inclusite: Herramienta software para la accesibilidad web.

Fabián Caballero Sandoval

Trabajo Fin de Master (TFM), curso 2012-2013
Master en Ingeniería del Software, Métodos formales y Sistemas de Información
Universidad Politécnica de Valencia – UPV
Cno. de Vera s/n, 46071 Valencia, España
facasan@masters.upv.es

Director: Joan J. Fons i Cors

Resumen

Existe una cantidad de personas con diferentes tipos de discapacidad (visual, cognitiva, física, etc.), las cuales no tienen acceso a la información en internet, ya sea por falta de que el usuario no tiene una herramienta software que le facilite ese acceso o por falta de que el contenido no sea accesible para dichos usuarios.

Algunas aplicaciones web intentan hacer lo más accesible posible su contenido por medio de estándares creados por la W3C [1] (World Wide Web Consortium), pero a veces con este tipo de estándares no llega a ser suficiente para que un usuario con algún tipo de discapacidad pueda navegar sobre esos sitios web.

La necesidad de que la Web sea universal y accesible por cualquier persona está presente desde sus inicios, ya que era un requisito contemplado en su diseño por su creador Tim Berners-Lee:

> *"The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect".*
>
> El poder de la Web está en su universalidad. El acceso por cualquier persona, independientemente de la discapacidad que presente es un aspecto esencial.
>
> Tim Berners-Lee, Director del W3C e inventor de la World Wide Web

Cuando lo pensamos en un primer momento quizás no nos demos cuenta del enorme potencial que puede tener la Web para las personas con discapacidad. Por ejemplo, antes de Internet, ¿cómo leían los periódicos las personas ciegas? La mayaría no lo hacían ya que las cintas de audio o los documentos en Braille eran caros y muy voluminosos. Hoy en día, la mayoría de periódicos publican sus contenidos para que puedan ser leídos por "lectores de pantalla".

La Web ofrece oportunidades sin precedentes a las personas con capacidades limitadas, pero si no se lleva cuidado, la falta de accesibilidad creará graves barreras que impedirán su uso.

A pesar del gran potencial que supone la Web para las personas con discapacidad, este potencial está aún en gran medida sin realizar.

En algunas páginas Web únicamente es posible navegar mediante el ratón, y sólo un pequeño porcentaje de contenido de vídeo o multimedia ha sido subtitulado para

sordos. Si el contenido de Internet sólo es accesible mediante el uso de un ratón, ¿qué hace la gente que no puede utilizar un ratón? ¿Y si los desarrolladores web utilizan gráficos en lugar de texto? Si los lectores de pantalla sólo pueden leer el texto, ¿cómo leen los gráficos a las personas ciegas?

Para solventar en gran parte estos problemas se ha creado una arquitectura y una herramienta software basada en un estándar creado por la W3C que estructura la web de un modo semántico para que se pueda navegar sobre su contenido por medio de interfaces adaptadas a cada tipo de discapacidad, en este caso el WAI-ARIA [2] es el estándar en el que se basa el proyecto para dar las pautas de cómo se debe estructurar la web e INCLUSITE es la herramienta que se presta como interfaz entre el usuario y la aplicación web.

Este proyecto pertenece a la empresa Company for Software and Development, S.A. (CSD) y yo hice parte del equipo de desarrollo. CSD ha obtenido subvenciones y ayudas públicas para la evolución de la solución y el desarrollo de nuevas funcionalidades (Fondos Europeos de Desarrollo Regional, Plan Avanzados, etc.).

ABSTRACT

There are a number of people with different types of disabilities (visual, cognitive, physical, etc..), Which do not have access to information on the Internet, either for lack of a user does not have a software tool that facilitates the access or lack of content is not accessible to those users.

Some web applications try to make as accessible as possible through content standards created by the W3C [1] (World Wide Web Consortium), but sometimes with such standards become not enough for a user with some kind of disabilities can surf on these websites.

The need for the Web is universal and accessible by anyone present from the beginning, as it was a requirement set forth in its design by its creator Tim Berners-Lee:

"The power of the Web is In Its universality. Access by everyone Regardless of disability is an essential aspect. "

Tim Berners-Lee, W3C Director and inventor of the World Wide Web

When we think at first we may not realize the enormous potential impact of the Web for people with disabilities. For example, before the Internet, read newspapers how blind people? Almost all did not because the audiotape or Braille documents were expensive and very bulky. Today, most newspapers publish their content to be read by "screen readers".

The Web offers unprecedented opportunities for people with limited capabilities, but does not take care, lack of accessibility will create serious barriers that prevent their use.

Despite the great potential that the Web for people with disabilities, this potential is still largely unrealized.

In some Web pages may only navigate using the mouse, and only a small percentage of video or multimedia content has been subtitled for the deaf. If the Internet content is only accessible by using a mouse, what do people who cannot use a mouse? What if web developers use graphics instead of text? If screen readers can only read text, how to read graphs for blind people?

To solve these problems largely have created an architecture and a software tool based on a standard created by the W3C web structure of a semantic way so that you can browse its content by means of interfaces tailored to each type of disability In this case the WAI-ARIA [2] is the standard that underpins the project to provide guidelines of how to structure the web and INCLUSITE is the tool that is provided as an interface between the user and the Web application.

This project belongs to Company for Software and Development, S.A. (CSD) and I became part of the development team. CSD has obtained subsidies and support for the development of the settlement and development of new functionalities (European Funds for Regional Development, Advanced Plan, etc.).

Tabla de contenido

Lista de Figuras

Lista de Tablas

1. Introducción

La accesibilidad web es la frase base para el inicio del trabajo de la integración de discapacitados al mundo del internet, se han creado pautas y recomendaciones por la Web Accesibility Initiative - WAI[1], estas recomendaciones están agrupadas de acuerdo a cuatro principios, perceptibilidad, operabilidad, comprensibilidad y robustez. Cada pauta de estas tiene puntos de verificación, y a su vez cada punto tiene asignada una prioridad la cual indica el impacto que tiene dicho punto sobre la accesibilidad en un sitio web.

La web se ha convertido en una red universal de conocimiento que abarca un amplio ámbito en cuanto se refiere a la adquisición y tratamiento de la información. Sin embargo, debido a diferentes motivos, hoy en día todavía existen limitaciones que no permiten acceder a esa información a una gran cantidad de personas, entre ellas, personas con discapacidad y personas de la tercera edad, aunque existan estándares que busquen esa accesibilidad e incluso que la ley obligue a los dueños de los sitios web a usarlos.

Otro concepto en relación a la accesibilidad son los principios de diseño para todos[2], que tienen como objetivo productos y entornos de fácil uso para la mayor cantidad posible de personas, sin necesidad de adaptarlos o rediseñarlos para casos especiales. Esos principios son:

- Igualdad de uso.
- Flexibilidad.
- Simple e intuitivo.
- Información fácil de percibir.
- Tolerancia a errores.
- Minimizar el esfuerzo físico.
- Dimensiones apropiadas.

También hay que destacar que aplicar los requerimientos de accesibilidad conlleva ventajas, además de permitir que personas con discapacidad puedan acceder a los contenidos web, ventajas tales como; simplifica el desarrollo, mejora la indexación en

[1] *WAI, Web Accessibility Initiative, http://www.w3.org/WAI/.*
[2] *http://www.sidar.org/recur/desdi/usable/dudt.php*

los buscadores ya que la información está más ordenada, aumenta la usabilidad gracias a su simplicidad, etc.

A la final se puede ver la accesibilidad web, no como una cantidad de requisitos a cumplir para un colectivo en concreto, sino como ayudas para la mejora de la calidad de un sitio web, ya que aportan beneficios y permiten que se esté preparado para futuras tecnologías web.

1.1. Motivación.

Actualmente hay millones de personas con discapacidad que no pueden hacer uso de la web, la mayoría de las aplicaciones web existentes presentan barreras de accesibilidad, lo que conlleva a la situación de no ofrecer la posibilidad de acceder a la información a aquellos usuarios con alguna discapacidad. Entonces, cuanto más sitios web accesibles y software estén disponibles, la cantidad de personas con discapacidad que puedan usar la web aumentará.

Crear un sitio web accesible puede ser tanto sencillo como difícil, ya que depende de muchos factores como por ejemplo, el tamaño y la complejidad del sitio, el contenido, etc. y además ofrece beneficios, como se ha mencionado antes, se mejora la indexación del contenido.

Gran parte de las características de accesibilidad se pueden implementar de forma sencilla sobre un sitio web, si se planean desde el inicio del desarrollo, pero ¿Qué pasa con aquellos sitios web que han sido desarrollados sin aplicar términos de accesibilidad sobre ellos? La modificación de un sitio web para hacerlo accesible puede ser muy laborioso y podría requerir un sobre esfuerzo, lo cual llevaría a un equipo de desarrollo decidir crear de nuevo el sitio web.

Por lo tanto la motivación es crear un herramienta software que sobre un sitio web ya desarrollado que no ofrezca accesibilidad o inclusive si lo ofrece, poner una capa de accesibilidad, sin necesidad de hacer modificaciones sobre el código fuente[3], haciendo entonces que la cantidad de barreras para acceder a la información sea más pequeña, permitiendo así la integración al mundo del internet a millones de personas.

[3] Con código fuente se hace referencia al lenguaje de programación sobre el cual fue desarrollado el sitio web, es decir, JSP, .NET, PHP, etc.

1.2. El problema.

Otro aspecto importante a considerar al respecto con accesibilidad son las ayudas técnicas, las cuales son elementos, software o hardware, que permiten usar alguna herramienta informática, en el contexto de este trabajo son elementos que permiten la comunicación entre un usuario con discapacidad y un sitio web, por ejemplo los lectores de pantalla, son una ayuda técnica que da la posibilidad a aquellas personas con discapacidad visual interactuar con un ordenador.

Entonces crear ayudas técnicas específicas para dar acceso a contenidos web, hace que la accesibilidad acarre beneficios sobre personas que presentan algún grado de discapacidad o incluso personas de tercera edad. Beneficios tales como:

- Los usuarios podrán acceder a una gran fuente de información y beneficios, que permitirá mejorar su calidad de vida.
- Se podría potenciar el teletrabajo, así los usuarios con discapacidad podrían introducirse al mundo laboral desde sus casas.
- Permitiría mejorar la claridad y la velocidad de navegación por el contenido del sitio web.

1.3. Solución

La herramienta **inclusite**, es una solución software que se basa 100% en la web para ofrecer accesibilidad, se trata de una interfaz inteligente que se amolda a las necesidades específicas de cada usuario permitiendo distintas modalidades de navegación dentro del contenido de un sitio web.

Para que un usuario use inclusite sobre una página web, no es necesario que tenga instalado alguna ayuda técnica en su ordenador, ya que inclusite actúa como una ayuda técnica, además intenta paliar las carencias perceptivas o motrices haciendo uso de los medios presentes en cualquier ordenador, como el micrófono o los altavoces.

Inclusite trabaja simultáneamente sobre los dos grandes frentes en los que la web actual presenta los mayores inconvenientes en términos de accesibilidad; la estructuración del contenido y la presentación del mismo.

Inclusite basa su funcionamiento en el añadido de atributos sobre las etiquetas HTML del sitio web, que son asignados por medio de reglas específicas para cada sitio, que se ejecutan sobre un analizador de HTML. El principal atributo es aquel llamado **role**, el cual define los elementos de una página web.

Luego de la ejecución de las reglas sobre el sitio web, el usuario dispondrá de la posibilidad de visualizar la web en su forma original o seleccionar la interfaz que se adapte a sus necesidades, creando así una capa de accesibilidad sobre el contenido permitiendo al usuario interactuar con él.

1.4. Organización de la tesis

El documento está estructurado en 9 capítulos de la siguiente manera:

- En el capítulo 2 se presenta el estado del arte sobre tecnología software que sirve como ayudas técnicas para uso de sistemas informáticos, seguido de una explicación del actual estándar de accesibilidad de la WAI.
- En el capítulo 3 se presenta el contexto tecnológico, se describen las diferentes tecnologías usadas para el desarrollo de la herramienta.
- En el capítulo 4 se muestran las fases de análisis y diseño para la arquitectura de la aplicación desarrollada. Se describe de una forma general el cómo debe funcionar la herramienta, se presenta una serie de diagramas de casos de uso sobre unas de las partes a desarrollar, y por último se explica cómo se aplica el estándar WAI-ARIA sobre la herramienta.
- En el capítulo 5 se describe el proceso de desarrollo de una de las interfaces con la cual inclusite permite a usuarios con discapacidad navegar por el contenido, también se habla de una interfaz que permite interactuar con formularios, se muestra una serie de tablas que describen las clases desarrolladas para esas funcionalidades.
- En capítulo 6 se da un ejemplo de cómo es el proceso de implantación de inclusite en un sitio web.
- En el capítulo 7 se describen las conclusiones obtenidas después del proceso de implementación de inclusite.
- En el capítulo 8 se habla sobre cosas a hacer y mejorar sobre la herramienta.
- En el capítulo 9 la biografía seguido de los anexos, que contienen el diagrama de clases con el cual se implemento el analizador de roles, y el código fuente

de cada una de las reglas que se uso para adaptar la página web de ejemplo del capítulo 6.

2. Estado del Arte

La frase “acortar la brecha digital” se queda en eso, en una frase si no nos damos cuenta de que en el mundo se está haciendo todo lo posible por tratar de que todas la tecnologías existentes; móviles, ordenadores, televisores, consolas de video juegos, etc. Puedan ser usadas por cualquier persona, y de eso se encarga el software.

Existen lectores de pantalla, ratones y teclados virtuales, incluso sistemas operativos adaptados para ser usados por personas discapacitadas. Actualmente la mayoría de empresas que están en el sector del desarrollo de software quieren acotar con aplicaciones de este tipo cada una de las discapacidades que puedan existir.

Si se hace referencia al ámbito del uso del ordenador, un gran número de usuarios con discapacidad son invidentes, por eso en general existe una buena cantidad de software para ellos, el cual les permite saber lo que hay en sus pantallas (Jaws, ORCA, etc.) mientras ésta es leída, y en relación con la accesibilidad web este tipo de aplicaciones necesitan por parte de las páginas web, que estén diseñadas para que permitan que los usuarios que usan estas tecnologías les puedan explotar sus funcionalidades dentro de este contexto.

A continuación se hablará un poco sobre este tipo de software que han servido en parte como inspiración para este proyecto, también respecto a la accesibilidad web y todo lo que conlleva en relación a ella como fundamento para desarrollar las funcionalidades de la aplicación final.

2.1. JAWS (Job Access With Speech)

Es un lector de pantallas que permite a los usuarios con algún tipo de discapacidad visual hacer accesible su ordenador el debe tener como sistema operativo alguna versión de Microsoft Windows [4], para tal finalidad. JAWS [3] convierte el contenido de la pantalla en sonido, permitiendo así al usuario acceder y navegar sobre él.

Para acceder a las diferentes funciones de JAWS se usa el teclado numérico de un teclado convencional, usando distintas combinaciones de estas teclas se puede controlar el contenido o las zonas de la pantalla que se quiera JAWS lea. Entre dichos

comandos está la posibilidad de navegar entre líneas o entre caracteres dentro del contenido, o incluso entre oraciones o párrafos.

Otra funcionalidad que caracteriza a este software es el hecho de que permite el acceso a internet a través del navegador internet explorer [5], cuando éste inicia te permite acceder a “tus favoritos” y escoger la página a la cual quieres entrar, además JAWS después de haber cargado la web anuncia si dentro del contenido hay iFrames*, enlaces, formularios, y luego lo empieza a leer.

Para la navegación dentro de la página web, JAWS da una serie de comandos que permiten navegar de una forma fácil dentro del contenido, por ejemplo **T** to para acceder a las tablas, **F** para los formularios, **H** para ir al encabezado y cosas así.

Los formularios en particular no son muy accesibles, pero JAWS permite interactuar con ellos, el usuario puede moverse a través de los diferentes campos y tipos, rellenarlos y seleccionar en el caso de los comboboxes o checkboxes, e incluso personalizarlos para cuando el usuario vuelva a usarlos. Esto último se hace por medio de los “PlaceMarkers”; que da al usuario la posibilidad de agregar dentro de una web, documento o incluso un formulario dentro de una web, las zonas que él considere relevantes, éstas pueden ser definidas como temporales o permanentes.

2.2. ORCA

Orca[6] es un software libre, flexible, extensible y eficaz para personas con discapacidad visual que usan el sistema operativo Linux, éste combina el uso del habla, braille* y la magnificación, haciendo que aquellas aplicaciones y herramientas que soportan AT-SPI* se vuelvan accesibles. Orca es desarrollado y liderado por el Accessibility Program Office de Sun Microsystem, Inc y la contribución de varios miembros de la comunidad de desarrolladores.

Cuando se dice que este software hace uso de la magnificación de pantalla hace referencia a adaptar la pantalla para el tipo de discapacidad visual que tenga el usuario, obviamente con discapacidad visual no se enfoca solo en las personas con ceguera total, si no también parcial o puede darse también el caso de daltónicos. Se permite aumentar el tamaño de la letra, el color, e incluso tiene configuraciones por defecto para determinados tipos de invalidez.

Otra característica de éste nombrada anteriormente es su extensibilidad, hay aplicaciones que se pueden instalar en el sistema operativo o potencial las funcionalidades del software, un ejemplo es el navegador web Firefox; él permite con el uso de Orca acceder al contenido web con combinaciones de teclas tal cual como lo permite hacer JAWS, deja navegar tabular mente entre los contenidos resaltados como importantes (encabezado, banners, formularios, enlaces, etc.) e incluso asignarles combinaciones de teclas para acceder directamente a ellos, obviamente que serán combinaciones que no afecten el funcionamiento normal de Orca.

A continuación una tabla de las diferentes funcionalidades de Orca junto con firefox:

Keystroke	**Action**
h, Shift+h	Jump to next and previous heading
1, Shift+1	Jump to next and previous heading level 1 (2...6 accomplish similar things)
i, Shift+i	Jump to next and previous list item
l, Shift+l	Jump to next and previous list
o, Shift+o	Jump to next and previous large object
q, Shift+q	Jump to next/previous blockquote
r, Shift+r	Jump to next and previous live region
t, Shift+t	Jump to next and previous table
u, Shift+u	Jump to next and previous unvisited link
v, Shift+v	Jump to next and previous visited link
y	Jump to last live region that was announced
Shift+Alt+Arrow	When in a table, jump to table cell in given arrow direction (Up, Down, Left, Right)
Shift+Alt+Home, Shift+Alt+End	When in a table, jump to first/last table cell
\	Advance the politeness level for the current live region
Shift+\	Set all live regions on page to politeness of 'off'
Orca+Fn, where n=1-9	Review up to the ninth previously announced live region

	message
unassigned	Goes to next/previous landmark
Alt_L+down	Expand combo box
ctrl+home/ctrl+end	Goto top/bottom of document
ctrl+left/right	goto previous/next Word
orcaKey+left/right	goto previous/next object
orcaKey+tab / orcaKey+shift+tab	goto next/previous form field
orcaKey+z	toggle structural navigation

Tabla 1: combinación de teclas para ejecutar comandos básicos con Orca dentro de firefox [7].

2.3. Accesibilidad Web

La accesibilidad es el estatus que se le da a un objeto, lugar o servicio que puede ser usado por cualquier persona, en este caso se hablará de la web, lo que lleva a una definición más concreta; cualquier persona con algún tipo de discapacidad podrá hacer uso de una aplicación web.

Para que una aplicación web sea accesible tiene que poseer un diseño basado en unas especificaciones que permitan al usuario discapacitado interactuar de una forma fluida con el contenido de la web e incluso aportar contenido, con esta idea se crea la **Iniciativa de Accesibilidad Web**, más conocida como **WAI** (Web Accesibility Initiative) la cual es una actividad desarrollada por W3C, donde se crean estrategias, pautas y recursos para ayudar a hacer la web accesible para personas con discapacidad[8], además trata de concientizar la importancia de la accesibilidad web y de abrir nuevos campos de investigación en relación a esta área.

WAI desarrolla las pautas para diseñar aplicaciones web accesibles bajo la idea de diferentes tipos de contextos de discapacidad:

- Personas que no pueden ver, oír, moverse.
- Personas que no pueden procesar algún tipo de información o toda fácilmente.

- Personas que no pueden usar un teclado o el ratón.
- Personas que por alguna situación no pueden ver, escuchar, o usar sus manos (conduciendo, cocinando, trabajando en zonas ruidosas, etc.).
- Personas que no hablan ni entienden de fluidamente el idioma en el que está el contenido.

Estos y más contextos son algunos tipos de situaciones que debe considerar un desarrollador durante el diseño de una página web.

Dichas pautas se denominan, **Pautas de Accesibilidad al Contenido Web** o bien sus siglas en inglés **WCAG** (Web Content Accessibility Guidelines), existen dos versiones de las WCAG, ahora se hablará de las más actuales las WCAG 2.0 que se basa en las WCAG 1.0 [9] y están diseñadas con el propósito de que se puedan usar en una amplia gama de tecnologías web actuales y futuras.

Para que estas pautas sean empleadas por desarrolladores o diseñadores web, se plantean varios niveles de orientación que en conjunto de alguna forma guían en el cómo crear contenido web más accesible: principios generales de la accesibilidad, pautas generales, criterios de conformidad verificables y técnicas suficientes y recomendables para satisfacer los criterios.

2.3.1. PRINCIPIOS DE LA ACCESIBILIDAD

Los principios generales de la accesibilidad es la base de las WCAG, estos principios proporcionan los pilares de la accesibilidad web:

1. **Perceptible:** Los componentes de información e interfaz de usuario deben ser presentados de una manera en que sean percibidos a primera vista por los usuarios.
    - Lo que se quiere decir es que no exista contenido invisible al usuario, por ejemplo, un menú desplegable, el cual funciona por medio de eventos del mouse*, cuando un usuario pasa el mouse sobre la opción de menú aparece un sub menú, si el usuario intenta acceder a él puede pasar que el mouse salga del rango donde causa que se muestre el sub menú haciéndolo desaparecer, causando al usuario una sensación de insatisfacción.
2. **Operable:** Los componentes de navegación e interfaz de usuario deben ser totalmente operables.

- Esto quiere decir que la interfaz no debería contener elementes habilitados que no den ninguna funcionalidad a la aplicación, o sea botones los cuales dicen ejecutar una acción y no la hacen, sea o no, por error de la aplicación, también evitar opciones de menú que dicen acceder a una zona de la web y redirigen a otra.

3. **Comprensible:** la información y la interacción del usuario con la interfaz deben ser entendibles.
    - Significa la facilidad del usuario al comprender el contenido y el funcionamiento de la web.
4. **Robusto:** El contenido debe ser lo suficientemente robusto para que pueda ser interpretado de una manera fiable por una amplia variedad de software de asistencia[4] o ayuda técnica.
    - Cumplir este principio hace que permita que los usuarios puedan acceder al contenido de la web con otras tecnologías como lectores de pantalla, esto se puede lograr por medio de especificaciones como WAI-ARIA de la cual se habla la sección 3.8.

En resumen se puede decir que si alguno de estos principios no son ciertos, puede que usuarios con discapacidad no puedan hacer uso de la web.

2.3.2. PAUTAS WACG 2.0

Como se ha mencionado antes, las pautas tienen como base los 4 principios de la accesibilidad, a continuación se explicarán cada una de ellas tratando de dar un ejemplo sencillo sobre cómo y dónde se deberían aplicar, además las pautas contienen una serie de criterios de conformidad que permiten usar las pautas en aquellas situaciones en las que existen requisitos y necesidades de evaluación, y con la finalidad de que se cumplan las necesidades en los diferentes grupos y situaciones, se establecen tres niveles de conformidad para la calificar una página web con referencia a dichos criterios: A (el mínimo), AA y AAA (el máximo).

Entonces se dice que una página web es de:

- **Nivel A:** Cuando la página web satisface todos los criterios de conformidad de nivel A, o proporciona una versión alternativa [10].

[4] Software de asistencia: hace referencia a cualquier programa de ordenador que ayude al usuario a usar todas o determinadas funcionalidades del sistema.

- **Nivel AA:** Cuando la página web satisface todos los criterios de conformidad de nivel A y AA, o proporciona una versión alternativa de nivel AA [10].
- **Nivel AAA:** Cuando la página web satisface todos los criterios de conformidad de nivel A, AA y AAA, o proporciona una versión alternativa de nivel AAA [10].

Para comenzar a continuación las pautas basadas al principio “perceptible”:

2.3.2.1. PERCEPTIBLE

Pauta 1.1 Alternativas textuales

Dar opciones textuales al contenido no textual, de modo que se puedan convertir a otros formatos de acuerdo a las necesidades del usuario.

La propuesta de esta pauta, es asegurarse de que todo contenido no textal también esté disponible en texto (no imágenes de texto), el cual cumple el mismo propósito, la ventaja de que exista también en texto, es que éste puede ser presentado en diferentes formas, y mejores para el usuario en relación a su discapacidad, tales como texto ampliados, braille, voz, símbolos o un lenguaje más simple.

Criterios de conformidad:

1.1.1 Contenido no textual: No todo contenido no textual puede tener una alternativa textual que cumpla el mismo propósito, las situaciones enumeradas a continuación son la excepción (Nivel A):

- **Controles, entradas de datos:** Este tipo de contenido siempre tiene un nombre o un texto al lado que describe su propósito.
- **Contenido multimedia basado en el tiempo:** Este tipo de contenido debe tener como alternativa textual al menos una identificación descriptiva del contenido no textual.
- **Pruebas, sensorial:** Si el contenido no textual es una prueba o ejercicio que no fuera valido si se presenta en forma de texto, o si tiene como objetivo principal crear una experiencia sensorial especifica (como el sonido de nota musical), entonces como alternativa textual proporcionará al menos una identificación descriptiva del contenido no textual.
- **CAPTCHA:** En este caso se dan alternativas textuales que describen e identifican el propósito del contenido no textual y se proporcionan otras

alternativas de CAPTCHA con modos de salida para distintos tipos de percepción sensorial, con el objetivo de adaptarse a las diferentes discapacidades.

- **Decoración, formato, invisible:** Si el contenido no textual solo tienen un propósito estético, no proporciona información o no tiene ninguna funcionalidad, entonces se implementa de tal forma que se ignore por software de asistencia.

Pauta 1.2 Contenidos multimedia basados en el tiempo

Proporcionar alternativas para los contenidos multimedia basados en el tiempo.

Esta pauta propone la forma de dar acceso al contenido multimedia basado en el tiempo y sincronizado*, esto incluye contenido que puede ser:

- Sólo audio.
- Sólo video.
- Audio-video.
- Audio y/o video combinado con interacciones.

Para hacer fácil la implementación de esta pauta, se dan los criterios de conformidad de una manera separada de acuerdo al tipo de contenido multimedia y permitiendo también predeterminar si es grabado o en directo, además se cumple la excepción sobre todos los criterios de que no es necesario aplicarlo cuando el contenido es un contenido multimedia alternativo al texto y está claramente identificado como tal, por ejemplo, en una web de un periódico en las plantillas que muestra un artículo en concreto, a veces están acompañados de un video, ese video debería tener un atributo dentro de su etiqueta que lo relaciona con el texto del artículo.

Criterios de conformidad:

1.2.1 Sólo audio y sólo video (grabado): Para contenido sólo audio grabado y video grabado (video sin información auditiva), se cumple lo siguiente (Nivel A):

- **Sólo audio grabado:** Proporcionar una alternativa que presente información equivalente para el contenido multimedia de sólo audio grabado.

- **Sólo video grabado:** Proporcionar una alternativa o una pista sonora que presente información equivalente al contenido multimedia de sólo video grabado.

1.2.2 Subtítulos (grabados): Proporcionar subtítulos para el audio grabado dentro del contenido multimedia sincronizado (Nivel A).

1.2.3 Audio-descripción o medio alternativo (grabado): Proporcionar una alternativa que presente información equivalente o una audio-descripción (ésta representa la información necesaria del video en partes donde no hay audio) para el contenido del video grabado en el contenido multimedia sincronizado (Nivel A).

1.2.4 Subtítulos (en directo): Proporcionar subtítulos para todo el contenido de audio en directo de los contenidos multimedia sincronizados, sin excepciones (Nivel AA).

1.2.5 Audio-descripción (grabado): Proporcionar una audio-descripción para todo el contenido de video grabado del contenido multimedia sincronizado, sin excepciones (Nivel AA).

1.2.6 Lenguaje de señas: Proporcionar una interpretación en lenguaje de señas para todo el contenido de audio grabado del contenido multimedia sincronizado, sin excepciones (Nivel AAA).

Este criterio sube el nivel de conformidad, debido a que el lenguaje de señas permite reflejar entonación, emociones y otro tipo de información auditiva a diferencia de los subtítulos.

1.2.7 Audio-descripción ampliada (grabada): Cuando las pausas del audio de primer plano son muy cortas para poder comunicar el significado del video por medio de la audio-descripción, se proporcionará una audio-descripción ampliada para todos los contenido del video grabado dentro del contenido multimedia (Nivel AAA).

En este caso se usan técnicas que permiten pausar el video mientras se escucha la audio-descripción y luego volverlo a reproducir cuando ésta haya terminado.

1.2.8 Medio Alternativo (grabado): Proporcionar una alternativa que presente un información equivalente, tanto para todos los contenidos multimedia sincronizados grabados como para todos los contenidos de sólo videos grabados (Nivel AAA). Esa alternativa podría leerse en algunas ocasiones como un libro que describe todo lo que está sucediendo en el contenido multimedia.

1.2.9 Sólo audio (en directo): Proporcionar una alternativa que presente información equivalente para el contenido de sólo audio en directo (Nivel AAA).

Una técnica para que este criterio se cumpla, puede ser el hacer uso de una persona que escuche lo que se dice y a la vez vaya escribiéndolo para ser presentado en formato de texto en ese mismo momento al usuario discapacitado.

Pauta 1.3 Adaptable.

Crear contenido que pueda presentarse de diferentes formas sin perder información o estructura.

El objetivo de esta pauta es asegurarse que toda la información está disponible en un formato para que así por medio de algún tipo de software pueda ser presentado a cualquier tipo de usuario.

Criterios de conformidad:

1.3.1 Información y relaciones: La información, estructura y relaciones comunicadas a través de la interfaz de usuario pueden ser procesadas por software o están disponibles como texto (Nivel A).

Este criterio es para asegurar de que sin importar que el usuario vea la información o la está escuchando pueda notar las diferencias del contenido, es decir, si dentro de una texto existe un enlace, visualmente se puede distinguir cambiando el color o el tamaño del texto, pero si se usa un lector de pantallas, ese texto debe dar la opción para que el lector de pantallas sepa que es un enlace o incluso un anclaje a otro contenido que está en la misma página.

1.3.2 Secuencia significativa: Cuando la secuencia en que se presenta el contenido afecta a su significado, se puede determinar por software la secuencia de lectura (Nivel A).

Este criterio pretende que la estructura del contenido tenga una forma o que permita al software de asistencia determinar una secuencia del contenido que tenga sentido para el usuario, ya que puede causar confusiones o desorientación a los usuarios.

1.3.3 Características sensoriales: Las instrucciones proporcionadas para entender y operar el contenido no dependen exclusivamente de las características sensoriales de los componente tales como su forma, tamaño, ubicación o sonido (Nivel A).

El objetivo de este criterio es asegurarse de que los usuarios puedan acceder a las instrucciones de un contenido sin importar que ellos no puedan percibir formas o tamaños, o usar la información sobre ubicación, por lo tanto las indicaciones deben ser bastante claras y el contenido deberá estar ordenado de tal forma que puedan ser captadas fácilmente.

Pauta 1.4 Distinguible.

Facilitar a los usuarios la forma de ver y oír el contenido, incluyendo la separación entre el primer plano y el fondo.

Esta es una pauta que se enfoca más en hacer más fácil la forma en que usuarios con discapacidades visuales y auditivas captan el contenido marcando la diferencia entre el contenido de primer plano y el de fondo.

Criterios de conformidad:

1.4.1 Uso del color: No usar el color como la única forma visual de trasmitir información, indicar una acción, solicitar una respuesta o distinguir un componente (Nivel A).

Aunque el color es un aspecto muy importante en el diseño de una web, debido a que proporciona usabilidad y accesibilidad ya que con él se puede transmitir un significado o importancia de ese contenido al usuario, pero para los usuarios que tienen problemas para percibir los diferentes colores esto no es útil, pero se pueden usar otras formas visuales para que esos contenidos puedan ser comprendidos.

1.4.2 Control del audio: En el caso de que la web tenga contenido auditivo que se reproduce automáticamente por más de 3 segundos, ésta debe proporcionar la manera de que pueda ser pausado o detenido, o también debe permitir controlar los sonidos que son independientes del nivel de volumen del sistema operativo (Nivel A).

