El color de los surrogates podria significar algo en especial? ¿Qué expresaria con el color de un surrogate?	tipo noticia
¿Los surrogates tendrán tamaño fijo o éste cambiará?	cambiara según click
¿Automáticamente o por interacción?	interaccion
¿Según el número de surrogates en pantalla o con independencia a éstos?	independencia

Figura 5.6 - Resultados entrevista sobre escenario 3

El número de surrogates que se muestren en este escenario dependerán de la búsqueda. Podrán aparecer desde 6 a 24 surrogates a la vez, pero siempre desde inicio, no irán apareciendo nuevos a no ser que cambiemos de página de resultados. Los surrogates en este escenario estarán fijos en el mapa, por lo que no requerirán métodos de desplazamiento. Solamente de posicionamiento en una posición de la matriz de dos dimensiones. En ambos casos requerirán también de una serie de métodos para el tratamiento del color y la transparencia, ya que serán propiedades que tendrán un significado concreto para cada caso.

5.3.2.4 Aportación de los surrogates al escenario 3

Los surrogates que se implementen en este escenario aportaran una nueva forma de consultar noticias. A los usuarios que utilicen esta interfaz se les presentará una nueva forma de buscar noticias por palabras clave con resultados más gráficos y novedosos.
De esta manera, los usuarios se pueden sentir más atraídos por las noticias presentadas de esta manera ya que al estar representadas mediante surrogates, generan más interés que no presentándolas en forma de listado. Esta nueva manera de buscar noticias garantiza informarse del tema que se pretende sin tener que consultar diferentes portales de noticias y hacer el esfuerzo de leer multitud de titulares en una cantidad de texto enorme y difícil de soportar cognitivamente hablando.

5.3.3 - Descripción detallada de los casos de uso comunes

En este apartado describiremos de forma detallada dos de los casos de uso comunes que hemos encontrado en los anteriores escenarios. De este modo, podremos observar como en todo caso, los surrogates están destinados a que el usuario pulse sobre ellos y muestren más información de la que representan en un principio.

1- El usuario quiere encontrar una información en concreto.

Caso de uso: Buscar una información en concreto
Contexto: El usuario dispone de tiempo para interactuar con la aplicación con el objetivo de encontrar una información en concreto
Actores primarios: Usuario que usa la interfaz
Actores secundarios: No tiene
Precondiciones: El usuario es conocedor de la información a grandes rasgos y quiere informarse en detalle.
Postcondiciones de éxito: El usuario encuentra la información y la consulta sin ningún problema.
Postcondiciones de fracaso: El usuario no encuentra la información.
Escenario exitoso principal:
1 - El sistema presenta el primer nivel de noticias.
2 - El usuario encuentra la noticia y le hace clic

3 - El sistema ofrece la noticia completa.

4 - El usuario lee la noticia y hace clic para cerrarla

5 - El sistema cierra la noticia y muestra el nivel de noticias donde ésta se encontraba.

Extensiones:

2.a El usuario no encuentra la noticia que busca

2.a.1 El usuario *pulsa uno de los botones para cambiar de nivel* de noticias.

2.a.2 Volvemos al paso 1 pero ofreciendo el nuevo nivel de noticias.

Includes:

Pulsa uno de los botones para cambiar de nivel

2- Un usuario quiere informarse de un tema en concreto.

Caso de uso: Informarse de un tema

Contexto: El usuario quiere informarse de un tema concreto y mostrara interés por todas las noticias que hagan referencia a dicho tema.

Actores primarios: Usuario que usa la interfaz

Actores secundarios: No tiene

Precondiciones: El usuario quiere informarse de un tema preconcebido.

Postcondiciones de éxito: El usuario ha encontrado noticias sobre ese tema y ha saciado su interés.

Postcondiciones de fracaso: El usuario no ha encontrado ninguna o las suficientes noticias sobre ese tema.

Escenario exitoso principal:

1 - El sistema presenta el primer nivel de noticias.

2 - El usuario encuentra una noticia del tema y le hace clic

3 - El sistema ofrece la noticia completa.

4 - El usuario lee la noticia y hace clic para cerrarla

5 - El sistema cierra la noticia y muestra el nivel de noticias donde ésta se encontraba.

Extensiones:

2.a El usuario no encuentra ninguna noticia del tema que busca en ese nivel

2.a.1 El usuario pulsa uno de los botones para cambiar de nivel de noticias.

2.a.2 Volvemos al paso 1 pero ofreciendo el nuevo nivel de noticias.

5.a El usuario encuentra otra noticia del tema en el mismo nivel

5.a.1 El usuario hace clic sobre la nueva noticia.

5.a.2 Volvemos al paso 3.

5.b El usuario no encuentra ninguna noticia más del tema en ese nivel

5.b.1 El usuario *pulsa uno de los botones para cambiar de nivel* de noticias.

5.b.2 Volvemos al paso 1 pero ofreciendo el nuevo nivel de noticias.

Includes:

Pulsa uno de los botones para cambiar de nivel

5.3.4 Conclusiones sobre los casos de uso

Después de haber analizado tres escenarios y sus casos de uso, hemos podido comprobar que los surrogates se pueden aplicar a estos tipos de proyecto y a otros de funcionalidad similar. De esta manera, los surrogates ofrecen un apoyo importante para mostrar la información sobre mapas y tablas dinámicas de información, entre otros.

De hecho, **facilitan en gran medida la manera de presentar** de forma sencilla y clara una información que de otra forma, sería mucho más costosa.

Además, los surrogates ofrecen una nueva manera de mostrar la información ya que se pueden manipular con fines determinados. La posición, el color, el tamaño del surrogate puede significar por sí mismo varias características que pueden guiar al usuario de manera más eficiente.

Asimismo, cabe decir que los surrogates ejercen como si fueran una puerta de enlace a una información más detallada, es decir, mostrando de inicio un detalle de ésta. De esta manera, los surrogates tendrán un método implícito en su concepto, que será el de hacer esta **transición de información resumida a información detallada** implementando métodos de redimensionamiento para poder ofrecer toda la información. El beneficio de todo esto es que el usuario se puede ver beneficiado de este apoyo a la visualización ya que se puede estandarizar ésta, y además de aportar información

5.4 Visualización. Prototipos de interfaz

En este apartado mostraremos los prototipos de los tres escenarios que describimos en el punto anterior. De esta manera podremos observar de manera gráfica el resultado de utilizar los surrogates.

5.4.1 Prototipo de interfaz para el escenario 1

Figura 5.7 - Uso de surrogates sobre GoogleMaps en escenario 1

En la figura 5.7 podemos observar cómo se representaría la información al pulsar sobre la zona de Barcelona. Aparece un surrogate con la noticia principal del punto escogido. Una vez apareciera el surrogate sobre el mapa, el usuario podría volver a pulsar sobre el surrogate para ampliar la información y leer la noticia detalladamente. Si con el surrogate fuera suficiente para saciar el interés del usuario sobre la noticia, podría pulsar sobre otro punto del mapa para que desapareciera el surrogate actual y apareciera otro sobre el punto seleccionado. También podría repetir sobre el mismo punto de manera que aparecería un nuevo surrogate sustituyendo al anterior.

5.4.2 Prototipo de interfaz para el escenario 2

Figura 5.8 - Uso de surrogates sobre GoogleMaps en escenario 2

En la figura 5.8 podemos observar cómo estaría compuesto un mashup formado por un mapa de España de GoogleMaps por una parte y la información sobre los partidos de la jornada de liga de primera división por otra. En este caso, el contenido de los surrogates se estaría captando en tiempo real y el color de éstos aportaría, con el apoyo de una leyenda, información extra sobre el estado del partido.

También se podrían aplicar otras funcionalidades para dar soporte a la visualización, como por ejemplo, la ampliación de un surrogate al ser seleccionado para aportar más información sobre un partido en concreto.

5.4.3 Prototipo de interfaz para el escenario 3

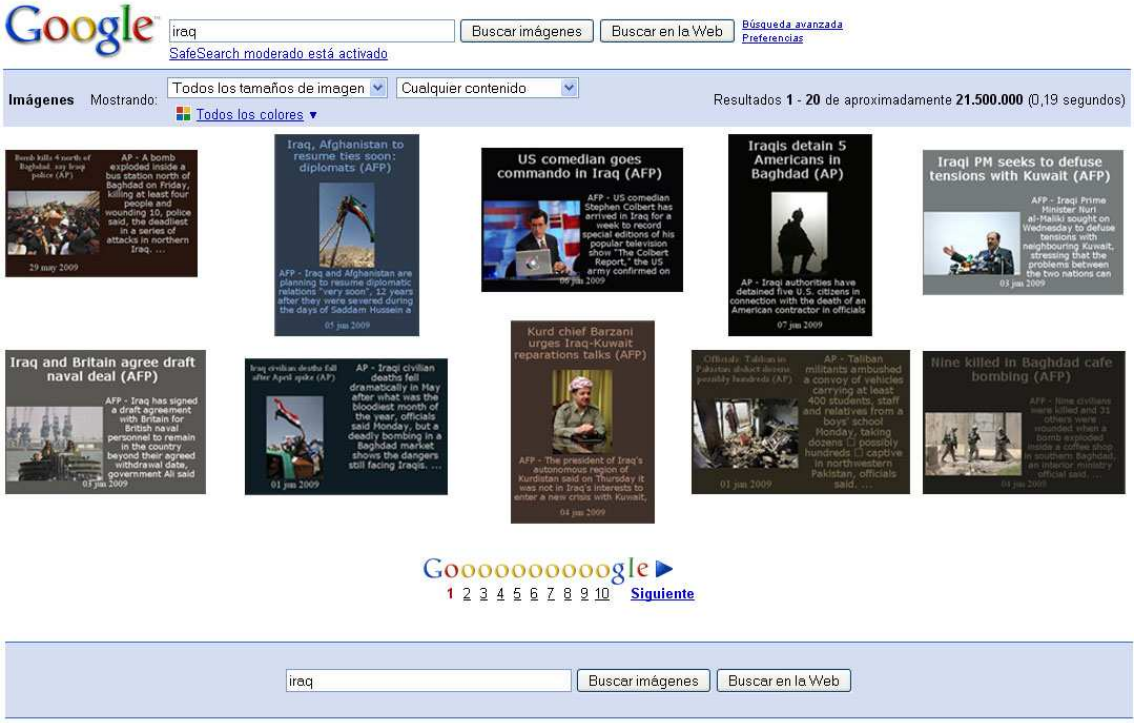

Figura 5.9- Uso de surrogates como resultados de una búsqueda en una tabla de imágenes

En la figura 5.9 podemos observar como los surrogates pueden dar soporte en una tabla de imágenes como resultados de una búsqueda en un buscador.

En este caso, el usuario introduciría las palabras clave para realizar dicha búsqueda y los resultados encontrados se mostrarían como surrogates.

Al pulsar sobre un surrogate, al igual que en los otros casos, podría ampliarse el tamaño de éste para mostrar detalladamente la información escogida por el usuario.

5.4.4 Prototipo de interfaz para el proyecto WakiTable

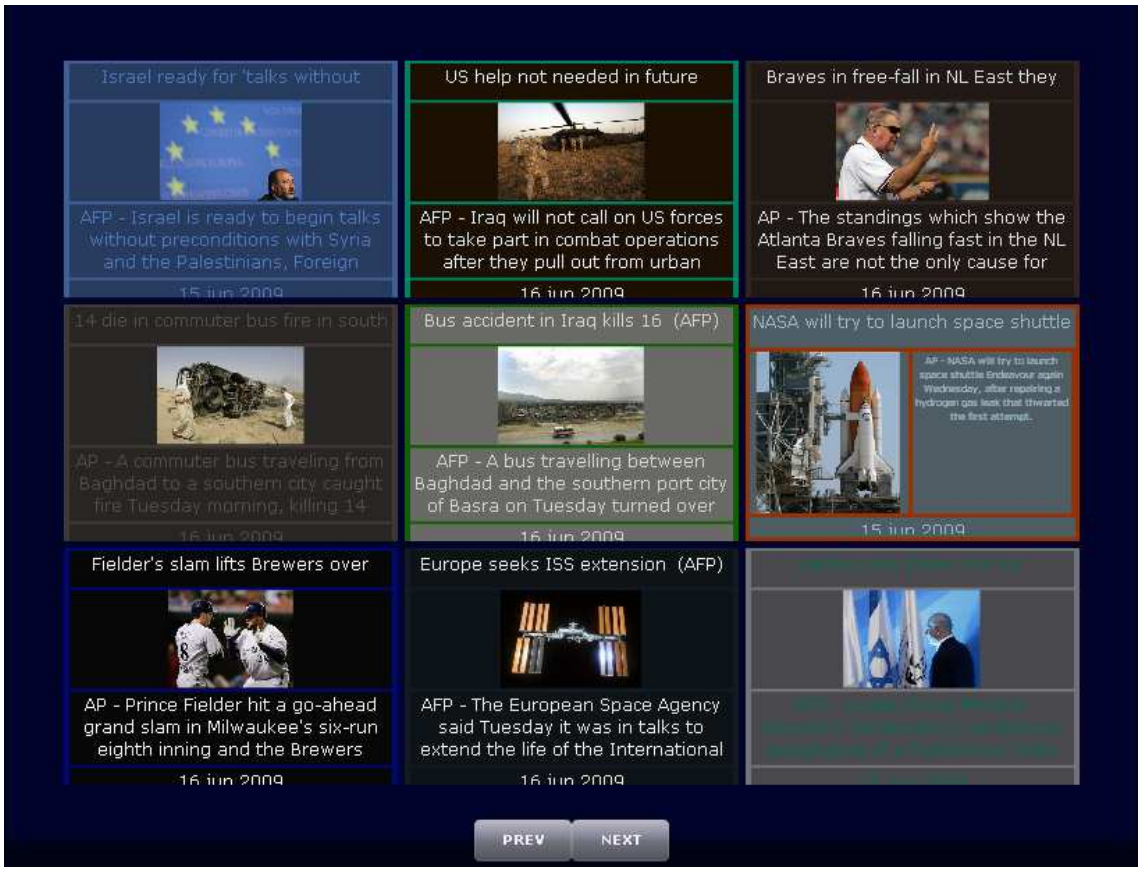

Figura 5.10 - Uso de surrogates en proyecto WakiTable

En la figura 5.10 podemos observar un nuevo proyecto sobre el que se utilizarán los surrogates. De hecho, en este caso, los surrogates estarán implementados en un proyecto de interfaz desarrollado en Flex y sobre el cual efectuaremos las pruebas de evaluación de los surrogates a nivel de colecciones. Constará de una matriz de surrogates sobre noticias diversas. Al igual que en los otros casos, al pulsar sobre un surrogate, éste se ampliará para mostrar la noticia a la que representa y aportar más información sobre la misma.

5.5 Modelo de interacción

Concluiremos el capítulo mostrando el modelo de interacción entre el usuario y la interfaz del proyecto WakiTable, al ser ésta la interfaz donde realizaremos las pruebas:

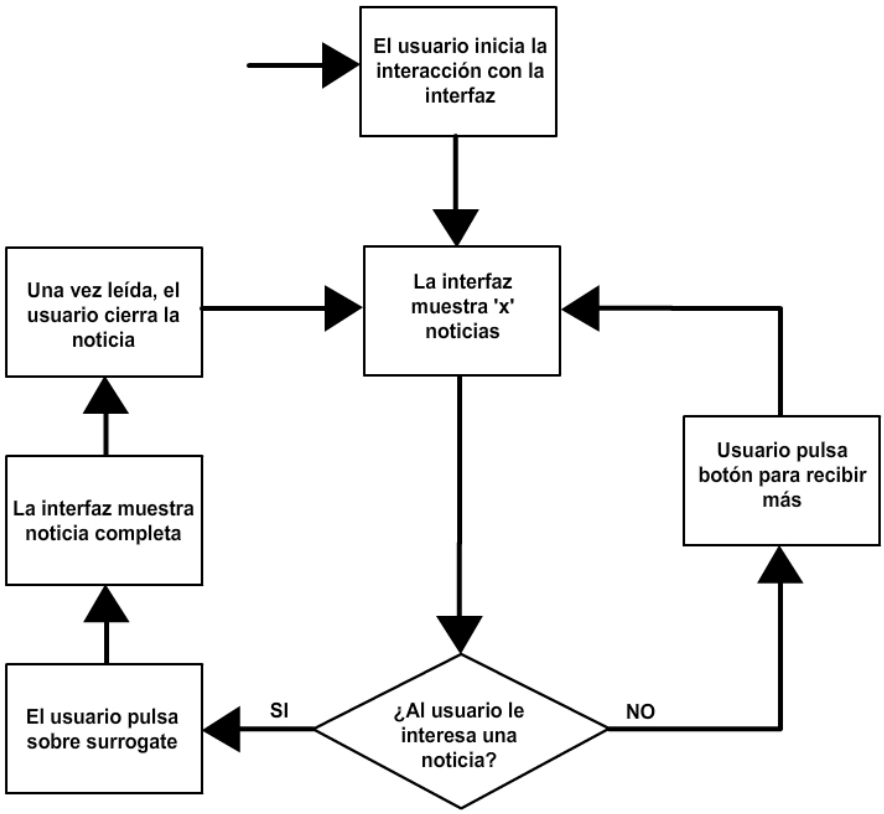

Figura 5.11 - Modelo de interacción

Como podemos ver en la figura 5.11, Wakitable será una interfaz que irá mostrando surrogates en una matriz. La interacción entre el usuario y ésta consistirá en que el usuario irá pulsando sobre los surrogates que han despertado su interés para informarse más sobre la noticia representada en éste.

6- Implementación de la solución

En el apartado *Descripción de la solución* se ha descrito la solución que se implementará para poder generar y utilizar surrogates en diferentes proyectos. Será en este apartado donde detallaremos el proceso de creación de surrogates a partir de noticias sindicadas y utilizarlos en todo proyecto que soporte el formato XML.

Nuestro sistema estará formado por tres componentes principalmente, el lector de fuentes web, el generador de surrogates y un paquete de clases para dar soporte a la lectura, dibujo y manipulación de los surrogates en proyectos donde se utilicen éstos. Los dos primeros componentes estarán integrados en un mismo proyecto Java llamado *SurrogateComposer*. Los resultados del primer componente (el lector) serán los datos de entrada para el segundo componente que generará los surrogates como resultado final. Para comprender el funcionamiento del sistema es necesario describir con anterioridad las estructuras de datos que se utilizarán en el programa.

6.1 Estructuras de datos

Las estructuras de datos servirán para organizar un conjunto de datos elementales con el objetivo de facilitar su manipulación. Disponemos de dos tipos de estructuras de datos.

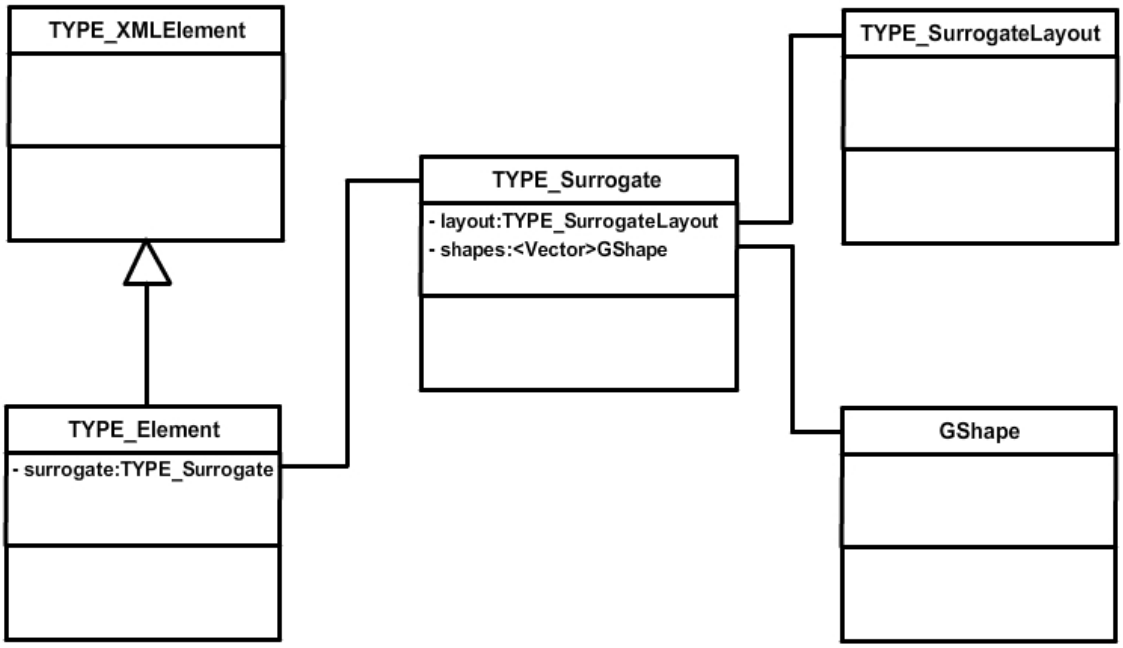

Figura 6.1 Diagrama UML de las estructuras de datos

6.1.1 Estructuras de almacenamiento

Las estructuras que describiremos en este apartado serán aquellas clases que darán soporte para el almacenamiento de los datos. Estas estructuras son las siguientes:

TYPE_XMLElement: Esta estructura de datos será sobre la que se iráalmacenando desde un principio la información leída desde Internet. Servirá para almacenar la información fundamental de una noticia, por lo que figurará la información siguiente: el identificador, el recurso de donde se recoge, la url de la noticia, el titular, el contenido, la fecha y por último y si dispone de ello: información sobre el tipo de elemento multimedia asociado, la ruta de donde está ubicado y una descripción del mismo. En el caso de las noticias este elemento será siempre una imagen.

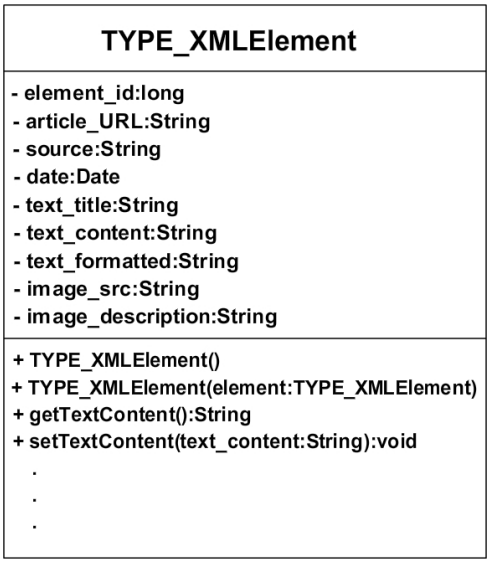

Figura 6.2 Atributos y métodos principales de la clase TYPE_XMLElement

En cuanto a los métodos de esta clase figurarán varios tipos de constructores, que dependiendo de los parámetros con los que se intente instanciar éste, se utilizarán unos u otros.

Por último, figurarán los métodos getters y setters de cada atributo de la clase y también varios métodos cuya finalidad será la de hacer un buen tratamiento de los datos, como puede ser la función getDateFormatted(), que convertirá una fecha completa (*EEE, dd MMM yyyy HH:mm:ss*) a una fecha simplificada (*dd MMM yyyy*) o la función generateID(), que calculara un único número identificador para cada noticia.

TYPE_Surrogate: Esta estructura de datos dispondrá de los atributos y métodos necesarios para almacenar la información referente a los componentes del propio surrogate. Destacarán un vector de GShapes destinado a almacenar la estructura del surrogate y un TYPE_SurrogateLayout para definir las características gráficas del mismo. Por último contendrá la información de la imagen asociada al surrogate.

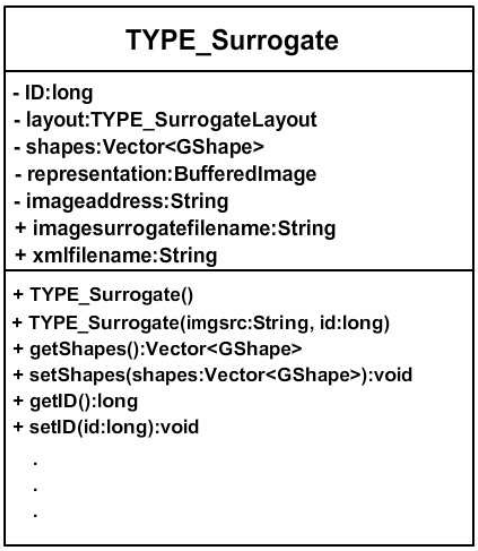

Figura 6.3 Atributos y métodos principales de la clase TYPE_Surrogate

Cabe destacar que los métodos de esta clase se reducen al constructor y los getters y setters de los atributos de la misma.

TYPE_Element: Este tipo de estructura hereda de TYPE_XMLElement, por lo que en ésta se incluye toda la información de TYPE_XMLElement anteriormente listada. Además se incluye un elemento más: un vector de TYPE_Surrogate. Este tipo de estructura es el más completo, ya que dispondrá de toda la información de la noticia recogida en TYPE_XMLElement más los soportes gráficos para su representación recogidos en TYPE_Surrogate.

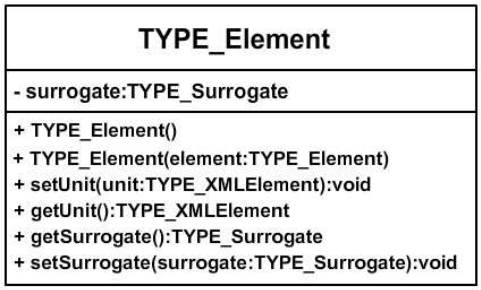

Figura 6.4 Atributos y métodos principales de la clase TYPE_Element

El constructor de esta clase heredará la información de su clase madre descrita al inicio.

6.1.2 Estructuras gráficas

Las estructuras que describiremos en este apartado serán aquellas clases que darán soporte para la representación gráfica de los surrogates.

TYPE_SurrogateLayout: Esta clase tendrá como objetivo definir varios aspectos gráficos del surrogate. De hecho, entre sus atributos figurarán la dimensión que tendrá el surrogate, un elemento Rectángulo destinado para cada componente que formará el surrogate: para el titulo, imagen, descripción y fecha. También figurarán los atributos destinados a almacenar el valor numérico de margin, que servirá para separar los rectángulos y por último también figurarán los atributos destinados a contener el valor del tamaño de las fuentes de los rectángulos que contengan texto.

Por otra parte, en esta clase se definirán una serie de métodos abstractos de tal manera que se implementarán en las clases que hereden de TYPE_SurrogateLayout: SurrogateLayout1, SurrogateLayout2, SurrogateLayout3, SurrogateLayout4 y SurrogateLayout5. Estas clases determinarán básicamente los tamaños y posicionamientos de los elementos del surrogate.

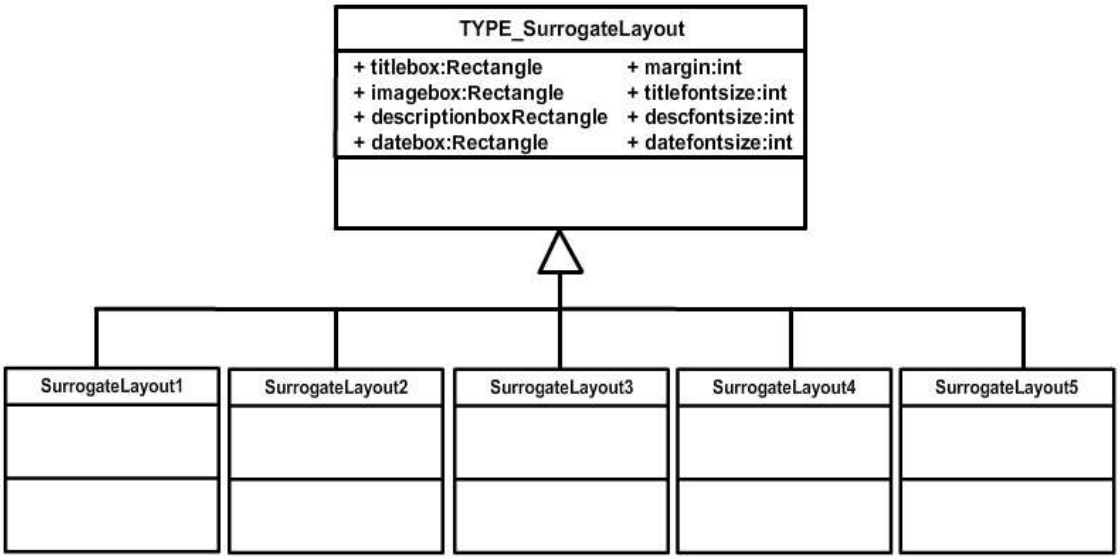

Figura 6.5 TYPE_SurrogateLayout y clases que heredan de ella.

SurrogateLayout1: Esta clase, al igual que el resto que heredan de TYPE_SurrogateLayout, tiene como objetivo aportar los valores que darán apariencia al surrogate. De hecho, en el programa están implementados estos cinco layouts para dar cinco aspectos diferentes al surrogate (ver apartado 3.4.4). En cada clase se definen

como atributos el valor del ancho y el alto del surrogate de manera estática y su constructor heredará el de su clase, por lo que se definirán todos los elementos de ella, y a continuación, se procederá a calcular los valores de los componentes que formarán el surrogate.

```
public SurrogateLayout1()   {
    super();
    width = (int)(width * default_xscale);
    height = (int)(height * default_yscale);
    int margin = SurrogateLayoutProp.margin;
    titlebox = new Rectangle(margin, margin,
                             width - 2*margin, (height/5)- (2*margin));

    imagebox = new Rectangle(margin, titlebox.height + 2*margin,
                            (width/2) - (2*margin), (height*7/10) - margin);

    descriptionbox = new Rectangle(imagebox.width + 2*margin, titlebox.height + 2*margin,
                           (width/2) - (2*margin), (height*7/10) - margin);

    datebox = new Rectangle(margin, descriptionbox.y + descriptionbox.height + margin,
                             width - 2*margin, (height/10)- margin);
}
```

Figura 6.6 Constructor de SurrogateLayout1. Cálculos para el tamaño y posicionamiento.

Como podemos observar en la figura 6.6, para calcular la posición (x,y) del primer elemento del surrogate (titlebox) solamente se aplica el valor del margen recogido de la clase SurrogateLayoutProp. En esta clase se definirán las constantes que se utilizarán por el resto del programa, como es el propio caso de SurrogateLayout1. Para acabar de calcular las dimensiones del Rectangle titlebox, el ancho y el alto, volvemos a usar el valor de margin, junto con el de width y height y otros cálculos adicionales.