Para las personas que usan un lector de pantalla, el hecho de entrar en una página web y que ésta reproduzca algún tipo de contenido auditivo, podría interferir en el modo de navegación del usuario de esta tecnología, incluso para encontrar dichos mecanismo de control de ese audio, de igual manera es importante que el usuario pueda controlar esos sonidos.

1.4.3 Contraste (mínimo): La presentación visual del texto e imágenes de texto* tiene una relación de contraste de al menos de 4.5:1 con respecto al fondo, excepto en los siguientes casos (Nivel AA):

- **Textos grandes:** Textos de tamaño grande e imágenes de texto de tamaño grande tendrán una relación de contraste de al menos 3:1 con respecto al fondo.
- **Incidental:** Los textos o imágenes de texto que son parte de un componente inactivo de la interfaz de usuario, que son solo decoración, que no son visibles para nadie o hacen parte de una imagen que contienen otros elementos visuales más significativos, no tienen requerimientos de contraste.
- **Logotipos:** Texto que hace parte de un logo o el nombre de una marca, no tiene un mínimo contraste requerido.

Cumplir este criterio facilitaría la lectura a personas con problemas de visión baja que no usan algún tipo de software que mejore el contraste.

1.4.4 Cambio del tamaño del texto: A excepción de los subtítulos y las imágenes de texto, todo el texto podrá ser ajustado sin software de asistencia hasta un 200 porciento sin que se pierda el contenido o la funcionalidad (Nivel AA). Este criterio representaría una ayuda más para aquellos usuarios con problemas de visión.

1.4.5 Imágenes de texto: Si con las tecnologías (ej. CSS) que se están utilizando se logra conseguir la presentación visual deseada sobre la página web, se deberá utilizar texto en vez de imágenes de texto para transmitir la información, excepto en los siguientes casos (Nivel AA).

- **Configurable:** La imagen de texto es visualmente configurable según los requisitos del usuario.
- **Esencial:** La forma de presentación de ese texto es esencial para la información que se transmite (ej. Logotipos).

1.4.6 Contraste (mejorado): Este criterio es similar al criterio 1.4.3, solo que en este caso la relación de contraste es de 7:1 como mínimo, y con respecto a los textos grandes sería de 4.5:1 (Nivel AAA).

1.4.7 Sonido de fondo bajo o ausente: Para el contenido de sólo audio grabado que:

1. Contiene habla en primer plano.
2. No es un CAPTCHA sonoro o un logo sonoro.

3. No es una vocalización que pretende principalmente ser una expresión musical como cantar.

Al menos debe cumplir uno de los siguientes casos:

- **Ningún sonido de fondo:** El audio no contiene sonido de fondo.
- **Apagar:** Los sonidos de fondo pueden ser apagados.
- **20db:** Los sonidos son como mínimo 20dbs (4 veces) más bajos que el audio en primer plano, con excepción de sonidos ocasionales.

Este criterio permitirá a usuarios con problemas auditivos diferenciar los sonidos de fondo con respecto a los de primer plano.

1.4.8 Presentación visual: Para la presentación visual de bloques de texto, se proporcionará un mecanismo para lograr lo siguiente (Nivel AAA):

1. Los colores de fondo y primer plano pueden ser elegidos por el usuario.
2. El ancho no debe ser mayor de 80 caracteres o signos (40 en caso del chino, japonés o coreano).
3. Texto no justificado (Alinear ambos márgenes izquierda y derecha).
4. El espacio entre líneas es de 1.5 dentro de los párrafos y dentro de los párrafos es por lo menos 1.5 veces más grande que el espacio entre líneas.
5. El texto se ajusta hasta un 200 porciento sin software de asistencia de tal modo que no necesite desplazamiento horizontal para leer una línea de texto.

Esto permite a algunos usuarios con discapacidad cognitiva, de lenguaje o de aprendizaje o incluso algunos con problemas de visión acceder a un texto más legible sin interferir en el diseño.

1.4.9 Imágenes de texto (sin excepciones): Las imágenes de texto sólo se deben usar como simple decoración o cuando una forma particular de presentación del texto es esencial para ser mostrada (Nivel AAA).

2.3.2.2. OPERABLES

Pauta 2.1 Accesible por teclado.

Proporcionar acceso a toda la funcionalidad mediante el teclado.

No existe otra forma de inserción que tenga esta la flexibilidad que tiene el teclado o su soporte universal y que además sea operable por personas con diferentes discapacidades, siempre y cuando las entradas por medio del teclado no sean dependientes del tiempo. En fin si toda la funcionalidad de la página web puede ser operable con el teclado, será fácil de usar tanto para los usuarios como para software de asistencia que genera salidas de combinaciones de teclas.

Criterios de conformidad

2.1.1 Teclado: Toda la funcionalidad del contenido puede ser operable a través de una interfaz de teclado sin importar los tiempos que requiera hacer una combinación de teclas, excepto cuando la funcionalidad interna requiere de una entrada dependiente del trayecto que efectúe el usuario con las teclas y no de la ultima que pulse (Nivel A).

2.1.2 Sin trampas de teclado: Si es posible mover el foco del teclado a un componente de la página usando una interfaz de teclado, entonces el foco puede ser movido de ese componente sólo con esa interfaz de teclado, y si se necesita algo más que las teclas de navegación o el tabulador, se le deberá informar al usuario el método apropiado para mover el foco (Nivel A).

Este criterio pretende asegurar que el usuario que usa el teclado no quede atrapado entre las sub secciones del contenido de la página web, esto se soluciona dándole al usuario una forma de salir de ellas.

2.1.3 Teclado (sin excepciones): Toda la funcionalidad del contenido se puede operar a través de una interfaz de teclado sin requerir una determinada velocidad entre pulsaciones (Nivel AAA).

Este criterio es igual que el 2.1.1, solo que en este pretende que aquellas excepciones que hay en el 2.1.1 se hagan accesibles por medio del teclado.

Pauta 2.2 Tiempo suficiente.

Proporcionar a los usuarios el tiempo suficiente para leer y usar el contenido.

Esta pauta pretende que los usuarios discapacitados que necesiten más tiempo para interactuar con el contenido para hacer determinada tarea puedan terminarla, sea eliminando las condiciones de tiempo o permitiendo agregarle más tiempo a dichos

usuarios para completar sus labores, obviamente habrán excepciones donde esto no podrá efectuarse.

Criterios de conformidad

2.2.1 Tiempo ajustable: Para cada límite de tiempo que es puesto por el contenido, se cumple al menos uno de los siguientes casos: (Nivel A)

- **Apagar:** El usuario podrá apagar el límite de tiempo antes que éste se acabe.
- **Ajustar:** El usuario puede ajustar el límite de tiempo en una rango amplio, al menos 10 veces mayor que el tiempo fijado por el contenido antes de que éste se acabe.
- **Extender:** Se le debe advertir al usuario antes que el tiempo termine y concederle 20 segundos para extender ese límite temporal con una acción simple (ej. Pulsando determinada tecla), y podrá extender ese tiempo como mínimo 10 veces más.
- **Excepción en tiempo real:** El límite de tiempo es un requisito que forma parte de una evento en tiempo real y no resulta posible ofrecer una alternativa a ese límite de tiempo (ej. Una subasta).
- **Excepción esencial:** Cuando el límite de tiempo es esencial, y si éste se extendiera, invalidaría la actividad.
- **Excepción de 20 horas:** Cuando el límite de tiempo es mayor a 20 horas.

Este criterio pretende asegurar que los usuarios con determinadas discapacidades puedan completar una tarea sin cambios inesperados en el contenido o contexto los cuales son el resultado de un límite de tiempo, eso sí, se aconseja que es mejor deshabilitar esos límites de tiempo a que sean configurables.

2.2.2 Pausar, parar, esconder: Para la información que se mueve, parpadea, se desplaza (ej. Imágenes en movimiento, contenido multimedia sincronizado, animaciones, juegos en tiempo real, desplazamientos de cotizaciones de bolsa) o se actualiza automáticamente (ej. Actualizaciones automáticas del clima, de noticias, mensajes), debe cumplir los siguiente: (Nivel A)

1. **Movimiento, parpadeo, desplazamiento:** Para toda la información que se mueve, parpadea o se desplaza, que:
    1. Comienza automáticamente.
    2. Dura más de cinco segundos.
    3. Se presenta en paralelo con otro contenido.

Tendrá que existir algún mecanismo para que el usuario pueda pararla, pausarla u ocultarla, a menos que esa información sea parte esencial de una actividad.

2. **Actualización automática:** Para toda la información que se actualiza automáticamente, que:
    1. Comienza automáticamente.
    2. Se presenta en paralelo con otro contenido.

    Tendrá que existir algún mecanismo para que el usuario pueda parar, pausar u ocultar, o controlar la frecuencia de actualización, a menos que dicha actualización se parte esencial de una actividad.

Este criterio se usa para evitar distracciones a los usuarios a la hora de navegar por el contenido de la web.

2.2.3 Sin tiempo: El tiempo no debe ser parte esencial del evento o actividad presentada por el contenido, excepto los contenidos multimedia sincronizados no interactivos y los eventos en tiempo real (ej. Una subasta) (Nivel AAA).

Con esto se pretende minimizar la aparición de contenido que requiere interacción temporizada, permitiendo a usuarios discapacitados interactuar con el contenido.

2.2.4 Interrupciones: El usuario podrá postergar o eliminar las interrupciones, excepto si las interrupciones implican una emergencia (Nivel AAA). Cuando se dice emergencias hace referencia a cosas como mensajes de alerta de emergencias civiles, o mensajes que previenen daño a la salud, seguridad o propiedad del usuario, incluyendo perdida de datos o conexión, etc.

2.2.5 Re autentificación: Cuando expira una sesión, el usuario podrá continuar su actividad sin perdidas de datos tras volver a autentificarse (Nivel AAA). Este criterio permitirá que las transacciones autenticadas hechas por los usuarios sean completadas aunque tenga límites de tiempo de inactividad o por alguna causa sacan de la sesión al usuario mientras se completa las transacción.

Pauta 2.3 Convulsiones

No diseñar contenido de un modo que se sepa puede causar ataques, espasmos o convulsiones.

Existen personas con trastornos compulsivos los cuales pueden sufrir un ataque provocado por el parpadeo de un contenido visual, por lo tanto esta pauta es más una medida de seguridad. En todo caso contiene dos criterios de conformidad para alcanzar en nivel más bajo de satisfacción o el más alto.

Criterios de conformidad

2.3.1 Umbral de tres destellos (flahes) o menos: Una página web no tendrá contenido que destelle más de tres veces en un periodo de un segundo, o el destello es menor que los umbrales de destello general o rojo******** (Nivel A).

Si algún contenido no satisface este criterio podrá interferir en la capacidad del usuario para usar el contenido de la página, por lo tanto todo contenido de una página web deberá satisfacer este criterio tanto si satisface o no otros criterios.

2.3.2 Tres destellos: La página web no tendrá contenido que destelle más de 3 veces por segundo (Nivel AAA).

Pauta 2.4 Navegable

Proporcionar mecanismos para ayudar a los usuarios a navegar, encontrar contenido y determinar dónde está

No es fácil para usuarios con discapacidades ubicarse u orientarse dentro del contenido de una página web, esta pauta pretende establecer una forma para ayudar a esos usuarios encontrar el contenido y permitirles hacer un seguimiento de su ubicación dentro de la web.

Criterios de conformidad

2.4.1 Evitar bloques: Existe un mecanismo para evitar los bloques de contenido en múltiples páginas web (Nivel A).

El propósito de este criterio es permitir a las personas que navegan de una forma secuencial******* tenga una forma de llegar directamente al contenido principal de la web sin tener que pasar por el resto del contenido que se repite en las páginas web, tales como, encabezados, pie de página, barras laterales, menús de navegación, etc.

2.4.2 Titulado de páginas: Las páginas web deben tener títulos que describen su temática o propósito (Nivel A).

La ventaja de que una página lleve un título es que permite al usuario identificar su ubicación actual sin necesidad de interpretar el contenido de la página web. También ayuda en que, en los resultados de búsqueda el usuario se le hará más fácil encontrar e identificar el contenido que necesita.

2.4.3 Orden del foco: Si se puede navegar secuencialmente[*****] por una página web y la secuencia de navegación afecta su significado u operatividad, los componentes deberán recibir el foco en un orden que no afecte su significado u operatividad (Nivel A).

Presentar el contenido en un orden, permite a los usuarios encontrar el contenido que es consistente con su significado y además es operable con el teclado, esto reduce confusiones y permite que los usuarios creen un modelo mental del contenido.

2.4.4 Propósito de los enlaces (en contexto): El propósito de cada enlace puede ser determinado por solo el texto del enlace, o desde el texto del enlace junto con su contexto determinado por software, excepto cuando el propósito de los enlaces resulte ambiguo para los usuarios en general (ej. Cuando se da una introducción de un artículo y al final de esa introducción hay un enlace que dice “Leer más…”). (Nivel A)

La intención de este criterio es dar al usuario con discapacidad la información necesaria sobre a dónde o que acción va ejecutar ese enlace para que él puede decidir con seguridad si lo quiere seguir o no.

2.4.5 Múltiples vías: Se tendrá que proporcionar más de un camino para localizar una página web dentro de un mismo sitio web, excepto cuando la página es el resultado o el paso intermedio de un proceso, como por ejemplo cuando se está rellanando un formulario que requiere de varios pasos (Nivel AA).

La finalidad de este criterio es proporcionar al usuario una forma en que al le parezca más fácil llegar al contenido, los sitios web pueden proporcionar site-maps, breadcrumb, etc. para este propósito. Por ejemplo un usuario con limitaciones cognitivas podría encontrar más fácil el usar una funcionalidad de búsqueda que usar el site-map.

2.4.6 Encabezados y etiquetas: Los encabezados o etiquetas describen el tema o propósito (Nivel AA).

El término etiqueta en este caso hace referencia al texto u otro componente con una alternativa textual que se presenta para identificar un componente dentro del contenido de la web, lo que hace que el usuario pueda entender el contenido de la web o como

está organizada la información. Lo mismo sucede cuando los encabezados son claros y descriptivos.

2.4.7 Foco visible: La interfaz de teclado con la cual se navega sobre el contenido deberá tener un indicador del foco del teclado que sea visible (Nivel AA), esto con el fin de tener una forma visual de localizar el foco del teclado.

2.4.8 Ubicación: Se proporcionará información de la ubicación del usuario dentro del sitio web (Nivel AAA).

Este criterio puede ayudar a personas con problemas de atención las cuales pueden llegar a confundirse al seguir una gran cantidad de pasos de navegación, también ayuda a usuarios en general entender el contenido o encontrar más información relacionada.

2.4.9 Propósito de los enlaces (sólo enlaces): Se proporcionará un mecanismo que permite identificar el propósito de cada enlace con sólo el texto del enlace, excepto cuando el propósito del enlace resulta ambiguo para los usuarios en general (Nivel AAA).

Este criterio tiene la finalidad de que muchos de los enlaces pueden ser entendidos con sólo su texto, incluso si están fuera del contexto, para que así cualquier software de asistencia los pueda enlistar de alguna forma y sean entendidos.

2.4.10 Encabezados de sección: Usar encabezados de sección para organizar el contenido (Nivel AAA).

El término encabezado se usa en el sentido general que incluye los títulos y otras formas de agregar encabezados a los diferentes contenidos, un ejemplo se puede plantear en un sitio web de un periódico, donde la portada tiene noticias de diferente tipo, y se puede sectorizar por medio de encabezados, deporte, economía, tecnología, etc. Estos encabezados permiten organizar el contenido cuando son páginas demasiado extensas.

2.3.2.3. COMPRENSIBLE

Pauta 3.1 Legible

Hacer que los contenidos textuales resulten legibles y comprensibles.

Existen usuario con discapacidades que perciben el texto de diferentes formas, puede ser táctil, visual, auditivo o incluso estas últimas juntos visual y auditivo, y con el fin de que estos usuarios no tengan dificultades a la hora de entender el contenido esta pauta pretende asegurarse de que el contenido se pueda leer y en determinados casos tenga disponible la información necesaria para que sea comprendido.

Criterios de conformidad

3.1.1 Idioma de la página: El idioma de cada página puede ser determinado por software (Nivel A).

El hecho de que la página web tenga una forma de identificar en qué idioma está, permite al software de asistencia u otras tecnologías gestionar el contenido en el idioma correcto, evitando así que se le muestre al usuario en el idioma erróneo. Un caso bastante notorio sería en los lectores de pantalla, si no existiera ese medio de saber el idioma, el lector de pantalla podría estar hablando en español, pero el contenido estar en inglés.

3.1.2 Idioma de las partes: El idioma de cada pasaje o frase del contenido puede ser determinado por software, excepto para nombres propios, términos técnicos, palabras de un idioma indeterminado y palabras o frases que se hayan convertido en parte natural del texto que los rodea (Nivel AA).

En algunos casos el contenido contiene palabras o frases que son extranjeras, es decir que están escritas y se pronuncian en otra lengua diferente a la establecida por la página web, pero que hacen parte de la forma de hablar para llenar un vacío semántico o como alternativa para otras expresiones, dichas expresiones no necesitan ser traducidas, pero si esas que están fuera de ese extranjerismo, así este criterio se asegura de que el contenido pueda ser comprendido por usuarios discapacitados haciendo posible que los software de asistencia u otras tecnologías puedan presentar le contenido acorde a las reglas de pronunciación de cualquier idioma.

3.1.3 Palabras inusuales: Existirá un mecanismo que permita identificar definiciones especificas de palabras usadas de un modo inusual o restringido, incluyendo expresiones y jergas (Nivel AAA).

Existen discapacidades las cuales hacen difícil entender un conjunto de palabras que no tienen un significado literal, y este criterio pretende que se proporciona una manera de saber dicho significado, puede ser por medio de un link a otra página que lo tenga, o que la misma página tenga un glosario donde encontrar esos significados.

3.1.4 Abreviaturas: Proporcionar mecanismo para identificar la forma extendida o el significados de las abreviaturas (Nivel AAA).

3.1.5 Nivel de lectura: Cuando un texto requiere un nivel de lectura más avanzado que el nivel mínimo de educación secundaria (de 7 a 9 años en un instituto educativo), después de quitar nombres propios y títulos, se deberá proporcionar un contenido suplementario o una versión del contenido que no necesite un nivel de lectura mayor al del mínimo de secundaria (Nivel AAA).

La intensión de este criterio no es que todo esté escrito de una forma universal en que sea fácil de entender, pero sí, hay formas de escribir las cosas de una forma clara y un lenguaje apropiado permitiendo que ese contenido sea más entendible como por ejemplo para personas con alguna discapacidad cognitiva.

3.1.6 Pronunciación: Existirá un mecanismo para identificar la pronunciación especifica de palabras donde su significado, dentro del contexto, es ambiguo si no se conoce la pronunciación (Nivel AAA).

La intensión de este criterio es ayudar a personas con alguna discapacidad visual o de compresión de lectura en aquellos casos donde su significado depende de la pronunciación, también se da el caso de que normalmente el algunos idiomas cuando una palabra bajo un contexto necesita un acento para generar una pronunciación diferente y esta no lo lleva, a veces de forma visual se tarda un poco en entender, por lo tanto con un lector de pantalla será aun más difícil.

Pauta 3.2 Predecible

Hacer que las páginas web aparezcan y trabajen de forma predecible.

Esta pauta pretende que el contenido de un conjunto de páginas de un sitio web se muestre y se comporte de una forma similar en cada una de ellas, haciendo fácil para usuarios con determinadas discapacidades acostumbrarse al contenido y hacer navegar dentro de él de forma fluida hasta encontrar lo que buscan.

Criterios de conformidad

3.2.1 En el foco: Cuando algún componente recibe el foco no debe iniciar ninguna tarea que cambie el contexto (Nivel A).

Este criterio solo pretende asegurarse que no se ejecute ninguna tarea cuando un determinado componente (ej. Un botón de un formulario) recibe el foco, solo hasta que el usuario pida se efectúe dicha acción. Este comportamiento evita que el usuario se pierda en los cambios de contexto, ya que es él, quien ha hecho una cantidad de procesos para que eso suceda.

3.2.2 En la entrada: El cambio de configuración de un componente de la interfaz de usuario no causa automáticamente un cambio de contexto a menos que el usuario haya sido avisado de ese comportamiento antes de usarlo (Nivel A).

Claramente la intensión de este criterio es asegurarse que la entrada de datos o la selección de algún componente de formulario tengan un comportamiento predecible, es decir que los cambios de contexto que generan algunos componentes sean por la ejecución de una acción por parte del usuario. Un ejemplo claro podría ser la selección de un checkbox que hace aparecer un conjunto de componentes nuevos, pero el texto que acompaña a este checkbox hace mención a que se va efectuar algún cambio (ej. ¿Quieres recibir noticias? Selecciona esta casilla y escoge tus temas de interés…).

3.2.3 Navegación coherente: Los mecanismos de navegación que son repetidos en múltiples páginas web dentro de un conjunto de páginas, aparecerán siempre en el mismo orden relativo cada vez que se repiten, a menos que el cambio sea iniciado por el usuario (Nivel AA).

La intensión de este criterio es permitir a los usuarios que interactúan con contenido repetido a través de las páginas web puedan predecir la ubicación del contenido que están buscando y lo encuentren rápidamente, haciendo la navegación más fluida.

3.2.4 Identificación coherente: Los componentes que tienen la misma funcionalidad dentro de un conjunto de páginas web son identificados coherentemente (Nivel AA).

La finalidad de este criterio es que aquellos botones, iconos, enlaces que tienen la misma funcionalidad dentro diferentes páginas web de un mismo sitio, tenga una forma de identificación dentro de las diferentes páginas web, es decir el hecho de que una funcionalidad tenga diferentes nombres en diferentes páginas web hace que el uso de la página sea más difícil, un ejemplo de que no se tiene que hacer sería tener un icono de PDF, el cual puede contiene un texto alternativo que dice “Descargar PDF” en un página web, pero en otra tiene otro texto que dice “Clica aquí para descargar”, este tipo de cosas confundiría a usuario discapacitados que usan lectores de pantalla.

3.2.5 Cambios por petición: Los cambios de contexto son iniciados por petición del usuario o se proporciona un mecanismo para detener tales cambios (Nivel AAA).

Este criterio pretende promover dar al usuario el control total sobre el comportamiento del contenido de una página web, lo cual ayuda a eliminar confusiones generadas por cambios inesperados de contextos, aunque no todo cambio automático de contexto es malo ya que en ocasiones dichos cambios ayudan a la usabilidad, aun así el usuario deberá tener la posibilidad de evitar que esos cambios se ejecuten.

Pauta 3.3 Asistencia en Entrada de datos

Ayudar a los usuarios a evitar y corregir errores.

Todos cometemos errores a la hora de introducir datos, pero para una persona discapacitada a veces detectar que ha cometido algún error es difícil. Esta pauta busca reducir la cantidad de errores que se pueden cometer en estos casos, incrementando la posibilidad de que los usuarios puedan detectarlos, y ayudarlos a corregirlos.

3.3.1 Identificación de errores: Si se detecta un error de entrada de datos automáticamente, el elemento con el error es identificado y el error es descrito al usuario en texto (Nivel A).

La finalidad de este criterio lleva a que el usuario pueda saber que ha pasado un error y determinar que está mal, otra cosa a considerar es que el mensaje descriptivo del error sea los más claro posible, hay usuarios que aunque tengan software de asistencia, en ocasiones se les puede hacer imposible encontrar el elemento con dicho error.

3.3.2 Etiquetas o instrucciones: Proporcionar etiquetas o instrucciones cuando el contenido requiere la introducción de datos por parte del usuario (Nivel A).

La gente con discapacidades confían en formularios documentados y en procesos que permiten interactuar con la página web, por ejemplo, los usuarios con ceguera que usan lectores de pantalla, gracias a las etiquetas junto al campo, sabrían que información tienen que introducir en cada uno, o para usuarios con discapacidad cognitiva, el uso de simples instrucciones visuales junto a los campos podrían ayudarlos.

3.3.3 Sugerencias ante errores: Si se detecta automáticamente un error en la entrada de datos y la sugerencias para corregirlo son conocidas, éstas se le proporcionarán al usuario, a menos que esto perjudicara la seguridad o el propósito del contenido (Nivel AA).

Aunque en el criterio 3.3.1 se pide que se describa el error, no se le da una sugerencia de cómo puede corregirlo, ésta es la finalidad de este criterio, es decir, asegurarse que el usuario reciba para la corrección de un error ocurrido por la entrada de datos si es posible.

3.3.4 Prevención de errores (Legal, financiero, datos): Para las páginas web que causan compromisos legales o transacciones financieras, que modifica o elimina datos controlables por el usuario en sistemas de almacenamiento de datos, o que envían las respuestas de un examen, al menos una de las siguientes cosas es verdad (Nivel AA):

1. Reversible: El envío es reversible.
2. Comprobado: Los datos introducidos por el usuario son comprobados para detectar errores y se le proporcionará al usuario la oportunidad de corregirlos.
3. Confirmado: Se proporcionará un mecanismo para revisar, confirmar y corregir la información antes de finalizar el envío.

Cumpliendo este criterio se puede evitar que usuarios discapacitados cometan errores que llevan a seria consecuencias debido a la ejecución de una acción que no puede ser revertida, por ejemplo, si un usuario discapacitado hace un compra, puede que haya metido por error una cantidad mayor a la que necesitaba, o al momento de poner la fecha a unos tickets de avión se equivoque y los compre para el día en que él o ella no va a viajar, el fin, este tipo de usuarios necesita saber si la información que van a enviar es la correcta.

3.3.5 Ayuda: Se proporciona ayuda relacionada al contexto (Nivel AAA).

Usar información relaciona a la función que se está o se quiere ejecutar, permitiría a los usuarios discapacitados encontrar la forma de cómo realizar las operaciones sin perderse en lo que están haciendo. Las ayudas relacionadas al contexto vienen bien cuando la etiqueta no describe suficientemente bien la funcionalidad y se espera que ésta sea obvia para ellos, además que la puedan consultar cuando lo necesiten.

3.3.6 Prevención de errores (Todos): Para las páginas web que requieran del usuario el envío de información, al menos se cumple una de las siguientes cosas (Nivel AAA):

1. Reversible: El envío es reversible.
2. Comprobado: Los datos introducidos por el usuario son comprobados para detectar errores y se le proporcionará al usuario la oportunidad de corregirlos.
3. Confirmado: Se proporcionará un mecanismo para revisar, confirmar y corregir la información antes de finalizar el envío.

Este criterio al igual que el 3.3.4 evita las consecuencias de por los errores de envío de datos erróneos, pero en este caso, se pretende que sea para todo tipo formulario.

2.3.2.4. ROBUSTO

Pauta 4.1 Compatible

Maximizar la compatibilidad con las aplicaciones de usuario actuales y futuras, incluyendo los software de asistencia.

El propósito de esta pauta es asegurarse que el contenido sea compatible con las diferentes tecnologías, para que ellas funcionen siempre correctamente, sobre todo cualquier software de asistencia, cumplir esta pauta, permite a dichos software interactuar y reorganizar el contenido para ser expuesto a los usuarios, y facilita el trabajo de desarrolladores para adaptar los software a nuevas tecnologías web.

Criterios de conformidad

4.1.1 Procesamiento: En los contenidos implementados por medio del lenguaje de marcas, los elementos tienen etiquetas de cierre y apertura completas, los elementos están anidados de acuerdo a las especificaciones, no contienen atributos duplicados y todos los identificadores son únicos, excepto cuando las especificaciones permitan estás características (Nivel A).

A veces el código HTML de un página web no puede ser bien interpretado por los navegadores web, debido a que el código tiene algunas etiquetas sin cerrar, este criterio pretende que los diseñadores web tenga en cuenta el no dejar etiquetas abiertas, aunque actualmente, los navegadores web usan técnicas para indexar dichos códigos. También se puede dar el caso en que existan etiquetas con un mismo identificador causando que algún software de asistencia que necesita de dicho identificador para funcionar correctamente cometa errores.

4.1.2 Nombre, role, valor: Para todos los componentes de la interfaz de usuario, el nombre y el role pueden ser determinados por software, los estados, propiedades y valores que pueden ser asignados por el usuario pueden ser especificados por software, y los cambios de estos elementos pueden ser consultados por las aplicaciones de usuario, incluyendo software de asistencia (Nivel A).

Suministrar nombres, identificadores, roles o valores a todos los componentes de la interfaz de usuario, hace compatible el sitio web con cualquier software de asistencia, como lectores de pantalla, reconocedores de voz, etc. usados por usuarios discapacitados.

3. Contexto Tecnológico

A continuación se explicará todo lo relacionado con las tecnologías, servicios, el estándar WAI-ARIA, etc. en relación al proyecto. Para comenzar, se hablará de cada uno de dichos aspectos y el porqué se ha usado para crear **inclusite.**

3.1. JavaScript

Debido a que **inclusite** es una aplicación web que se adapta al contenido de otra aplicación web y necesita la velocidad de respuesta de ejecución en la parte cliente se decide usar JavaScript para implementar su funcionamiento.

3.1.1. ¿Qué es JavaScript?

JavaScript [12] es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.

Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

3.1.2. Cómo incluir JavaScript en documentos XHTML

La integración de JavaScript y XHTML es muy flexible, ya que existen al menos tres formas para incluir código JavaScript en las páginas web.

El código JavaScript se encierra entre etiquetas **<script>** y se incluye en cualquier parte del documento. Aunque es correcto incluir cualquier bloque de código en

cualquier zona de la página, se recomienda definir el código JavaScript dentro de la cabecera del documento (dentro de la etiqueta **<head>**).

Para que la página XHTML resultante sea válida, es necesario añadir el atributo **type** a la etiqueta <script>. Los valores que se incluyen en el atributo type están estandarizados y para el caso de JavaScript, el valor correcto **es text/javascript**.

Este método se emplea cuando se define un bloque pequeño de código o cuando se quieren incluir instrucciones específicas en un determinado documento HTML que completen las instrucciones y funciones que se incluyen por defecto en todos los documentos del sitio web.

El principal inconveniente es que si se quiere hacer una modificación en el bloque de código, es necesario modificar todas las páginas que incluyen ese mismo bloque de código JavaScript.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio documento</title>
<script type="text/javascript">
  alert("Un mensaje de prueba");
</script>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

Figura 1: Cómo se implementa JavaScript dentro del documento XHTML

3.1.3. Definir JavaScript en un archivo externo

Las instrucciones JavaScript se pueden incluir en un archivo externo de tipo JavaScript que los documentosXHTML enlazan mediante la etiqueta <script>. Se pueden crear todos los archivos JavaScript que sean necesarios y cada documento XHTML puede enlazar tantos archivos JavaScript como necesite.

Ejemplo:

Archivo **codigo.js**

```
alert("Un mensaje de prueba");
```

Figura 2: código JavaScript

Documento XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio documento</title>
<script type="text/javascript" src="/js/codigo.js"></script>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

Figura 3: Enlazar archivo .js al documento XHTML

Además del atributo type, este método requiere definir el atributo src, que es el que indica la URL correspondiente al archivo JavaScript que se quiere enlazar. Cada etiqueta <script> solamente puede enlazar un único archivo, pero en una misma página se pueden incluir tantas etiquetas <script> como sean necesarias.

Los archivos de tipo JavaScript son documentos normales de texto con la extensión .js, que se pueden crear con cualquier editor de texto como Notepad, Wordpad, EmEditor, UltraEdit, Vi, etc.

La principal ventaja de enlazar un archivo JavaScript externo es que se simplifica el código XHTML de la página, que se puede reutilizar el mismo código JavaScript en todas las páginas del sitio web y que cualquier modificación realizada en el archivo JavaScript se ve reflejada inmediatamente en todas las páginas XHTML que lo enlazan.