Repitiendo este proceso y utilizando las posiciones calculadas de los elementos Rectangle anteriores, se calcularán todos los valores necesarios para disponer de un surrogate bien formado y proporcionado.

Por último, en SurrogateLayout1, se implementarán los métodos abstractos que se definieron en su clase madre. Tales métodos serán simplemente los getters de cada elemento de TYPE_SurrogateLayout.

GShape: Esta clase tiene por objetivo almacenar los valores necesarios para poder dibujar gráficamente un surrogate. Entre sus atributos figuran elementos de tipo gráfico importados desde las librerías gráficas, como Area, Rectangle, Color y RoundRectangle2D y también figurarán otros atributos como el id del GShape; el tipo (que puede ser main, img, txt o vid, dependiendo del tipo de recurso dedicado a esta área) y children, que indicará el número de GShapes que dependerán de él.

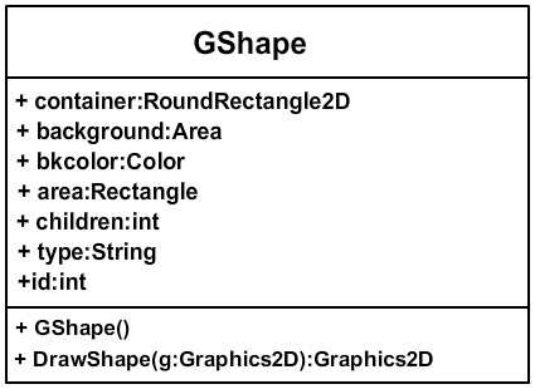

Figura 6.7 Atributos y métodos principales de GShape

Esta clase es muy simple. Lo único a destacar es que el constructor instanciará a los elementos gráficos de ésta. En primer lugar a container, después a area y por último a background.

GShapeText: Esta clase será la que se utilizará cuando el GShape contenga información de tipo textual. Heredará los atributos de GShape y tiene añadido otros para almacenar la información de tipo texto. Estos atributos servirán para almacenar el contenido y sus propiedades como el color, el tipo de fuente y el tamaño del texto.
Dispondrá de una serie de métodos enfocados a tratar las propiedades del texto.

GShapeImage: Esta clase será la que se utilizará cuando el GShape contenga información de tipo imagen. Heredará los atributos de GShape y tiene añadido otros para almacenar la información de tipo imagen. Entre ellos están la imagen, el area y la ruta de la misma.
En cuanto a métodos solamente dispondrá del constructor, que llamará al de su clase madre sin más.

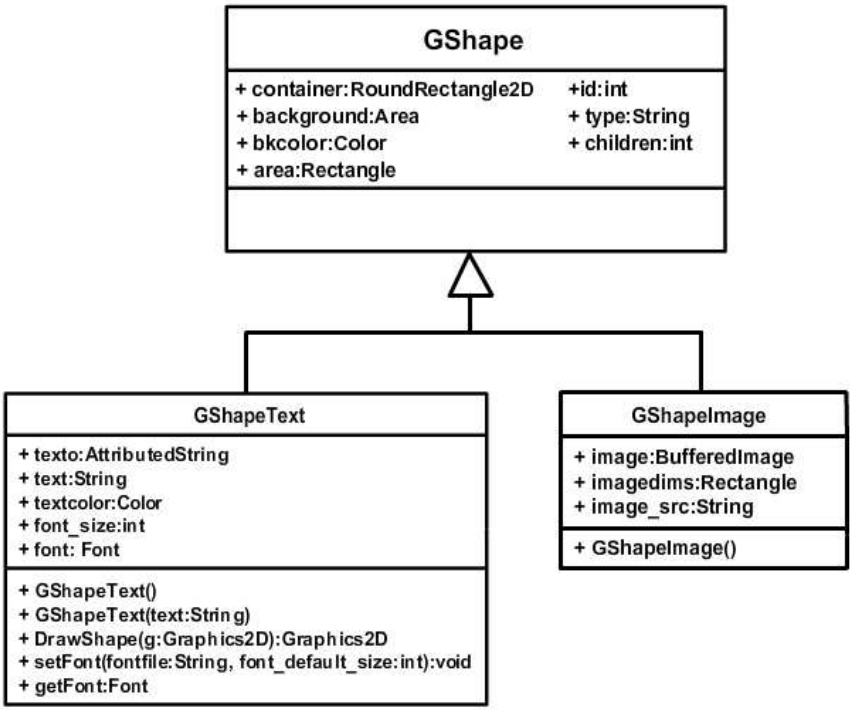

Figura 6.8 Herencia clases GShape

En el gráfico 6.8 podemos observar los atributos que tendrá cada tipo de GShape después de aplicar la herencia entre ellos.

Para acabar este punto cabe destacar que se ha creado un GshapeVideo para apoyar el uso futuro de videos en los surrogates, pero no se ha llegado a implementar.

6.2 Funcionamiento general del programa

Una vez se han descrito las estructuras de datos que se usarán en el programa y antes de entrar en detalle con el primer bloque del mismo, debemos describir su funcionamiento general.

La clase principal es SurrogateComposer. Dicha clase declarará tres vectores, uno para cada estructura de datos fundamental: uno para los TYPE_XMLElemetns; otro para los TYPE_Surrogates y por último, otro con los TYPE_Elements y determinará el funcionamiento del programa, ya que será desde donde se invocarán a las clases y métodos necesarios para lograr el objetivo del mismo.

En primer lugar se instanciará un objeto de la clase SurrogateComposer desde la función principal del programa. Al hacer esto, se instanciarán a su vez los tres vectores de cada tipo, ya que será la función del constructor de la clase. Una vez completado este primer paso, el programa mostrará un mensaje donde dejará escoger entre dos opciones:

la primera, si deseamos crear los surrogates de manera online, es decir, conectándonos a las fuentes web o si por el contrario, la segunda, deseamos crearlos mediante archivos XML previamente guardados por otra ejecución anterior del programa. En caso de ejecutar el programa por vez primera en nuestro sistema, solamente podremos generar surrogates escogiendo la primera opción.

Una vez escogida dicha opción, la clase SurrogateComposer llamará a un conjunto de métodos de diferentes clases cuyo objetivo será el de leer la información de las fuentes web y almacenarla en objetos de tipo TYPE_XMLElement y almacenarlos en archivos XML para que de esta manera, podamos crear en el futuro surrogates de manera offline aprovechando estos archivos de formato XML.

Una vez completado este primer bloque del programa, cuando tengamos la información volcada en el vector de TYPE_XMLElements, se llamará al segundo bloque, el del componedor de surrogates. En este segundo bloque se crearán los TYPE_Elements después de haberle añadido a la información de la noticia una estructura para que pueda ser representada en pantalla y se creará también el propio surrogate como imagen al utilizar todos estos datos de soporte gráfico.

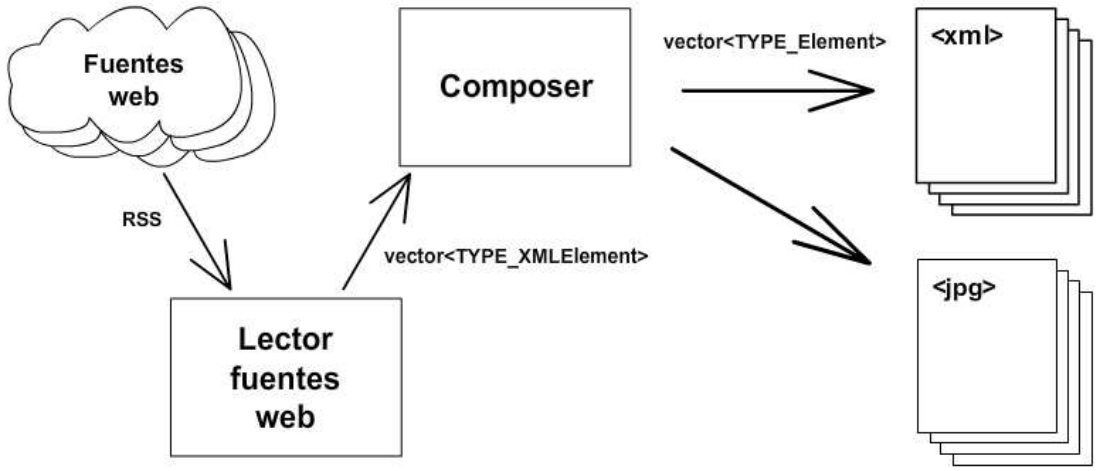

Figura 6.9 Proceso del programa

Así pues, una vez descrito todo el funcionamiento del programa, pasaremos a detallar en profundidad los dos bloques principales que componen el proyecto *SurrogateComposer*.

6.3 Lector de feeds online

Una vez se han descrito las estructuras de datos y el funcionamiento general del programa, en este apartado se describirá el bloque destinado a la lectura de las noticias disponibles en fuentes web y su posterior almacenaje tanto en un vector de TYPE_XMLElements, como en archivos de formato XML.

A continuación, en la figura 6.10, se muestra un diagrama con las clases que participarán en este proceso:

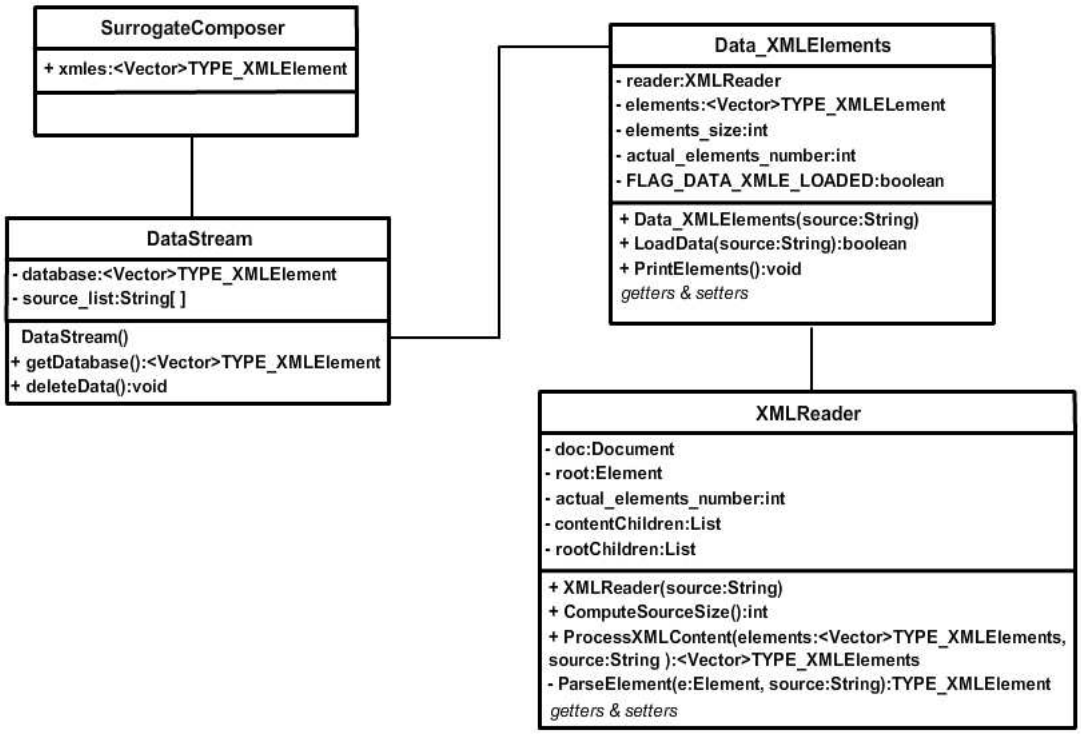

Figura 6.10 UML del Parser o Lector

Como hemos comentado anteriormente, después de instanciarse un objeto SurrogateComposer llamado composer en la función principal del programa y de esta manera haber dejado preparado a su vez el vector destinado a almacenar los TYPE_XMLElements, se instanciará un objeto de la clase DataStream.

DataStream: Esta clase tendrá como atributos un vector de TYPE_XMLElements y un array que contendrá las direcciones de las fuentes web de las que queremos recoger los feeds, en nuestro caso, los feeds disponibles en YahooNews. En primer lugar se instanciará el vector y acto seguido se llenará el array con las direcciones que nos

interesen. Para cada dirección llamaremos a la clase DATA_XMLElements, que gestionará la lectura de los *feeds* disponibles en dichas direcciones.

```
public class DataStream {

    private Vector<TYPE_XMLElement> database;
    private String[] source_list = new String[6];

    public DataStream()
    {
        database = new Vector<TYPE_XMLElement>();

        this.source_list[0] = "http://rss.news.yahoo.com/rss/world/";
        this.source_list[1] = "http://rss.news.yahoo.com/rss/iraq/";
        this.source_list[2] = "http://rss.news.yahoo.com/rss/mideast/";
        this.source_list[3] = "http://rss.news.yahoo.com/rss/science/";
        this.source_list[4] = "http://rss.news.yahoo.com/rss/entertainment/";
        this.source_list[5] = "http://rss.news.yahoo.com/rss/tech/";
        for (int i = 0; i < source_list.length; i++)    // load each source in a database section
        {
            DATA_XMLElements dados = new DATA_XMLElements(source_list[i]);
            database.addAll(dados.getElements());
            System.out.println("DATASTREAM---> Colleccionando Articulos de la fuente " + this.source_list[i]);
            System.out.println("Numero de articulos en base de dados: " + getDatabase().size());
        }
    }
```

Figura 6.11 Acciones de la clase DataStream

DATA_XMLElements y XMLReader: El objetivo común de estas clases será el de leer la información de las fuentes web. DATA_XMLElements será la clase encargada de gestionar la lectura, mientras de XMLReader dispondrá de los métodos necesarios para manipular las APIs JDOM y SAXBuilder, cuya finalidad será la de leer documentos XML.

DATA_XMLElements dispondrá de cinco atributos. El primero de ellos será de nuevo un vector de XML_Elements para almacenar las diferentes noticias de la fuente web. También dispondrá de variables de tipo entero que recogerán información sobre el número de noticias leídas. El cuarto atributo será un XMLReader, cuyo objetivo será leer las noticias y almacenar la información de éstas. Por último, el quinto y último atributo de esta clase será una variable de tipo boolean que servirá como indicador sobre si se ha acabado las operaciones de lectura o no.

Por su parte, los atributos de XMLReader serán los elementos necesarios para la correcta lectura de ficheros XML. Destacarán atributos de tipo Document, de tipo Element y un par de tipo List.

A continuación mostraremos la secuencia de acciones entre las dos clases para la correcta lectura y rellenado del vector de TYPE_XMLElements:

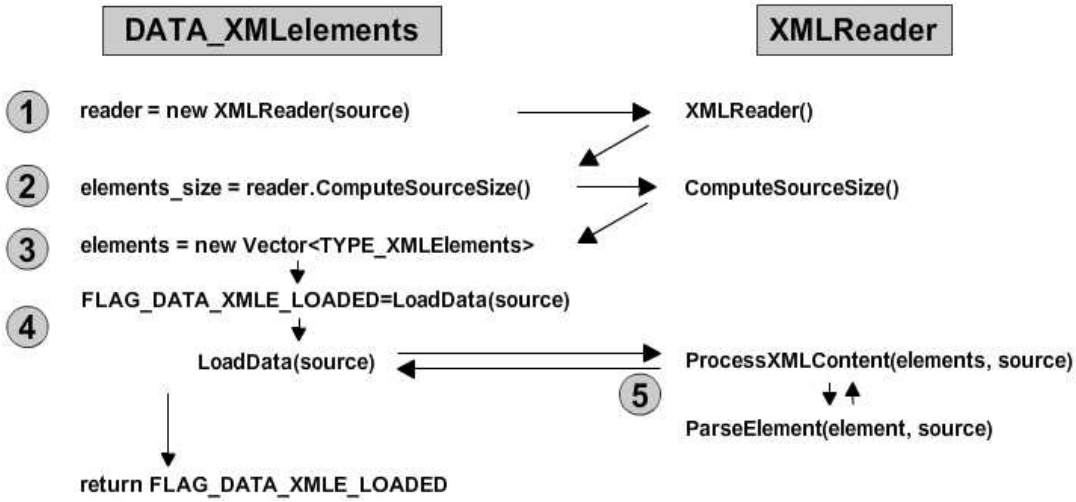

Figura 6.12 Secuencia de acciones para la lectura de noticias

1. La primera acción de DATA_XMLElements será la de instanciar al objeto reader de la clase XMLReader. Este objeto usará las APIs JDOM y SAXBuilder para la lectura de las fuentes web. En primer lugar instanciará a *sb*, un objeto de la clase SAXBuilder. Acto seguido, mediante la sentencia *this.doc = sb.build(source)* se cargará el archivo XML a leer en el objeto *sb* (será la dirección de la fuente web, por ejemplo, http://rss.news.yahoo.com/rss/world/). Una vez cargado éste, la sentencia *this.root = this.doc.getRootElement()* obtendrá el valor del primer nodo del archivo XML (en nuestro caso *channel*). Por último, una vez recogido el nodo raíz se recogerán en un objeto List sus nodos hijos con su contenido (en nuestro caso *ítem* y el contenido de éstos: *title,link, source, pudDate, media,* entre otros).
2. En este paso se calcularán el número de noticias (*item*s) que habrán en la fuente web leída.
3. Se preparará el vector de tipo TYPE_XMLElemet para ser rellenado por objetos de este tipo en las próximas acciones.
4. La función *LoadData()* se ocupará de que toda la información de las noticias leídas en el objeto reader, se plasmen en el vector elements. Para ello hará uso de la función *ProcessXMLContent()* que se describirá en el punto 5. Una vez completado este vector, se recogerá el número de noticias incluidas en éste y se volverá a la clase DataStream para que ésta acabe el proceso de lectura.
5. *ProcessXMLContent()* se ocupara de ir llamando a *ParseElement()* por cada noticia leída y ésta acabará rellenando la información en objetos

TYPE_XMLElement. A medida que se vayan conformando estos objetos, se incluirán en el vector elements y ya se dispondrá de toda la información bien almacenada.

Resumiendo este proceso de otra forma, *ParseElement()* se ocupará de componer cada noticia de la fuente web en un objeto de tipo TYPE_XMLElement.
ProcessXMLContent() se ocupará de que se vayan añadiendo estos objetos al vector de ese tipo de objetos y una vez completado éste, se calculará su tamaño y se volverá a la clase DataStream para finalizar este primer paso de lectura.
DataStream informará por pantalla de cuantas noticias se han recogido para cada una de las direcciones especificadas en el array *source_list* y devolverá la acción a la función main del programa. De esta manera, se habrá finalizado el proceso de lectura.

6.4 Guardar objetos TYPE_Element en archivos XML

Antes de pasar al bloque dedicado a componer surrogates a partir de los objetos anteriormente leídos y guardados en un vector, el programa llevará a cabo un paso previo tal y como ya se especificó en puntos anteriores, que será el de guardar estos objetos TYPE_XMLElement en formato XML por si en ejecuciones futuras se querrán componer surrogates de manera offline utilizando dichos archivos.
Para ello, desde la función principal se llamará al método *saveDataOffline()*. Esta función a su vez llamará al método *writeXMLelement()* de la clase TYPE_XMLElementWriter para cada elemento del vector.

TYPE_XMLElementWriter: Esta clase recibirá por parámetro un elemento de tipo TYPE_XMLElement. Será una clase dedicada a formar el archivo XML de la noticia, pero también guardará de manera local, la imagen asociada a la noticia en el directorio *typexmls.media*.

En primer lugar se especificará el directorio donde se guardará el archivo. Si éste no está creado, se creará. Acto seguido, se comprobará si el objeto TYPE_XMLElement tiene asociada una imagen. De ser así, se procederá a su guardado mediante los métodos implementados en la clase ImageManip *readImage()* y *SaveImage()*.

Una vez completados estos primeros pasos, se procederá a la creación propiamente dicha del archivo XML. Para crear un archivo XML primero será necesario definir todos los nodos y todos los contenidos de los mismos para posteriormente, proceder a su guardado final.

Así pues, primero se instanciará un elemento de la clase Element y se especificará la etiqueta raíz o root del archivo:

Element root = new Element ("xml_element");

A continuación se crearán los atributos del nodo padre *xml_element* de la siguiente manera:

```
root.setAttribute("id",Long.toString(element.getElement_id()));
root.setAttribute("url", element.getArticle_URL());
root.setAttribute("source",element.getSource());
```

Para crear el nodo hijo *text* que a su vez dispondrá de cuatro nodos hijos más, se procederá de la siguiente manera:

```
Element text_elem=new Element("text");
        Element date = new Element("pubdate");
        date.setText(element.getDateFormattedLong().toString());
        Element text_title=new Element("text_title");
        text_title.setText(element.getText_title());
        Element text_content=new Element("text_content");
        text_content.setText(element.getText_content());
        Element text_formatted=new Element("text_formatted");
        text_formatted.setText(element.getText_formatted());
text_elem.addContent(date);
text_elem.addContent(text_title);
text_elem.addContent(text_content);
text_elem.addContent(text_formatted);
root.addContent(text_elem);
```

Para completar el archivo XML añadiremos la información sobre la imagen con la etiqueta *media* con una serie de atributos (*type* y *url*) y texto procediendo de la forma habitual:

```
if(!element.getImage_src().equals(null)) {
        Element elimg=new Element("media");
        elimg.setAttribute("type","image");
        elimg.setAttribute("url",element.getImage_src());
        elimg.setText(element.getImage_description());
        root.addContent(elimg);
}
```

Finalmente, una vez ya hemos establecido todo el contenido del archivo de formato XML bien formado tenemos que acabar de guardarlo mediante la clase Document. Esto se acabará de completar de manera sencilla con la siguiente sentencia:

Document doc = new Document(root);

Para guardar este *Document* localmente se tendrá que hacer uso de las clases Format, XMLOutputter, File y FileOutputStream. Cada archivo se guardará con el nombre "xml_element" seguido del id de su objeto TYPE_XMLElement asociado.

Para acabar este punto es importante decir que para no ocupar espacio innecesario en memoria, una vez se hayan guardado los TYPE_XMLElements tanto en el vector xmls de SurrogateComposer como en archivos de formato XML de manera local, se ha de vaciar toda la información recogida en la instancia de DataStream, que recordamos, también tenía el vector de TYPE_XMLElements lleno. Haremos esto mediante el método *deleteData()* de DataStream, que vaciará el vector de TYPE_XMLElements.

6.5 Recuperar Vector TYPE_XMLElements de manera offline

Como se especifica en el punto sobre el funcionamiento general del programa, una vez se ha capturado la información de manera online y se ha guardado localmente, en próximas ejecuciones del programa se dará la opción de generar los surrogates de manera offline a partir de los archivos XML guardados. En este punto especificaremos cómo el programa puede llevar a cabo esta acción.

Para este caso, se instanciará la clase OfflineDataStream. Dicha clase tendrá como único atributo el vector donde se almacenarán los TYPE_XMLElements. A su vez, y de igual forma que DataStream, dispondrá de dos métodos destinados a devolver dicho vector (cuando lo tengamos lleno) y a vaciarlo (para optimizar los recursos del sistema donde se ejecute el programa).

Las acciones que se ejecutarán en el constructor de esta clase serán las siguientes:

1. Se instanciará el vector *database* donde se irán almacenando los TYPE_XMLElements en puntos siguientes.
2. Se especificará el directorio donde están guardados los archivos XML.
3. Se creará un array llamado *children* donde se especificarán todos los archivos XML del directorio.
4. Para cada posición del array, es decir, para cada archivo XML, se llamará a la función *readXMLelement()* de la clase TYPE_XMLElementReader.
5. Se guardarán los objetos TYPE_XMLElement devueltos en el vector *database*

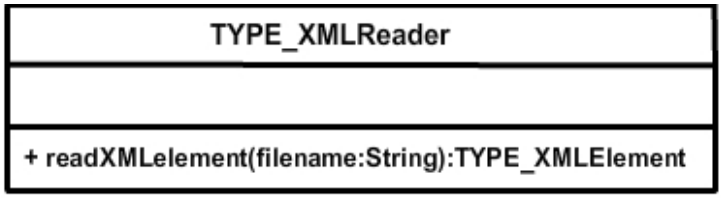

Figura 6.13 TYPE_XMLReader

TYPE_XMLElementReader: Esta clase solamente contendrá la función *readXMLElement()*. Se le pasará por parámetro el nombre del archivo del directorio donde están almacenados los XML y devolverá un objeto de tipo TYPE_XMLElement. En esta función, al igual que en XMLReader hará uso de las APIs JDOM y SAXBuilder para soportar la lectura de los XML.

En primer lugar se declarará el objeto element de tipo TYPE_XMLElement que devolverá la función. A continuación se instanciará el objeto de la clase SAXBuilder y el Document a leer. Una vez hechos estos pasos, se procederá al montaje del objeto *element* a raíz de la recogida de datos del archivo. A continuación se mostrará un ejemplo de cómo se ejecutará dicha tarea:

```
TYPE_XMLElement element  = new TYPE_XMLElement;
SAXBuilder builder=new SAXBuilder(false);
Filefl=new File(properties.SurrogateProp.type_xml_directory+filename);
Document doc=builder.build(fl);
Element root=doc.getRootElement();
element.setElement_id(Long.parseLong(root.getAttributeValue("id")));
```

element.setArticle_URL(root.getAttributeValue("url"));

element.setSource(root.getAttributeValue("source"));

En las sentencias anteriores podemos observar cómo se está transfiriendo la información desde el archivo XML al objeto TYPE_XMLElement. Primero se recoge la información desde el archivo XML mediante métodos *get* y se almacena ésta en el objeto TYPE_XMLElement mediante métodos *set.*

Para concluir con este punto, mostraremos el diagrama UML de todo el proceso de lectura de manera offline que se ha descrito hasta el momento:

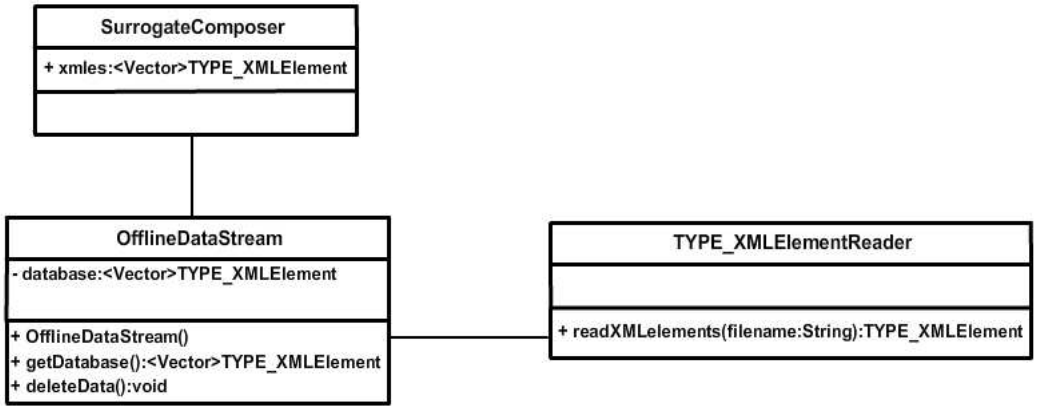

Figura 6.14 UML sobre lectura de archivos de manera offline

6.6 Generador de surrogates

6.6.1 Descripción del proceso

En este apartado especificaremos la manera en que se compondrán los surrogates. Hasta el momento hechos visto todo el procedimiento necesario para leer la información tanto de manera online desde fuentes web como de manera offline desde archivos XML. Por lo tanto, ya se haya recogido de una manera u otra, dispondremos de un vector de TYPE_XMLElements lleno de noticias. Tal y como se mencionó en puntos anteriores, los resultados que se obtendrán en este apartado serán dos: añadir información sobre la estructura gráfica para representar al surrogate en el archivo XML y obtener el surrogate totalmente compuesto como una imagen de formato JPG.

A continuación mostraremos un diagrama UML con las clases participantes en el proceso de generación de un surrogate:

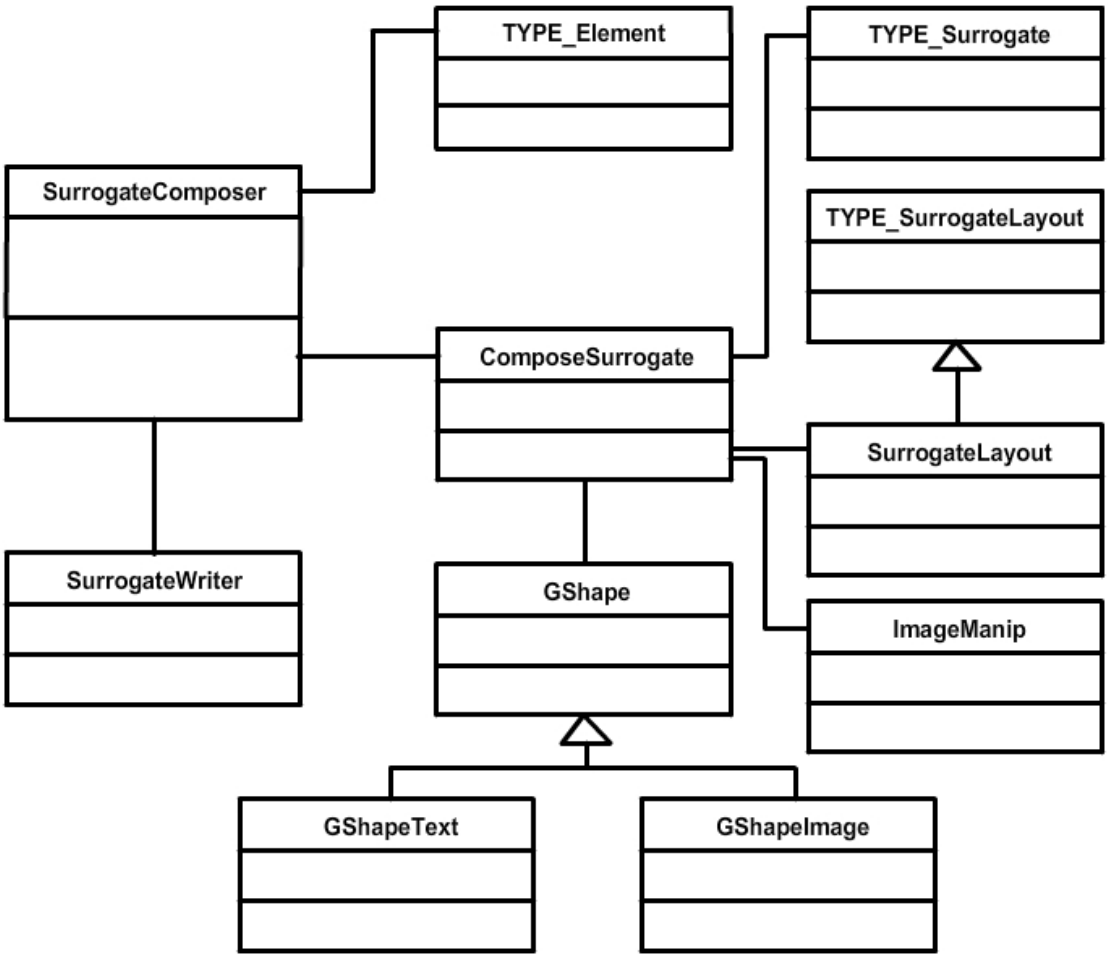

Figura 6.15 UML sobre la composición de los surrogates

Tal y como se ha indicado al inicio de este punto, el programa principal dispone del vector de TYPE_XMLElements completo después de realizar los procesos especificados en los puntos anteriores. Será en este momento donde pasará la acción del programa a la función *generateSurrogates()*, que gestionará todo el proceso para generar los TYPE_Elements y las imágenes finales de los surrogates, que será el siguiente:

1. En primer lugar dará la opción de escoger qué tipo de layout o apariencia querremos dar a los surrogates. (Consultar punto 3.4.4 sobre tipos de surrogates).
2. Para cada objeto del vector de tipo TYPE_XMLElement *xmles* realizará las siguientes acciones llamando a los métodos de diferentes clases especificadas en el diagrama 5.15 para completar el proceso de generación de surrogates:
    i. Declarará al objeto *element* de tipo TYPE_Element donde al final del proceso contendrá toda la información del surrogate.
    ii. Llamará al método *composeXMLSurrogate()* para que gestione el proceso de composición del surrogate con el layout especificado y realice los cálculos necesarios.
    iii. Pasará información del objeto TYPE_XMLElement al objeto TYPE_Element.
    iv. Generará el surrogate en forma de imagen JPG.
    v. Añadirá la información del objeto TYPE_Surrogate al objeto TYPE_Element.
    vi. Escribirá el objeto *element* a un archivo XML de manera que ya dispondremos de toda la información para poder representar un surrogate gráficamente a partir de un archivo XML.
3. Finalmente creará un archivo XML como un índice de todos los archivos XML *element* para que otros programas hagan uso de éstos.

Después de describir de forma general todo el proceso de creación de los surrogates, detallaremos cada una de sus partes más relevantes. A continuación describiremos las operaciones que se producirán en la clase ComposeSurrogate.

6.6.2 Composición del Surrogate

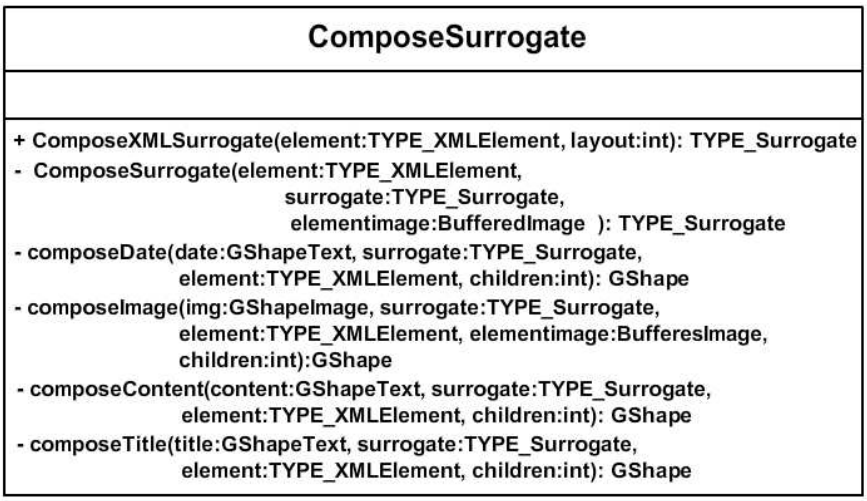

Figura 6.16 Clase ComposeSurrogate

ComposeSurrogate: A esta clase se llega desde la función de la clase principal del programa *generateSurrogates()* habiéndole pasado por parámetro un elemento TYPE_XMLElement del vector que contiene a todos éstos y el layout escogido. Por lo que se ejecutarán las sentencias descritas en el método público de esta clase.

En primer lugar se declarará el objeto TYPE_Surrogate *surrogate* y se le dará el mismo id que el elemento *element*. Acto seguido se comprobará si el TYPE_XMLElement dispone de una imagen asociada. De ser así, cargará dicha imagen en un objeto BufferedImage desde el directorio donde se guardaron las imágenes de las noticias (en el directorio *typexmls.media*). Llegados a este punto, según la opción de layout escogida y de si *element* dispone de imagen o no, se escogerá una clase SurrogateLayout u otra para definir la estructura del surrogate. Pueden darse los casos siguientes:

- Opción de layout 1 e imagen asociada: se escogerá SurrogateLayout1
- Opción de layout 1 e imagen asociada: se escogerá SurrogateLayout2
- Opción de layout 1 y no imagen asociada: se escogerá SurrogateLayout3
- Opción de layout 2 e imagen asociada: se escogerá SurrogateLayout4
- Opción de layout 2 y no imagen asociada: se escogerá SurrogateLayout3
- Opción de layout 3 e imagen asociada: se escogerá SurrogateLayout5
- Opción de layout 3 y no imagen asociada: se escogerá SurrogateLayout3

Mediante la sentencia siguiente se le dará al objeto *surrogate* la estructura definida en cada tipo de SurrogateLayout (consultar punto 6.1.2):

surrogate.setLayout(new SurrogateLayout1());

Una vez establecida toda la información sobre los componentes que tendrá el surrogate, su posicionamiento y proporciones, se llamará a la función *composeSurrogate()* pasándole por parámetro los elementos *element*, *surrogate* y *elementimage* para que se acabe de gestionar todo el proceso de creación del objeto TYPE_Surrogate.

composeSurrogate: Lo primero que se declarará en esta función será un vector de GShapes llamado *shapes* y una variable de tipo entero llamada *children*. El vector servirá para almacenar los GShapes de los elementos del surrogate más el GShape principal, mientras que la variable *children* servirá para contabilizar cuántos son los hijos que dependerán del GShape principal una vez finalizado el proceso.
A continuación se detallará cómo se van a ir creando todos estos GShapes. Como ya hemos dicho, el primero será el principal:

```
int children = 0;
Vector<GShape> shapes = new Vector<GShape>();
GShape main = new GShape();
main.type = "main";
main.id = 0;
main.container = new RoundRectangle2D.Double(0,0,
                                    surrogate.getLayout().getWidth(),
                                    surrogate.getLayout().getHeight(),
                                    15,15);
main.area = new Rectangle(SurrogateLayoutProp.margin, SurrogateLayoutProp.margin,
                          surrogate.getLayout().getWidth()-2*SurrogateLayoutProp.margin,
                          surrogate.getLayout().getHeight()-2*SurrogateLayoutProp.margin);

main.bkcolor=Color.decode("0xFFFFFF");
main.composeStainBackground();
shapes.add(0, main);
surrogate.setShapes(shapes);
```

Figura 6.17 Declaración y asignación de valores del GShape principal o main

Se puede observar en la figura 6.17 toda la asignación de valores al GShape principal: el atributo type contendrá "main"; el id de este GShape padre será "0"; se darán los valores a los elementos *container* y *area* a partir de la información calculada previamente en SurrogateLayout; y finalmente se dará el color de fondo.
Una vez asignada toda esta información, se incluirá el GShape principal al vector *shapes* y éste a su vez, al objeto *surrogate.*

Una vez incluido al vector *shapes* el GShape principal, se procederá a ir incluyendo los GShapes de los elementos que formarán el surrogate. En nuestro caso serán tres ó cuatro dependiendo de si la noticia dispone de imagen asociada o no. En caso de disponer de ella los GShapes serán los siguientes: el dedicado al título; el dedicado a la descripción; el dedicado a la fecha y el dedicado a la imagen. Los tres primeros serán de tipo GShapeText, ya que contendrán texto, mientras que el último será de tipo GShapeImage.

```
if (!element.getText_title().equals(null) && !element.getText_title().equals("")){
    children = children + 1;
    GShapeText title = new GShapeText(element.getText_title());
    title.text=element.getText_title();
    title.setTextProperties("000000", 80);
    shapes.add(children, composeTitle(title, surrogate, element, children));
    surrogate.setShapes(shapes);
}
```

Figura 6.18 Declaración y asignación de valores del GShapeText

```
private static GShape composeTitle(GShapeText title,
        TYPE_Surrogate surrogate, TYPE_XMLElement element, int children) {
    title.type = "txt";
    title.id = children;
    title.bkcolor = surrogate.getShapes().get(0).bkcolor;
    title.container = new RoundRectangle2D.Double(
            surrogate.getLayout().getTitlebox().x,
            surrogate.getLayout().getTitlebox().y,
            surrogate.getLayout().getTitlebox().width,
            surrogate.getLayout().getTitlebox().height,
            8,8);
    title.area = new Rectangle(5,5,
            surrogate.getLayout().getTitlebox().width - 10,
            surrogate.getLayout().getTitlebox().height - 10);

    title.font = new Font("verdana.ttf",Font.BOLD, 14);
    title.font_size = 14;
    title.text = element.getText_title();
    return title;
}
```

Figura 6.19 Subfunción ComposeTitle

Como podemos observar, las dos figuras muestran el procedimiento para componer el GShape del título del surrogate. La figura 6.18 comienza con un condicional para comprobar que el surrogate dispone de este elemento. De ser así, se llevarán a cabo una serie de acciones similares a lo que se hizo con el GShape *main*. En primer lugar, la variable *children* se incrementará en uno e indicará que será un hijo del GShape *main.* Se instanciará un nuevo GShapeText al que se le pasará el texto del título como

parámetro desde el elemento TYPE_XMLElement *element.* A continuación se aportarán las propiedades del texto y antes de incluir este GShape al vector *shapes* se llamará a la función *composeTitle()* para acabar de especificar las propiedades del GShape. Una vez completadas estas acciones se incluirá al vector *shapes* y éste, de igual manera que *main,* al surrogate.

Este proceso se repetirá para cada componente de manera similar. De hecho, para los GShape de la descripción y la fecha será prácticamente el mismo procedimiento al ser éstos de tipo texto. Para el GShape dedicado a la imagen, el proceso cambiará ligeramente, ya que el tratamiento de ésta se hará de manera distinta ya que requerirá de una serie de cálculos. Estos cálculos estarán dirigidos a comprobar la necesidad de escalar la imagen del surrogate o no y en caso afirmativo, saber en qué proporción. Para realizar dicha tarea, se usará la clase de apoyo ImageManip.

Una vez se ha completado todo el proceso de componer cada GShape e incluirlo en el vector *shapes* del objeto *surrogate*, se acabará el proceso de componer el TYPE_Surrogate con las siguientes dos líneas de código:

```
surrogate.getShapes().get(0).children = children;
surrogate.setID(element.getElement_id());
```

La primera especificará cuántos GShape hijos tendrá el GShape principal y la segunda acabará especificando el mismo id del objeto *element* al objeto *surrogate* para mantener la unidad de dichas estructuras.

6.6.3 Generación de los surrogates en imágenes en formato JPG

En este apartado se describirá el proceso en el que se generarán los surrogates en formato JPG a partir de toda la información procesada a lo largo de todo el programa hasta este momento.

Solamente se necesitará un paso previo para comenzar con la creación del surrogate como imagen, que será el de volcar la información recogida en el objeto

TYPE_XMLElement a otro de tipo TYPE_Element mediante la función *setUnit()* de la clase TYPE_Element.
Una vez realizado este paso solo hace falta hacer una comprobación: si el surrogate dispone de imagen asociada o no. Dependiendo de esto se llamará a una clase u otra destinada a este propósito: ComposeImgSurrogate o ComposeTxtSurrogte.
En ambos casos se pasarán por parámento el objeto *element* y el objeto *surrogate* previamente rellenados en los pasos que se han descrito hasta este momento.

ComposeImgSurrogate: Esta clase se encargará de gestionar todo el proceso para la creación del surrogate en formato JPG. Para ello, utilizará una serie de métodos implementados en clases de apoyo del paquete *surrogate.compose_utils* para realizar todos los cálculos sobre colores, proporciones y demás características para llevar a cabo dicha tarea. Estas clases de apoyo serán ImageManip, ColorCompute y TextManip
Como ya se especificó anteriormente y como su nombre indica, esta clase creará la imagen de los surrogates que tengan una imagen asociada. Para llevar a cabo esta tarea se seguirá el siguiente procedimiento:

En primer lugar y mediante la función *formatTitleBoxV()* se encuadrará el texto del título a su área correspondiente. El siguiente paso será cargar en un objeto BufferedImage la imagen asociada al surrogate mediante la función *loadImageFromHD()*. El tercer paso será el de calcular el color de fondo que se le dará al surrogate a través del método *getSurrBackColor()* que extraerá el valor medio de la imagen cargada anteriormente. Acto seguido se calculará el color de la fuente con *getTextColor()*. Este método tratará de asignar un valor con el que dar contraste respecto al color de fondo anteriormente calculado de manera que el texto se pueda leer claramente.
Una vez efectuados estos pasos, se procederá a crear la imagen del surrogate y sobre la cual se irán añadiendo los diferentes elementos que lo compondrán. Para ello se creará el objeto de tipo BufferedImage *background*. En un principio, solamente se instanciará el objeto dándole las dimensiones especificadas en el layout, por lo que hasta el momento estará vacía. A continuación se instanciará un elemento Graphics al que se le asociará *background* y a partir de aquí se empezará a dibujar la imagen propiamente dicha. Lo primero que se hará será darle el color de fondo a la imagen. Una vez dado éste, se irán incluyendo los diferentes elementos por los que este compuesto el

surrogate. Como podemos ver en la figura 5.20 en primer lugar se incluirá el titulo; en segundo lugar la descripción; en tercer lugar la imagen y por último, la fecha.

```
BufferedImage background = new BufferedImage(surrogate.getLayout().getWidth(), surrogate.getLayout().getHeight()
                                        ,BufferedImage.TYPE_INT_BGR);
Graphics g = background.getGraphics();
g.setColor(areacolor);
g.fillRect(0,0, surrogate.getLayout().getWidth(), surrogate.getLayout().getHeight());
Graphics2D graphics = (Graphics2D) g;
graphics = TextManip.writeTitleText(graphics, surrogate.getLayout().titlebox,
        surrogate.getLayout().getTitleFont(), element.getText_title(), textcolor, areacolor);

graphics = TextManip.writeTitleText(graphics, surrogate.getLayout().descriptionbox,
        surrogate.getLayout().getDescFont(), element.getText_content()+ " ...", textcolor, areacolor);

background = addImageToBackground(background, elementimage, surrogate);

graphics = TextManip.writeDescText(graphics, surrogate.getLayout().datebox,
        surrogate.getLayout().getDateFont(), element.getDateFormatted(), textcolor, areacolor);

surrogate.setRepresentation(background);
```

Figura 6.20 Proceso de creación del surrogate como imagen JPG

Una vez se han compuesto todos los elementos sobre la imagen *background*, ésta se almacenará en el atributo de tipo BufferedImage *representation* de la estructura TYPE_Surrogate mediante la siguiente línea:

Surrogate.setRepresentation(background);

Por último, se le dará un id a la imagen generada y se guardará localmente mediante el método *saveImage()* facilitado por la clase ImageManip.