3.1.4. Incluir JavaScript en los elementos XHTML

Este último método es el menos utilizado, ya que consiste en incluir trozos de JavaScript dentro del código XHTML de la página:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio documento</title>
</head>

<body>
<p onclick="alert('Un mensaje de prueba')">Un párrafo de texto.</p>
</body>
</html>
```

Figura 4 Incluir código JavaScript en los elementos XHTML

El mayor inconveniente de este método es que *ensucia* innecesariamente el código XHTML de la página y complica el mantenimiento del código JavaScript. En general, este método sólo se utiliza para definir algunos eventos y en algunos otros casos especiales, como se verá más adelante.

3.1.5. Etiqueta noscript

Algunos navegadores no disponen de soporte completo de JavaScript, otros navegadores permiten bloquearlo parcialmente e incluso algunos usuarios bloquean completamente el uso de JavaScript porque creen que así navegan de forma más segura.

En estos casos, es habitual que si la página web requiere JavaScript para su correcto funcionamiento, se incluya un mensaje de aviso al usuario indicándole que debería activar JavaScript para disfrutar completamente de la página. El siguiente ejemplo muestra una página web basada en JavaScript cuando se accede con JavaScript activado y cuando se accede con JavaScript completamente desactivado.

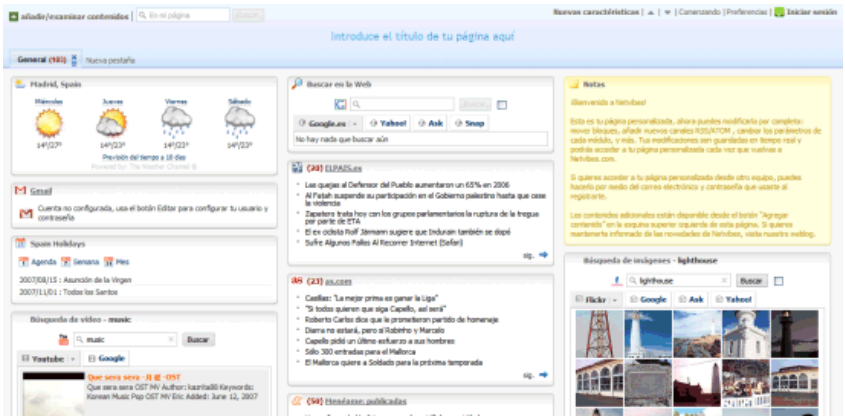


Figura 5: Imagen de www.Netvibes.com con JavaScript activado

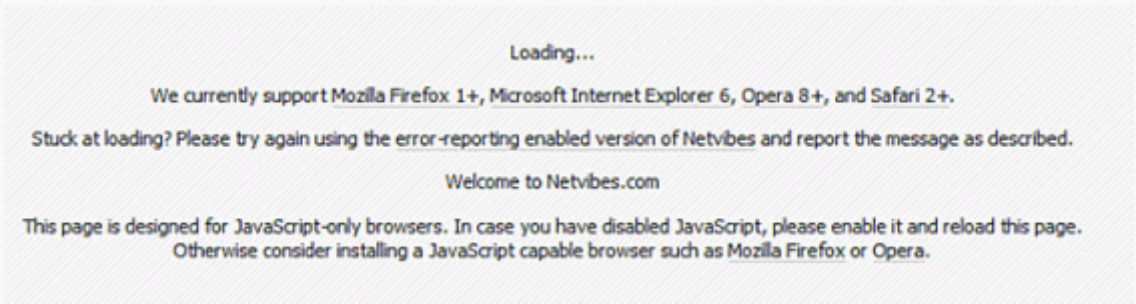


Figura 6: Imagen de www.Netvibes.com con JavaScript desactivado

El lenguaje HTML define la etiqueta <noscript> para mostrar un mensaje al usuario cuando su navegador no puede ejecutar JavaScript. La siguiente figura muestra un ejemplo del uso de la etiqueta <noscript>:

```
<head> ... </head>
<body>
<noscript>
  <p>Bienvenido a Mi Sitio</p>
  <p>La página que estás viendo requiere para su funcionamiento el uso de JavaScript.
Si lo has deshabilitado intencionadamente, por favor vuelve a activarlo.</p>
</noscript>
</body>
```

Figura 7: uso de la etiqueta noscript

3.1.6. Sintaxis

La sintaxis de un lenguaje de programación se define como el conjunto de reglas que deben seguirse al escribir el código fuente de los programas para considerarse como correctos para ese lenguaje de programación.

La sintaxis de JavaScript es muy similar a la de otros lenguajes de programación como Java y C. Las normas básicas que definen la sintaxis de JavaScript son las siguientes:

- **No se tienen en cuenta los espacios en blanco y las nuevas líneas**: como sucede con XHTML, el intérprete de JavaScript ignora cualquier espacio en blanco sobrante, por lo que el código se puede ordenar de forma adecuada para entenderlo mejor (tabulando las líneas, añadiendo espacios, creando nuevas líneas, etc.)
- **Se distinguen las mayúsculas y minúsculas**: al igual que sucede con la sintaxis de las etiquetas y elementos XHTML. Sin embargo, si en una página XHTML se utilizan indistintamente mayúsculas y minúsculas, la página se visualiza correctamente, siendo el único problema la no validación de la página. En cambio, si en JavaScript se intercambian mayúsculas y minúsculas el script no funciona.
- **No se define el tipo de las variables**: al crear una variable, no es necesario indicar el tipo de dato que almacenará. De esta forma, una misma variable puede almacenar diferentes tipos de datos durante la ejecución del script.
- **No es necesario terminar cada sentencia con el carácter de punto y coma (;)**: en la mayoría de lenguajes de programación, es obligatorio terminar cada sentencia con el carácter ;. Aunque JavaScript no obliga a hacerlo, es conveniente seguir la tradición de terminar cada sentencia con el carácter del punto y coma (;).
- **Se pueden incluir comentarios**: los comentarios se utilizan para añadir información en el código fuente del programa. Aunque el contenido de los comentarios no se visualiza por pantalla, si que se envía al navegador del usuario junto con el resto del script, por lo que es necesario extremar las precauciones sobre la información incluida en los comentarios.

  JavaScript define dos tipos de comentarios: los de una sola línea y los que ocupan varias líneas.

  Ejemplo de comentario de una sola línea:

```
// a continuación se muestra un mensaje
alert("mensaje de prueba");
```

Figura 8: ejemplo de comentario de una línea en código JavaScript

Los comentarios de una sola línea se definen añadiendo dos barras oblicuas (//) al principio de la línea.

Ejemplo de comentario de varias líneas:

```
/* Los comentarios de varias líneas son muy útiles
cuando se necesita incluir bastante información
en los comentarios */
alert("mensaje de prueba");
```

Figura 9: ejemplo de comentario de varias líneas en código JavaScript

Los comentarios multi-línea se definen encerrando el texto del comentario entre los símbolos /* y */.

3.1.7. Posibilidades y limitaciones

Desde su aparición, JavaScript siempre fue utilizado de forma masiva por la mayoría de sitios de Internet. La aparición de Flash disminuyó su popularidad, ya que Flash permitía realizar algunas acciones imposibles de llevar a cabo mediante JavaScript.

Sin embargo, la aparición de las aplicaciones AJAX programadas con JavaScript le ha devuelto una popularidad sin igual dentro de los lenguajes de programación web.

En cuanto a las limitaciones, JavaScript fue diseñado de forma que se ejecutara en un entorno muy limitado que permitiera a los usuarios confiar en la ejecución de los scripts.

De esta forma, los scripts de JavaScript no pueden comunicarse con recursos que no pertenezcan al mismo dominio desde el que se descargó el script. Los scripts tampoco pueden cerrar ventanas que no hayan abierto esos mismos scripts. Las ventanas que se crean no pueden ser demasiado pequeñas ni demasiado grandes ni colocarse fuera de la vista del usuario (aunque los detalles concretos dependen de cada navegador).

Además, los scripts no pueden acceder a los archivos del ordenador del usuario (ni en modo lectura ni en modo escritura) y tampoco pueden leer o modificar las preferencias del navegador.

Por último, si la ejecución de un script dura demasiado tiempo (por ejemplo por un error de programación) el navegador informa al usuario de que un script está consumiendo demasiados recursos y le da la posibilidad de detener su ejecución.

A pesar de todo, existen alternativas para poder saltarse algunas de las limitaciones anteriores. La alternativa más utilizada y conocida consiste en firmar digitalmente el script y solicitar al usuario el permiso para realizar esas acciones.

3.1.8. JavaScript y navegadores

Los navegadores más modernos disponibles actualmente incluyen soporte de JavaScript hasta la versión correspondiente a la tercera edición del estándar ECMA-262.

La mayor diferencia reside en el *dialecto* utilizado, ya que mientras Internet Explorer utiliza JScript, el resto de navegadores (Firefox, Opera, Safari, Chrome) utilizan JavaScript.

Este tipo de diferencias generaba problemas a la hora del desarrollo de las funcionalidades de **inclusite**; cuando se probaban las funcionalidades en ocasiones no funcionaba en Internet Explorer ya que en JScript no existía la función que se llamaba a ejecutar.

3.2. AJAX

En algunas aplicaciones existe contenido que interactúa con el cliente, es decir que hay contenido que se actualiza dependiendo de la ejecución de la acción de un usuario e incluso en ocasiones esa actualización de contenido puede darse de forma automática por la aplicación web.

Ajax permite que **inclusite** detecte esos cambios de contenido dados por las acciones de un usuario permitiendo acceder a ese nuevo contenido.

El término AJAX [13] se presentó por primera vez en el artículo *Ajax: A New Approach to Web Applications*[5] publicado por Jesse James Garrett el 18 de Febrero de 2005. Hasta ese momento, no existía un término normalizado que hiciera referencia a un nuevo tipo de aplicación web que estaba apareciendo.

En realidad, el término AJAX es un acrónimo de ***Asynchronous JavaScript + XML***, que se puede traducir como "JavaScript asíncrono + XML".

El artículo define AJAX de la siguiente forma:

Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes.

Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

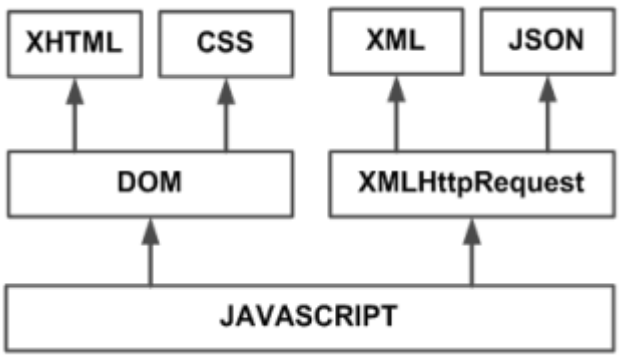


Figura 10: Tecnologías agrupadas bajo el concepto de AJAX

Desarrollar aplicaciones AJAX requiere un conocimiento avanzado de todas y cada una de las tecnologías anteriores.

En las aplicaciones web tradicionales, las acciones del usuario en la página (pinchar en un botón, seleccionar un valor de una lista, etc.) desencadenan llamadas al

[5] http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario.

En el siguiente esquema, la imagen de la izquierda muestra el modelo tradicional de las aplicaciones web. La imagen de la derecha muestra el nuevo modelo propuesto por AJAX:

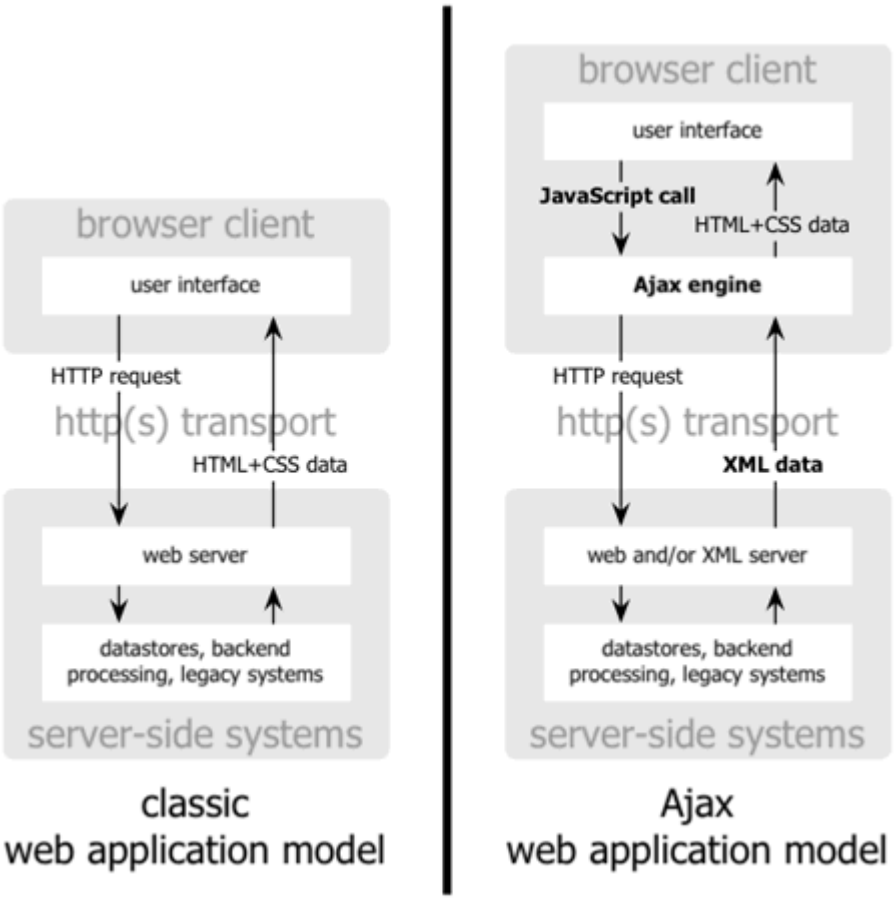


Figura 11: Comparación gráfica del modelo tradicional de aplicación web y del nuevo modelo propuesto por AJAX

Esta técnica tradicional para crear aplicaciones web funciona correctamente, pero no crea una buena sensación al usuario. Al realizar peticiones continuas al servidor, el usuario debe esperar a que se recargue la página con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, su uso se convierte en algo molesto

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

El siguiente esquema muestra la diferencia más importante entre una aplicación web tradicional y una aplicación web creada con AJAX. La imagen superior muestra la interacción síncrona propia de las aplicaciones web tradicionales. La imagen inferior muestra la comunicación asíncrona de las aplicaciones creadas con AJAX.

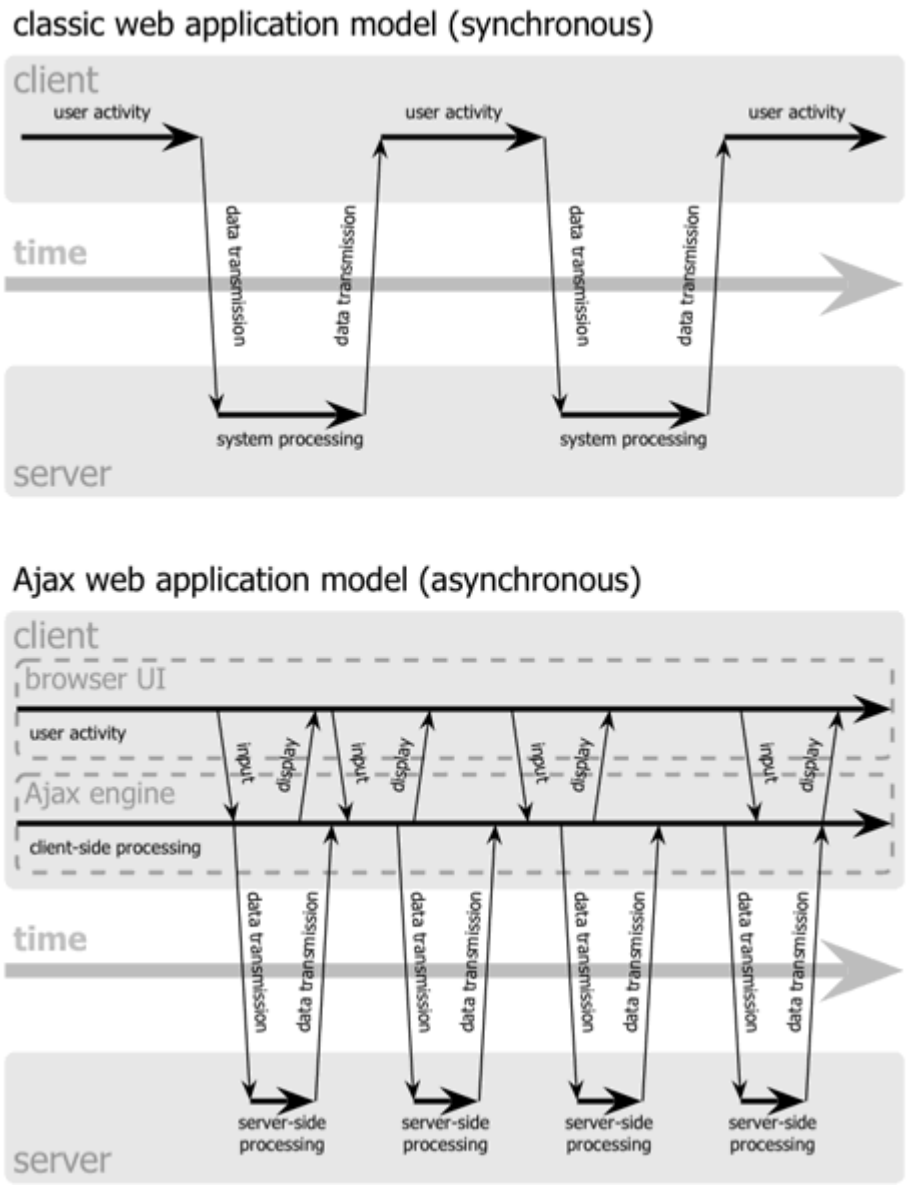


Figura 12: Comparación entre las comunicaciones síncronas de las aplicaciones web tradicionales y las comunicaciones asíncronas de las aplicaciones AJAX

Las peticiones HTTP al servidor se sustituyen por peticiones JavaScript que se realizan al elemento encargado de AJAX. Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX. En este caso, la interacción del usuario tampoco se ve interrumpida por recargas de página o largas esperas por la respuesta del servidor.

Desde su aparición, se han creado cientos de aplicaciones web basadas en AJAX. En la mayoría de casos, AJAX puede sustituir completamente a otras técnicas como Flash. Además, en el caso de las aplicaciones web más avanzadas, pueden llegar a sustituir a las aplicaciones de escritorio.

3.3. JSON

JSON [14] (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, se presentan de estas formas:

Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con “{“ (llave de apertura) y termine con “}” (llave de cierre). Cada nombre es seguido por “:” (dos puntos) y los pares nombre/valor están separados por “,” (coma).

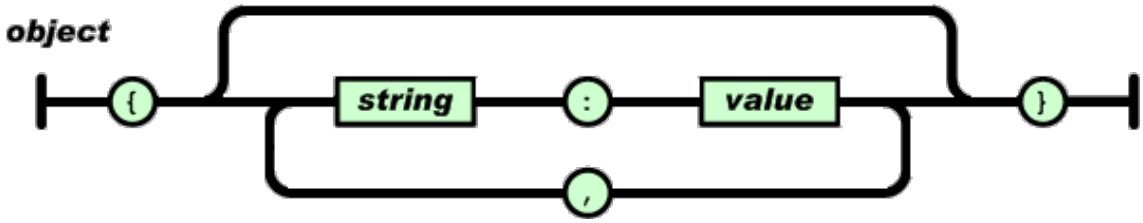


Figura 13: estructura de JSON nombre\valor

Un arreglo es una colección de valores. Un arreglo comienza con “[“ (corchete izquierdo) y termina con “]” (corchete derecho). Los valores se separan por “,” (coma).

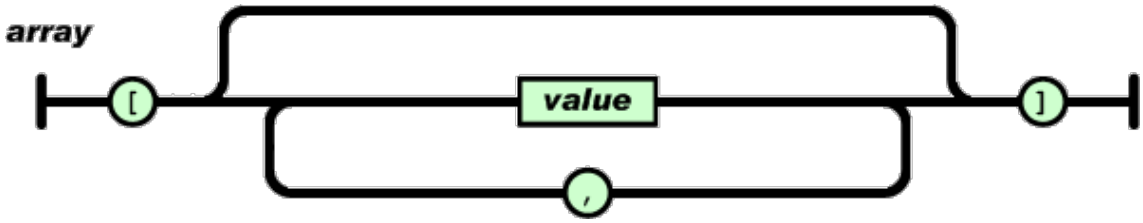


Figura 14: estructura de JSON lista ordenada de valores

Un valor puede ser una cadena de caracteres con comillas dobles, o un número, o true o false o null, o un objeto o un arreglo. Estas estructuras pueden anidarse.

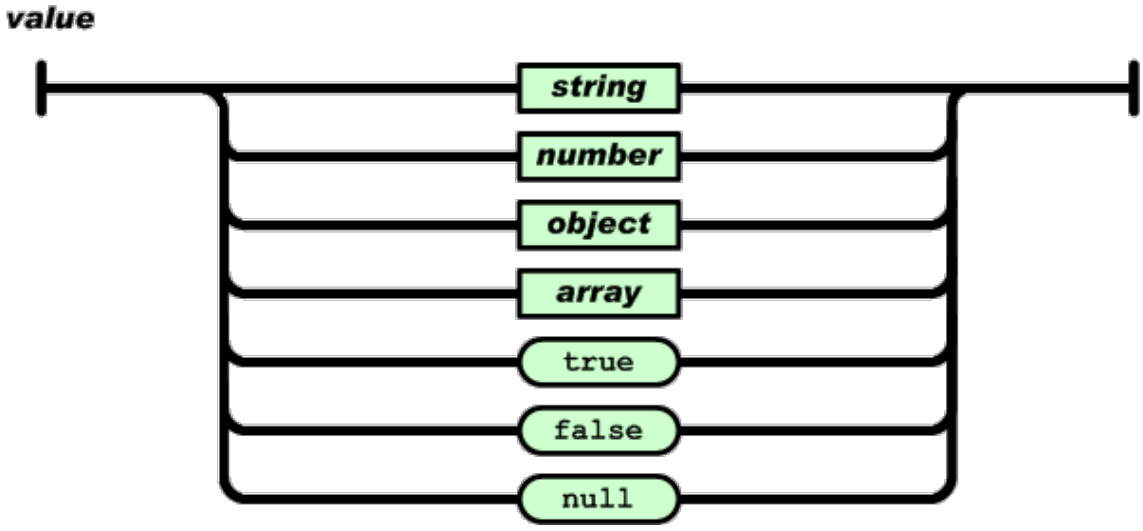


Figura 15: tipo de valores que soporta JSON

Una cadena de caracteres es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape. Un carácter está representado por una cadena de caracteres de un único carácter. Una cadena de caracteres es parecida a una cadena de caracteres C o Java.

Un número es similar a un número C o Java, excepto que no se usan los formatos octales y hexadecimales. Los espacios en blanco pueden insertarse entre cualquier par de símbolos.

3.4. ActionScript Adobe Flash

Adobe ActionScript [23] es el lenguaje de programación de la Plataforma Adobe Flash. Originalmente desarrollado como una forma para que los desarrolladores programen de forma más interactiva. La programación con ActionScript permite mucha más eficiencia en las aplicaciones de la plataforma Flash para construir animaciones de todo tipo, desde simples a complejas, ricas en datos e interfaces interactivas.

La versión más extendida actualmente es ActionScript 3.0, que significó una mejora en el manejo de programación orientada a objetos al ajustarse mejor al estándar ECMA-262 y es utilizada en las últimas versiones de Adobe Flash y Flex y en anteriores versiones de Flex. Desde la versión 2 de Flex viene incluido ActionScript 3, el cual mejora su rendimiento en comparación de sus antecesores, además de incluir nuevas características como el uso de expresiones regulares y nuevas formas de empaquetar las clases.

3.5. SQUID

Squid [18] es un popular programa de software libre que implementa un servidor proxy y un *dominio* para caché de páginas web, publicado bajo licencia GPL. Tiene una amplia variedad de utilidades, desde acelerar un servidor web, guardando en caché peticiones repetidas a DNS y otras búsquedas para un grupo de gente que comparte recursos de la red, hasta caché de web, además de añadir seguridad filtrando el tráfico. Está especialmente diseñado para ejecutarse bajo entornos tipo Unix.

Squid ha sido desarrollado durante muchos años y se le considera muy completo y robusto. Aunque orientado principalmente a HTTP y FTP es compatible con otros protocolos como Internet Gopher. Implementa varias modalidades de cifrado como TLS, SSL, y HTTPS.

Squid posee las siguientes características:

- **Proxy y Caché de HTTP, FTP, y otras URL:** Squid proporciona un servicio de Proxy que soporta peticiones http, HTTPS y FTP a equipos que necesitan acceder a Internet y a su vez provee la funcionalidad de caché especializado en el cual almacena de forma local las páginas consultadas recientemente por los usuarios. De esta forma, incrementa la rapidez de acceso a los servidores de información Web y FTP que se encuentra fuera de la red interna.
- **Proxy para SSL:** Squid también es compatible con SSL (*Secure Socket Layer*) con lo que también acelera las transacciones cifradas, y es capaz de ser configurado con amplios controles de acceso sobre las peticiones de usuarios.
- **Jerarquías de caché:** Squid puede formar parte de una jerarquía de caches. Diversos proxys trabajan conjuntamente sirviendo las peticiones de las páginas. Un navegador solicita siempre las páginas a un sólo proxy, si este no tiene la página en la caché hace peticiones a sus hermanos, que si tampoco las tienen las hacen a su/s padre/s... Estas peticiones se pueden hacer mediante dos protocolos: HTTP e ICMP.
- **ICP, HTCP, CARP, *caché digests:*** Squid sigue los protocolos ICP, HTCP, CARP y *caché digests* que tienen como objetivo permitir a un proxy "preguntarle" a otros proxys caché si poseen almacenado un recurso determinado.
- **Caché transparente:** Squid puede ser configurado para ser usado como proxy transparente de manera que las conexiones son enrutadas dentro del proxy sin configuración por parte del cliente, y habitualmente sin que el propio cliente conozca de su existencia. De modo predefinido Squid utiliza el puerto 3128 para atender peticiones, sin embargo se puede especificar que lo haga en cualquier otro puerto disponible o bien que lo haga en varios puertos disponibles a la vez.
- **WCCP:** A partir de la versión 2.3 Squid implementa WCCP (*Web Cache Control Protocol*). Permite interceptar y redirigir el trafico que recibe un router hacia uno o más proxys caché, haciendo control de la conectividad de los mismos. Además permite que uno de los proxys caché designado pueda

determinar cómo distribuir el tráfico redirigido a lo largo de todo el array de proxys caché.

- **Control de acceso:** Ofrece la posibilidad de establecer reglas de control de acceso. Esto permite establecer políticas de acceso en forma centralizada, simplificando la administración de una red.
- **Aceleración de servidores HTTP:** Cuando un usuario hace petición hacia un objeto en Internet, este es almacenado en el caché, si otro usuario hace petición hacia el mismo objeto, y este no ha sufrido modificación alguna desde que lo accedió el usuario anterior, Squid mostrará el que ya se encuentra en el caché en lugar de volver a descargarlo desde Internet. Esta función permite navegar rápidamente cuando los objetos ya están en el caché y además optimiza enormemente la utilización del ancho de banda.
- **SNMP:** Squid permite activar el protocolo SNMP, este proporciona un método simple de administración de red, que permite supervisar, analizar y comunicar información de estado entre una gran variedad de máquinas, pudiendo detectar problemas y proporcionar mensajes de estados.
- **Caché de resolución DNS:** Squid está compuesto también por el programa *dnsserver*, que se encarga de la búsqueda de nombres de dominio. Cuando Squid se ejecuta, produce un número configurable de procesos *dnsserver*, y cada uno de ellos realiza su propia búsqueda en DNS. De este modo, se reduce la cantidad de tiempo que la caché debe esperar a estas búsquedas DNS.

El proxy caché es una manera de guardar los objetos solicitados de Internet (por ejemplo, datos como páginas web) disponibles vía protocolos HTTP, FTP y Gopher en un sistema más cercano al lugar donde se piden. Los navegadores web pueden usar la caché local Squid como un servidor proxy HTTP, reduciendo el tiempo de acceso así como el consumo de ancho de banda. Esto es muchas veces útil para los proveedores de servicios de Internet para incrementar la velocidad de sus consumidores y para las redes de área local que comparten la conexión a Internet.

Debido a que también es un proxy (es decir, se comporta como un cliente en lugar del cliente real), puede proporcionar un cierto grado de anonimato y seguridad. Sin embargo, también puede introducir problemas significativos de privacidad ya que puede registrar mucha información, incluyendo las URL solicitadas junto con otra información adicional como la fecha de la petición, versión del navegador y del sistema operativo, etc.

Un programa cliente (por ejemplo, un navegador) o bien tiene que especificar explícitamente el servidor proxy que quiere utilizar (típico para consumidores de ISP) o bien podría estar usando un proxy sin ninguna configuración extra. A este hecho se le denomina caché transparente, en el cual todas las peticiones HTTP son interceptadas por squid y todas las respuestas guardadas en caché. Esto último es típico en redes corporativas dentro de una red de acceso local y normalmente incluye los problemas de privacidad mencionados previamente.

Squid tiene algunas características que pueden facilitar establecer conexiones anónimas. Características tales como eliminar o modificar campos determinados de la cabecera de peticiones HTTP de los clientes. Esta política de eliminación y alteración de cabeceras se establece en la configuración de Squid. El usuario que solicita páginas a través de una red que utiliza Squid de forma transparente, normalmente no es consciente de este proceso o del registro de información relacionada con el proceso.

3.6. TTS

TTS [19], el habla sintética es una voz artificial (no pregrabada), generada mediante un proceso de sintetización del habla.

La síntesis de habla es la producción artificial de habla humana. Un sistema usado con este propósito recibe el nombre de sintetizador de habla y puede llevarse a cabo en software o en hardware. La síntesis del habla se llama a menudo en inglés text-to-speech (TTS), en referencia a su capacidad de convertir texto en habla. Sin embargo, hay sistemas que en lugar de producir voz a partir de texto lo hacen a partir de representación lingüística simbólica en habla.

La *calidad* de una voz sintética vendrá dada por:

- Su *inteligibilidad*: ¿con qué facilidad/dificultad es entendida?
- Su *naturalidad*: ¿en qué medida se asemeja a la voz *real* de un humano?

Un sistema texto a voz se compone de dos partes: un front-end y un back-end. A grandes rasgos, el front-end toma como entrada texto y produce una representación lingüística fonética. El back-end toma como entrada la representación lingüística simbólica y produce una forma de onda sintetizada.

El front-end desempeña dos tareas principales. Primero, toma el texto y convierte partes problemáticas como números y abreviaturas en palabras equivalentes. Este proceso se llama a menudo *normalización de texto* o *pre-procesado*. Entonces asigna una transcripción fonética a cada palabra, y divide y marca el texto en varias unidades prosódicas, como frases y oraciones. El proceso de asignar transcripciones fonéticas a las palabras recibe el nombre de conversión *texto a fonema* (TTP en inglés) o *grafema a fonema* (GTP en inglés). La combinación de transcripciones fonéticas e información prosódica constituye la *representación lingüística fonética*.

La otra parte, el back-end, toma la representación lingüística simbólica y la convierte en sonido. El back-end se llama a menudo sintetizador.

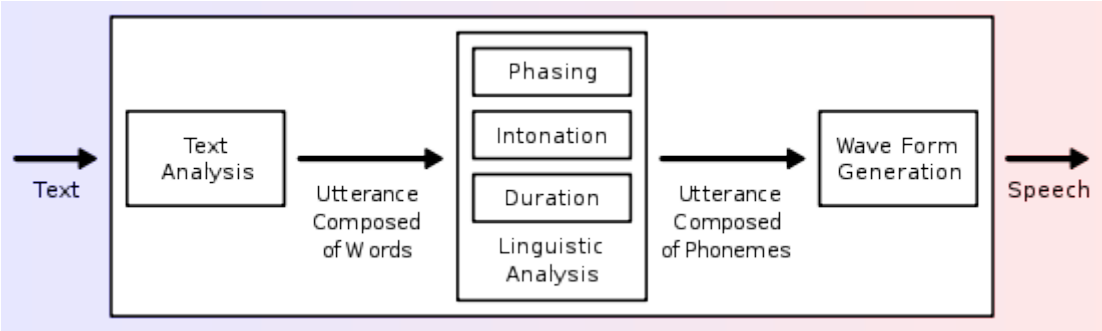


Figura 16: Un típico sistema de TTS.

3.7. ASR

ASR [20] conocida por su nombre en inglés Automatic Speech Recognition, o en español reconocimiento automático del habla, es una rama de la inteligencia artificial que tiene como objetivo permitir la comunicación hablada entre seres humanos y orenadores. El problema que se plantea en un sistema de ASR es el de hacer cooperar un conjunto de informaciones que provienen de diversas fuentes de conocimiento (acústica, fonética, fonológica, léxica, sintáctica, semántica y pragmática), en presencia de ambigüedades, incertidumbres y errores inevitables para llegar a obtener una interpretación aceptable del mensaje acústico recibido.

Un sistema de reconocimiento del habla es una herramienta computacional capaz de procesar la señal de voz emitida por el ser humano y reconocer la información contenida en ésta, convirtiéndola en texto o emitiendo órdenes que actúan sobre un proceso. En su desarrollo intervienen diversas disciplinas, tales como: la fisiología,

la acústica, el procesamiento de señales, la inteligencia artificial y la ciencia de la computación.

Las fuentes de conocimiento sintáctico, semántico y pragmático dan lugar al modelo del lenguaje del sistema. Cuando la representación de la Sintaxis y de la Semántica tiende a integrarse, se desarrollan sistemas de ASR de gramática restringida para tareas concretas.

El reconocimiento de la gramática restringida trabaja reduciendo las típicas frases reconocidas a un tamaño más pequeño que la gramática formal. Este tipo de reconocimiento trabaja mejor cuando el hablante proporciona respuestas breves a cuestiones o preguntas específicas: las preguntas de “si” o “no”, al elegir una opción del menú, un artículo de una lista determinada, etc. La gramática especifica las palabras y frases más típicas que una persona diría como respuesta rápida y después asocia esas palabras o frases a un concepto semántico. Por ejemplo, un “si” puede entenderse cuando se oye un “sip”, “vale”, “yes” o “okey”, y un “no” con un “nop”, “nada” o “en absoluto”.

Si el hablante dice algo que gramaticalmente no tiene sentido, el reconocimiento fallará. Normalmente, si el reconocimiento falla, la aplicación incitará al usuario a repetir lo que ha dicho y el reconocimiento se intentará de nuevo. Si el sistema está correctamente diseñado y es repetidamente incapaz de entender al usuario (debido a que no se ha entendido bien la pregunta, un acento cerrado, interferencias o demasiado ruido alrededor), se retirará y desviará la llamada a otro operador. La investigación muestra que las llamadas a las que se las pide replantear la pregunta o cuestión una y otra vez, en poco tiempo se frustran y se agitan.

Aunque en teoría cualquier tarea en la que se interactúe con un ordenador puede utilizar el reconocimiento de voz, actualmente las siguientes aplicaciones son las más comunes:

- **Dictado automático**: El dictado automático es, en el 2007, el uso más común de las tecnologías de reconocimiento de voz. En algunos casos, como en el dictado de recetas médicas y diagnósticos o el dictado de textos legales, se usan corpus especiales para incrementar la precisión del sistema.
- **Control por comandos**: Los sistemas de reconocimiento de habla diseñados para dar órdenes a un computador (p.ej. "Abrir Firefox", "cerrar ventana") se llaman Control por comandos. Estos sistemas reconocen un vocabulario muy reducido, lo que incrementa su rendimiento.

- **Telefonía**: Algunos sistemas PBX permiten a los usuarios ejecutar comandos mediante el habla, en lugar de pulsar tonos. En muchos casos se pide al usuario que diga un número para navegar un menú.
- **Sistemas portátiles**: Los sistemas portátiles de pequeño tamaño, como los relojes o los teléfonos móviles, tienen unas restricciones muy concretas de tamaño y forma, así que el habla es una solución natural para introducir datos en estos dispositivos.
- **Sistemas diseñados para discapacitados**: Los sistemas de reconocimiento de voz pueden ser útiles para personas con discapacidades que les impidan teclear con fluidez, así como para personas con problemas auditivos, que pueden usarlos para obtener texto escrito a partir de habla. Esto permitiría, por ejemplo, que los aquejados de sordera pudieran recibir llamadas telefónicas.

3.8. WAI ARIA

WAI ARIA [21] es una iniciativa del W3C que responde a la necesidad de proporcionar accesibilidad a las aplicaciones web enriquecidas. Pretende ser una ayuda en el contenido dinámico de los interfaces actuales de las aplicaciones web, que cada día son más parecidos a los entornos de escritorio y que su propio funcionamiento puede interferir en los productos de apoyo.

Esta tecnología tiene como principal objetivo aportar información acerca de las diferentes partes que constituyen los contenidos dinámicos generados, normalmente, por medio de *scripts*. Toda esta información será utilizada por los productos de apoyo para la interacción con el usuario final.

Los documentos técnicos de WAI ARIA están desarrollados por el grupo de trabajo del W3C PFWG (Protocols and Formats Working Group). El W3C pone a disposición varios documentos sobre WAI ARIA, entre ellos, destaca la especificación técnica[6].

WAI ARIA proporciona una serie de atributos que funcionan como identificadores de las diferentes partes de la aplicación que interactúa con el usuario. También se

[6] http://www.w3.org/TR/wai-aria/.

incluyen mapeo de controles y eventos para la accesibilidad de las APIs (Application Programming Interfaces).

WAI ARIA dispone de roles que describen tanto los *widgets* (componentes con funcionalidad propia de las interfaces de escritorio o web) de la aplicación como la estructura de la página web, como por ejemplo: los encabezados y las regiones. También dispone de varias propiedades como los estados de los *widgets*, las regiones activas de actualización de contenidos y sobre características *drag-and-drop*. A su vez, provee una manera de navegar mediante teclado dentro de los componentes.

Al usar esta tecnología se encuentran una serie de ventajas:

- Al añadir valor semántico al contenido y a los *widgets* se puede situar al usuario exactamente en donde está. Además de su facilidad de implementación ya que son los atributos los que los define.
- Las actualizaciones de contenido dinámico, normalmente realizado con tecnologías de *script,* son notificadas al usuario por medio de su producto de apoyo.
- Accesibilidad de los *widgets* por medio del teclado.
- Se dota de información de cómo se utiliza un *widget* y qué tipo de datos proporciona.

En cambio, se puede identificar una desventaja manifiesta, ya que al aplicar esta tecnología, los documentos web que se desarrollen no validarán tanto en HTML 4 como en XHTML 1.0. El primero no soporta espacio de nombres, por lo que no se podrá incluir los atributos específicos de WAI ARIA. Con XHTML 1.0, su validación no sería posible porque no estaría incluido en el esquema que define al lenguaje.

Una posible solución sería incluir los atributos de WAI ARIA mediante DOM, es decir, incluir con JavaScript, dinámicamente, los atributos en los elementos destinados a ellos. Dicha solución haría dependiente de la tecnología *script* cualquier tipo de implementación.

Existe otra solución que podría resolver el problema: la utilización de XHTML 1.1. Este lenguaje es una especificación de lenguaje modular y extensible que permitiría personalizar el DTD incorporando el esquema de WAI ARIA.

HTML fue creado para la presentación de documentos textuales con formatos específicos para su función, como puedan ser los encabezados, marcadores de párrafos o cualquier otro tipo de elementos que tuvieran que ver con el formateo de texto. Esto es así desde que en las primeras versiones se incluyó el elemento `IMG` para la incorporación de elementos de imágenes dentro de los documentos.

El lenguaje fue desarrollado para un modelo de cliente-servidor, donde el usuario pedía el documento y desde un servidor se lo proporcionaba. Esta comunicación está orientada de forma secuencial, es decir, un usuario envía una petición a un servidor, normalmente de un documento HTML, y el servidor lo procesa para, finalmente, enviarlo al usuario final.

Con la evolución de la web, las páginas han ido evolucionando hacia las llamadas aplicaciones web, mucho más parecidas a los entornos de escritorio que tienen un funcionamiento más dinámico con respecto a los contenidos. Para ello, se han utilizado tecnologías web como JavaScript, y más concretamente AJAX, que de unos años hasta hoy ha tenido un crecimiento sustancial.

Con estas técnicas, al hacer operaciones y procesamientos desde el lado del cliente, se afecta de forma notable a la accesibilidad de los sitios web, ya que los productos de apoyo no reconocen los componentes o *widgets* insertados.

Estas tecnologías pueden cargar datos sin tener que actualizar toda la página web, es decir, existen regiones activas que envían peticiones al servidor en un segundo plano. Muchos productos de apoyo no reconocen estas áreas activas o, simplemente, no pueden reconocer que había información nueva, ya que la propia página no había sido actualizada por completo desde el lado del servidor.

Existen en HTML componentes de interfaz, como los existentes para los formularios, que son insuficientes para la gran cantidad de funcionalidades aparecidas en los últimos años. Para ello se han recurrido a técnicas de *scripting* o de objetos para suplir esta carencia. También la presentación o el estilo de los mismos ha sido, en ocasiones, modificado para que fuese más atractivo. Es el caso de los *radiobuttons* en HTML que, en muchas ocasiones, son personalizados en *sliders* con varios estados o posiciones que apuntan a un determinado valor.

También está presente el problema de la independencia de dispositivo, ya que algunos componentes de las aplicaciones web enriquecidas no son accesibles desde el teclado, orientando su uso al ratón. Esta característica entraña un grave problema de

accesibilidad, dado que muchas personas con discapacidad física son incapaces de utilizar el ratón.

WAI ARIA propone a los desarrolladores una serie de soluciones destinadas a hacer accesibles *widgets*, áreas activas y demás componentes enriquecidos que se encuentran en la mayoría de las aplicaciones web en la actualidad.

Para ello describen roles y propiedades con la finalidad de dotar de información a los productos de apoyo y para que interactúen adecuadamente con los componentes más normales de las aplicaciones web.

Los roles pueden ser de dos tipos:

- Los primeros describen el tipo de *widget* del que se trata. Los *widgets* pueden ser de varios estilos, desde menús desplegables en forma de árbol, hasta *sliders,* pasando por barras de medición de progreso.
- El segundo tipo de rol es aquel que define la estructura de la página web. Estos roles pueden definir elementos de encabezado o incluso tablas o *grid*, es muy común, también, definir diferentes regiones.

Las propiedades pueden describir el estado de los propios *widgets*: un *widget* que funcione como un *checkbox* puede tener un estado de *checked,* por ejemplo. También se utilizan para definir regiones activas donde se actualizarán contenidos proporcionando políticas de aviso de actualización. Otra funcionalidad de las propiedades de WAI ARIA es aquella destinada a describir los *drag-and-drop,* definiendo las fuentes *drag* y los objetivos *drop*.

Como se ha comentado antes, los roles dentro de WAI ARIA sirven para describir los *widgets,* así como para definir elementos propios de las páginas web. Estos atributos proporcionan información a los productos de apoyo sobre los componentes *widgets*.

Es evidente que, por definición, las etiquetas de HTML tienen un rol predefinido, el mismo que el propio elemento indica. Un rol de WAI ARIA ayuda a concretar la funcionalidad de la etiqueta en la que va definida y se podría afirmar que la sustituye. A continuación se ve un ejemplo en forma de código:

```
<ul role="toolbar" tabindex="0">
  <li id="boton1">Copiar</li>
  <li id="boton2">Pegar</li>
  <li id="boton3">Cortar</li>
</ul>
```

Figura 17: Código HTML con atributo role.

El código anterior representa un *widget* sencillo. En este caso el desarrollador ha querido representar una barra de herramientas con las tres funcionalidades más populares de la edición de texto, como son Cortar, Pegar y Cortar. Se intuye que estas opciones están dotadas de una funcionalidad mediante *script*, como pueda ser JavaScript. Por defecto, el elemento `<UL>` tiene de por sí un significado propio como etiqueta de lista. En el contexto de la página web se sabe que tendrá una función muy determinada, como barra de herramientas, ahí entra en valor el atributo de WAI ARIA `role` que viene a renombrar su definición concretándola aún más, en este caso con el valor `toolbar`. Existen otros roles predefinidos como `log`, `progressbar` o `timer` (se puede encontrar la lista completa en la siguiente dirección: http://www.w3.org/TR/wai-aria/roles).

Los roles también pueden definir regiones de un documento web. Es común poder identificar partes sustanciales de una página web, más allá de los predefinidos por las etiquetas de HTML dentro de un nivel de precisión más alto. Los roles *Landmark* son un tipo que permite dotar de un significado a ciertas regiones que tienen una funcionalidad o propósito bien marcado y relevante para el uso de cualquier usuario. Las regiones destinadas a la navegación o a la búsqueda de términos pueden ser consideradas como *Landmarks.* Algunos tipos de roles *Landmarks* son por ejemplo: `contentinfo`, `form`, `banner` o `search`.

Existen otros tipos de roles que definen estructuras dentro del documento o página web. No suelen representar elementos interactivos con los usuarios y un ejemplo pueden ser los valores como `group`, `region` o `row`.

También se pueden encontrar roles llamados abstractos, pero no se han de utilizar por parte de los desarrolladores web ya que sirven para completar la ontología en la especificación WAI ARIA.

Para cualquier desarrollador web puede resultar fácil implementar este atributo en una página web bien estructurada. Es común proporcionar nombres semánticos a ciertos elementos en los atributos `id` o `class`. Estos atributos dan pistas del propósito del contenido de dichos elementos. Se verá con el siguiente ejemplo:

```
<div id="navegacion">
      <! - contenido de navegacion -- >
</div>
<div id="contenidoPrincipal">
      <!—contenido principal -- >
</div>
<div id="anuncios">
<!—contenido anuncios-- >
</div>
```

Figura 18: Código HTML de una página web bien estructurada.

En este ejemplo podemos ver tres bloques `<DIV>` cuyos atributos *`id`* poseen nombres semánticos. Este tipo de atributos pueden servir para guiarnos e incluir atributos *role* de WAI ARIA en el documento HTML. Se verá como quedaría el código:

```
<div id="navegacion" role="navigation">
      <! - contenido de navegacion -- >
</div>
<div id="contenidoPrincipal" role="main">
      <!—contenido principal -- >
</div>
<div id="anuncios" role="banner">
<!—contenido anuncios-- >
</div>
```

Figura 19: Código HTML con el atributo role en sus componentes.

En este ejemplo de código se han añadido los atributos de WAI ARIA *`role`* que aportan un valor semántico que se ha recogido de los atributos *`id`*. Se verá que el `<DIV>` con *`id`* navegación puede corresponder con un *`role`* de tipo *`navigation`*, y, como a su vez, el `<DIV>` con *`id contenidoPrincipal`* puede corresponder al role con valor *`main`*. Aunque ésta no es una técnica fidedigna, y siempre se ha de corresponder con el criterio de los desarrolladores, aporta pistas y muestra la visibilidad de la correspondencia semántica de algunos atributos y su aprovechamiento para las técnicas de WAI ARIA.

4. Análisis de inclusite

En este apartado se hablará sobre el análisis llevado con el fin de crear **inclusite.**

Se inicia con una visión general de lo que es y pretende inclusite, luego se redacta los requerimientos que debe cumplir el sistema, seguido se explicará su arquitectura, en el cuarto punto se muestra los diagramas de casos de uso de la interacción del usuario con la herramienta (configuración e interfaces de entrada y salida), como último punto el uso de WAI-ARIA como estándar base para el funcionamiento de inclusite.

4.1. Visión general

La mayoría de sitios web se limitan a la adaptación de contenido web no accesible a normativas tales como WAI, donde la idea principal radica en hacer la Web más accesible para todos los usuarios independientemente de las circunstancias y los dispositivos involucrados a la hora de acceder a la información. Partiendo de esta idea, una página accesible lo sería tanto para una persona con discapacidad, como para cualquier otra persona que se encuentre bajo circunstancias externas que dificulten su acceso a la información (en caso de ruidos externos, en situaciones donde nuestra atención visual y auditiva no están disponibles, pantallas con visibilidad reducida, etc.)[22].

La plataforma inclusite es una herramienta que pretende dar acceso a los contenidos a los usuarios con diferentes discapacidades en cualquier sitio web que esté adaptado de una forma que no es percibida ni por los usuarios sin discapacidades, para esa adaptación se usa como base WAI-ARIA.

Uno de los principales objetivos de inclusite es ofrecer el contenido al usuario discapacitado de la misma forma en que lo recibe el usuario habitual, cuando inclusite está activo, éste le presta al usuario una interfaz para que pueda interactuar con el contenido, navegar por el sitio web, etc. lo especial de la interfaz es que es adecuada a las capacidades de cada usuario, además inclusite no necesita de que el usuario final aporte un hardware o software adicional, diferentes a los habituales, es decir pueden hacer uso de un micrófono, un teclado, unos parlantes o auriculares, etc. los cuales actualmente, vienen incluidos en un ordenador.

Dado a que es difícil saber a priori los requisitos necesarios para hacer posible la navegación de un usuario en un sitio web que tenga inclusite, se hace uso de una configuración previa por parte del usuario, la cual en algunos casos se necesitará la ayuda de un tercero. Esta configuración se guardará para cuando el usuario vuelva hacer uso de inclusite, o incluso si entra a otro dominio diferente a ese donde ha configurado por primera vez.

Otra característica de inclusite es que enriquece el aspecto de la página web con elementos visuales o sonoros dependiendo de la interfaz que esté usando el usuario, facilitándole aun más la navegación.

Inclusite es una tecnología no intrusiva, es decir la estructura original del sitio web no se modifica más que añadir un script para que funcione, tal como lo hace Google Analytics, cuando se inicia inclusite en el sitio web, este inicia un servicio que agrega atributos a las etiquetas HTML del sitio web, las cuales permiten a inclusite saber la semántica de la web.

4.2. Requerimientos Front-end inclusite

Como parte de desarrollo de una aplicación siempre estará presenta la parte de diseño, pero para que exista una parte de diseño deben existir unos requerimientos previos, por lo tanto a continuación, se plantearan lo que se quiere que haga la aplicación llamada inclusite.

El principal objetivo del sistema, que desde ahora se le llamará inclusite; es darle acceso al contenido de un sitio web a un usuario con alguna discapacidad, sea visual, motriz o intelectual, usando las herramientas que ordenador de gamma baja pueda tener.

Debido a que inclusite no será usado por usuarios con determinado tipo de discapacidad, se debe prestar al usuario una forma de escoger la manera de navegar sobre el contenido de la web, es decir que por medio de una configuración previa, permitir que el usuario tenga una interfaz adaptada a su discapacidad. Para comodidad de los usuarios se debe guardar la configuración hecha por ellos, para que la siguiente vez que entren a un sitio web adaptado inclusite se active automáticamente.

Existirá un servicio TTS que reproduzca el contenido de la web, para tal caso, el usuario también podrá decidir si este servicio se activa o no. Las alternativas que el usuario tendrá para navegar por el contenido un sitio web, son que pueda usar el teclado, por comandos de voz (para tal caso habrá un servicio ASR), o si el usuario no puede moverse y tampoco puede hablar, inclusite tendrá que dar la opción de que dicho usuario pueda ejecutar acciones emitiendo sonidos.

Por usabilidad inclusite tiene que prestar la manera de navegar por el contenido de una forma lógica a través de la semántica de la web, de esta forma el usuario podrá saber en qué área de la página se encuentra, además el sistema inclusite, debe permitir que el usuario entre y salga de dichas áreas. Si llegado el caso, el usuario se siente perdido dentro del contenido, el sistema debe permitir que el usuario pueda ir a una área que establezca la plantilla general de la web, es decir; que el usuario en esa área general encuentre el encabezado, el contenido principal, el pie de página o cualquier otra cosa que exista dentro de la estructura general del documento.

Normalmente cuando un usuario navega por un sitio web e intenta llegar a un contenido en concreto o de interés para él se hacen varios saltos por varias páginas o una página del sitio, inclusite debe prestar la posibilidad de que exista un área por defecto, además si el usuario está en otra área que no sea esa por defecto, él podrá con un solo comando llegar a ella.

Además que el usuario pueda acceder a texto e imágenes en el contenido, también se le debe dejar interactuar con contenido multimedia, rellenar y enviar formularios.

En el caso que se use el servicio TTS, al usuario se le debe indicar de una forma auditiva las opciones de navegación que tiene de acuerdo la estructura del contenido de la web, en el contenido multimedia y en formularios, incluso si él lo desea que se repitan, tal cual permite repetir las opciones de navegación, que admita también un comando para repetir la lectura del contenido.

Como toda herramienta software, inclusite también permitirá al usuario acceder a una ayuda dependiendo de la configuración que tenga, tanto en la navegación como en los contenidos multimedia y formularios. Para terminar el usuario también podrá desactivar el sistema.

4.3. Arquitectura de inclusite

La idea principal de inclusite es recoger el tráfico de la web de tipo accesible (llamaremos tráfico accesible a todo aquel tráfico de usuarios que necesitan la web accesible, independientemente de la interfaz final a utilizar), y pasarlo por un motor de reglas que es un analizador de HTML, con esto se pretende modificar en primera instancia la REQUEST (petición a servidor) y la RESPONSE (respuesta de servidor) en su fase final y analizar el HTML para hacerlo accesible.

El cambio del HTML se quiere modificar basándose en un motor de reglas para la inserción de estos atributos, TAGS, etc., de tipo accesible basados en el estándar WAI-ARIA, estas reglas dictarán en primera instancia la página y en instancias inferiores los TAGS de esa pagina o atributos a modificar, así al final de la ejecución de las reglas, se obtiene un HTML accesible, que se devolverá al usuario.

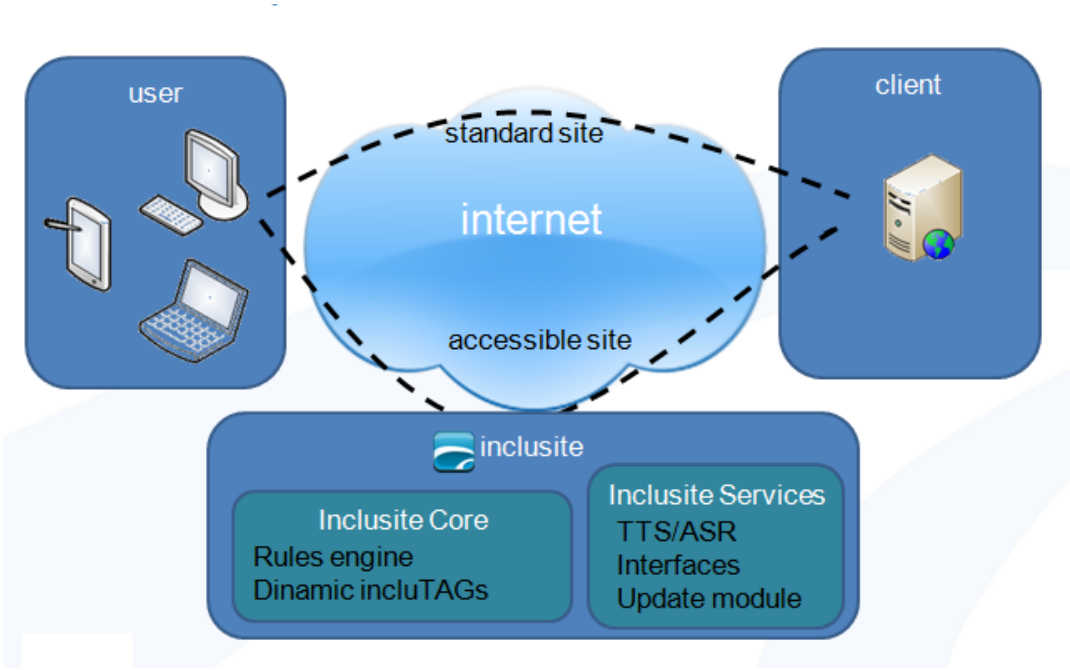


Figura 20: Arquitectura lógica de inclusite.

Después de pasar por el motor de reglas, de acuerdo a la configuración que allá escogido el usuario se prestarán los servicios correspondientes.

Por otra parte la arquitectura física viene determinada por la arquitectura lógica y por la premisa de que el sistema resultante sea escalable. Se tiene 2 grandes bloques; el servidor proxy y el servidor de servicios (TTS, ASR, JS, Analizador HTML).

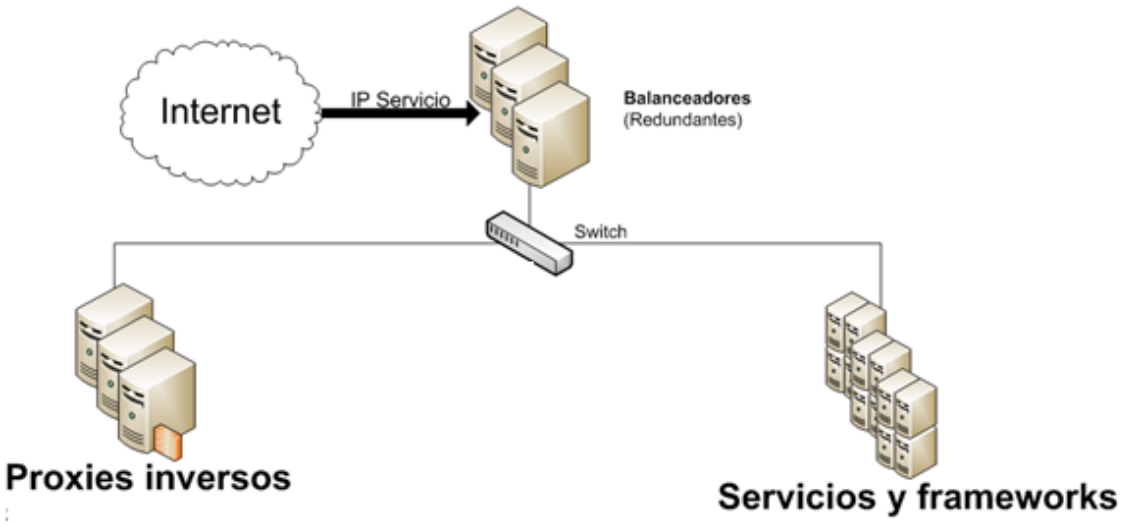


Figura 21: Arquitectura física de inclusite.

Para una mejor comprensión de la solución, podemos ver el flujo de llamadas y respuestas en base a la arquitectura mencionada, para ver cómo interaccionan los diferentes bloques.

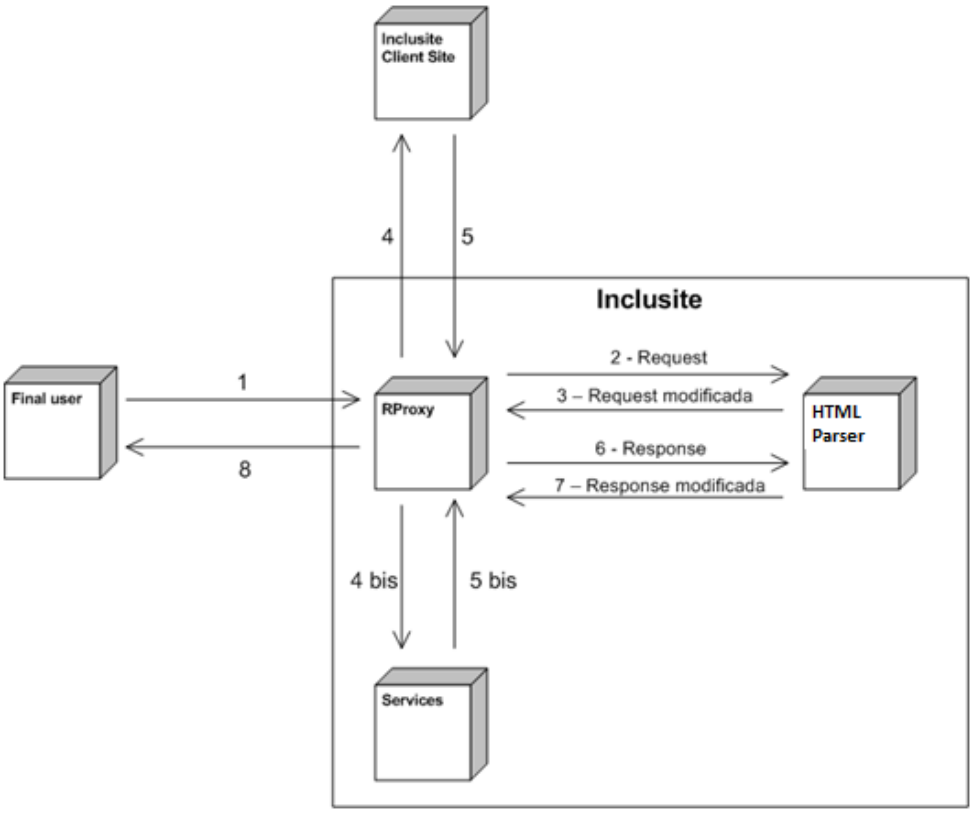


Figura 22: Flujo de llamadas a la arquitectura inclusite.

1 - El usuario visita la web del cliente y pincha la opción de usar Inclusite (eligiendo una interfaz de entrada y una de salida). La petición llega al proxy.

2 - El proxy envía la petición al analizador de HTML.

3 - El analizador de HTML devuelve la petición modificada en base a unas reglas.

4 - En función de la petición devuelta por el analizador de HTML se hace una petición a la web del cliente (la página web que quiere visitar el usuario).

5 - Respuesta de la web del cliente.

4 bis - En función de la petición devuelta por el analizador de HTML se hacen peticiones a los servicios (Frameworks, TTS, ASR).

5 bis – Respuestas de los servicios.

6 - Se envía la respuesta al analizador de HTML.

7 - El analizador de HTML devuelve la respuesta modificada, en base a unas reglas. En esta respuesta se añaden los interfaces de salida que ha elegido el usuario.

8 - Se envía la respuesta al usuario.

4.4. Casos de uso

Después de establecer la arquitectura del sistema y tomando en cuento los requerimientos establecidos, a continuación se presentan los diagramas de casos de uso de la funcionalidades que inclusite prestará a los usuarios, desarrollados en la fase de análisis. Para comenzar el listado de actores en el sistema.

Actores:

- **Usuario:** Persona con o sin discapacidad que hará uso de inclusite.
- **Servicio:** Es uno o más servicios (TTS y ASR) prestados por el back-end del sistema.

4.4.1. Diagrama de casos de uso general del sistema

La siguiente figura muestra el diagrama de casos de uso del funcionamiento general de inclusite. Luego se dará una descripción de cada uno de los casos de uso.

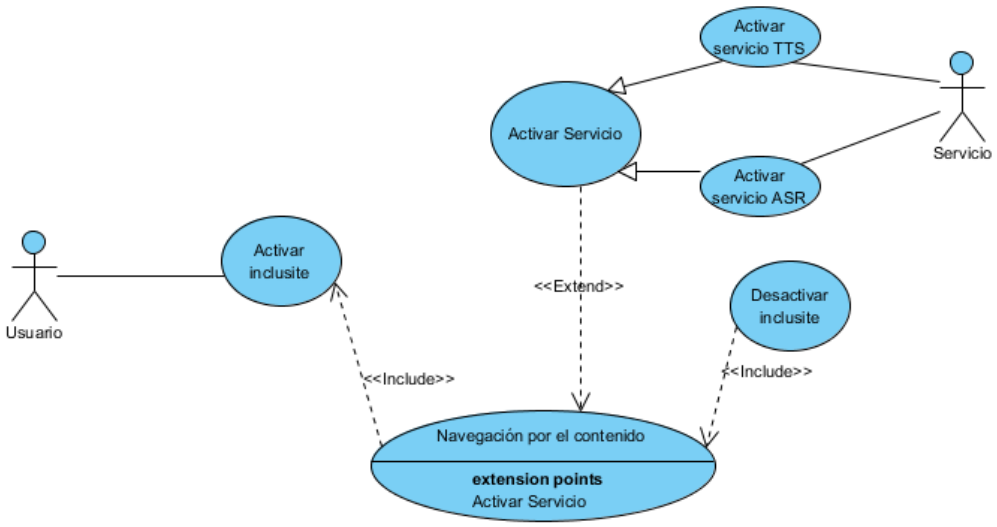


Figura 23: Diagrama caso de uso funcionamiento general de inclusite.

Caso de uso	Activar inclusite.
Actores	Usuario
Propósito	Activar las funcionalidades de inclusite.
Resumen	El usuario escoge usar inclusite para navegar por el contenido de la web, donde se debe hacer una configuración previa para que el usuario elija las interfaces de navegación de acuerdo a sus necesidades, una vez escogida las interfaces, se carga la página con las ayudas visuales que presta inclusite.
Precondiciones	La página web debe tener el script que permia activar inclusite.
Post-condiciones	El sistema queda listo para recibir órdenes.

Tabla 2: Descripción textual del caso de uso Activar inclusite.

Caso de uso	Navegación por el contenido
Actores	Usuario
Propósito	Permitir la navegación por el contenido de la web usando inclusite.
Resumen	Cuando inclusite esta activo, éste interpreta de una forma semántica el contenido, dando opciones numéricas junto con el nombre del área de la página web al usuario para que él escoja a donde quiere acceder.
Precondiciones	Inclusite debe estar activado en el sitio web.
Post-condiciones	Se cargan nuevas opciones mientras el usuario navega.

Tabla 3: Descripción textual del caso de uso navegación por el contenido.

Caso de uso	Desactivar inclusite.
Actores	Usuario.
Propósito	Desactivar las funcionalidades de inclusite.
Resumen	Si el usuario desea que la página web se vea sin la capa de accesibilidad que presta inclusite éste puede desactivarlo y acceder al contenido de forma normal.
Precondiciones	Inclusite debe estar activado
Post-condiciones	Se carga la web sin inclusite.

Tabla 4: Descripción textual del caso de uso desactivar inclusite.

Caso de uso	Activar servicios
Actores	Servicio.
Propósito	Leer el contenido e interpretar los comandos de voz enviados por el usuario.
Resumen	Este caso de uso destaca el uso de los servicios que presta inclusite en el back-end del sistema, para el caso del TTS leer el contenido al cual accede el usuario y anunciarle las opciones de navegación que tiene, por otro lado el servicio ASR, se encargará de interpretar las órdenes enviadas con la voz por el usuario.
Precondiciones	El usuario debe haber escogido en la configuración alguna interfaz que haga uso de los servicios. En el caso del servicio ASR, el usuario tendrá que dar permisos a inclusite para usar el micrófono.
Post-condiciones	Ninguna.

Tabla 5: Descripción textual del caso de uso Leer contenido/Interpretar ordenes.

4.4.2. Diagrama de casos de uso configuración

La siguiente figura muestra el diagrama de casos de uso de la configuración previa que el usuario debe hacer antes de iniciar a navegar con inclusite.

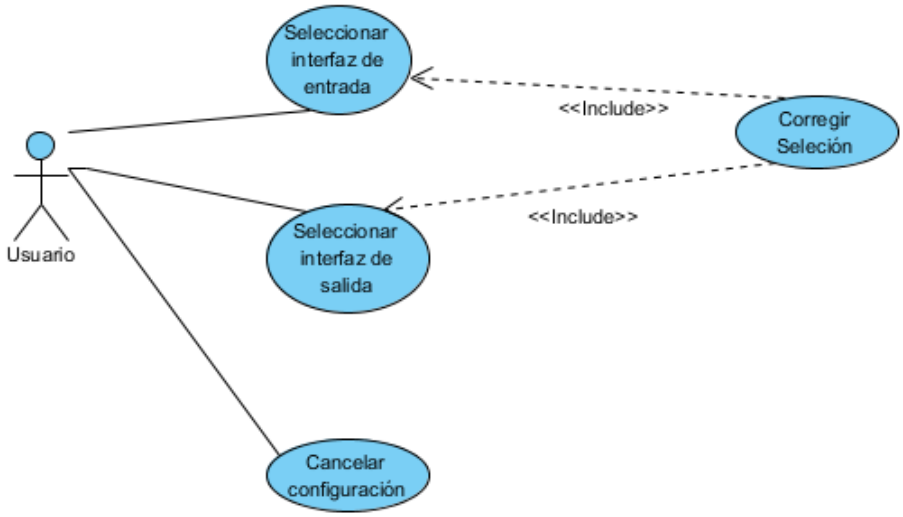


Figura 24: Diagrama de casos de uso de la configuración de inclusite.

Caso de uso	Seleccionar interfaz de entrada
Actores	Usuario
Propósito	Seleccionar de qué manera el usuario le quiere emitir los comandos de navegación a inclusite.
Resumen	Se enlistan las opciones a escoger de las diferentes interfaces de entrada que tendrá inclusite y el usuario escoge una de ellas.
Precondiciones	El usuario debe haber pedido que se activara inclusite.
Post-condiciones	Se activa la interfaz de entrada seleccionada.

Tabla 6: Descripción textual del caso de uso seleccionar interfaz de entrada.

Caso de uso	Seleccionar interfaz de salida.
Actores	Usuario
Propósito	Dar la opción al usuario de activar o no, la interfaz de salida de voz.
Resumen	El en segundo paso de configuración el usuario podrá escoger si se reproduce el contenido y las opciones de navegación de la página web en formato audio, usando el servicio TTS.
Precondiciones	La ventana de configuración debe estar activa y el usuario debe haber escogido una interfaz de entrada.
Post-condiciones	Se activa o no la interfaz de salida de voz. En el caso que la interfaz de entrada sea la de sonido o la de voz, se accederá a un tercer paso para dar permisos a inclusite al uso del micrófono.

Tabla 7: Descripción textual del caso de uso seleccionar interfaz de salida.

Caso de uso	Corregir selección.
Actores	Usuario
Propósito	Permitir que el usuario cambie las opciones escogidas.
Resumen	En ocasiones los usuarios se equivocan al escoger una opción, en la configuración se debe permitir dar un paso atrás ya sea para escoger de nuevo la interfaz de entrada o la de salida.
Precondiciones	Tener activa la ventana de configuración de inclusite e ir al menos en el segundo paso de configuración.
Post-condiciones	Volver al paso anterior de la configuración de la fase actual.

Tabla 8: Descripción textual del caso de uso Corregir selección.

Caso de uso	Cancelar configuración.
Actores	Usuario.
Propósito	Permitir que el usuario cancele la configuración de inclusite.
Resumen	La ventana de configuración está abierta y el usuario desea no configurar inclusite, entonces usa un comando que le permite cerrar la ventana de configuración y volver a la página web actual sin la capa de accesibilidad que presta inclusite.
Precondiciones	Tener abierta la ventana de configuración de inclusite.
Post-condiciones	Cargar la página web actual sin inclusite.

Tabla 9: Descripción textual del caso de uso Cancelar configuración.

4.4.3. Diagrama de casos de uso navegación sobre el contenido.

La siguiente figura muestra el diagrama de casos de uso de la navegación del usuario sobre el contenido de la página web usando inclusite, el caso de uso Activar servicio ya ha sido descrito en el apartado anterior.

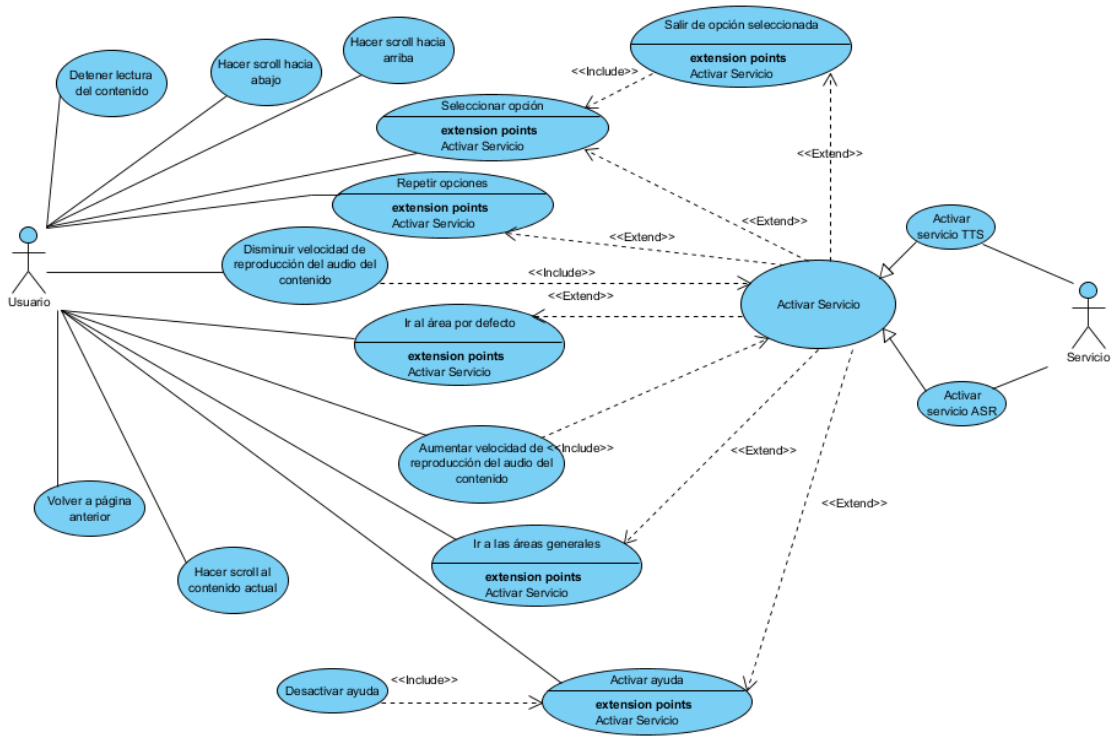


Figura 25: Diagrama casos de uso de la navegación por el contenido con inclusite.

Caso de uso	Seleccionar opción.
Actores	Usuario.
Propósito	Ejecutar la acción de acuerdo a la opción seleccionada por el usuario.
Resumen	Al usuario durante la navegación se la darán una serie de opciones numéricas que identifica de una forma semántica cada área de la página web para navegar sobre el contenido, el usuario selecciona alguna de dichas opciones y se accede al

	área correspondiente a la opción escogida.
Precondiciones	El sistema inclusite debe estar activo y configurado.
Post-condiciones	Se asignarán nuevas opciones numéricas de acuerdo al área donde se encuentra si ésta tiene áreas hijas en su jerarquía, de lo contrario se lee el contenido de la página web.

Tabla 10: Descripción de caso de uso Seleccionar opción.

Caso de uso	Salir de opción seleccionada.
Actores	Usuario.
Propósito	Salir del área actual en el cual se encuentra el usuario dentro el contenido de la página web.
Resumen	El usuario entra a un área de las opciones numéricas de navegación ofrecidas por inclusite, luego sale de ella ejecutando algún comando que le permita efectuar esta acción.
Precondiciones	El usuario debe haber accedido al menos a un nivel de profundidad dentro del árbol jerárquico de las áreas creadas por inclusite.
Post-condiciones	Volver a un nivel superior dentro la jerarquía de áreas desde la actual en la que se encuentra el usuario.

Tabla 11: Descripción caso de uso Salir de opción seleccionada.

Caso de uso	Repetir opciones.
Actores	Usuario.

Propósito	Permitir que el usuario recargue las opciones de navegación actual.
Resumen	El usuario ejecuta un comando que recarga las opciones de navegación del área actual donde se encuentra, también recarga las ayudas visuales que pone inclusite sobre el contenido, si el servicio TTS está activo, se repiten las opciones anunciadas.
Precondiciones	El servicio inclusite debe estar activo y configurado.
Post-condiciones	Se recargarán opciones y funciones de inclusite.

Tabla 12: Descripción textual del caso de uso Repetir opciones.

Caso de uso	Ir al área por defecto.
Actores	Usuario.
Propósito	Llevar al usuario al área por defecto del contenido de la página web.
Resumen	Se debe permitir al usuario ir a la zona por defecto o la que se considere de mayor importancia dentro del documento HTML de la página web, la cual debe estar identificada con algún atributo para que inclusite pueda llegar a ella. El usuario estará en cualquier área dentro del contenido de la página web y decide ejecutar el comando que lo lleva al área por defecto.
Precondiciones	Tener activo y configurado el servicio inclusite.
Post-condiciones	Si existe un área por defecto se asigna como área actual dentro de la navegación de inclusite.

Tabla 13: Descripción textual del caso de uso Ir al área por defecto.

Caso de uso	Ir a las áreas generales.
Actores	Usuario.
Propósito	Se permitirá que el usuario pueda volver al nivel más alto de la jerarquía de áreas desde cualquier profundidad en la que se encuentre dentro del árbol de navegación.
Resumen	El usuario está navegando por el contenido de la página web usando inclusite, y éste decide ir a las áreas generales de la estructura de la página web, ejecuta entonces el comando.
Precondiciones	Tener activo y configurado el servicio inclusite.
Post-condiciones	El usuario vuelve al nivel más alto de la jerarquía de áreas y se recargan de nuevo las opciones de navegación.

Tabla 14: Descripción textual de caso de uso Ir a las áreas generales.

Caso de uso	Activar ayuda.
Actores	Usuario.
Propósito	Mostrar al usuario una ayuda breve de la navegación con inclusite.
Resumen	El usuario está navegando por el contenido de la página web y ejecuta un comando que lanza una ayuda sobre la forma en que se debe navegar y los diferentes comandos que puede ejecutar en relación a la interfaz de entrada que tiene activa.
Precondiciones	Tener activo y configurado el servicio inclusite.
Post-condiciones	Se abrirá una ventana con el contenido de la ayuda.

Tabla 15: Descripción textual del caso de uso Activar ayuda.

Caso de uso	Desactivar ayuda.
Actores	Usuario.
Propósito	Cerrar la ventana de ayuda de inclusite.
Resumen	Cuando el usuario tiene activa la ayuda de inclusite, este podrá cerrarlo con el mismo comando que la abrió.
Precondiciones	Tener abierta la ventana de ayuda de inclusite.
Post-condiciones	Se recargan todas las opciones de navegación.

Tabla 16: Descripción textual del caso de uso Desactivar ayuda.

Caso de uso	Hacer scroll hacia arriba.
Actores	Usuario.
Propósito	Hacer un desplazamiento hacia arriba cuando la página tiene scroll.
Resumen	El usuario está navegando por el contenido de la página web usando inclusite, entonces el usuario ejecuta un comando que hace activar el desplazamiento de la barra scroll del navegador hacia arriba.
Precondiciones	Tener activado y configurado inclusite.
Post-condiciones	Hace el desplazamiento hacia arriba.

Tabla 17: Descripción textual del caso de uso Hacer scroll hacia arriba.

Caso de uso	Hacer scroll hacia abajo.
Actores	Usuario.
Propósito	Hacer un desplazamiento hacia abajo cuando la página tiene scroll.
Resumen	El usuario está navegando por el contenido de la página web usando inclusite, entonces el usuario ejecuta un comando que hace activar el desplazamiento de la barra scroll del navegador hacia abajo.
Precondiciones	Tener activado y configurado inclusite.
Post-condiciones	Hace el desplazamiento hacia abajo.

Tabla 18: Descripción textual del caso de uso Hacer scroll hacia abajo.

Caso de uso	Hacer scroll al contenido actual.
Actores	Usuario.
Propósito	Hacer un desplazamiento hacia el área actual.
Resumen	El usuario está navegando por el contenido de la página web usando inclusite, y hecho acciones que conllevan un desplazamiento en sentido horizontal sobre el contenido, entonces decide desplazarse visualmente hacia el área actual en la cual se encuentra dentro del árbol de navegación del contenido ejecutando el comando correspondiente.
Precondiciones	Tener activado y configurado inclusite.
Post-condiciones	Hace el desplazamiento hacia el área actual.

Tabla 19: Descripción textual del caso de uso Hacer scroll al contenido actual.

Caso de uso	Disminuir velocidad de reproducción del audio del contenido.
Actores	Usuario.
Propósito	Permitir que el usuario escuche la reproducción del audio del contenido más lento.
Resumen	Cuando el servicio TTS está activo, el contenido y las opciones de navegación son escuchadas por el usuario, entonces él puede por medio de un comando disminuir la velocidad de reproducción de ese audio.
Precondiciones	Tener activo y configurado inclusite de un modo donde se use el servicio TTS.
Post-condiciones	La reproducción de audio es más lenta.

Tabla 20: Descripción textual del caso de uso Disminuir velocidad de reproducción del audio del contenido.

Caso de uso	Aumentar velocidad de reproducción del audio del contenido.
Actores	Usuario.
Propósito	Permitir que el usuario escuche la reproducción del audio contenido más rápido.
Resumen	Cuando el servicio TTS está activo, el contenido y las opciones de navegación son escuchadas por el usuario, entonces él puede por medio de un comando aumentar la velocidad de reproducción de ese audio.
Precondiciones	Tener activo y configurado inclusite de un modo donde se use el servicio TTS.

Post-condiciones	La reproducción de audio es más rapida.

Tabla 21: Descripción textual del caso de uso Aumentar velocidad de reproducción del audio del contenido.

Caso de uso	Detener lectura del contenido.
Actores	Usuario.
Propósito	Para la reproducción del audio del contenido.
Resumen	Cuando el servicio TTS está activo, el contenido y las opciones de navegación son escuchadas por el usuario, entonces él puede por medio de un comando detener la reproducción de ese audio.
Precondiciones	Tener activo y configurado inclusite de un modo donde se use el servicio TTS.
Post-condiciones	La reproducción de audio del contenido es detenida.

Tabla 22: Descripción textual del caso de uso Detener lectura del contenido.

Caso de uso	Volver a página anterior
Actores	Usuario.
Propósito	Dar una vuelta átras en la navegación en el sitio web.
Resumen	Cuando el usuario esté navegando y se desplace de una página a otra él puede retornar a la página anterior ejecutando el comando correspondiente.
Precondiciones	Tener activado y configurado inclusite.

Post-condiciones	Retorna a la página anterior y recarga funciones de inclusite.

Tabla 23: Descripción textual del caso de uso Volver a página anterior.

4.4.4. Diagrama de casos de uso interfaz multimedia.

La siguiente figura muestra el diagrama de casos de uso de la interacción del usuario sobre el contenido multimedia de la página web usando inclusite, cuando se dice contenido multimedia, se hace referencia a vídeos y audios.

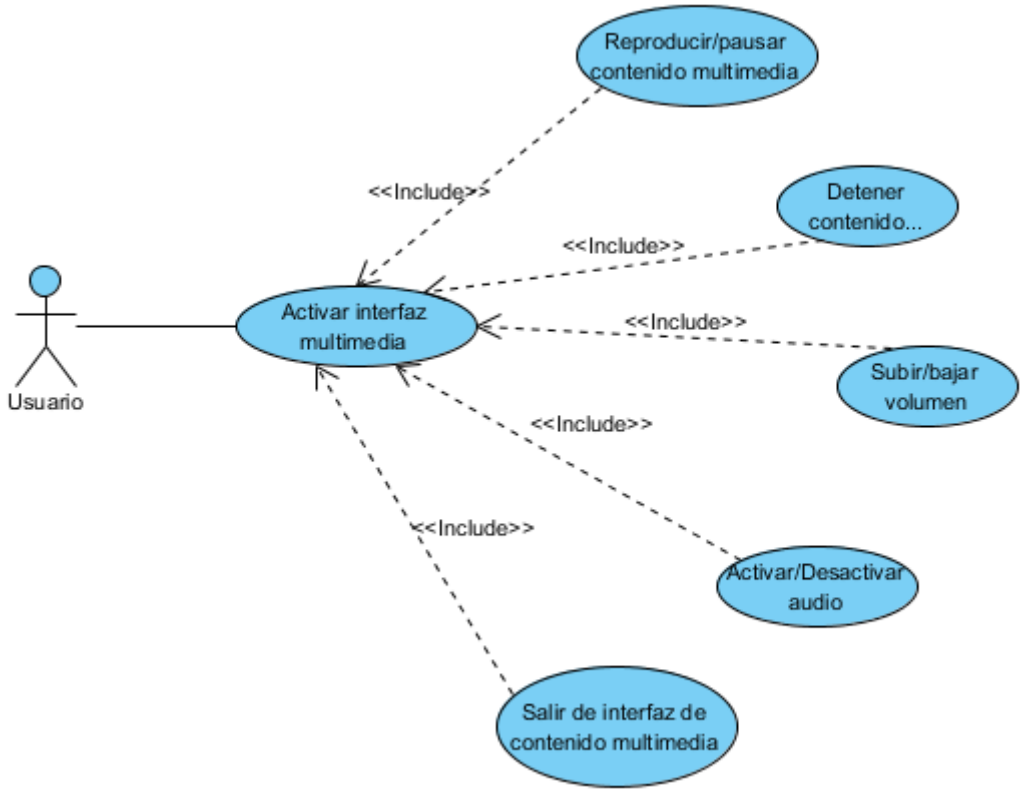


Figura 26: Diagrama de casos de uso de interfaz multimedia.

Caso de uso	Activar interfaz multimedia.
Actores	Usuario.
Propósito	Activar la interfaz con la cual el usuario puede manipular contenido multimedia.

Resumen	Dentro de la navegación por el contenido web, es probable encontrarse vídeos o audios, cuando el usuario se encuentra con algún contenido de este tipo ejecuta la opción para acceder y se activa una interfaz que le permite interactuar con el contenido.
Precondiciones	Tener activo y configurado inclusite, luego seleccionar una opción donde el contenido sea de tipo multimedia.
Post-condiciones	Se activa la interfaz multimedia.

Tabla 24: Descripción textual del caso de uso Activar interfaz multimedia.

Caso de uso	Reproducir/pausar contenido multimedia.
Actores	Usuario.
Propósito	El usuario puede reproducir y/o pausar el contenido multimedia.
Resumen	Cuando se lanza la interfaz multimedia el usuario ejecuta un comando para reproducir el contenido si éste está parado o pausarlo si se está reproduciendo.
Precondiciones	Tener la interfaz multimedia activa.
Post-condiciones	Se reproduce o se pausa el contenido.

Tabla 25: Descripción textual del caso de uso de Reproducir/pausar contenido multimedia.

Caso de uso	Detener contenido multimedia.
Actores	Usuario.

Propósito	Detener la reproducción del contenido multimedia.
Resumen	El usuario está interactuando con el contenido multimedia y éste se está reproduciendo, él ejecuta un comando para detener la reproducción y vuelve a reiniciarse a espera que le pidan reproducir de nuevo.
Precondiciones	Tener la interfaz de reproducción activa.
Post-condiciones	El contenido multimedia se detiene y vuelve al inicio de su línea temporal.

Tabla 26: Descripción textual del caso de uso Detener contenido multimedia.

Caso de uso	Subir/baja volumen.
Actores	Usuario.
Propósito	Permitir que el usuario suba o baje el volumen del contenido multimedia.
Resumen	El usuario está reproduciendo algún contenido multimedia, y ejecuta un comando que hace que el volumen del contenido se disminuya, luego de un rato decide subir el volumen ejecutando el comando que efectúa esta acción.
Precondiciones	Tener la interfaz multimedia activa.
Post-condiciones	Se baja o se sube el volumen del contenido multimedia.

Tabla 27: Descripción textual del caso de uso Subir/bajar volumen.

Caso de uso	Activar/desactivar audio.

Actores	Usuario.
Propósito	Activar o desactivar el audio de un contenido multimedia.
Resumen	El usuario tiene activa la interfaz multimedia, el contenido se esté reproduciendo y ejecuta una acción que silencia el audio, luego de un rato ejecuta una acción que vuelve activar el audio.
Precondiciones	Tener activa la interfaz multimedia.
Post-condiciones	Se desactiva o se activa el audio del contenido multimedia.

Tabla 28: Descripción textual del caso de uso de Activar/desactivar audio.

Caso de uso	Salir de interfaz de contenido multimedia.
Actores	Usuario.
Propósito	Salir de la interfaz multimedia.
Resumen	El usuario está reproduciendo un contenido multimedia, éste se termina y el usuario ejecuta una acción la cual cierra la interfaz y vuelve a cargar la interfaz de navegación en la página web.
Precondiciones	Tener activa la interfaz multimedia.
Post-condiciones	El usuario vuelve al contenido de la página web con la interfaz de navegación que tenga activa.

Tabla 29: Descripción textual del caso de uso Salir de interfaz de contenido multimedia.

4.4.5. Diagrama de casos de uso navegación sobre formularios.

La siguiente figura muestra el diagrama de casos de uso de la navegación del usuario sobre los formularios y sus campos, de la página web usando inclusite.

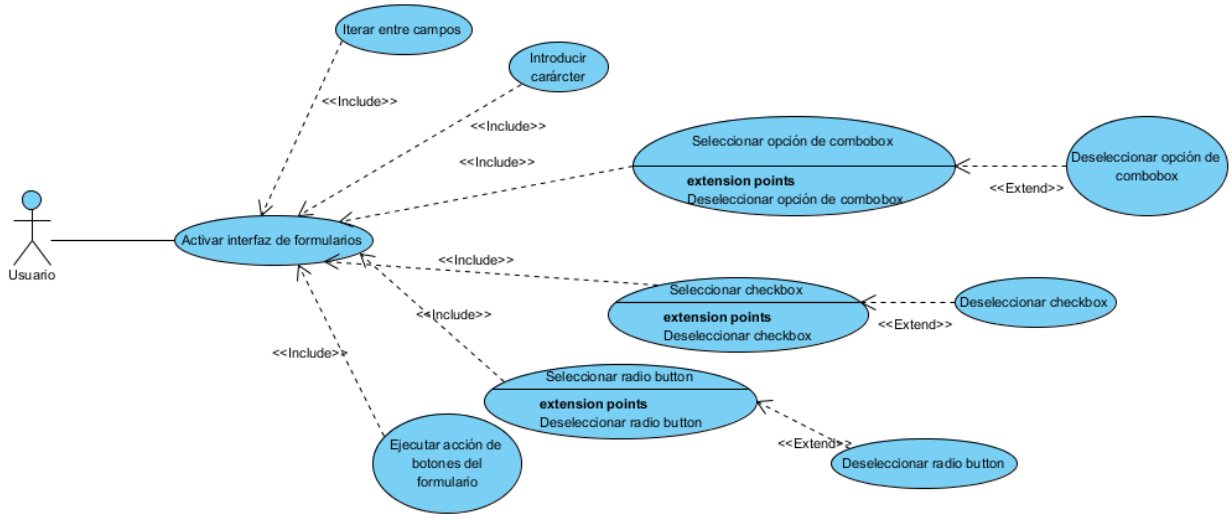


Figura 27: Diagrama de casos de uso de la navegación en formularios.

Caso de uso	Activar interfaz de formularios.
Actores	Usuario.
Propósito	Activar la interfaz con la cual el usuario puede interactuar con los campos de un formulario.
Resumen	Dentro de la navegación por el contenido web, es probable encontrarse con formularios, sea para registrarse, identificarse, hacer búsquedas o simplemente para hacer comentarios, cuando el usuario con algún contenido de este tipo se activa una interfaz que le permite introducir datos a los campos de los formularios en relación a la interfaz de navegación que tenga activa.

Precondiciones	Tener activo y configurado inclusite, luego seleccionar una opción donde el contenido sea un formulario.
Post-condiciones	Se activa la interfaz de formularios.

Tabla 30: Descripción textual del caso de uso de Activar interfaz de formulario.

Caso de uso	Iterar entre campos.
Actores	Usuario.
Propósito	Permitir que el usuario pueda iterar entre los campos de un formulario.
Resumen	El usuario mientras navega por el contenido web entra a un área donde el contenido es un formulario, se activa la interfaz de formulario y se puede desplazar hacia adelante o hacia atrás entre los campos del formulario.
Precondiciones	La interfaz de formularios está activa.
Post-condiciones	Ninguna.

Tabla: Descripción textual del caso de uso Iterar entre campos.

Caso de uso	Introducir carácter.
Actores	Usuario.
Propósito	Permitir que el usuario introduzca caracteres en los campos de texto.
Resumen	El usuario entra un formulario, se activa la interfaz de formularios, donde el primer campo, es un campo de texto, el

	usuario introduce los caracteres que desee por medio de la interfaz.
Precondiciones	Tener activa la interfaz de formularios.
Post-condiciones	Los caracteres introducidos a través de la interfaz de formularios son puestos en el campo de texto de la página web.

Tabla 31: Descripción textual del caso de uso Introducir carácter.

Caso de uso	Seleccionar opción de combobox.
Actores	Usuario.
Propósito	Permitir que el usuario pueda seleccionar una o más de las opciones que tenga un listado de combobox.
Resumen	El usuario entra a un formulario, se activa la interfaz de formularios, iterando entre los campos se encuentra con un combobox, las lista de opciones que éste contenga son mostradas al usuario a través de la interfaz, el usuario puede seleccionar una o varias opciones, según lo permita el combobox de la página web.
Precondiciones	Tener activa la interfaz de formularios.
Post-condiciones	Las selecciones sobre el combobox hechas por el usuario a través de la interfaz son puestas en el combobox de la página web.

Tabla 32: Descripción textual del caso de uso de Seleccionar opción de combobox.

Caso de uso	Deseleccionar opción de combobox.

Actores	Usuario.
Propósito	Permitir que el usuario pueda deseleccionar una o más de las opciones que estén seleccionadas en el listado de combobox.
Resumen	El usuario entra a un formulario, se activa la interfaz de formularios, iterando entre los campos se encuentra con un combobox, las lista de opciones que éste contenga son mostradas al usuario a través de la interfaz, el usuario puede deseleccionar una o varias opciones ya seleccionadas en el combobox
Precondiciones	Tener activa la interfaz de formularios y tener opciones del combobox seleccionadas
Post-condiciones	Las deselecciones sobre el combobox hechas por el usuario a través de la interfaz son efectuadas en el combobox de la página web.

Tabla 33: Descripción textual del caso de uso de Deseleccionar opción de combobox.

Caso de uso	Seleccionar checkbox.
Actores	Usuario.
Propósito	Permitir que el usuario pueda seleccionar un checkbox.
Resumen	El usuario entra a un formulario y se activa la interfaz de formularios, iterando entre los campos se encuentra con checkbox, el usuario selecciona el checkbox.
Precondiciones	Tener activa la interfaz de formularios.
Post-condiciones	La selección sobre el chcekbox hecha por el usuario a través de la interfaz de formularios es efectuada en el checkbox de la

	página web.

Tabla 34: Descripción textual del caso de uso Seleccionar checkbox.

Caso de uso	Deseleccionar checkbox.
Actores	Usuario.
Propósito	Permitir que el usuario pueda deseleccionar un checkbox.
Resumen	El usuario entra a un formulario y se activa la interfaz de formularios, iterando entre los campos se encuentra con checkbox que está seleccionado, el usuario deselecciona el checkbox.
Precondiciones	Tener activa la interfaz de formularios y tener un checkbox seleccionado.
Post-condiciones	La deselección sobre el checkbox hecha por el usuario a través de la interfaz de formularios es efectuada en el checkbox de la página web.

Tabla 35: Descripción textual del caso de uso Deseleccionar checkbox.

Caso de uso	Seleccionar radio button.
Actores	Usuario.
Propósito	Permitir que el usuario pueda seleccionar un radio button.
Resumen	El usuario entra a un formulario y se activa la interfaz de formularios, iterando entre los campos se encuentra con radio button, el usuario selecciona el radio button.

Precondiciones	Tener activa la interfaz de formularios.
Post-condiciones	La selección sobre el radio button hecha por el usuario a través de la interfaz de formularios es efectuada en el radio button de la página web.

Tabla 36: Descripción textual del caso de uso Seleccionar radio button.

Caso de uso	Deseleccionar radio button.
Actores	Usuario.
Propósito	Permitir que el usuario pueda deseleccionar un radio button.
Resumen	El usuario entra a un formulario y se activa la interfaz de formularios, iterando entre los campos se encuentra con radio button que está seleccionado, el usuario deselecciona el radio button.
Precondiciones	Tener activa la interfaz de formularios y que exista un radio button seleccionado.
Post-condiciones	La deselección sobre el radio button hecha por el usuario a través de la interfaz de formularios es efectuada en el radio button de la página web.

Tabla 37: Descripción textual del caso de uso Deseleccionar radio button.

4.5. WAI-ARIA inclusite

A la hora de desarrollar inclusite, se tenía que pensar en cómo se podría establecer una estructura estándar sobre todas las páginas web, para que así inclusite permitiera al usuario navegar e interactuar con ellas, para tal caso se investigo y se llego a la conclusión de usar WAI-ARIA (véase sección 3.8).

Como se ha dicho antes, WAI-ARIA dispone de roles que permiten estructurar la página web de tal forma que da la versatilidad a la hora de navegar sobre ella, además que le da ese valor semántico permitiendo al usuario saber en qué parte o área de la página se encuentra, estas y más cualidades han permitido crear diferentes funcionalidades de inclusite, tales como identificar una área por defecto, establecer la estructura general de la web, etc.

El uso de WAI-ARIA en inclusite es solo una base para así crear un mapeo estructural de una página web a conveniencia de las funcionalidades que pretende prestar inclusite; se usan entonces algunos roles existentes, para así, además proporcionar a la página web adaptada un nivel de conformidad con respecto a las pautas establecidas en WCAG, por consiguiente se ha creado un modelo que se sigue para establecer la estructura de navegación sobre el contenido que tendrá la web cuando use inclusite. A continuación se explicará las entidades dentro del diagrama y el uso de sus atributos. Para ver el diagrama completo, ver el ANEXO 1.

Para empezar la entidad raíz que solo se encuentra en el cuerpo de la página web sería el role **document**, y a partir de ahí se desencadenan los diferentes roles.

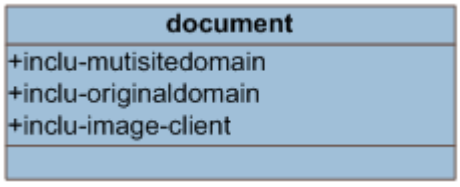


Figura 28: Entidad document.

El atributo **incu-multisitedomain**; establece a que entorno de inclusite debe conectarse para pedir los servicios; entorno hace referencia al mantenimiento evolutivo y correctivo del software, **inclu-originaldomain**; es el dominio original de la página que usa inclusite y por ultimo **inclu-image-client**; es la URL a una imagen, si el cliente desea poner su logo en la configuración de inclusite.

Aunque el siguiente nivel de entidades establezca una conexión directa con la entidad document, hay que resaltar dos de las cuales pueden llegar a estar a un nivel similar al de la jerarquía DOM, que son el role **main** y el role **complementary**, pero también es significativo que un complementary puede estar dentro de un role main, estos dos pueden contener dentro de ellos el resto de entidades que se encuentran dentro del modelo en cualquier zona de la página web, este concepto facilita a inclusite encontrar la semántica de la estructura del contenido de la página web.

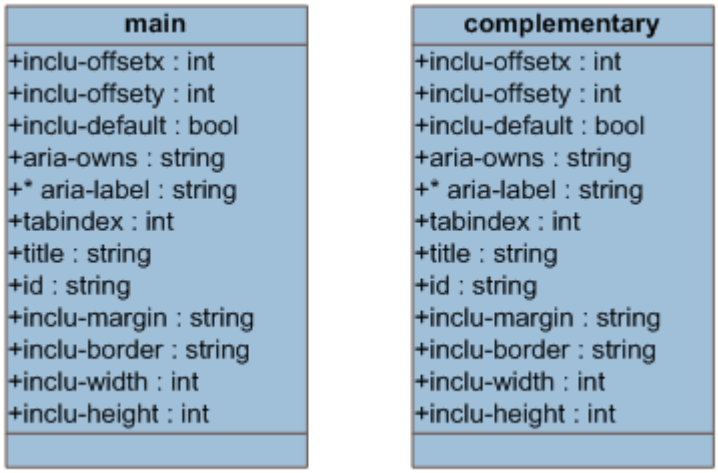


Figura 29: roles main y complementary.

Los siguiente atributos a describir también los contienen los otros roles, existen algunos roles que tendrán atributos de más que se usan en especial para su uso o incluso que tenga menos, también se dejaron atributos del propio WAI-ARIA ya que prestan funcionalidad a inclusite.

inclu-offsetx	Este permite aumentar o disminuir la posición X de los elementos visuales que aporta inclusite.
inclu-offsety	Este permite aumentar o disminuir la posición Y de los elementos visuales que aporta inclusite.
aria-owns	Identifica un elemento o varios en orden a definir de forma visual, funcional o contextual la relación padre e hijo cuando la jerarquía del DOM no puede hacerlo [21].
aria-label	Define el valor de una cadena que etiqueta a un elemento

	[21].
tabindex	Asigna un orden de tabulación para dar foco a los elementos del DOM, siendo el primero el número de menos valor, si no tiene valor el orden corresponderá al orden lógico del documento.
title	Actúa de la misma forma que el aria-label.
Id	Es el identificador del elemento.
inclu-margin	Da el estilo margin a los elementos visuales que aporta inclusite.
inclu-border	Da el estilo border a los elementos visuales que aporta inclusite.
inclu-width	Da el estilo width a los elementos visuales que aporta inclusite.
inclu-height	Da el estilo height a los elementos visuales que aporta inclusite.
inclu-platform	Se usa en el role video para establecer a que plataforma pertenece el video, si es de youtube o de vimeo se usa dichos reproductores, si no es ninguno de los anteriores se usa el reproductor de inclusite.
inclu-url	Se usa en el role video para establecer la ubicación del archivo multimedia a reproducir.

Tabla 38: atributos de los roles.

Los otros roles y no menos relevantes son los siguientes, cabe destacar que algunos no están en WAI-ARIA, y han sido creados para dar más funcionalidades en los formularios a inclusite:

navigation	Se usa para identificar los elementos que prestan como función, navegar sobre la página web, es decir menús, sub menús, etc.
contentinfo	Se usa para identificar el área donde exista información sobre el documento padre, normalmente se usa en los pie de página, haciendo mención sobre derechos de autor, políticas de privacidad, etc.
article	Se usa para identificar aquellas áreas donde el contenido es la información a la cual quiere llegar el usuario.
banner	Se usa para identificar aquellas áreas donde hay publicidad o anuncio.
search	Se usa para identificar un área donde exista un formulario básico de búsqueda.
form	Se usa para identificar las áreas donde exista un formulario.
item	Es un role que es hijo del role navigation, el cual permite identificar los enlaces de menú, permitiendo a inclusite, tratarlo de una forma diferente a un enlace normal para así no perder la usabilidad del menú.
text	Se usa para identificar esas áreas donde hay texto fuera del anidamiento lógico del documento, usando el aria-owns para asociarlo a un role article, banner o contentinfo.
img	Se usa para identificar las imágenes que contiene un role article, banner o contentinfo
video	Se usa para identificar el contenido multimedia que contiene un role article, banner o contentinfo.
link	Se usa para identificar los enlaces que contiene un role article, banner, contentinfo, main o complementary

input	Se usa para identificar los campos de texto de un formulario.
button	Se usa para identificar los botones de un formulario
combobox	Se usa identificar las listas desplegables de un formulario.
radiogroup	Se usa para identificar un grupo de radio buttons de un formulario.
radio	Se usa para identificar un radio button de un formulario
textbox	Se usa para identificar los text-area de un formulario.
checkboxgroup	Se usa para identificar un grupo de checkbox en un formulario.
checkbox	Se usa para identificar un checkbox en un formulario.
password	Se usa para identificar campos de texto de tipo “password” en un formulario.

Tabla 39: roles de inclusite.

5. Implantación inclusite

El desarrollo de inclusite ha tenido dos arquitecturas, en la primera se planteó de tal forma que parecía ser demasiado robusta, no permitía flexibilidad a la hora de crear nueva funcionalidades, nuevas interfaces, etc. además que la usabilidad y la navegación del usuario en la página web era bastante tediosa, también se tenía el problema de que se debía tener el código fuente original de la página web donde se insertaban unas etiquetas que permitían a inclusite analizar la semántica de la web.

Gracias a esa primera versión se decidió hacer un análisis más profundo sobre cómo hacer que inclusite mejorara la usabilidad, que fuera más flexible, que los tiempos de respuesta a los servicios fueran más cortos, compatibilidad con los diferentes navegadores del mercado, compatibilidad con las librerías javascript que pudiera usar cada página que decidiera usar inclusite.

A continuación se hablará de las dos versiones, de la primera se habla, solo para que se vea la evolución tan grande que ha tenido el sistema en la segunda versión, como se ha mencionado en algunas partes de este documento inclusite dispone de un back end y un front end, el back end presta los servicios ASR y TTS o algún otro que se necesite prestar en un futuro, la otra parte, el front end, es donde he participado en el desarrollo del proyecto y es de lo cual hablaré en esta sección del documento.

5.1. Implementación inclusite v1.0

Para esta primera versión de inclusite se necesitaba de un lenguaje de programación que se ejecutara sin importar sobre que tecnología estaba desarrollado el sitio web donde inclusite fuera implementado, por lo tanto se decidió usar javascript, cuyo lenguaje se ejecuta en el lado del cliente, además que todos los navegadores web pueden interpretarlo. Para más facilidad a la hora de desarrollo se decide entonces usar la librería jQuery[7] hecha en javascript.

La otra tecnología usada en el front end, para poder detectar el micrófono del usuario final y además implementar las clases que le permitieran navegar sobre el contenido, esa tecnología era Adobe Flash a base del lenguaje de programación ActionScript, con

[7] http://jquery.com/

esto si se configuraba el micrófono, el usuario podría comunicar las ordenes a inclusite si hiciera uso de las interfaces que así lo requieran.

5.1.1. Analisis de la estructura del DOM e implantación de inclusite.

Una vez decididas las tecnologías a usar, como primer planteamiento a desarrollar fue cómo se iba a analizar la estructura del DOM para darle un sentido semántico a través de inclusite, para tal caso, se decide insertar dentro del código fuente de la página web unas etiquetas en forma de comentarios que llevarían información del tipo contenido (un menú, pie de página, cabecera, etc.), identificadores que permitieran saber de dónde asciende ese contenido, el llamado a una función en javascript si el contenido lo necesitará o re direccionara a otra página si fuera un enlace y un título. Cabe resaltar que para esta primera versión no se hizo uso de WAI-ARIA.

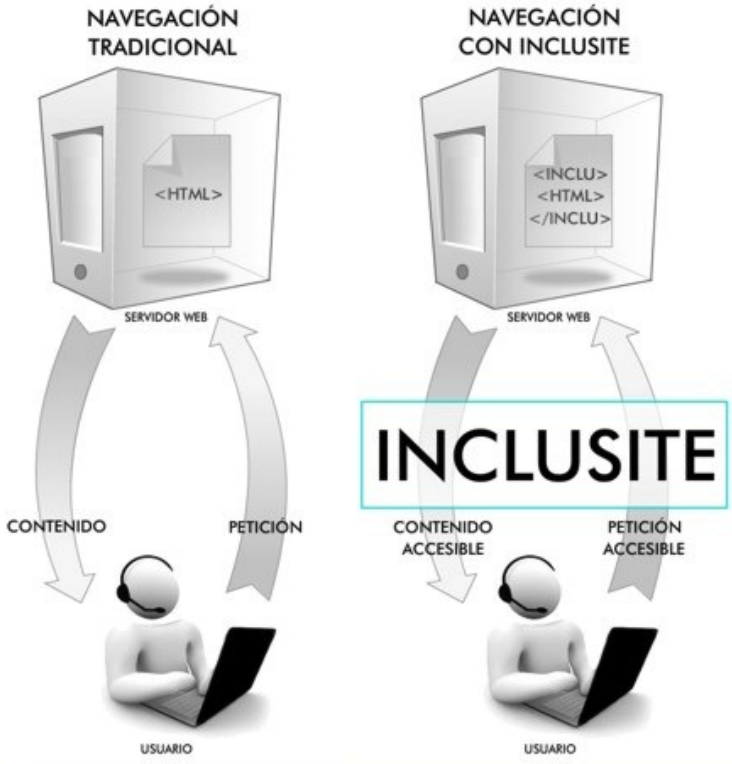


Figura 30: Usuario usando una web usar inclusite y otro usando inclusite.

A continuación un ejemplo visual de cómo se ubicaban las inclutags dentro del documento HMTL en un menú de navegación.

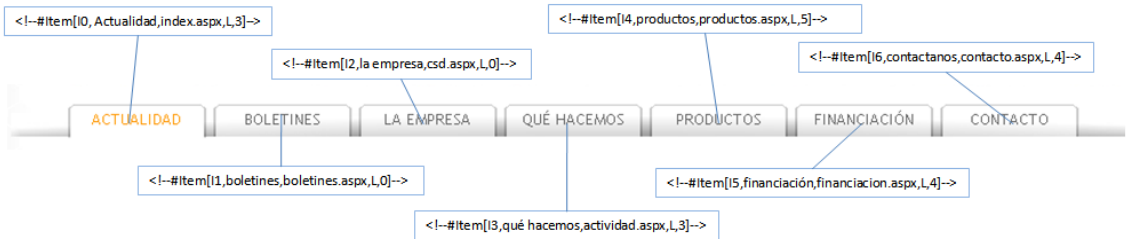


Figura 31: Menú de navegación de una página web, indicando que inclutag debería tener cada ítem del menú.

El código HTML del menú quedaría de esta forma:

```
<div id="header">
  <ul id="menu">
    <li id="inicio_li"><!--#Item[I0,Actualidad,index.aspx,L,3]--><a href='index.aspx' class='active'>Actualidad</a></li>
    <li id="boletines_li"><!--#Item[I1,boletines,boletines.aspx,L,0]--><a href="boletines.aspx">Boletines</a></li>
    <li id="csd_li"><!--#Item[I2,la empresa,csd.aspx,L,0]--><a href="csd.aspx">La empresa</a></li>
    <li id="actividad_li"><!--#Item[I3,qué hacemos,actividad.aspx,L,3]--><a href="actividad.aspx">Qué hacemos</a></li>
    <li id="productos_li"><!--#Item[I4,productos,productos.aspx,L,5]--><a href="productos.aspx">PRODUCTOS</a></li>
    <li id="financiacion_li"><!--#Item[I5,financiación,financiacion.aspx,L,4]--><a href="financiacion.aspx">FINANCIACIÓN</a></li>
    <li id="contacto_li"><!--#Item[I6,contactanos,contacto.aspx,L,4]--><a href="contacto.aspx">CONTACTO</a></li>
  </ul>
</div>
```

Figura 32: Código HTML de un menú de navegación con inclutags.

Luego de puestas las etiquetas, se implementa el código para que se inicie el análisis sobre el documento, además tenía que tener como resultado el contenido que había dentro de las inclutags para así generar un archivo XML con el contenido y la ubicación de éste dentro del DOM.

5.1.2. La navegación

Una vez logrado el análisis del DOM, el siguiente planteamiento fue el cómo debía navegar el usuario sobre el contenido. Se le había puesto a cada inclutag una serie de identificadores que permitían establecer cuáles eran los hijos de determinada inclutag o cual era su padre, así se decidió que la navegación fuera en forma de árbol.

Para la navegación se planteó hacer accesible las áreas mediante valores numéricos, de acuerdo al nivel en el que estuvieran dentro del árbol de navegación. Ya se tenía un XML que proveía la estructura del contenido de la página web en relación a la ubicación de las inclutags, con esto se podía usar ActionScript para analizar el XML resultante y así poner navegar sobre el contenido.

5.1.3. Interfaces de entrada

En el análisis de inclusite se planteo que tenía que haber tres interfaces de entrada, la interfaz de entrada hace referencia en la forma en que el usuario le emite las órdenes a inclusite para que éste las ejecute. Cada uno de estas interfaces debían tener las mismas funcionalidades además de permitir navegar sobre el contenido, cada interfaz tiene como propósito prestar accesibilidad al contenido al usuario con determinada discapacidad. Las interfaces de entrada serían:

- **Interfaz de teclado:** Con las tecnologías de asistencia actuales, los usuarios con pérdida de visión total pueden hacer uso de lectores de pantalla, que por medio de comandos enviados desde el teclado ejecuta acciones, incluso tiene una configuración avanzada que le permite al usuario navegar por el contenido de forma tabular,. Con base a lo anterior se implementa esta interfaz para dar soporte a dichos usuarios, los cuales se esperan que usen la interfaz de salida de voz de la cual se hablará más adelante.

- **Interfaz de voz:** Del mismo modo, alguien que tenga reducida su movilidad hasta el punto de no poder operar con un ratón, podría optar por emitir comandos de voz para llevar a cabo la navegación, la implementación de esta interfaz es una llamada al servicio ASR el cual interpreta y devuelve la acción a ejecutar por inclusite.

- **Interfaz de sonido:** Existen otros colectivos, que debido a parálisis severas, tienen limitada el habla; para ellos se diseña una interfaz minimalista que permite la navegación por modulación de sonidos, bien sea soplando sobre el micrófono o simplemente emitiendo sonidos con una mínima intensidad para ser detectados.

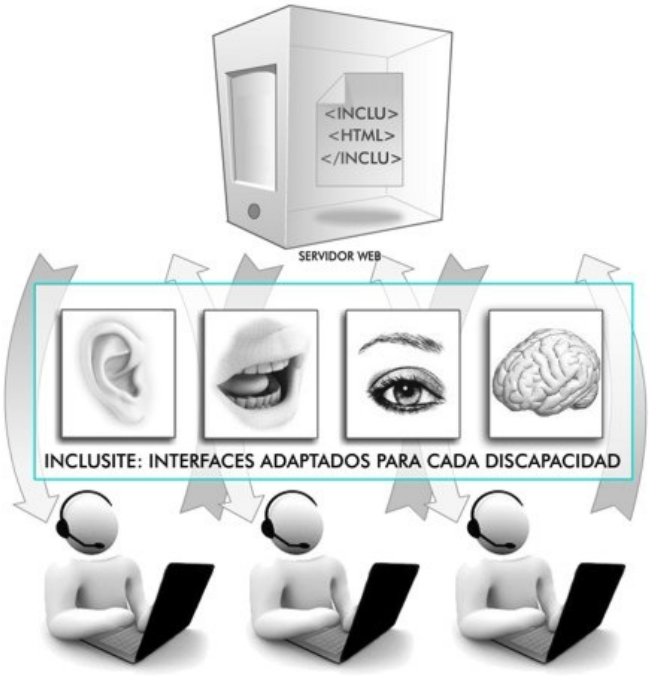


Figura 33: Usuarios usando inclusite.

5.1.4. Interfaces de salida

Como se dijo anteriormente, los usuarios con pérdida de visión total hacen uso de lectores de pantalla, por lo tanto para darle más accesibilidad a estos usuarios o a cualquier otro, se tenía que implementar una interfaz de salida con audio, que anunciara las opciones de navegación y leyera el contenido de la página web cuando el usuario llegara a la zona de interés para él.

La implementación de esta interfaz es una simple llamada al servicio TTS, pasándole el contenido desde el nodo del XML en el que se encuentra el usuario, para así recibir del servicio un audio de dicho contenido o de opciones de navegación, si es el caso.

5.1.5. Interfaces de formulario y contenido multimedia.

Para esta versión, no se había decidido que tecnología usar, cómo implementar, cómo identificar las áreas de la web donde el usuario se podía topar con algún formulario o contenido multimedia.

5.1.6. Configuración

La configuración de inclusite, consiste en un par de pasos que preparaban a inclusite para determinar de qué interfaz de entrada vendrían los comandos y que debía hacer con el contenido, el usuario elige una interfaz de entrada y escogía si deseaba o no que se leyera el contenido de la página web.

En el ***Paso1*** elegiremos como queremos comunicarnos con la página web:

Figura 34: Primer paso de configuración de inclusite versión 1.

En el ***Paso2*** elegiremos como queremos que la página web se comunique con nosotros:

Figura 35: Segundo paso de configuración de inclusite versión 1.

5.2. Implementación inclusite v2.0

Para la segunda versión de inclusite, se seguía manteniendo la idea de usar javascript y flash como tecnologías bases del front end del sistema, pero durante la implantación de inclusite v1.0 en algunos sitios webs, se notó varios aspectos a cambiar sobre la arquitectura de inclusite y su funcionamiento en ellos; con algunos se tuvo problemas de compatibilidad con la librería jQuery, afectando el funcionamiento normal de los sitios web donde inclusite era usado, así se decidió usar javascript de forma normal, sin ninguna librería, para que en futuras implantaciones en sitios web, no afectaran las funciones de la web, si ese sitio web hiciera uso de alguna librería como jQuery.

También se quiso reducir el uso de actionscript, por lo tanto esas clases que se habían creado para cada inclutag, serían eliminadas y se implementarían en javascript, pero se quería también, eliminar el hecho de que se tuviera que tener el código fuente para insertar las inclutags dentro del contenido, entonces nos dimos a la tarea de investigar más a fondo temas de accesibilidad web, y allí nos encontramos con el WAI-ARIA, el cual nos aporto la idea de usar el atributo role y otros más dependiendo de si nos daba o no funcionalidad en inclusite, para insertarlos dentro de las etiquetas HTML del documento, de ahí se crea el diagrama que se puede ver en el **ANEXO 1** y está explicado en la sección 4.5.

Para insertar los roles y evitar tener el código fuente del sitio web, otro grupo de desarrolladores crean un analizador de HTML en java, el cual por medio de reglas especificas para cada sitio web, se pondrían los roles y otros atributos sobre las etiquetas HTML.

5.2.1. La nueva arquitectura de inclusite v2.0

Mientras se solventaba el problema de cómo modificar el HTML que recibe el usuario sin necesidad de tener el código fuente del sitio web, se quería solucionar la robustez de la primera versión del front end de inclusite, la cual no permitía agregar nuevas funcionalidades sin modificar el código del sistema de una forma bastante prudente, entonces se quería que la nueva arquitectura del front end fuera lo suficientemente flexible para que cuando se agregará una nueva funcionalidad, una nueva inclutag o una nueva interfaz, se hiciera, simplemente agregando un par de líneas o incluso ninguna sobre el código ya existente.

Para tal caso se diseñaron varios diagramas de clases, que con el tiempo se fueron modificando hasta obtener el siguiente:

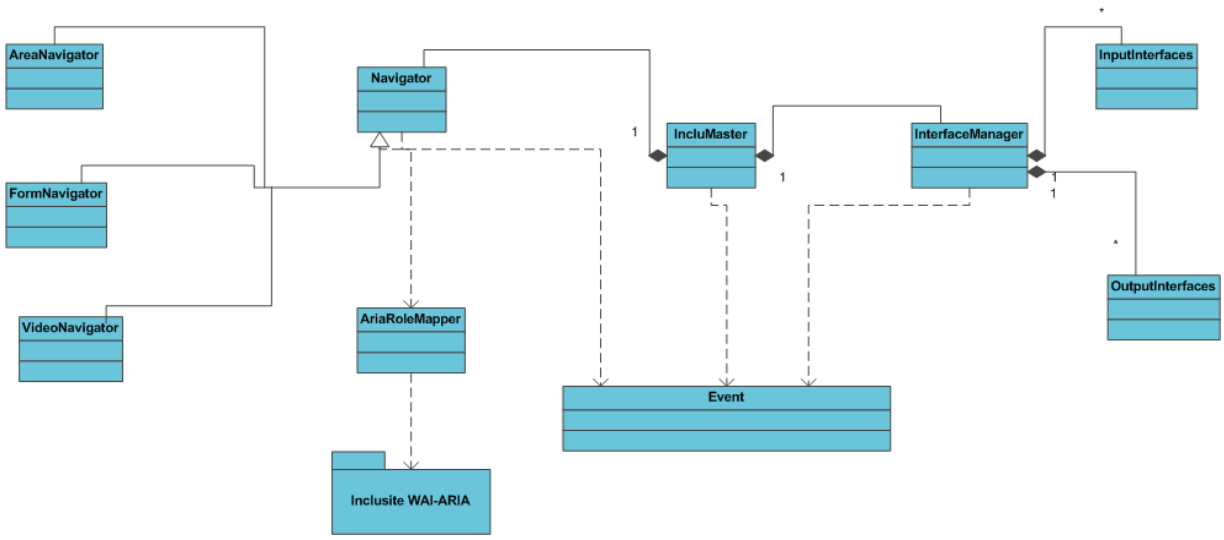


Figura 36: Diagrama de clases del front end inclusite.

A continuación se explica brevemente la función de cada componente del diagrama.

- **IncluMaster:** Esta clase es la más importante del sistema, ya que se encarga de recibir los eventos y comandos de la interfaz de entrada, los envía al navigator, el navigator ejecuta la acción y devulve los resultados al IncluMaster, donde éste le dice al InterfaceManager que interfaces de salida tienen que mostrarse al usuario, además cuando inclusite es configurado o se carga una página web es éste el que arranca todo el sistema.

- **InterfaceManager:** Esta clase se encarga de recibir los comandos provenientes de la interfaz de entrada, determinar que eventos tiene relacionados esa acción, luego espera la respuesta del IncluMaster, la interpreta, luego activa y adapta las interfaces de salida correspondientes.

- **Navigator:** Es una clase abstracta la cual ejecuta los comandos enviados por la interfaz de entrada y controla los eventos disparados por esas acciones, dentro del contexto en que se encuentre; es decir si está solo navegando por el contenido o sobre formularios o contenido multimedia. AreaNavigator, FormNavigator, MultimediaNavigator; son especializaciones de la clase Navigator, donde se independiza para dar funcionalidad en el tipo de contenido al que corresponde cuando el usuario esté navegando sobre alguno de su tipo.

- **AriaRoleMapper:** La labor de esta clase es mapear todo el documento HTML en busca de los roles del inclusite WAI-ARIA, para así crear un árbol, el cual es usado por el navigator para interactuar y desplazarse dentro del contenido.

- **Event:** El objetivo de esta clase es coger los eventos que pueda disparar el navegador sobre un componente de la página web y ejecutarlos en caso que fuera necesario cuando el usuario de inclusite esté interactuando con él, esto incluye eventos de teclado, de ratón, de carga de la página, etc.

- **AreaNavigator:** Es usada para la navegación por el contenido y áreas de la página web, ejecutar acciones de enlaces, desplazarse hacia arriba, hacia abajo, o al área actual donde se encuentra el usuario dentro del árbol navegación y demás funcionalidades que pretenda prestar inclusite.

- **FormNavigator:** Es usada para la navegación por los campos de los formularios, ejecutar comandos de inserción de caracteres, borrado, selección y des selección de opciones sobre checkbox, radio buttons y combobox, además de ejecutar las acciones de los botones, tales como enviar o reiniciar el formulario.

- **MultimediaNavigator:** Es usada para la ejecución de los comandos que envía el usuario cuando se entra a un contenido multimedia, permitiéndole interactuar con dicho contenido.

- **InputInterfaces, OutputInterfaces:** Hacen referencia a las interfaces que estuvieran activas, o que se ordenara que se activaran, como en el caso de las interfaces de contenido multimedia o formularios que dependen del contenido para activarse.

  La interfaces de entrada o interfaces de navegación son; una de teclado, una por voz y otra por sonidos. Las interfaces de salida se adapta como consecuencia de un evento disparado por las interfaces de entrada, las interfaces son; una para sonido, una para poner componente visuales dentro del área actual en la que se encuentra el usuario, la interfaz de formularios, la de contenido multimedia y la de voz.

5.2.2. Implementación Interfaz de sonido

Se me fue asignado desarrollar la interfaz de entrada de sonido, la cual trata de permitir que el usuario pueda enviar comandos a inclusite por medio de sonidos.

En la primera versión de inclusite, esta interfaz estaba hecha en actionscript, la cual se trataba de una barra con un conjunto de opciones, donde se podía escoger una, emitiendo un sonido cuando por medio de una iteración cíclica sobre ellas cambiaba de color. Como uno de los propósitos sobre la segunda versión era disminuir el uso de flash, mi labor fue entonces, desarrollar esta interfaz en javascript.

Para comenzar se tenía que crear la barra de opciones, la barra estaba compuesta por unos comandos por defecto, que permitirán al usuario, desplazarse hacia arriba o abajo dentro de la página web, repetir las opciones, salir del área actual, etc. pero también tenía su parte dinámica que eran las opciones de navegación, dependiendo del nivel del árbol donde se encontrara el usuario, la barra se tenía que crear con el número de opciones que tenía ese nivel, pero salió la cuestión de dónde se iban a sacar esas opciones, si según la nueva arquitectura, esa información solo la tendría como resultado de un proceso hecho a partir de un comando emitido desde una interfaz de entrada.

Entonces se decide separar lo que en la primera versión era una interfaz de entrada, en una interfaz de salida y otra de entrada, la de salida se encargaría de pintar la barra de opciones y comandos, mientras la de entrada detectaría los sonidos emitidos por el usuario sobre alguna opción o comando, y enviaría lo escogido a procesar, para obtener una nueva respuesta, de la cual dependería la interfaz de salida para crear de nuevo la barra.

Cabe destacar también que con el tiempo se fuero agregando nuevos métodos y variables a la interfaz de entrada, para que permitiera prestar funcionalidad a nuevas interfaz de salida, tales como la de contenido multimedia y formularios.

A la clase de la interfaz de salida se le da el nombre de **SoundOutputInterface** la cual queda estructurada de la siguiente manera:

Variables	Descripción
silencelevel	Determina el nivel de silencia máximo

	que debe soportar el micrófono, para evitar que cualquier leve ruido, que no ha emitido el usuario, ejecute una acción.
classcellactive	Nombre de la clase del estilo del componente HTML de la barra de sonido o teclado virtual, que define si la celda está activa.
classcellselected	Nombre de la clase del estilo del componente HTML de la barra de sonido o teclado virtual, que define si la celda ha sido seleccionada.
classrowselected	Nombre de la clase del estilo del componente HTML del teclado virtual, que define si una fila ha sido seleccionada.
classcellactivemultimedia	Nombre de la clase del estilo del componente HTML de la barra de sonido para contenido multimedia, que define si una celda está activada.

Tabla 40: Descripción de variables de la clase SoundOutputInterface.

Métodos	Descripción
SoundOutputInterface	Constructor de la interfaz de salida de sonido.
createSoundbarContainer	Crea un div dentro del código HTML de la página web, el cual contendrá la barra de navegación por sonido.

createSoundBarItems	Crea el componente HTML que contiene las opciones de navegación para la barra.
createSoundBarCommands	Crea el componente HTML que contiene los comandos que puede ejecutar el usuario dentro de la barra.
loadItems	Carga las áreas actuales para crear las opciones numéricas de la barra de sonido.
iterateCells	Se encarga de manejar la iteración entre las opciones de la barra de sonido.
addCellsStyle	Busca el valor de la celda seleccionada y asigna el estilo.
cleanCellsStyle	Limpia las clases asociadas a la celda activa.
destroy	Elimina la interfaz.
OutEvent	Recibe el evento enviado por la interfaz de entrada y gestiona la reconstrucción de la barra de sonido.
adapterOut	Adapta el evento a los datos que necesita la interfaz, este método es invocado desde el InterfacesManager.

Tabla 41: Descripción de métodos de la clase SoundOutputInterface.

A la clase de la interfaz de entrada se le da el nombre de **SoundInputInterface** la cual queda estructurada de la siguiente manera:

Variables	Descripción
maxitems	Determina el número máximo de opciones de navegación que se mostrarán en la barra.
idBarItems	Identificador del componente HTML de la barra de sonido, donde se encuentran las opciones numéricas de navegación.
idBarCommands	Identificador del componente HTML de la barra de sonido, donde se encuentran los comandos.
speed	Velocidad de iteración entre las celdas.
idSoundbar	Identificador del componente HTML que contiene la barra de navegación por sonido.
itemList	Lista de áreas sobre las que se puede navegar.

Tabla 42: Descripción de variables de la clase SoundInputInterface.

Métodos	Descripción
SoundIputInterfaceFlash	Contructor de la clase SoundInputInterface, además gestiona la captura de los eventos de la interfaz.
createFlashObject	Crea un objecto Flash para capturar los sonidos que el son emitidos a través del micrófono del ordenador del usuario.

toActive	Recupera el valor de la celda seleccionada de la barra de soplido y ejecuta el evento que este valor desencadena.
formOutputConnector	Detecta si la interfaz de salida es la FormOutputInterface y cambia su comportamiento.
sendFormEvents	Envía el evento de la tecla seleccionado en la interfaz de formularios.
toActiveForm	Recupera el valor de la celda seleccionada del teclado virtual del formulario y ejecuta el evento que este valor desencadena.
destroy	Destruye la interfaz de entrada.
onEvent	Subscribe el listener de eventos, para ser detectado por el IncluMaster o el Navigator activo en ese momento.

Tabla 43: Descripción de métodos de la clase SoundInputInterface.

Como resultado del desarrollo de esta interfaz, a continuación se muestra las siguientes imágenes, donde se ve la barra de navegación con las opciones y comandos que se pueden ejecutar. La celda que es de color gris, hace referencia a la iteración circular que se hace sobre la barra para que el usuario pueda escoger lo que quiere ejecutar, cuando ese cambio de color este sobre la celda deseada él podrá emitir el sonido para seleccionarla.

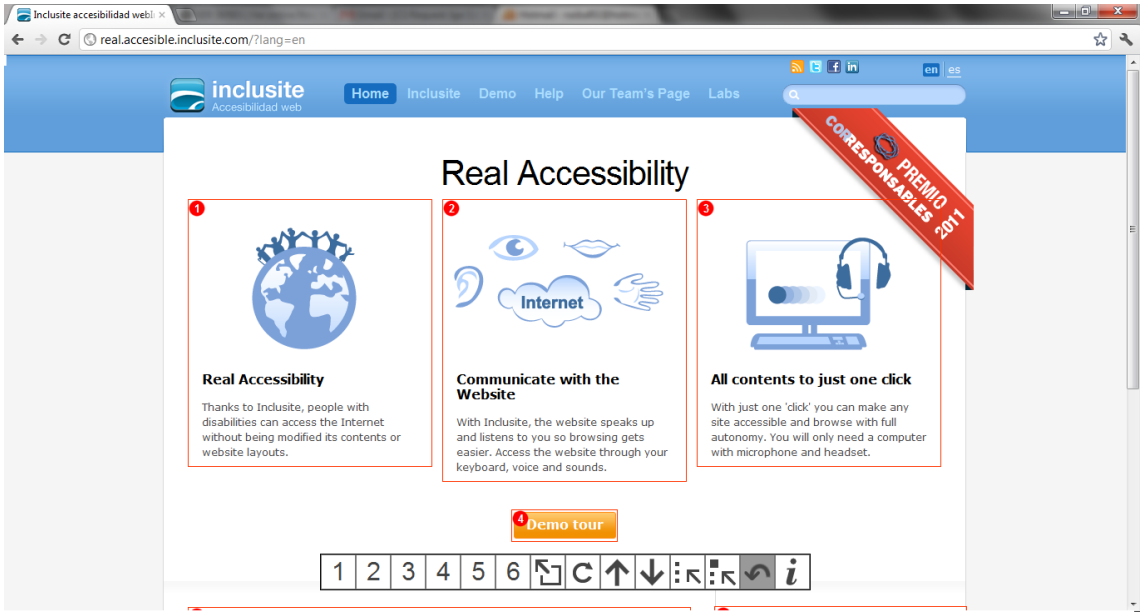


Figura 37: Barra de navegación en un sitio web.

Esta barra estará dividida en tantos segmentos numerados como Identificadores de área tenga el nivel del árbol de navegación sobre el contenido de la página. A lo largo de esta barra se desplazará de forma constante un indicador visual como se ve en la siguiente imagen.

Figura 38: Barra de navegación con indicador visual sobre un comando.

Deberá emitir un sonido de cierta intensidad cuando el indicador se encuentre sobre la opción que desee.

Los comandos que puede ejecutar el usuario se explican a continuación:

 Permite al usuario salir del área actual, es decir subir de nivel sobre el árbol de navegación.

 Permite al usuario desplazarse hacia arriba dentro de la página web.

 Permite al usuario desplazarse hacia abajo dentro de la página web.

 Permite al usuario ir a las áreas de primer nivel del árbol de navegación. Un diseño básico de una página web, se basa en tener un encabezado, un pie de página y la zona donde se encuentra el contenido relevante, se podría decir que estás son las área de primer nivel del árbol de navegación que implementa inclusite.

 Permite al usuario ir al área por defecto dentro del contenido de la página web.

 Permite al usuario reiniciar la iteración del indicador visual sobre la barra, además hace que la interfaz de salida de voz, repetida lo que estaba anunciando.

5.2.3. Implementación interfaz de formularios

Ésta es una interfaz de difícil implementación, ya que se tuvo que pensar cómo hacer usable esta interfaz para cada una de las interfaces de navegación implementadas, tema que actualmente seguimos discutiendo en el equipo de desarrollo.

Como primera instancia para empezar el desarrollo de esta interfaz, se propuso hacer una interfaz visual que se dispararía cuando se hiciera foco sobre el campo del formulario, sin importar el tipo de interfaz de navegación que tuviera activa el usuario.

La interfaz de formularios es definida coma una de salida, es decir, respondería a los comandos y eventos que ejecutara la interfaz de entrada escogida por el usuario; debido a que cada evento disparado, ejecutaría una acción diferente cuando se está navegando sobre el contenido, que cuando se está navegando sobre los campos de un formulario, es decir; inclusite usa la tecla D para llevar al usuario al área por defecto en el contenido de la página web, pero dentro de un campo de texto de un formulario, esta tecla debería asignar la letra D sobre el campo.

A base de lo anterior se decide entonces desarrollar una clase que permita navegar dentro de los formularios y gestionar de forma diferente los eventos y comandos de las interfaces de entrada.

A la clase de navegación se le da el nombre de **FormNavigator** la cual queda estructurada de la siguiente manera:

Variables	Descripción
acceptFormRole	Expresión regular para filtrar los tipos de role **form** y **search**.
keyCancelEvents	Array de teclas para cancelar su comportamiento por defecto desde el objeto IncluMaster.
previousArea	Almacena el campo anterior durante la navegación dentro por los campos del formulario.
comboInputValue	Es el valor seleccionado del combobox.
comboArrayCharacters	Se usa para construir el teclado predictivo que filtra las opciones por letra en el combobox.
comboCurrentOptions	Array de las opciones del combobox, como resultado de la filtración por letras.
accentBuffer	Almacena el carácter para asignar el acento a las vocales.
areaForm	Para almacenar el área que corresponde al role **form** o **search.**
selectionIndex	Para saber la posición actual dentro de la lista de opciones en un combobox, grupo de radio buttons o checkbox.
isChanged	Booleano que indica si se debe disparar el evento **onchange** en el campo actual del formulario.

eventers	Array de eventos a gestionar por la clase FormNavigator.

Tabla 44: Descripción de variables de la clase FormNavigator.

Métodos	Descripción
FormNavigator	Constructor de la clase FormNavigator.
onMove	Gestiona la navegación entre los campos de un formulario.
onEnd	Se encarga de salir de la interfaz de formulario y lanzar la interfaz de navegación activa.
setVoiceAdapter	Procesa la petición al servicio TTS para la interfaz de salida de voz.
readElement	Recupera los datos insertados en el campo de formulario actual y lo anuncia por medio de la interfaz de salida de voz.
writeCharacter	Escribe el carácter insertado por el usuario en el campo de texto.
deleteCharacter	Borra un carácter insertado en un campo de texto.
optionsFormBar	Gestiona las teclas de comandos que tienen los diferentes teclados virtuales.
onEnter	Gestiona el comportamiento del evento ENTER sobre cada uno de los campos.

Tabla 45: Descripción de métodos de la clase FormNavigator.

Para el desarrollo de esta funcionalidad se crea la clase **FormOutputInterface**, la cual se encargará de crear cada una de los componentes visuales en relación al campo del formulario. La estructura de la clase es la siguiente:

Variables	Descripción
isSoundActive	Bolean que indica si está activa la interfaz de sonido.
datatype	Indica que tipo de teclado virtual debe crear la interfaz.
iteratingCellsFlag	Bolean que cuando está a **true**, con la interfaz de sonido, el indicador visual itera sobre las columnas de las filas, si está en **false**, itera sobra las filas.
capsFlag	Detecta si se ha pulsado la tecla **CAPS** para cambiar el teclado a mayúsculas o a minúsculas.
symbolFlag	Bolean que cuando está a true indica que se ha creado el teclado con símbolos.
accentFlag	Bolean que cuando está a true indica que se ha creado un teclado con teclas con acentos.
formElementName	Indica el nombre del elemento actual del formulario.
checkedFlag	Indica si un checkbox o radiobutton está o no seleccionado.

formCurrentElement	Campo actual en el que se encuentra el usuario dentro de un formulario.
idSoundBar	Identificador del componente HTML de la barra de sonido.

Tabla 46: Descripción de variables de la clase FormOutputInterface.

Métodos	Descripción
FormOutputInterface	Constructor de la clase FormOutputInterface.
setFormEvents	Asigna el evento a ejecutar sobre cada tecla.
setInterfaceBehavior	Define el comportamiento de la interfaz en relación a la interfaz de entrada.
createVirtualKeyboard	Crea el código HTML de los diferentes teclados virtuales.
setComboKeyboard	Crea el teclado predictivo del combobox en relación a la lista de opciones.
iterateCells	Gestiona la iteración del indicador visual entre las columnas del teclado.
iterateRows	Gestiona la iteración del indicador visual entre las filas del teclado.
executeCommand	Interpreta las ordenes enviadas desde la interfaz de entrada.
checkCombobox	Ejecuta la acción visual de seleccionar

	una opción dentro de un listado.
checkRadioCheckbox	Ejecuta la acción visual de seleccionar un radio button o un checkbox.
onOffSymbol	Activa y desactiva el teclado de símbolos.
onOffAccent	Activa y desactiva el teclado de acentos.
onOffCapsLock	Activa y desactiva el teclado de mayúsculas.
OutEvent	Gestiona los comandos emitidos por la interfaz de entrada para generar el teclado virtual.
destroy	Elimina el código HTML de la interfaz cuando se sale de un formulario.

Tabla 47: Descripción de métodos de la clase FormOutputInterface.

Ya que todos los campos de un formulario no son iguales, esa interfaz visual que se dispararía tendría que ser diferente para cada uno ellos, por lo tanto se hicieron bosquejos de qué debería tener cada una, aparte se especifico su comportamiento para hacerlo más usable, a continuación se mostrará el resultado del desarrollo de cada una de esas interfaces visuales, el cómo deben funcionar sobre cada interfaz de navegación y cuál es su aporte de usabilidad y accesibilidad que le puede prestar al usuario que usa inclusite para rellenar un formulario.

5.2.3.1. Campos de texto

Para los campos de texto se crea un teclado virtual, el cual se comporta de las siguientes maneras en relación a la interfaz de entrada:

- **Interfaz de Teclado:** Con esta interfaz de entrada, cuando se está en un campo de texto la interfaz de formularios no presta funcionalidad, aun así ésta se activa, insertando los datos en el campo de texto de la interfaz.

- **Interfaz de voz:** Permite al usuario insertar caracteres uno a uno hasta completar la palabra deseada, incluye símbolos, acentos, números, letras en mayúscula o minúscula, etc. al igual que con la interfaz de teclado también insertando los datos en el campo de texto de la interfaz.

- **Interfaz de sonido:** Sobre esta interfaz de entrada es donde más funcionalidad presta la interfaz de formularios, en ella se desplaza un indicador visual por las filas del teclado, cuando se selecciona una fila por medio de un sonido, el indicador inicia un desplazamiento por las columnas de la fila (o por decirlo sobre las teclas), además agrega un botón que permite salir de la fila por si el usuario se ha equivocado.

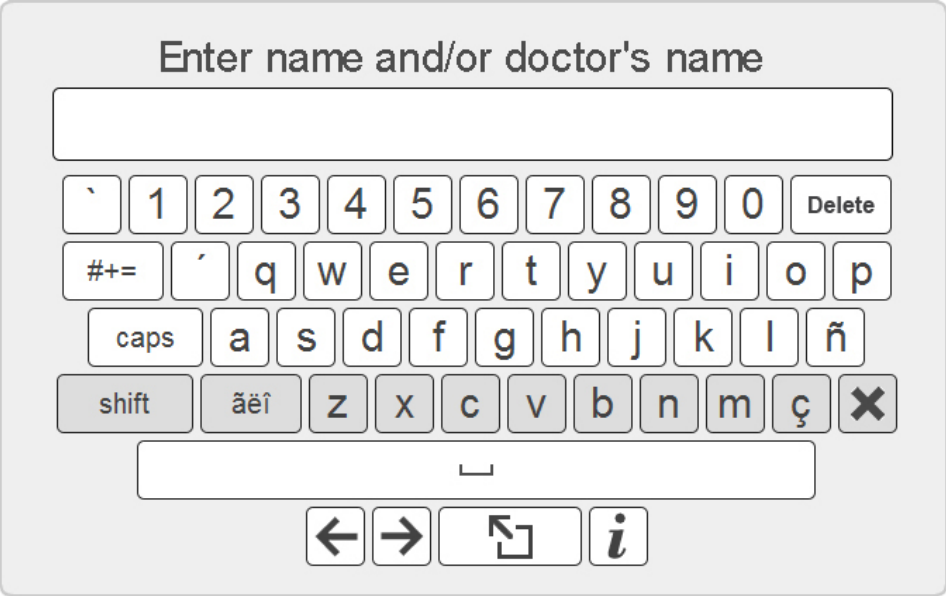


Figura 39: Teclado virtual de interfaz de formularios para campos de texto, indicador visual iterando sobre las filas del teclado.

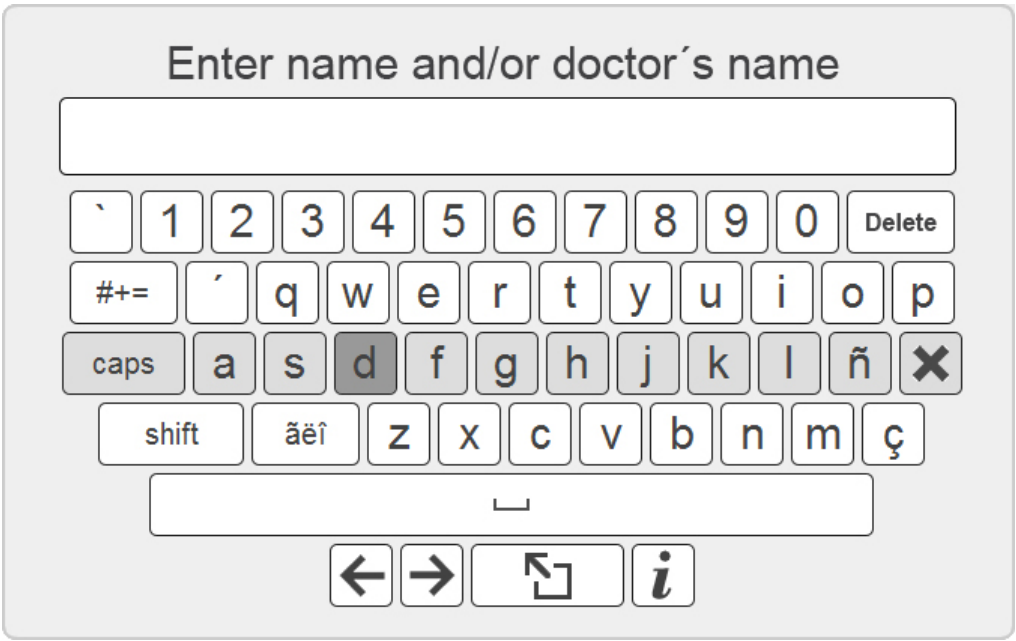


Figura 40: Teclado virtual de interfaz de formularios para campos de texto, indicador visual iterando sobre las teclas de una fila del teclado.

Las imágenes anteriores, son una muestra de cómo funciona el teclado virtual con la interfaz de sonido, con las otras dos interfaz de entrada no se hace la iteración del indicador sobre el teclado, pero se permite hacer uso del rato para pulsar sobre las teclas.

También cabe destacar que cuando la interfaz de salida de voz está activa, se puede escuchar lo que está escrito en el campo y los caracteres que se están introduciendo.

5.2.3.2. Combobox

Para el listado de opciones, o más popularmente conocido como combobox, la interfaz de formularios crea un teclado virtual con comandos específicos para poder navegar sobre las opciones y poder seleccionar una o varias, según lo permita el componente web. Además de los comandos para navegar, muestra un teclado predictivo con las letras necesarias para ir filtrando las opciones, hasta encontrar la deseada.

Si la interfaz de salida de voz está activa, cada vez que se pase por una opción, ésta es escuchada por el usuario.

El comportamiento de la funcionalidad sobre el combobox con cada interfaz de entrada es el siguiente:

- **Interfaz de teclado:** Con las teclas de navegación hacia arriba y hacia abajo, el usuario se puede desplazar dentro de las diferentes opciones, también puede pulsar las letras para ir creando un filtro dentro de la lista. Luego de encontrar la opción deseada, con la tecla ENTER es seleccionada.

- **Interfaz de voz:** Se puede navegar entre las opciones diciendo los comandos “subir” y “bajar” para navegar entre la lista de opciones, además diciendo una letra puede iniciar a filtrar las opciones. Luego de encontrada la opción, el usuario con el comando SELECCIONAR podrá escogerla.

- **Interfaz de sonido:** Sobre esta interfaz, realiza la misma iteración con el indicador visual sobre los comandos y letras que da el teclado. Para navegar dentro de la lista de opciones, solo hay que entrar en la fila donde están las flechas en dirección hacia arriba y abajo, también pueden seleccionar las letras que ofrece el teclado para filtrar las opciones. Luego de encontrada la opción deseada, el usuario deberá seleccionar la tecla que dice MARCAR para escogerla.

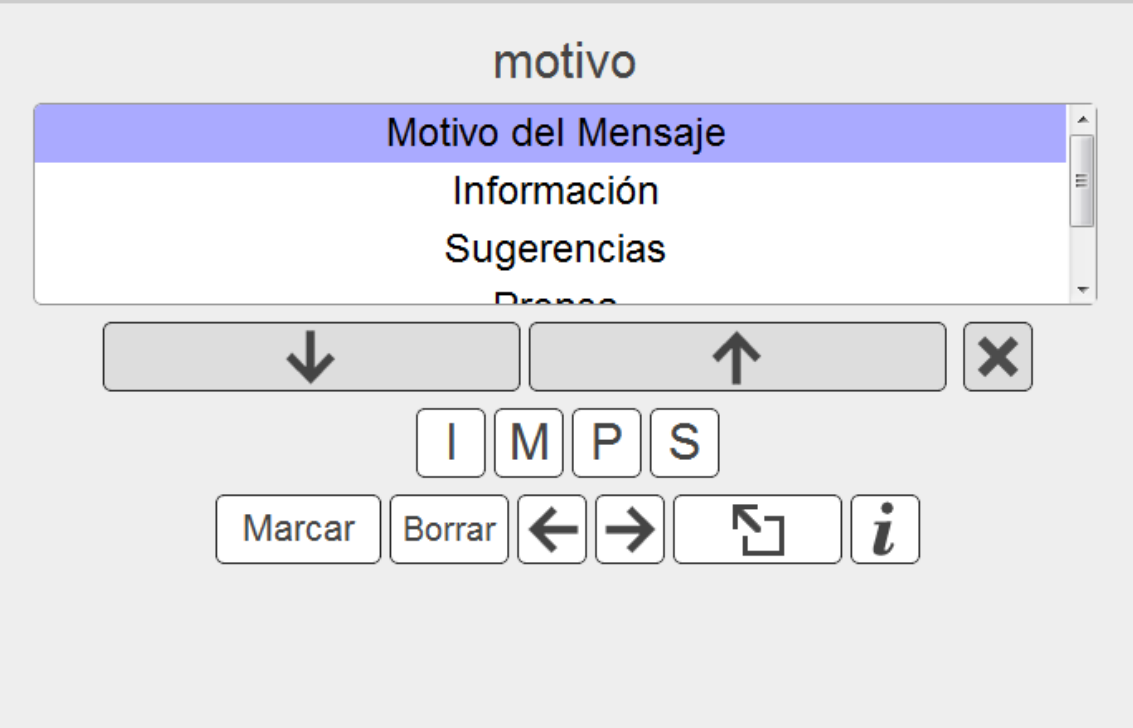


Figura 41: Teclado virtual de interfaz de formularios para combobox, indicador visual iterando sobre una fila del teclado.

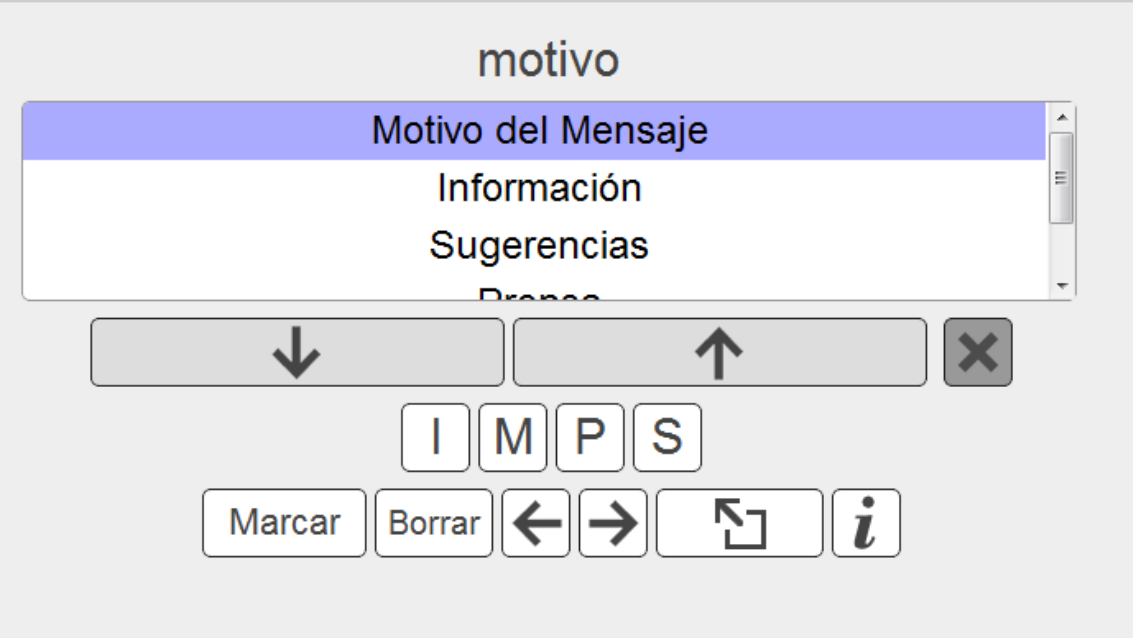


Figura 42: Teclado virtual de interfaz de formularios para combobox, indicador visual iterando sobre las teclas de una fila del teclado.

5.2.3.3. Radiobuttons y checkbox

Para un conjunto de radiobuttons o checkbox, la interfaz de formularios crea un teclado virtual con comandos específicos para poder navegar sobre las opciones y poder seleccionar una o varias, según lo permita el componente web.

Si la interfaz de salida de voz está activa, cada vez que se pasa por una opción, ésta es escuchada por el usuario.

El comportamiento de la funcionalidad sobre los radiobuttons o checkbox, con cada interfaz de entrada es el siguiente:

- **Interfaz de teclado:** Al igual que con los combobox, el usuario puede desplazarse con las teclas de navegación hacia arriba o abajo para seleccionar la opción que quiera. Luego de encontrar la opción deseada, con la tecla ENTER es seleccionada.

- **Interfaz de voz:** Se puede navegar entre las opciones diciendo los comandos “subir” y “bajar” para navegar entre la lista. Luego de encontrada la opción, el usuario con el comando SELECCIONAR podrá escogerla.

- **Interfaz de sonido:** Sobre esta interfaz, realiza la misma iteración con el indicador visual sobre los comandos. Para navegar dentro de la lista de opciones, solo hay que entrar en la fila donde están las flechas en dirección hacia arriba y abajo. Luego de encontrada la opción deseada, el usuario deberá seleccionar la tecla que dice MARCAR para escogerla.

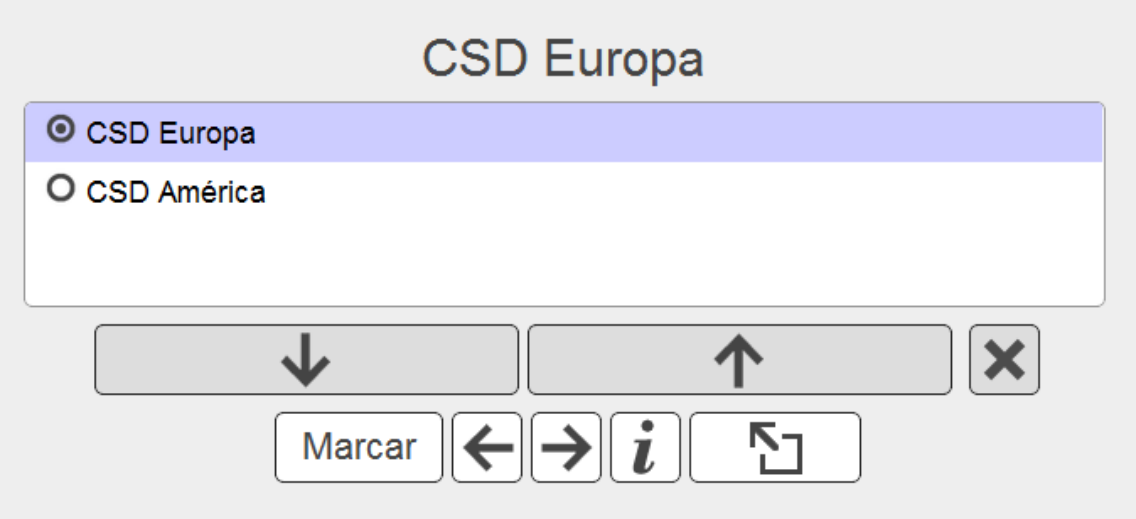


Figura 43: Teclado virtual de interfaz de formularios para radio buttons, indicador visual iterando sobre las filas del teclado.

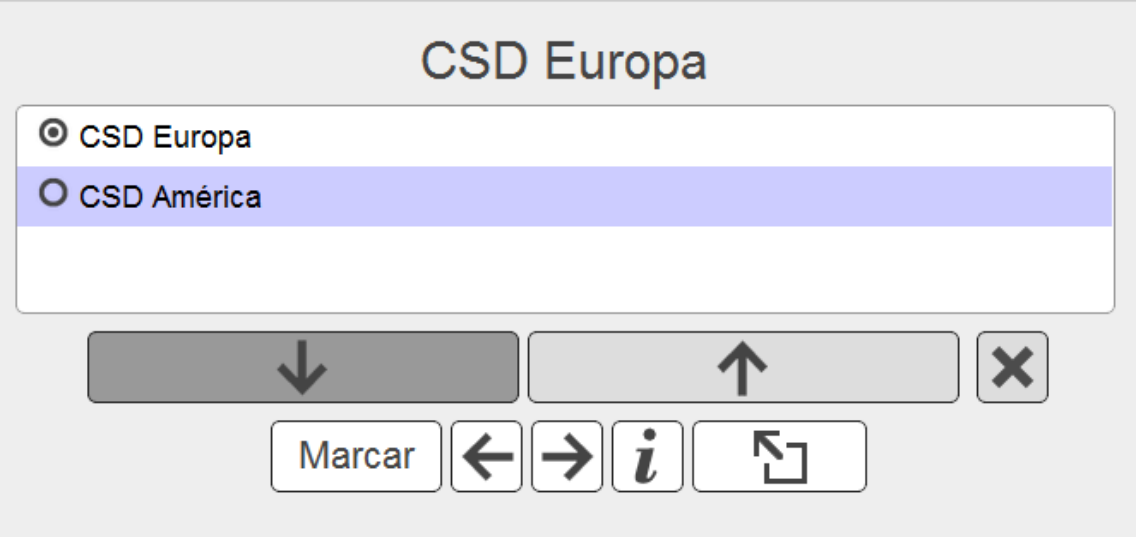


Figura 44: Teclado virtual de interfaz de formularios para radio buttons, ejecutando el comando hacia abajo para desplazarse en la lista de opciones.

Figura 45: Teclado virtual de interfaz de formularios para radio buttons, ejecutando el comando MARCAR para seleccionar la opcion.

5.2.3.4. Botones

En relación a los botones que pueda tener un formulario, tales como aquel que hace el envío del formulario o borra todo los campos del formulario dejándolos en su estado por defecto, la interfaz de formularios simplemente presta la opción para se ejecute la acción del botón.

Normalmente, dichos botones tienen nombres en de acuerdo a la acción que ejecuta el botón (ej. enviar, borrar, buscar, etc.), si el usuario tiene activa la interfaz de salida de voz, el nombre que tenga el botón será anunciado al usuario.

El comportamiento de la funcionalidad de los botones, con cada interfaz de entrada es el siguiente:

- **Interfaz de teclado:** Con la interfaz de entrada de teclado, el usuario simplemente tiene que pulsar la tecla ENTER para ejecutar la acción del botón.
- **Interfaz de voz:** Con esta interfaz, el usuario debe decir el comando ENTRAR para así ejecutar la acción del botón.
- **Interfaz de sonido:** Sobre esta interfaz, realiza la misma iteración con el indicador visual sobre los comandos. Para ejecutar la opción del botón, el usuario deberá seleccionar la tecla que dice INTRO.

Figura 46: Teclado virtual de interfaz de formularios para botones, selecciona el comando INTRO para ejecutar la acción del botón.

6. Implantación de inclusite en un sitio web

En este capítulo se explicará el proceso de cómo se implanta inclusite en un sitio web. Para comenzar; hay que destacar que el uso de inclusite sobre un sitio web, no precisará el cambio de su apariencia, con solo agregar una etiqueta ***script*** en el código HTML del sitio (de una forma similar que con google analitics), con la ubicación del archivo javascript que inicializa las funciones de inclusite, y algunos parámetros que llevan información del dominio del sitio web.

Las tecnologías que se usan para construir sitios web dinámicos son muy variadas. Por citar algunos ejemplos, se pueden encontrar páginas desarrolladas en PHP, JSP, ASP.NET, etc. pero al final todas esas tecnologías acaban generando código HTML que es interpretado por el navegador web. Inclusite recoge este principio para incluir en las etiquetas HTML, algunos atributos sin que afecte la re indexación del contenido, haciendo esto, como el punto clave para que inclusite sea independiente del lenguaje que se haya usado para programar el sitio web.

El primer paso, es la estructuración del contenido de la página de una forma semántica, y así ofrecérselo al usuario final de una manera lógica. ¿Cómo se consigue dicha semántica? Asignando los roles estandarizados por el WAI-ARIA, en las etiquetas dentro del código HTML.

Para prestar una mejor explicación de cómo se implanta inclusite en un sitio web, a continuación se muestra un ejemplo del proceso de cómo se asignan los roles, el código de las reglas que asignan los roles, etc.

6.1. Asignando roles

Actualmente los sitios web son diseñados con diferentes plantillas, a base de esa idea, se puede establecer que etiquetas del código HTML llevarán roles y qué tipo de role llevarán, entonces se crean reglas para cada una de esas plantillas. **Las reglas** no son más que clases java con un método ***main*** que ejecuta modificaciones sobre el

contenido de una página web, también existen casos que la regla depende de la URL para ejecutarse o no.

Para evitar que las reglas se ejecuten en cualquier sitio web que haga peticiones al servicio, por medio de un archivo de configuración se determina en relación a un dominio, qué reglas se pueden ejecutar. Y dentro de las reglas existe una condición que determina cuales urls dentro del dominio pueden ejecutarla.

Para empezar se crea una regla para aquellas plantillas que aparecen en todas la páginas del sitio web; por ejemplo, la zonas de encabezado, pie de página, aquella zona que determina donde estará el contenido principal, alguna barra lateral, etc.

A continuación una imgen de una página web con un remarcado en las zonas que se muestran también en otras páginas del sitio web.

Figura 47: Ejemplo página web.

En la imagen anterior se pude observar en remarcado en rojo, el encabezado de la página web, ese encabezado se le asignará por medio de la regla el role ***complementary*** ya que dentro contiene un menú de navegación, un formulario de búsqueda, imágenes y otros enlaces de interés.

Al menú de navegación, se le asigna el role ***navigation*** y a cada una de sus opciones, se le asigna el role ***item***, al formulario el role ***form***, a la zona donde se encuentran los enlaces Accesibilidad, Versión solo texto, Mapa de la web se le puede denominar

como un ***conteninfo*** o ***navigation***, en este ejemplo se le asigna el role conteninfo y a estos enlaces el role ***link***.

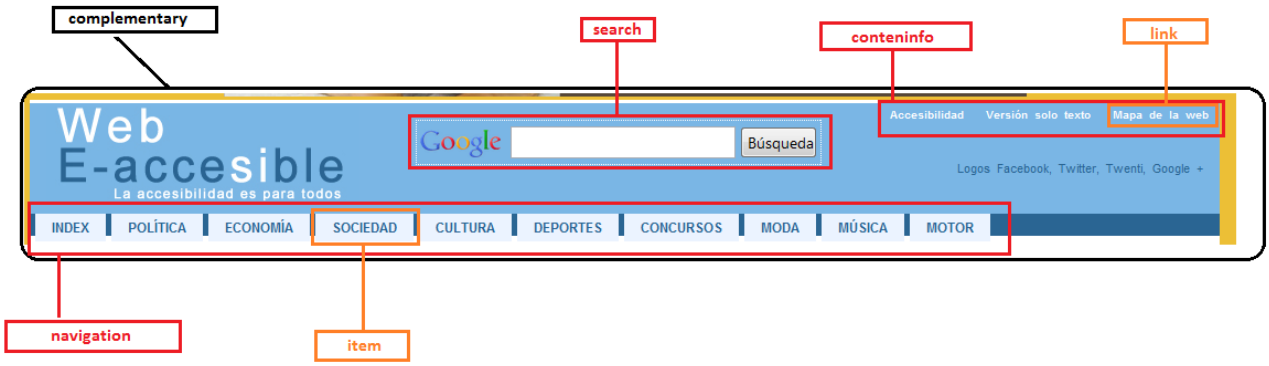


Figura 48: Asignación de roles en el encabezado.

Además de asigna el atributo role a las etiquetas HTML de la página web, por medio de las reglas también se fijan los atributos necesarios para que el usuario pueda navegar sobre las opciones de una forma determina, por ejemplo, para saber el nombre del área a la cual va entrar se ponen el atributo ***aria-label***, o si existe un área a la cual se le quiere dar más relevancia (el menú de navegación), pero ésta, se encuentra dentro del código HTML escrita después de áreas no tan relevantes, entonces se le puede dar más prioridad con el atributo ***tabindex***, y otra cantidad de situaciones que llevan a usar diferentes atributos, que serán entonces implementados bajo el criterio del desarrollador de la regla, como por ejemplo ***inclu-offsetx*** que se encargaría de desplazar, el cuadrado rojo que remarca la zona, en n pixeles sobre el eje x.

A continuación se mostrarán una serie de imágenes de cómo se ve la asignación de los roles sobre el código HTML del contenido del encabezado y el menú de navegación de la página web.

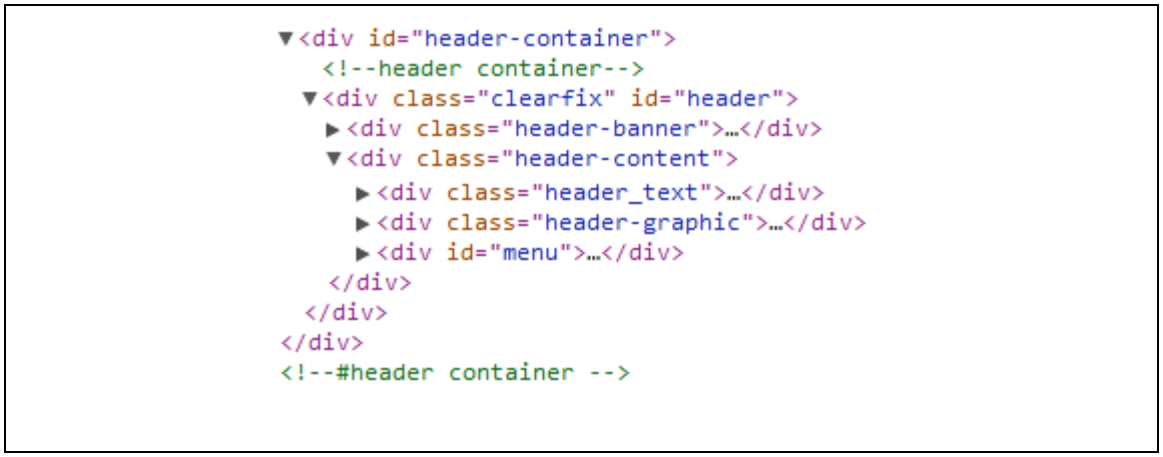


Figura 49: Código HTML comprimido del encabezado sin asignación de roles.

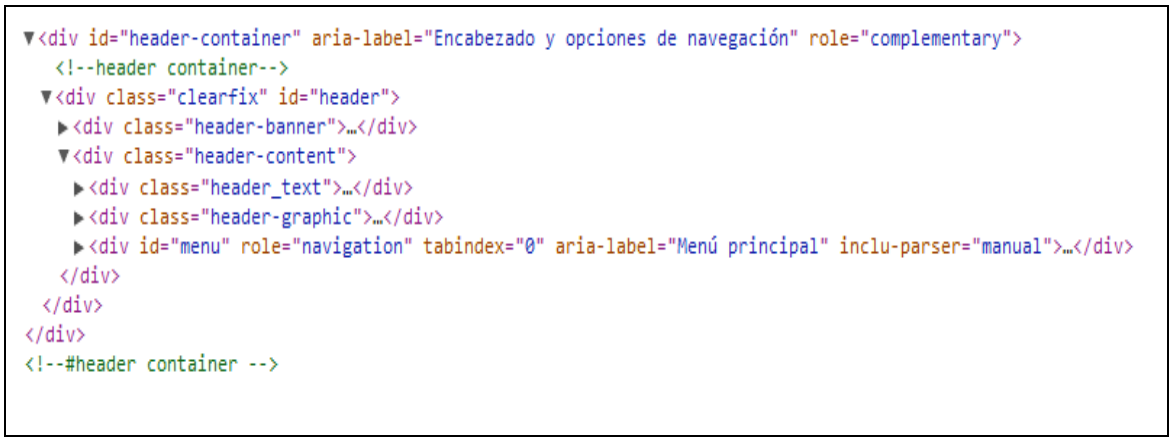


Figura 50: Código HTML comprimido del encabezado con roles asignados.

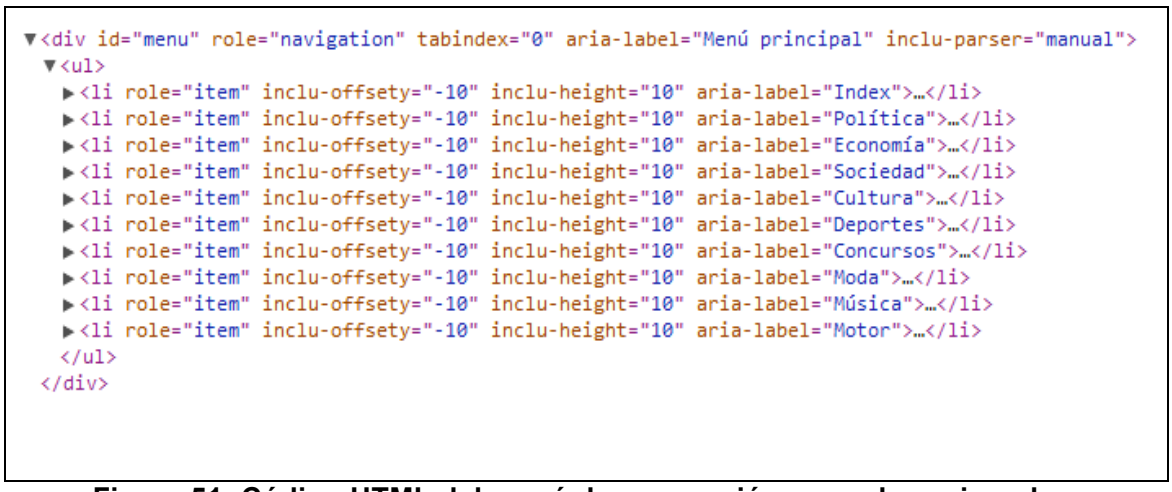


Figura 51: Código HTML del menú de navegación con roles asignados.

Parte del código de la regla que asigna los roles al menú se muestra a continuación, como primera parte el método **main** de la regla, él cual recibe como parámetro de

entrada un objeto de tipo HttpMessage de la librería creada por un Desarrollador de inclusite, para el análisis del código HTML de las páginas web, la cual pasándole la URL con una expresión regular, se determina si ejecuta o no la regla sobre la URL que hace petición al servicio.

```
public void main(HttpMessage httpMessage){

    if (httpMessage.isUrl("^http\:\/\/webexperimental\.csd\.com\.es\/webexperimental\d\/es\/.*")) {
        String HTMLBODY = httpMessage.getBody();
        document = Jsoup.parse(HTMLBODY);
        body = document.body();

        setNavigation();

        httpMessage.setBody(document.outerHtml());
    }

}
```

Figura 52: Código del método main de una regla.

```
private void setNavigation(){
    Element header = getElement(body,"div#header-container");
    HashMap<String,String> attsItemsMap = new HashMap<String,String>();
    HashMap<String,String> attsMap = new HashMap<String,String>();
    attsItemsMap.put("inclu-offsety","-10");
    attsItemsMap.put("inclu-height","10");
    attsMap.put("aria-label","Menú principal");
    attsMap.put("tabindex","0");
    setNavigation(header, "div#menu", attsMap, "li", attsItemsMap);
    attsMap.clear();
    attsMap.put("aria-label","Menú informativo");
    setNavigation(header, "div#flag-content", attsMap, "li", attsItemsMap);
}
```

Figura 53: Parte del código de la regla que asigna roles al menú de navegación.

En la figura anterior se recupera un objeto de tipo **Element** de la librería Jsoup[8], luego se crean dos objetos de tipo **HashMap** que contendrán nombres y valores de los atributos que se podrán en las etiquetas. Seguido de lo anterior se llama un método llamado **setNavigation** que recibe como parámetros el objeto Element donde se buscarán nuevos objetos que representan las etiquetas HTML, para asignarles los atributos que contienen los HashMaps attsMap y attsItemsMap.

[8] http://jsoup.org/

El código completo de la regla se puede ver en el **ANEXO 3**.

Para el área de la barra lateral al igual que el encabezado, se le pone el role complementary, luego al formulario y al banner de publicidad se les asigna los roles form y ***banner***, repectivamente.

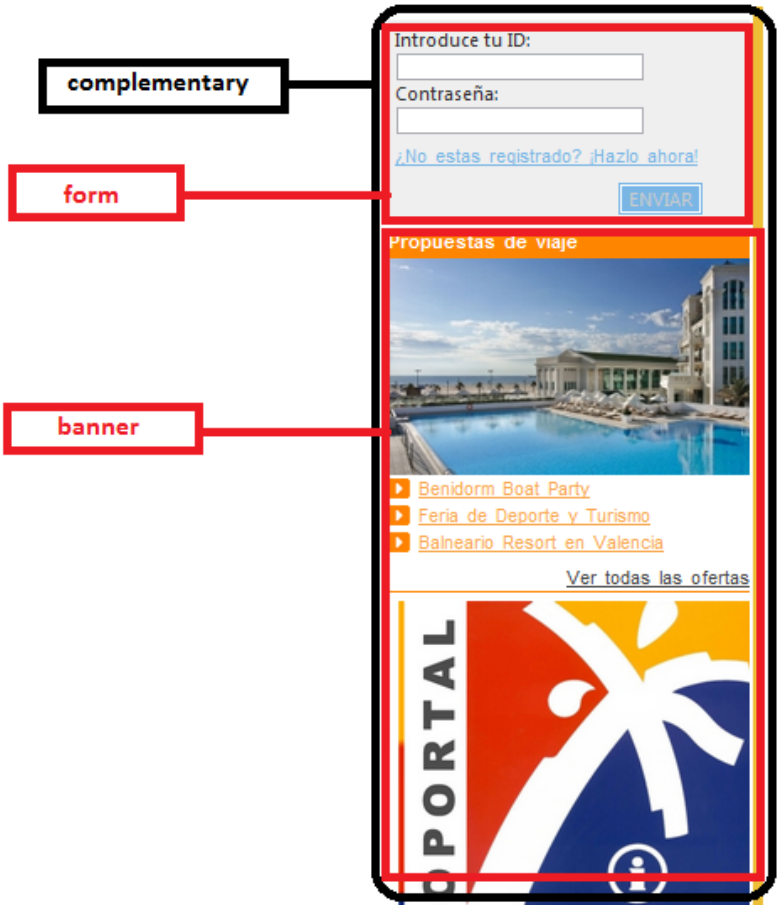


Figura 54: Asignación de roles a la barra lateral.

En el área principal de la página web simplemente se agrega el role ***main***, dentro de esta primera regla que quiere abarcar todo el contenido general que hay en la página web, en las siguientes reglas, se asignan los roles del área del contenido principal, las cuales dependen de la información que prestan, el diseño de la web, etc. la mayoría de contenido dentro del área principal se le asignan los roles ***article, img, link*** y ***complementary*** y en algunos casos existen formularios para hacer comentarios sobre algún artículo o publicación de la web, entonces se usa también el role **form**. El role article es aquel que remarca el contenido o la información relevante a la cual quiere llegar el usuario.

Figura 55: Asignación de role article a un contenido.

Las reglas que asignan estos roles sobre este sitio web, son las siguientes:

- **WebexGeneralTemplate:** Regla que asigna roles y atributos a las etiquetas HTML que aparecen en todo el sitio web, tales como las que representan barras laterales, pie de página, encabeza, zona de contenido, etc. **Ver ANEXO 2.**

- **WebexNavigation:** Regla que asigna el role **navigation** y atributos a la etiqueta que representa el menú de navegación de la página web. **Ver ANEXO 3.**

- **WebexIndex:** Regla que se encarga de asignar roles y atributos a las etiquetas de la página web de inicio del sitio web. La mayoría de veces los sitios web contienen una plantilla para paginas de inicio tanto del sitio web, como para sus diferentes contenidos. **Ver ANEXO 4.**

- **WebexContentGeneric:** Regla que asigna roles y atributos a las etiquetas de la plantilla para contenido que aparece en páginas diferentes a la de inicio. **Ver ANEXO 5.**

- **WebexForm:** Regla que asigna el role **form** y atributos a las etiquetas que tienen formulario dentro del sitio web. **Ver ANEXO 6.**

- **WebexModelService:** Regla que asigna roles y atributos a etiquetas que representan contenido dinámico, es decir ese contenido que se actualiza en el lado del cliente sin hacer petición al servidor del sitio web. **Ver ANEXO 7.**

Como resultado del paso de la implantación de inclusite sobre el sitio web experimental, se muestra como el ejemplo como quedaría el menú de navegación de la página.

Figura 56: sitio web usando inclusite, remarcando el menú de navegación.

7. Conclusiones

Aunque existan estándares que se encargan de orientar a los diseñadores y desarrolladores de sitios o aplicaciones web, a crear contenido accesible, las limitaciones para que un usuario pueda interactuar con él siguen siendo altas, ya que si el usuario no dispone de alguna ayuda técnica no puede acceder, y muchas de estas ayudas son de pago y algunas de un coste bastante alto.

Entonces crear herramientas que permitan acceder a los contenidos web sin necesidad de una previa instalación por parte del usuario y sean usadas por sitios web, reduce dichas limitaciones, haciendo así que esa web proporcione un acceso equitativo, además de una igualdad de oportunidades a personas que sufren algún tipo de discapacidad.

Los sitios web accesibles pueden ayudar a personas con discapacidad a integrarse y a que participen más activamente en la sociedad, también hay que tener en cuenta, que junto a la accesibilidad va enlazada la usabilidad, la cual se encarga de hacer fácil la navegación dentro del sitio, y esto hace que los visitantes vuelvan a éste, ya que es más probable que un usuario vuelva a una página de fácil acceso a una que no lo es.

Durante el desarrollo de inclusite, siempre se ha estado pensando en la usabilidad de las interfaces, además de prestar usabilidad a la página web donde inclusite es usado. En muchas ocasiones un usuario entra a una página web y por la sobre carga de contenido no sabe por dónde iniciar a navegar por él, inclusite presenta al usuario la estructura de la web de forma semántica sin importar la cantidad de información que haya, lo que conlleva a que el aprendizaje del manejo de la web sea mucho más rápido e intuitivo.

Por otra parte, cabe destacar que el uso de metodologías a la hora del desarrollo de un proyecto software, estas hacen que el producto resultante sea flexible ante posibles cambios, que la evolución del producto se documente para futuros integrantes del equipo de desarrollo, etc. en el caso del la implementación de inclusite se uso SCRUM

8. Trabajo futuro

En esta sección del trabajo se presentan ideas que se tienen planteadas para mejorar inclusite:

- Mejorar las buenas prácticas para implantar inclusite en un sitio web.
- Automatizar en un porcentaje alto la implantación de inclusite, sobre todo en un sitio web que use el nuevo estándar de HTML5.
- Hacer compatible inclusite con lectores de pantalla.
- Mejorar la usabilidad de las diferentes interfaces del sistema.
- Crear una interfaz para que inclusite pueda ser usado en los navegadores web de los smartphones.
- Crear una interfaz que reconozca expresiones del rostro o parpadeo, para permitir al usuario usar la cámara web del ordenador como otra herramienta para interactuar sobre un sitio web con inclusite.
- Crear un sitio web que pueda ser utilizado por usuarios discapacitados como buscador de información haciendo uso de inclusite (solo buscaría información en los sitios web que tienen implantado inclusite, para no llevar al usuario a una página donde no pueda navegar, además que existe un programa de pruebas, donde se ha implantado inclusite en la wikipedia[9], lo cual haría que la búsqueda de información sea en un ámbito grande).

[9] http://es.wikipedia.org.accesible.inclusite.com

9. Bibliografía

[1] http://www.w3.org/

[2] http://www.w3.org/WAI/intro/aria

[3] http://www.freedomscientific.com

[4] http://www.microsoft.com

[5] http://windows.microsoft.com/es-ES/internet-explorer/products/ie/home

[6] http://live.gnome.org/Orca

[7] http://live.gnome.org/Orca/Firefox

[8] http://www.w3.org/WAI/gettingstarted/Overview.html

[9] http://www.w3.org/TR/WAI-WEBCONTENT/

[10] http://accesibilidadweb.dlsi.ua.es/?menu=niveles-2.0

[11] http://www.w3.org/TR/WCAG20/

[12] http://www.librosweb.es/javascript/capitulo1.html

[13] http://www.librosweb.es/ajax/capitulo1.html

[14] http://www.json.org/json-es.html

[15] http://greasyspoon.sourceforge.net/index.html

[16] http://www.icap-forum.org

[17] http://jsoup.org/

[18] http://en.wikipedia.org/wiki/Squid_(software)

[19] http://en.wikipedia.org/wiki/Speech_synthesis

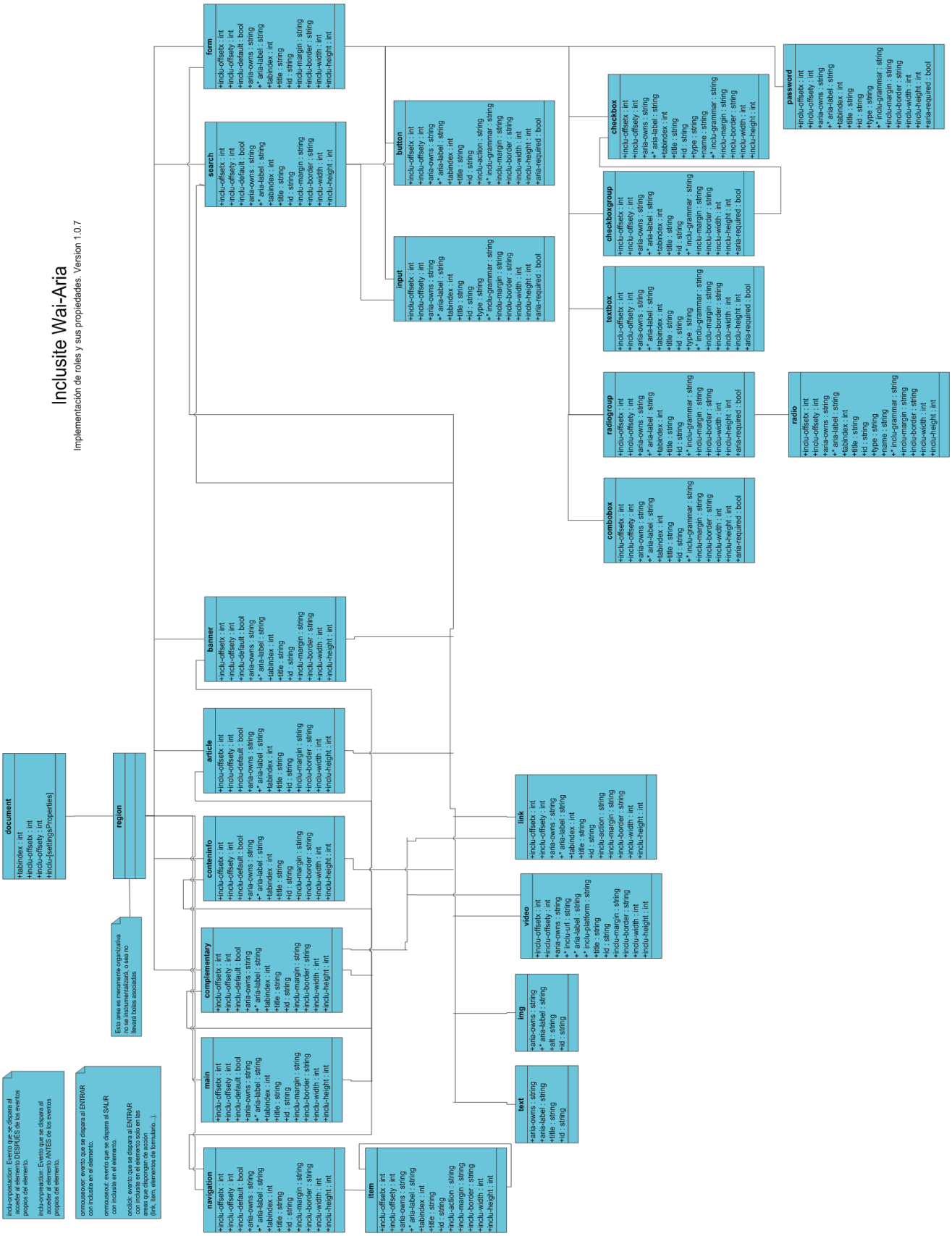
[20] http://en.wikipedia.org/wiki/Speech_recognition

[21] http://www.inteco.es/Accesibilidad/difusion/Manuales_y_Guias/WAI_ARIA

[22] http://www.w3c.es/divulgacion/guiasbreves/accesibilidad

[23] http://es.wikipedia.org/wiki/ActionScript

ANEXO 1. INCLUSITE WAI-ARIA



ANEXO 2. Regla WebexGeneralTemplate

```
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.inclusite.htmlparser.HttpMessage;

public class WebexGeneralTemplate {

 public Document  document = null;
 public Element body = null;

 public void main(HttpMessage httpMessage){

  if (httpMessage.isUrl("^http\:\/\/webexperimental\.csd\.com\.es\/webexperimental\d\/es\/.*")) {
            String HTMLBODY = httpMessage.getBody();
            document = Jsoup.parse(HTMLBODY);
            body = document.body();
            body.attr("role","document");
            body.attr("onhelp","return false;");
            body.attr("style",body.attr("style")+"width:auto;");
            Elements content = body.select("div#container,#footer-container");
            content.attr("style","width: 80%;min-width:80%;margin: 0 auto;");
            content = body.select("div#franja");
            content.attr("style","width: 72%;margin: 0 auto;");

            setInclusiteAreas();
            httpMessage.setBody(document.outerHtml());
  }
 }

private void setInclusiteAreas(){

  setInclusiteGlobalAreas(body.select("div#header-container"),"complementary","Encabezado y opciones
de navegación","","",false);
  setInclusiteGlobalAreas(body.select("div#body-container"),"main","Contenido","","",false);

  Elements content = body.select("div#content-main");
  content.attr("tabindex","0");
  setInclusiteGlobalAreas(body.select("div#content-main"),"complementary","Contenido
principal","","",false);
  setInclusiteGlobalAreas(body.select("div#content-column"),"complementary","Contenido barra
lateral","","",false);
  setInclusiteGlobalAreas(body.select("div#franja"),"contentinfo","Información","","",false);
  setInclusiteGlobalAreas(body.select("div#footer-container"),"contentinfo","Pie de
p&aacute;gina","","",false);

  setInclusiteGlobalAreas(body.select("div#formulario_registro"),"complementary","Accede como
usuario","","",false);
  setInclusiteGlobalAreas(body.select("div#banner1"),"banner","","div#banner1 p.titulo","",false);
  setInclusiteGlobalAreas(body.select("div#banner2"),"banner","","div#banner2 img","",false);
}

private void setInclusiteGlobalAreas(Elements elems,String roleArea,String titulo,String queryTitle,String
id,boolean isDefault){
```

```
  if(elems.size()>0){
        for(int i=0;i<elems.size();i++){
              Element elem = elems.get(i);
              setInclusiteGlobalArea(elem,roleArea,titulo,queryTitle,id,isDefault);
        }
  }
}


private void setInclusiteGlobalArea(Element elem,String roleArea,String titulo,String queryTitle,String
id,boolean isDefault){

        String label = titulo;
        if(!queryTitle.equals("")){
              if(body.select(queryTitle).text().equals("")){
                    label = titulo + " " + body.select(queryTitle).attr("alt");
              }else{
                    label = titulo + " " + body.select(queryTitle).text();
              }
        }
        elem.attr("aria-label",label);
        elem.attr("role",roleArea);
        if(id.length() > 0 ) elem.attr("id",id);
        if(isDefault)elem.attr("inclu-default","true");
  }
}
```

ANEXO 3. Regla WebexNavigation

```
import org.inclusite.htmlparser.HttpMessage;
import org.jsoup.*;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import java.util.HashMap;

public class WebexNavigation{

 public Document  document = null;
 public Element body = null;

 public void main(HttpMessage httpMessage){

  if (httpMessage.isUrl("^http\:\/\/webexperimental\.csd\.com\.es\/webexperimental\d\/es\/.*")) {
            String HTMLBODY = httpMessage.getBody();
            document = Jsoup.parse(HTMLBODY);
            body = document.body();

            setNavigation();

            httpMessage.setBody(document.outerHtml());
  }

 }

private void setNavigation(){
  Element header = getElement(body,"div#header-container");
  HashMap<String,String> attsItemsMap = new HashMap<String,String>();
  HashMap<String,String> attsMap = new HashMap<String,String>();
  attsItemsMap.put("inclu-offsety","-10");
  attsItemsMap.put("inclu-height","10");
  attsMap.put("aria-label","Menú principal");
  attsMap.put("tabindex","0");
  setNavigation(header, "div#menu", attsMap, "li", attsItemsMap);
  attsMap.clear();
  attsMap.put("aria-label","Menú informativo");
  setNavigation(header, "div#flag-content", attsMap, "li", attsItemsMap);
}


private Element setNavigation(Element region, String navName, HashMap<String,String> attsMap, String
itemIterator, HashMap<String,String> attsMapItems)
{
  Element navegador = getElement(region, navName);
  if (navegador != null){
            setToElement("navigation", navegador, attsMap);
            if (itemIterator != null && !itemIterator.isEmpty()) {
                      Elements items = getElements(navegador, itemIterator);
                      if(items != null && items.size() > 0) {
                                for( Element itemElement : items)
                                {
                                Element pd = itemElement.select("img").first();
```

```
                        if (pd != null){
                                String label = pd.attr("alt");
                                attsMapItems.put("aria-label",label);
                        }else{
                                attsMapItems.put("aria-label", itemElement.text());
                        }
                        setToElement("item", itemElement, attsMapItems);
                        }
                }
        }
  }
  return navegador;
}

private Element getElement(Element ElementToFind, String navName) {
  Elements elems = getElements(ElementToFind, navName);
  if (elems == null || elems.size() < 0) { return null; }
  return elems.first();
}

}
```

ANEXO 4. Regla WebexIndex

```
import org.inclusite.htmlparser.HttpMessage;
import org.jsoup.*;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import java.util.HashMap;

public class WebexIndex{

 public Document  document = null;
 public Element body = null;
 public void main(HttpMessage httpMessage){

  if (httpMessage.isUrl("^http\:\/\/webexperimental\.csd\.com\.es\/webexperimental\d\/es\/(index).*")) {
            String HTMLBODY = httpMessage.getBody();
            document = Jsoup.parse(HTMLBODY);
            body = document.body();

            modifyContent();
            setInclusiteContent();

            httpMessage.setBody(document.outerHtml());
  }
 }

private void setInclusiteContent(){
  HashMap<String,String> attsMap = new HashMap<String,String>();
  attsMap.put("aria-label","Titulares");
  setToElements("complementary","div#main-news", body,attsMap);
  setToElements("complementary","div#highlight div.type-section span", body,null);
  setToElements("complementary","div.right-column", body,null);
  setToElements("article","div.right-column div.news", body,null);

  setInclusiteToContent(body.select("div#news-container div.noticia"),"article","","div#news-container
div.noticia div.h3","","","","","","",false);

  setInclusiteToContent(body.select("div#highlight div.noticias-dia>div"),"article","","div#highlight
div.noticias-dia>div div.h2","","","","20","",false);
  Elements tabs = body.select("div#highlight div.type-section span");
  setIdOrAriaOwnsToContent(tabs,"itab_","",true);
  setAction(tabs,"a[href]");
  setIdOrAriaOwnsToContent(body.select("div#highlight div.noticias-dia>div"),"","itab_",false);

  setInclusiteToContent(body.select("div.little-news"),"article","","div.videos-title","-10","10","","","",false);


}

private void setAction(Elements elems,String query){
  if(elems.size()>0){
            for(Element elem : elems){
                  Element link = (elem.select(query).size()>0)?elem.select(query).first():null;
                  if(link != null)elem.attr("inclu-action",link.attr("href"));
```

```
		}
	}
}

private void setIdOrAriaOwnsToContent(Elements elems,String id,String ariaOwns,boolean hastitle){
	if(elems.size()>0){
		int index = 0;
		for(Element elem : elems){
			if(id.length()>0)elem.attr("id",id + String.valueOf(index++));
			if(ariaOwns.length()>0)elem.attr("aria-owns",ariaOwns + String.valueOf(index++));
			if(hastitle)elem.attr("aria-label",elem.text());
		}
	}
}

private void setInclusiteToContent(Elements elems,String roleArea,String titulo,String queryTitle,String
offsetx, String width,String offsety, String height,String ariaOwns,boolean isDefault){
	if(elems.size()>0){
		for(int i=0;i<elems.size();i++){
			Element elem = elems.get(i);
			String label = titulo;
			if(!queryTitle.equals("")){
				label = titulo + " " + body.select(queryTitle).get(i).text();
			}
			elem.attr("aria-label",label);
			elem.attr("role",roleArea);
			if(isDefault)elem.attr("inclu-default","true");
			setOffset(elem,offsetx,width,offsety,height);
		}
	}
}


private void setOffset(Element elem,String offsetx, String width,String offsety, String height){
	if(offsetx.length() >0)elem.attr("inclu-offsetx",offsetx);
	if(offsety.length() >0)elem.attr("inclu-offsety",offsety);
	if(width.length() >0)elem.attr("inclu-width",width);
	if(height.length() >0)elem.attr("inclu-height",height);
}

private void setToElements(String role, String navName, Element elementToFind,
HashMap<String,String> attsMap) {
	Elements elements = getElements(elementToFind, navName);
	for(Element element : elements)
	{
		setToElement(role, element, attsMap);
	}
}

private Element setToElement(String role, String navName, Element ElementToFind,
HashMap<String,String> attsMap) {
	Element element = getElement(ElementToFind, navName);
	return setToElement(role, element, attsMap);
}

private Element setToElement(String role, Element element, HashMap<String,String> attsMap) {
	if (element != null && role != null && !role.isEmpty() && !element.html().equals("")){
		element.attr("role", role);
```

```
            if(attsMap != null) {
                  for(String key : attsMap.keySet())
                  {
                        String value = attsMap.get(key);
                        if (value != null && !value.isEmpty()) element.attr(key, value);
                  }
            }
    }
    return element;
}

private Element setToElement(String role, String navName, HashMap<String,String> attsMap) {
    Element element = getElement(null, navName);
    return setToElement(role, element, attsMap);
}



private Element getElement(Element ElementToFind, String navName) {
    Elements elems = getElements(ElementToFind, navName);
    if (elems == null || elems.size() < 0) { return null; }
    return elems.first();
}

private Elements getElements(Element ElementToFind, String navName) {
    if (navName == null || navName.isEmpty()) { return null; }
    if (ElementToFind != null) {
            return ElementToFind.select(navName);
    }
    return document.select(navName);
}

private void modifyContent(){
    Element content = getElement(null,"div#titulares");
    if(content != null){
            Element lastHour = getElement(content,"div.ultima-hora");
            Elements noticias = getElements(content,"div.noticia,hr");
            String article = "<div role='article' aria-label='última hora'>" + lastHour.toString() +
elementHtmlToString(noticias) + "</div>";
            content.html(article);
    }

}

private String elementHtmlToString(Elements elems){
    String html = "";
    for(Element elem:elems){
            html = html + elem.toString();
    }
    return html;
}
}
```

ANEXO 5. Regla WebexContentGeneric

```
import org.inclusite.htmlparser.HttpMessage;
import org.jsoup.*;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import java.util.HashMap;

public class WebexContentGeneric{

 public Document  document = null;
 public Element body = null;
 public void main(HttpMessage httpMessage){

  if
(httpMessage.isUrl("^http\:\/\/webexperimental\.csd\.com\.es\/webexperimental\d\/es\/(politica|economia|s
ociedad).*")) {
            String HTMLBODY = httpMessage.getBody();
            document = Jsoup.parse(HTMLBODY);
            body = document.body();

            modifyContent();
            setInclusiteContent();
            setLabelToContent();

            httpMessage.setBody(document.outerHtml());
  }
 }

private void setInclusiteContent(){
  HashMap<String,String> attsMap = new HashMap<String,String>();
  attsMap.put("aria-label",body.select("div#content-main div.top-bar div.title").text());
  setToElements("article","div#content-main div.top-bar", body,attsMap);

  attsMap.clear();
  String label = body.select("div#contenido-interior div.noticia-seccion div.title-main-section").text();
  attsMap.put("aria-label",label);
  attsMap.put("inclu-width","-15");
  Element  content = setToElement("complementary","div#contenido-interior div.noticia-seccion",
body,attsMap);
  label = content.select("p.titulo").text();
  attsMap.put("aria-label",label);
  attsMap.put("inclu-width","0");
  setToElement("article","div#main_art", content,attsMap);

  attsMap.clear();
  attsMap.put("aria-label","Otras Noticias");
  attsMap.put("inclu-offsetx","570");
  attsMap.put("inclu-width","-570");
  Element colNoticias = setToElement("complementary","div#columna-noticias", body,attsMap);
  attsMap.clear();
  attsMap.put("inclu-offsetx","0");
  attsMap.put("inclu-width","0");
  setToElements("article","div.col-noticia", colNoticias,attsMap);
```

```
}

private void setLabelToContent(){
  Elements noticias = body.select("div#columna-noticias div.col-noticia");
  for(Element noticia : noticias){
        noticia.attr("aria-label",noticia.select("div.h2").text());
  }

}
private void setToElements(String role, String navName, Element elementToFind,
HashMap<String,String> attsMap) {
  Elements elements = getElements(elementToFind, navName);
  for(Element element : elements)
  {
        setToElement(role, element, attsMap);
  }
}

private Element setToElement(String role, String navName, Element ElementToFind,
HashMap<String,String> attsMap) {
  Element element = getElement(ElementToFind, navName);
  return setToElement(role, element, attsMap);
}

private Element setToElement(String role, Element element, HashMap<String,String> attsMap) {
  if (element != null && role != null && !role.isEmpty()){
        element.attr("role", role);
        if(attsMap != null) {
              for(String key : attsMap.keySet())
              {
                    String value = attsMap.get(key);
                    if (value != null && !value.isEmpty()) element.attr(key, value);
              }
        }
  }
  return element;
}

private Element setToElement(String role, String navName, HashMap<String,String> attsMap) {
  Element element = getElement(null, navName);
  return setToElement(role, element, attsMap);
}


private Element getElement(Element ElementToFind, String navName) {
  Elements elems = getElements(ElementToFind, navName);
  if (elems == null || elems.size() < 0) { return null; }
  return elems.first();
}
private Elements getElements(Element ElementToFind, String navName) {
  if (navName == null || navName.isEmpty()) { return null; }
  if (ElementToFind != null) {
        return ElementToFind.select(navName);
  }
  return document.select(navName);
}

private void modifyContent(){
```

```
  Element content = getElement(null,"div#contenido-interior div.noticia-seccion");
  Elements ps = getElements(content,"div.title-main-section, p");
  String article = "<div role='article' id='main_art'>" + elementHtmlToString(ps) + "</div>";
  Element comments = getElement(content,"div.comentarios");
  content.html(article + comments.toString());

}

private String elementHtmlToString(Elements elems){
  String html = "";
  for(Element elem:elems){
          html = html + elem.toString();
  }
  return html;
}
}
```

ANEXO 6. Regla WebexForm

```
import org.inclusite.htmlparser.HttpMessage;
import org.jsoup.*;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import java.util.HashMap;

public class WebexForm{