Para los surrogates que no dispongan de imagen asociada será la clase ComposeTxtSurrogate la encargada de realizar todo este proceso.

Tras finalizar todo este proceso, la acción del programa volverá a la función *generateSurrogates()* de SurrogateComposer donde se acabará de añadir el objeto *surrogate* al objeto *element*. De esta manera, ya tendremos el objeto *element* totalmente completado con toda la información proveniente de TYPE_XMLElements y de TYPE_Surrogate, por lo que restará hacer el último paso: guardar los objetos/noticias *elements* en archivos XML.

6.6.4 Generación de los archivos XML de los surrogates

En este apartado detallaremos brevemente el último paso para poder disponer de los surrogates en archivos XML. Para ello se invocará a la clase SurrogateWriter.

SurrogateWriter: esta clase tendrá un funcionamiento muy similar al de la clase TYPE_XMLElementsWriter, ya que los métodos utilizados para el guardado en archivos XML de la información contenida en las estructuras de datos serán prácticamente iguales. A diferencia que TYPE_XMLElementsWriter, esta clase guardará los datos almacenados en un elemento TYPE_Element, por lo que además de efectuar el guardado de la parte proveniente de TYPE_XMLElements, tendrá que hacerlo también de la información guardada en TYPE_Surrogate.

La primera parte de esta función estará destinada a guardar la información que ya se guardó en el TYPE_XMLElement. Es decir, se definirá el elemento *root* o raíz con el nombre de "element" (en vez de "type_xmlelement"). A este nodo padre se le agregarán tres atributos: la id del elemento, la url del artículo y la dirección de la fuente web de donde se ha recogido la información.

A continuación se definirá el nodo hijo "text", que a su vez contendrá los hijos "pubdate", "text_title", "text_content" y "text_formatted". Estos nodos contendrán la información sobre la fecha de publicación del artículo, el titular, la descripción y el texto con formato de la misma.

También se agregará un nodo hijo a "element" llamado "media". El contenido de éste será el mismo que ya se definió en TYPE_XMLElementWriter, es decir, definir el tipo de media asociado (image), la url de la imagen y la descripción de la misma.
Es a partir de este momento que la función definirá acciones nuevas, ya que se dispondrá a escribir la información contenida en la estructura *surrogate* del TYPE_Element.

```
TYPE_Surrogate surrogate=element.getSurrogate();
Element surr=new Element("surrogate");
surr.setAttribute("id", Integer.toString((int)surrogate.getID()));
surr.setAttribute("layout",surrogate.getLayout().getClass().getName());
surr.setAttribute("margin",((Integer)surrogate.getLayout().margin).toString());

for(int i=0;i<surrogate.getShapes().size();i++) {
    GShape gs=surrogate.getShapes().get(i);

    Element shape=new Element("GShape");
    shape.setAttribute("id", Integer.toString(gs.id));
    shape.setAttribute("type", gs.type);
    shape.setAttribute("n", Integer.toString(gs.children));

    Element position=new Element("position");
    position.setAttribute("x", Integer.toString((int)gs.container.getX()));
    position.setAttribute("y", Integer.toString((int)gs.container.getY()));
    position.setAttribute("width", Integer.toString((int)gs.container.getWidth()));
    position.setAttribute("height", Integer.toString((int)gs.container.getHeight()));

    Element bkcolor=new Element("bkcolor");
    bkcolor.setAttribute("r", Integer.toString(gs.bkcolor.getRed()));
    bkcolor.setAttribute("g", Integer.toString(gs.bkcolor.getGreen()));
    bkcolor.setAttribute("b", Integer.toString(gs.bkcolor.getBlue()));

    Element area=new Element("area");
    area.setAttribute("x", Integer.toString((int)gs.area.getX()));
    area.setAttribute("y", Integer.toString((int)gs.area.getY()));
    area.setAttribute("width", Integer.toString((int)gs.area.getWidth()));
    area.setAttribute("height", Integer.toString((int)gs.area.getHeight()));
```

Figura 6.21 Proceso de escritura de los GShapes de un surrogate

Como se puede observar en la figura 6.21, para escribir en formato XML la parte destinada a la estructura del surrogate, primero de todo se ha de instanciar un objeto TYPE_Surrogate pasándole la información del mismo mediante la línea:

TYPE_Surrogate surrogate=element.getSurrogate()

Una vez hecho esto, toda la información estará volcada en el objeto surrogate y a partir de ahí iremos escribiendo las etiquetas correspondientes. La primera de ella será "surrogate", que se le añadirán tres atributos: el identificador del mismo; el tipo de layout que se ha aplicado para componer el surrogate y el valor del margen que se ha usado para realizar los cálculos sobre posicionamiento y proporción del mismo

Una vez bien formada esta etiqueta, mediante una estructura *for-loop* se irá añadiendo la información de los GShapes del surrogate. Cabe destacar que dentro de este *for* se irán cogiendo cada uno de los GShapes guardados en el vector *shapes* de la estructura de datos TYPE_Surrogate. De esta manera, el primer GShape que se escribirá en el archivo XML será el correspondiente al *main* o principal. A continuación mostramos un ejemplo de la salida de un GShape de tipo "main":

```
<GShape id="0" type="main" n="4">
        <position x="0" y="0" width="240" height="260" />
         <bkcolor r="255" g="255" b="255" />
         <area x="5" y="5" width="230" height="250" />
         <backgroundshape nbpoints="144" xcoords=" 85...0" ycoords="30 ...0" />
 </GShape>
```

En caso de que el GShape sea de tipo GShapeText, la salida de este tipo sería la siguiente:

```
<GShape id="1" type="txt" n="0">
        <position x="5" y="5" width="230" height="42" />
        <bkcolor r="255" g="255" b="255" />
        <area x="5" y="5" width="220" height="32" />
        <backgroundshape nbpoints="0" xcoords="85...0" ycoords="...0" />
       <font size="14" font="Dialog" bold="0" italic="0" underline="0"
              justify="center" color="0" transparency="0" />
      <text margin="5">Ejemplo </text>
 </GShape>
```

Podemos observar en este caso que el valor del atributo *n* de la etiqueta GShape es "0" a diferencia del valor "4" del GShape de tipo *main.* Esto es así ya que este GShapeText no tiene ningún hijo que dependa de él. En este tipo de GShape hay disponibles un par de etiquetas más, la dedicada a las propiedades de la fuente que se empleará para el texto que figurará entre las etiquetas <text> y </text>.

Por último y para completar este apartado, mostraremos un ejemplo de la salida de un GShapeImage:

```
<GShape id="3" type="img" n="0">
  <position x="5" y="73" width="190" height="161" />
  <bkcolor r="14" g="16" b="13" />
  <area x="5" y="5" width="180" height="151" />
  <backgroundshape nbpoints="0" xcoords="85... 0" ycoords="30...0" />
  <img margin="5">C:/newsme\typexmls/media\20168694.jpg</img>
</GShape>
```

En este caso observamos como la etiqueta exclusiva de este tipo de GShape es <img>. Entre esta etiqueta se adjuntará la ruta de la imagen donde está almacenada localmente.

6.7 Servicios de apoyo

En este apartado mencionaremos los métodos de las clases de apoyo que se han ido utilizando en *SurrogateComposerSimple* y que han servido para lograr el objetivo de generar los surrogates. Básicamente se han utilizado las clases ColorCompute e ImageManip del paquete *surrogates.compose_utils*. A continuación detallaremos brevemente todos los métodos que se han usado especificando su función.

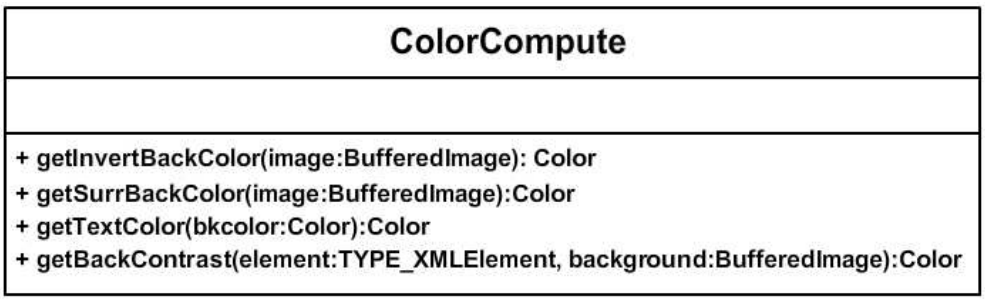

Figura 6.22 Clase ColorCompute

ColorCompute: en esta clase se han implementado los métodos siguientes:

- *getSurrBackColor():* Función que devuelve el color medio de una imagen. Se pasa por parámetro la imagen sobre la que calcular y se devuelve el color.
- *getInvertBackColor():* Función que extrae las componentes RGB del pixel de la imagen que se pasa por parámetro y se calculan sus inversas para devolverlas como objeto Color.
- *getTextColor():* Función a la que se le pasa por parámetro el color de fondo del surrogate y devuelve un color que contraste con éste para dárselo al texto.
- *getBackContrast():* Función que calcula y devuelve un color que contraste con la imagen que se le pasa porparámetro.

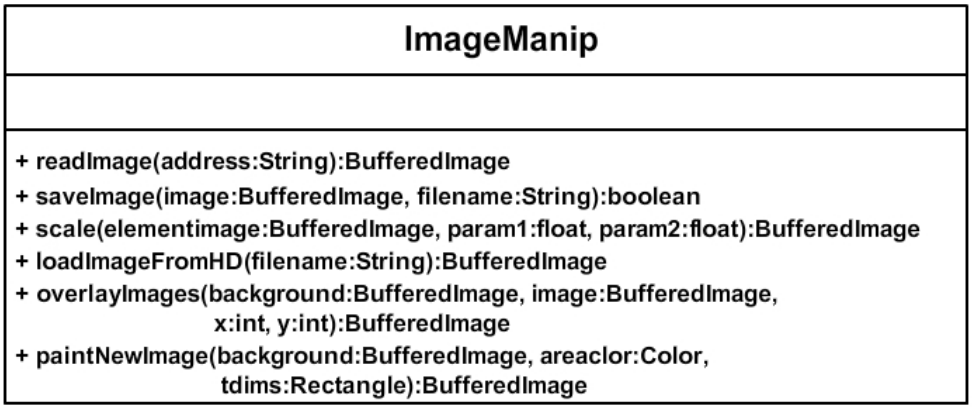

Figura 6.23 Clase ImageManip

ImageManip: en esta clase se han implementado los métodos siguientes:

- *readImage():* Función dedicada a la lectura de imágenes. Se pasa por parámetro la dirección de la imagen a ser leída y devuelve dicha imagen como BufferedImage.
- *saveImage():* Función dedicada al almacenamiento de imágenes. Se pasa por parámetro la imagen a guardar y el nombre con el que la queremos guardar. Devuelve *boolean* cuando se completa dicha acción.
- *scale():* Función dedicada al escalado de una imagen. Se pasan por parámetro la imagen a escalar, un factor de escala para el ancho de la imagen y otro factor de escala para el alto de la imagen. Devuelve la nueva imagen escalada en un objeto BufferedImage.
- *loadImageFromHD():* Función que carga una imagen guardada localmente. Se pasa por parámetro el nombre de la imagen y devuelve la imagen leída como un objeto BufferedImage.
- *overlayImages():* Función que añade una imagen a otra. Se pasa por parámetro la imagen de fondo,la imagen a incluir a éste y la coordenada (x,y). Devuelve el background con la imagen incluida.
- *paintNewImage():* Función que crea una nueva imagen de fondo a la que se agregarán los diferentes elementos del surrogate. Se utilizará en la clase ComposeTxtSurrogate.

6.8 Paquete Flex

Después de haber descrito en los puntos anteriores todo el procedimiento para generar los surrogates mediante el lector de feeds y el generador de surrogates (en formato XML y en formato JPG) en este apartado se describirá el desarrollo de un paquete de clases en ActionScript que tendrá por objetivo ofrecer métodos de lectura, dibujado y manipulación de los surrogates en interfaces que soporten la lectura de archivos XML, como es el caso de interfaces desarrolladas en Flex (por ejemplo, WakiTable. Ver apartado 5.4.4).

En definitiva, este paquete de clases estará orientado a dar el soporte gráfico necesario para dibujar surrogates de manera que se pueda implementar en cualquier otro proyecto fácilmente, haciendo que este sea altamente interoperable.

Antes de pasar a la descripción de las clases que forman dicho paquete, a continuación se mostrará el diagrama UML sobre la estructura del sistema:

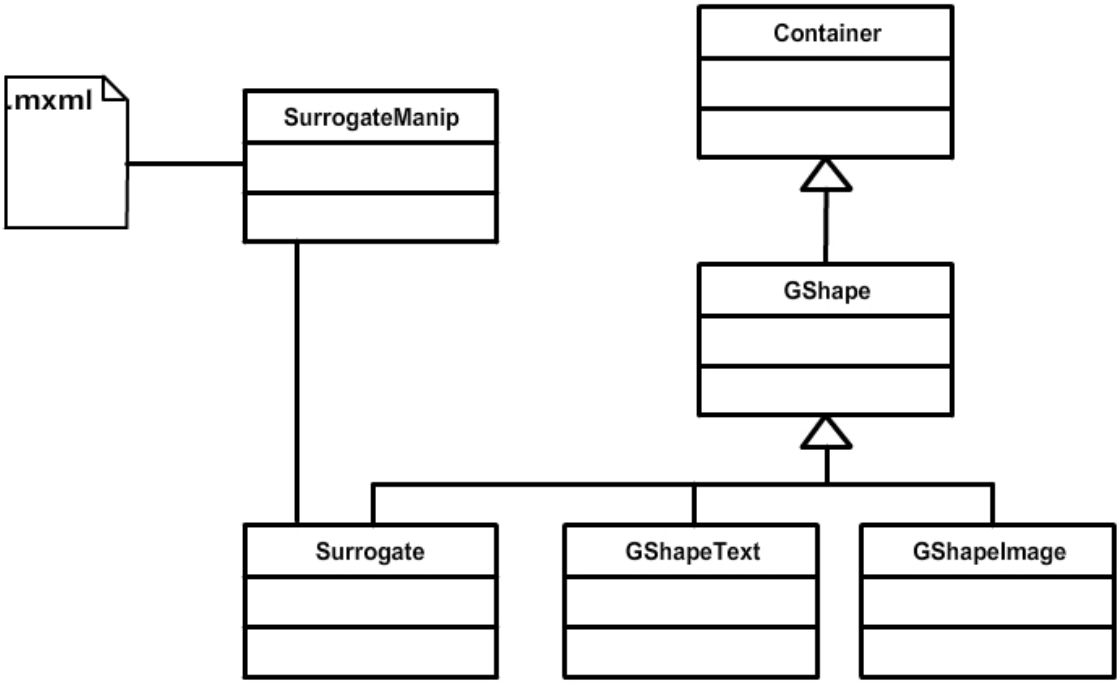

Figura 6.24 Diagrama UML de las clases del paquete Flex

Como se puede observar en la figura 6.24 el programa Flex se tendrá que conectar con la clase SurrogateManip, que se encargará de gestionar los métodos de lectura y dibujado de los surrogates mediante servicios HTTPService. Estos servicios leerán los

archivos de formato XML para que una vez acabada la lectura de éstos lanzar las funciones para el dibujado de los surrogates. (*readCollection() y readSurrogate()*).

6.8.1 Estructuras Gráficas

Como se menciona al inicio del punto 6.6, el objetivo de *SurrogateComposer* es el de generar objetos de tipo TYPE_Element. Estos objetos están estructurados de manera que por una parte almacenan el contenido de una noticia y por otra, almacenan la información sobre la estrucutra gráfica que se tendrá que hacer servir para representar al surrogate. Como se puede observar en la figura 6.25, dicha información es la que se almacena en el nodo *surrogate* del archivo en formato XML y es sobre la que construiremos las clases para poder representarla gráficamente. En esta estructura tendremos toda la información necesaria sobre un surrogate. Desde su estructura, hasta su contenido.

```
<surrogate id="25602816" layout="surrogates.layouts.SurrogateLayout1" margin="5">
      <GShape id="0" type="main" n="4">
            <position x="0" y="0" width="420" height="300" />
            <bkcolor r="52" g="56" b="58" />
            <area x="5" y="5" width="410" height="290" />
      </GShape>
      <GShape id="1" type="txt" n="0">
            <position x="5" y="5" width="410" height="50" />
            <bkcolor r="178" g="178" b="178" />
            <area x="5" y="5" width="400" height="40" />
            <font size="14" font="Dialog" bold="0" italic="0" underline="0"
                  justify="center" color="0" transparency="0" />
            <text margin="5">Titular</text>
      </GShape>
      <GShape id="2" type="txt" n="0">
            ...
      </GShape>
      <GShape id="3" type="img" n="0">
            <position x="5" y="60" width="200" height="205" />
            <bkcolor r="36" g="39" b="40" />
            <area x="5" y="5" width="190" height="195" />
            <img margin="5">C:\newsme\typexmls\media\25602816.jpg</img>
      </GShape>
      <GShape id="4" type="txt" n="0">
            ...
      </GShape>
 </surrogate>
```

Figura 6.25 Estructura gráfica de un surrogate

Como se puede observar en la figura 6.25, el valor *n* del primer GShape, el main, tendrá el valor de 4. Eso querrá decir que el Surrogate estará formado por un Contenedor principal y dentro de él, se incluirán cuatro contenedores más. Los GShapes descritos a continuación de éste.

Las clases que definiremos en esta sección serán las encargadas de recoger toda esta información para ir formando gráficamente estos contenedores y dibujar al fin los surrogates en una interfaz.

GShape: Esta clase heredará de la clase Container, que nos permitirá dibujar contenedores por pantalla una vez le asignemos los valores con los que queremos que se nos muestre. Los atributos básicos por tal que podamos dibujar por pantalla un contenedor serán los siguientes: la posición (coordenadas x e y); el ancho y el alto (width y height); y el color del fondo del contenedor. Los GShape de tipo main instanciarán al constructor de esta clase para dibujar el contenedor que contendrá al resto de GShapes hijos, que serán GShapeText o GShapeImage. Esta clase también implementará un método sencillo para su escalado

GShapeText: Esta clase tendrá un doble objetivo. Dibujar un contenedor y añadir a éste el texto correspondiente al elemento de la noticia (titular, descripción o fecha). Esta clase heredará de GShape para cumplir el primer objetivo de pintar por pantalla el contenedor y a su vez, añadirá un elemento TextArea sobre el que se aplicarán los diferentes atributos especificados en el archivo XML. De esta manera, se dibujará un contenedor con el texto que nos interese. Esta clase también implementará un método sencillo para su escalado.

GShapeImage: Al igual que GShapeText, esta clase tendrá el objetivo de dibujar el contenedor por una parte y añadir la imagen del surrogate por otra. Para ello, también heredará de la clase GShape. A esta clase se le añadirá un elemento *image* y un elemento *VBox* donde colocarla. Esta clase también implementará un método sencillo para su escalado.

Surrogate: Esta clase heredará también de GShape, ya que deberá dibujar por pantalla un contenedor. Este contenedor será el que albergará al resto, es decir, los creados por GShapeText y GShapeImage. Para ello, en esta clase se definirá un vector de GShapes, donde a medida que se vayan creando los diferentes contenedores, se iran incluyendo en uno “padre”.

6.8.2 Pasos para componer un surrogate

Para componer un surrogate se seguirá el siguiente procedimiento:

1. El programa llamará a la función *readSurrogate()*, que en primer lugar leerá los datos especificados en el GShape con id=0 y de tipo main. Una vez leídos los datos correspondientes al GShape padre se instanciará a la clase Surrogate pasando por parámetro los atributos *x, y, width, height* y *bkcolor* leídos.

```
var num_gshapes:int = elegetter.lastResult.element.surrogate.GShape[0].n;
var width:int = elegetter.lastResult.element.surrogate.GShape[0].position.width;
var height:int = elegetter.lastResult.element.surrogate.GShape[0].position.height;
var bkcolor:String = "#"+ elegetter.lastResult.element.surrogate.GShape[0].bkcolor.r +
    elegetter.lastResult.element.surrogate.GShape[0].bkcolor.g +
    elegetter.lastResult.element.surrogate.GShape[0].bkcolor.b;
var x:int = elegetter.lastResult.element.surrogate.GShape[0].position.x;
var y:int = elegetter.lastResult.element.surrogate.GShape[0].position.y;

var surrogate:Surrogate = new Surrogate(num_gshapes, width, height, bkcolor, x, y);
```

Figura 6.26 Lectura de los valores del GShape main

2. El constructor de la clase Surrogate se encargará a su vez de utilizar el de la clase GShape mediante la sentencia *super()*, por lo que ya se habrá formado el Container padre. Una vez realizado este paso, el objeto Surrogate instanciará a su vector de GShapes para ir incluyendo en este, todos los GShapes que formen parte de este Container padre en los pasos siguientes.

3. Si nos fijamos en la primera línea anteriormente descrita, se almacenará en la variable *num_gshapes* el número de elementos hijos que tendrá el Container padre. A continuación se irán formando éstos mediante un bucle. El procedimiento será similar al desarrollado en el primer punto. En primer lugar se leerán los valores de los atributos básicos para formar un container (x, y, width, height y background). En segundo lugar y dependiendo del tipo de GShape que sea (de texto o de imagen) se leerán los valores de los atributos que formarán sus elementos específicos (el del textArea en caso de que el GShape sea de tipo texto o el del VBox en caso que sea imagen). En tercer lugar, se llamará al método *addGshapeText()* o *addGShapeImage()* de las clases GShapeText o GShapeImage según sea el caso.

4. Los métodos *addGshapeText()* y *addGShapeImage()* llamarán a su vez al constructor de su clase correspondiente, que creará el container específico a partir de los valores leídos y pasados por parámetro.

```
public function addGShapeText(gshWidth:int, gshHeight:int, bkcolor:String, gshX:int,
gshY:int, txtText:String, txtWidth:int, txtHeight:int, txtFontColor:int, txtFontSize:int,
txtFontAlign:String, txtFontUnderline:int, txtFontBold:int, txtFontItalic:int, txtFontFont:String):void
{

    var gshape:GshapeText = new GshapeText(gshWidth, gshHeight, bkcolor, gshX, gshY, txtText,
            txtWidth, txtHeight, txtFontColor, txtFontSize, txtFontAlign, txtFontUnderline,
            txtFontBold, txtFontItalic, txtFontFont);
    shapes.push(gshape);
    this.addChild(gshape);
}
```

Figura 6.27 Métodos addGShapeText

Una vez creado este GShape, se incluirá al vector del surrogate

5. Cuando acaba todo el proceso de ir generando los containers hijos del padre, el objeto surrogate estará completado, por lo que la última acción de la función *readSurrogate()* será incluir al surrogate en el vector de surrogates definido en la clase SurrogatesManip para que de esta forma, una vez compuestos todos los surrogates, se pueda enviar este vector de surrogates a la interfaz y se pinten por pantalla.

Un gran beneficio de este paquete de clases es su fácil integración en cualquier proyecto en el que se tenga la intención de representar surrogates desarrollado en un lenguaje que soporte XML. Esto puede darse, por ejemplo, en proyectos que utilicen Java, Javascript o el mismo Flex.

7 - Pruebas y resultados

En este apartado se describirán las dos pruebas que se han realizado para probar que el uso de los surrogates ofrece varios beneficios. La primera prueba tratará a nivel de elementos, es decir, si usuarios no familiarizados con los surrogates son capaces de identificar a éstos como representaciones de noticias, comprobar si los surrogates les ayudan a recordar éstas y otros aspectos relacionados únicamente con su apariencia.

La segunda prueba tratará a nivel de colecciones. Se entrevistarán a los usuarios para que prueben una interfaz que disponga de un conjunto de surrogates y se extraerán sus opiniones al respecto.

Dichas pruebas estarán enfocadas a calcular la eficacia y aceptación de los surrogates por parte de los usuarios.

7.1 Prueba 1

Como decíamos, esta prueba servirá para evaluar a los surrogates a nivel de elementos. El propósito de esta prueba será comprobar si los usuarios identifican a los surrogates como representaciones de noticias o artículos; si éstos son una buena representación y si ayudan a recordarlas.

Para llevar a cabo esta prueba no se precisó de ordenador ya que se consideró que para extraer los resultados esperados la mejor opción era hacerlo en papel para que la familiarización y manipulación con los surrogates fuera más cercana. Por una parte se imprimieron en papel siete surrogates con temáticas diversas y por otra parte sus artículos completos en forma de YahooNews. La prueba se estructuró en tres partes:

1. Se dejó que los usuarios se familiarizaran con los surrogates durante unos minutos, para posteriormente preguntarles que explicaran todo lo que recordarán de cada uno de ellos.
2. Se hizo una prueba consistente en asociar cada surrogate con su artículo. En un principio se dispusieron en una zona de la mesa los surrogates boca abajo, ocultando su contenido, y lo mismo con los artículos en otra zona contigua a los surrogates. El procedimiento a seguir consistió en destapar un surrogate, volver a taparlo, destapar uno de los artículos y volverlo a tapar en caso de no haber coincidencia. Y así sucesivamente hasta conseguir completar todas las parejas. El objetivo de este proceso fue el de comprobar si los usuarios eran capaces de

recordar y asociar los surrogates a sus artículos correspondientes a medida que iban desvelando la información y si este proceso se hacia de una manera ágil y sin errores.

3. Una fase final de preguntas cerradas y abiertas sobre los surrogates, para que los usuarios expusieran sus impresiones acerca de éstos y extraer de esta manera más información sobre lo que pensaban de ellos.

La prueba se realizó a 10 personas de edades y profesiones variadas que se podrían incluir en 3 subgrupos:

- un primer grupo formado por 2 personas jubiladas, dos de ellas habituadas al uso de Internet y que consultan noticias a diario.
- un segundo grupo formado por 4 personas con un margen de edades comprendido entre los 28 y 33 años. Estos usuarios disponen de estudios superiores y están familiarizados con el uso de Internet y leen noticias a diario.
- un tercer grupo formado por otras 4 personas con un margen de edades comprendido entre los 24 y 26 años. Estos usuarios disponen de estudios superiores y están familiarizados con el uso de Internet y leen noticias a diario.

El documento y las noticias que sirvieron para realizar las pruebas se adjuntan en el anexo (Punto 10.2 y 10.3). A continuación se detallarán los resultados recogidos en las diferentes partes de la prueba.

Sobre la primera parte de la prueba cabe destacar que después de analizar los surrogates durante varios minutos, todos los usuarios identificaron los surrogates como representaciones de noticias. A su vez, un hecho común de todos los usuarios es que supieron explicar a grandes rasgos la temática del artículo cuando se les fue preguntando sobre ellos uno a uno. No hubo casos erróneos en ese sentido. Por otra parte, el 90% de los usuarios supieron dar prácticamente todos los detalles concretos que aparecen en los surrogates con temas de más actualidad o relevantes. Esta cifra baja hasta el 60% para temas menos relevantes o que no levantaron tanto interés entre los usuarios.

Aún así, se recogió un dato muy interesante y es que para las noticias más relevantes, los usuarios supieron dar más información que la descrita en el surrogate. Esto es debido a que los usuarios ya sabían información sobre el tema de la noticia previamente y al ver el surrogate lo asociaron a dicho tema ampliando de esta manera la información descrita en el mismo.

Sobre la segunda parte de la prueba, se recogieron datos muy positivos, ya que solo hubo un usuario del primer grupo que cometió un error a la hora de asociar los surrogates a los artículos. El tiempo para ejecutar cada movimiento de asociación fue de de entre unos 8 a 10 segundos aproximadamente. (entiéndase como movimiento de asociación al procedimiento que incluye destapar un surrogate, ver su contenido, volverlo a tapar, levantar un artículo y volverlo a tapar en caso de no producirse asociación).

Sobre la tercera parte de la prueba se extrajeron los siguientes resultados:

- Todos los usuarios opinaron que los surrogates fueron una buena representación del artículo al que representaban.
- Cuatro de los usuarios consideraron la imagen como el elemento más determinante para saber la temática de un surrogate. Otros cuatro usuarios consideraron que el titular y la imagen fueron igual de determinantes, ya que la imagen ayudaba mucho, pero a veces, por si sola no se podía saber el tema del que se hablaba en el surrogate. Finalmente, los dos usuarios restantes consideraron que el titular es el elemento clave para conocer el tema de una noticia.
- Ningún usuario sacrificaría ningún elemento del surrogate.
- Dos de los usuarios encuentra a faltar conocer la categoría de la noticia (si es económica, deportiva, política, etc.)
- Todos los usuarios opinan que la imagen ayuda mucho para conocer el tema de la noticia, ya que les introduce un poco en el contexto de éste, pero no la encuentran determinante ya que por si sola puede significar varias cosas.
- Todos los usuarios creen que los surrogates disponen de un número de palabras adecuado.

- Cuatro de los usuarios opina que la disposición de los elementos del surrogate es la adecuada, mientras que los seis usuarios restantes opinan que la fecha debería estar en una posición más visible, como por ejemplo, encima de la imagen.
- La totalidad de los usuarios creen que los surrogates tienen mucha potencia en cuanto a aplicación se refiere. Expusieron varios ejemplos sobre la aplicación de éstos y prácticamente todos los aplicarían a cualquier producto que se oferte en Internet aplicando la imagen del producto, su nombre, descripción y precio. Otro ejemplo que pusieron fue el de aplicar los surrogates a hoteles y restaurantes sobre GoogleMaps. Para concluir, un último ejemplo que se puso fue el de que los surrogates ejercieran de fichas educativas. En general, todos los usuarios se mostraron satisfechos con los surrogates y los encontraron de gran aplicación.

7.2 Prueba 2

Esta prueba siguió estando orientada a evaluar los surrogates a nivel de elementos, pero esta vez se efectuó con ordenador en vez de en papel como se hizo en la primera prueba. De todas formas, el propósito de la prueba fue el de confirmar si los usuarios identifican a los surrogates como representaciones de noticias o artículos; si éstos son una buena representación de las mismas y si ayudan a recordarlas. Todo esto aplicándolos a otro entorno, en concreto en una interfaz por ordenador, mostrándose varios surrogates al mismo tiempo.

Como decíamos, los surrogates se integraron en un prototipo de interfaz para poder representarlos por ordenador. Para ello se utilizó el paquete Flex descrito en el punto sobre implementación. En concreto se utilizó la interfaz WakiTable descrita en el punto (5.4.4). Esta interfaz consiste en un espacio donde colocar los surrogates en una matriz de tres filas por tres columnas. Como únicos elementos de control de la interfaz se dispondrán dos botones. Uno para que se muestren los nueve surrogates siguientes en la matriz sustituyendo a los ya colocados y otro para volver a los anteriores. Dicha interfaz se nutrió de un número elevado de surrogates especificados en un archivo *collection.xml* previamente generados por *SurrogateComposer* para que la navegación por la misma fuera suficientemente extensa. Las noticias ofrecidas en la interfaz fueron de temas actuales y diversos.

La prueba se estructuró en cuatro partes:

1. Una primera parte donde se dejó que los usuarios se familiarizaran con la interfaz durante unos minutos.
2. Esta segunda parte de la prueba consistió en hacer que los usuarios navegaran con detenimiento por las páginas de la interfaz. Al cabo de un par de páginas exploradas se les dio la orden de detenerse y a continuación se les preguntó si recordaban haber visto tres surrogates específicos y que dieran todos los detalles posibles sobre éstos. El objetivo de dicha prueba fue para comprobar si los usuarios eran capaces de recordar las noticias que habían visto con anterioridad.
3. En la tercera parte de la prueba, se les ordenó a los usuarios que buscaran un surrogate especifico aportándoles un dato determinado. Para realizar dicha prueba se utilizó el método de hablar en voz alta para poder entender cómo realizaron la búsqueda. El objetivo de este proceso fue el de comprobar si los usuarios eran capaces encontrar rápidamente un surrogate y el método para conseguirlo.
4. Una fase final con preguntas abiertas para que los usuarios expusieran sus impresiones acerca de los surrogates.

Antes de pasar a mostrar los resultados de la prueba cabe destacar que ésta se realizó a un representante de cada grupo especificado en el punto anterior. Es decir, a un usuario jubilado y habituado al uso del ordenador; un usuario de 31 años familiarizado con el uso de Internet y un tercer usuario de 25 años habituado también al uso de Internet.

Los resultados de la prueba fueron los siguientes:

Sobre la primera parte de la prueba, la destinada a la familiarización de los surrogates, cabe destacar el hecho que todos los usuarios identificaron a los surrogates como los elementos en papel que vieron en la primera prueba. De esta manera se probó que los surrogates se pueden identificar en una interfaz gráfica.

Sobre la segunda parte de la prueba, decir que dos de los tres usuarios supieron explicar las noticias con todo detalle, mientras que el usuario restante sólo supo explicar el tema general de las mismas aportando algún detalle. Por lo tanto, los resultados de

esta prueba se podrían considerar como satisfactorios ya que se confirma que los surrogates ayudan a recordar las noticias.

Los resultados de la tercera prueba fueron muy positivos, ya que todos los usuarios supieron encontrar los surrogates especificados con rapidez. De hecho, para los surrogates a buscar que no estaban disponibles en la primera página, los usuarios saltaron de página prácticamente al instante, ya que recordaban que esos surrogates los habían leído en páginas posteriores. Cabe comentar que los surrogates con imagen asociada se encontraron más rápidamente que los que no, siendo los propios usuarios los que dijeron que la imagen ayudaba mucho a recordar e identificar el surrogate.

En la cuarta parte de la prueba y tras discutir con ellos sobre la interfaz y los surrogates se comentaron los aspectos siguientes:

- Consideraron adecuado el número de surrogates mostrados por pantalla.
- Hubo críticas sobre los colores de algunos surrogates ya que el contraste del texto con el fondo no era el suficiente y no se les permitía leer el texto de forma óptima, teniéndose que esforzar en exceso.
- También comentaron que los surrogates sin imagen asociada resultaban menos llamativos, haciendo perder el interés por éstos.
- También hubo críticas sobre algún componente que se presentaba cortado, dificultando de esta manera su lectura.

Aún así, los usuarios aseguraron tener buenas sensaciones con la aplicación y les gustó esta nueva manera de navegar por noticias. La encontraron innovadora y fácil de usar, por lo que se mostraron interesados en ésta desde un primer momento.

8 - Conclusiones

8.1 Conclusiones

Una vez llegados a este punto del trabajo, podemos decir que hemos desarrollado un sistema complejo pero eficaz para generar surrogates. Hemos podido disfrutar de las ventajas que ofrece la Web 2.0 y su filosofía de la compartición de la información, ya que sin la sindicación de noticias, no hubiera sido posible realizarlo. El objetivo del proyecto no era generar surrogates por el mero hecho de generarlos utilizando las herramientas disponibles, sino que el verdadero propósito de este proyecto ha sido desarrollar todo un sistema de generación de éstos con el objetivo claro de facilitar al usuario la tarea de explorar por noticias mediante éstos, ya que han sido pensados para llevar a cabo esta tarea correctamente.
Tras mantener un feedback con los usuarios durante las pruebas hemos comprobado que los surrogates aportan los siguientes puntos:

- Los surrogates son una buena manera de representar noticias. El número de elementos que componen un surrogate se considera adecuado y la información mostrada en ésta se considera la justa para llegar a mostrar interés por la noticia completa.
- Se han identificado a los surrogates como sustitutos de las noticias. Es decir, los usuarios han detectado la función de "puerta de enlace" de los surrogates. Han sabido entender que eran una muestra de una noticia y que ésta se podía ampliar mediante interacción con ellos.
- Ayudan a **recordar** las noticias. Al presentar una información muy representativa de la noticia, los usuarios se acuerdan de los detalles y cifras mencionados en ésta. Además, si la noticia es muy concreta sobre un tema que ya se conoce, los usuarios **reconocen** y asocian dicha noticia a ese tema.
- Facilitan la búsqueda de una noticia en una colección. La imagen, junto con el titular, se consideran elementos clave para facilitar la búsqueda de un artículo sobre un tema en concreto dentro de una colección de surrogates.
- Gran potencial en cuanto a aplicación se refiere. Se considera que los surrogates son una buena manera de representar no sólo noticias, sino que también cualquier producto o elemento.

- La interacción con los surrogates se considera sencilla, poco costosa de aprender y muy estimable cuando el único objetivo de un usuario es informarse.

Una vez hemos llegado a estas conclusiones sobre los surrogates, queda decir que tras todo este proceso se ha conseguido buena parte de los beneficios que esperábamos conseguir con los surrogates, por lo que este proyecto ha sido muy ventajoso para reforzar, si cabe, todas las afirmaciones que se han dado sobre ellos.

También se ha comprobado la potencia que ofrece el formato XML ya que gracias a éste es posible manipular información ajena para conseguir resultados innovadores mediante la sindicación de noticias.
Como resultado de esta "filosofía" basada en la compartición de la información, hemos sido capaces de aplicar un concepto nuevo en cuanto a la representación de la información se refiere.

Un último punto a comentar es que se ha conseguido desarrollar un paquete de clases en ActionScript "universal". Dicho paquete se ha implementado con éxito en varios proyectos fácilmente, por lo que se ha conseguido definir un paquete bien formado e interoperable para dar soporte gráfico a los surrogates.

8.2 Posibles Limitaciones

Por otra parte, los resultados de este proyecto pueden presentar ciertas limitaciones que a priori se escapen de nuestro control:

1. Que la información no esté bien definida en la fuente RSS de la que nos nutrimos. Por ejemplo, si el contenido de los *feeds* que hacemos servir contienen errores ortográficos, éstos se mostrarán en los surrogates. También puede darse el caso que el contenido este mal etiquetado, por lo que en la descripción podemos encontrar el título y demás problemas de esta índole.

2. Que se coloque una cantidad enorme de surrogates en una interfaz. Ésta perderá la eficacia y las capacidades cognitivas del usuario se desbordarán por lo que no se conseguirá obtener los beneficios esperados de los surrogates en dicha exploración.

8.3 - Trabajo futuro

Todo y haber conseguido desarrollar un método de generación de surrogates bastante eficaz al que poder dar a escoger varias opciones de apariencia (o layout), aún queda mucho trabajo por hacer para el desarrollo de surrogates. A continuación se muestran algunos puntos sobre los que se podrían tratar de mejorar los métodos y clases ya implementados o incluso puntos nuevos a adjuntar al proyecto:

- Mejorar los métodos implementados en las clases sobre la gestión del color y el tamaño del texto, ya que a veces los surrogates presentan colores no muy acordes con la imagen que tienen asociada y/o el texto no contrasta demasiado con el fondo dificultando así su lectura.
- Desarrollo de nuevos métodos para mejorar la interacción con los surrogates. Por ejemplo:
  - Métodos para asignar una transparencia determinada a un surrogate según la fecha de la noticia.
  - Métodos para asignar un color determinado según el tema o la fuente web de de la noticia representada en el surrogate
- Desarrollo de nuevas clases para soportar video en los surrogates
- Mejorar el diseño gráfico de los surrogates para hacerlos más atractivos.

9 - Bibliografía y referencias

[1] George A. Miller

The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information, (1956), *The Psychological Review*.

The cognitive revolution: a historical perspective.

[2] Andruid Kerne

The Collage Machine

http://ecologylab.cse.tamu.edu/research/publications/kerneCast01Proceedings241-244.pdf

[3] Ley de Fitts

http://es.wikipedia.org/wiki/Ley_de_Fitts

[4] Ludwig Mies van der Rohe

http://es.wikipedia.org/wiki/Ludwig_Mies_van_der_Rohe

[5] RSS

http://es.wikipedia.org/wiki/RSS

[6] XML

http://es.wikipedia.org/wiki/XML

http://www.whatisrss.com/

[7] Mashup

http://es.wikipedia.org/wiki/Mashup_(aplicaci%C3%B3n_web_h%C3%ADbrida)

http://www.webmashup.com

[8] W3C

http://www.w3c.org

[9] OJD Interactiva

Benchmark de Internet. Auditoria online.

http://www.ojdinteractiva.es/

[10] Alexa

The Web Information Company

http://www.alexa.com/

[11] Yahoo News

http://news.yahoo.com/

10 - Anexos

10.1 Anexo 1. Ejemplos de las estructuras de datos usadas en los archivos

Ejemplo de resultado de un archivo que almacena la estructura TYPE_XMLElement:

```
<?xml version="1.0" encoding="UTF-8"?>
<xml_element id="107396496" url="http://us.rd.yahoo.com/dailynews/rss/world/
      *http://news.yahoo.com/s/ap/20090616/ap_on_re_la_am_ca/brazil_plane"
      source="http://rss.news.yahoo.com/rss/world/">
<text>
      <pubdate>mar, 16 jun 2009 00:00:00</pubdate>
      <text_title>French: Search for Flight 447 to continue  (AP) </text_title>
      <text_content>AP - The arduous mid-Atlantic search for the remains of Air France...
      <text_formatted> content="The arduous mid-Atlantic search for the remains of Air France...
 </text>
 <media type="image" url="C:\newsme\typexmls\media\107396496.jpg">Workers unload ...</media>
</xml_element>
```

Figura Anexo 1. Contenido real del archivo xmlelement_107396496.xml

Ejemplo de resultado de un archivo que almacena la estructura TYPE_Element:

```
<?xml version="1.0" encoding="UTF-8"?>
<element id="107396496" url="http://us.rd.yahoo.com/dailynews/rss/world/
        *http://news.yahoo.com/s/ap/20090616/ap_on_re_la_am_ca/brazil_plane" source="http://rss.news.yahoo.com/rss/world/">
    <concepts />
    <text>
        <pubdate>mar, 16 jun 2009 00:00:00</pubdate>
        <text_title>French: Search for Flight 447 to continue  (AP) </text_title>
        <text_content>AP - The arduous mid-Atlantic search for the remains of Air France Flight 447 ...</text_content>
        <text_formatted></text_formatted>
    </text>
    <media type="image" url="C:\newsme\typexmls\media\107396496.jpg">
        Workers unload debris, belonging to crashed Air France flight 447...
    </media>
    <surrogate id="107396496" layout="surrogates.layouts.SurrogateLayout2" margin="5">
        <GShape id="0" type="main" n="4">
            <position x="0" y="0" width="300" height="420" />
            <bkcolor r="35" g="31" b="27" />
            <area x="5" y="5" width="290" height="410" />
        </GShape>
        <GShape id="1" type="txt" n="0">
            <position x="5" y="5" width="290" height="65" />
            <bkcolor r="35" g="31" b="27" />
            <area x="5" y="5" width="280" height="55" />
            <font size="14" font="Dialog" bold="0" italic="0" underline="0" justify="center" color="473e36" transparency="100" />
            <text margin="5">French: Search for Flight 447 to continue  (AP) </text>
        </GShape>
        <GShape id="2" type="txt" n="0">
            <position x="5" y="255" width="290" height="130" />
            <bkcolor r="35" g="31" b="27" />
            <area x="5" y="5" width="280" height="120" />
            <font size="12" font="Dialog" bold="0" italic="0" underline="0" justify="center" color="473e36" transparency="100" />
            <text margin="5">AP - The arduous mid-Atlantic search for the remains of Air France Flight 447 ...</text>
        </GShape>
        <GShape id="3" type="img" n="0">
            <position x="5" y="75" width="290" height="175" />
            <bkcolor r="35" g="31" b="27" />
            <area x="5" y="5" width="280" height="165" />
            <img margin="5">C:\newsme\typexmls\media\107396496.jpg</img>
        </GShape>
        <GShape id="4" type="txt" n="0">
            <position x="5" y="390" width="290" height="35" />
            <bkcolor r="35" g="31" b="27" />
            <area x="5" y="5" width="280" height="25" />
            <font size="12" font="Dialog" bold="0" italic="0" underline="0" justify="center" color="473e36" transparency="100" />
            <text margin="5">16 jun 2009</text>
        </GShape>
    </surrogate>
</element>
```

10.2 Anexo 2. Cuestionario de la prueba 1

Experimento 1. Surrogates a nivel de elementos

Cuestionario

Fase 1: Familiarización a los surrogates

En esta fase mostraremos los 7 surrogates y se efectuarán las siguientes preguntas:

- ¿A qué crees que representan estos elementos?
    - Respuesta esperada: articulo/noticia. En caso de error explicar qué es.

- ¿De qué crees que habla el artículo/noticia?
    - Comparar puntos que menciona entrevistado con contenido del artículo y sumar coincidencias importantes.

Fase 2: Prueba Memory

En esta fase enseñaremos los artículos asociados a los surrogates uno a uno y después los propios surrogates. Los pondremos boca abajo y el usuario tendrá que ir asociarlos como si se tratara del juego Memory.

Comprobar cuanto tardan y número de errores

Fase 3: Preguntas sobre los surrogates

1. ¿Crees que los surrogates son una buena representación del artículo al que representan?
2. ¿Qué elemento crees que es el más determinante para una buena asociación?
3. ¿Sacrificarías algún elemento?
4. ¿Añadirías otro elemento? ¿Cuál?
5. ¿La imagen ayuda mucho o poco para saber la temática del artículo?
6. ¿Crees que hay muchas palabras o pocas para poder saber de qué va la noticia?
7. ¿Consideras que tiene una disposición adecuada el surrogate?
8. ¿Consideras que es una buena representación del artículo?
9. En tu opinión, ¿para qué pueden servir los surrogates?

10.3 Anexo 3. Surrogates y artículos usados en las pruebas

En este anexo mostraremos los surrogates que se hicieron servir en las pruebas: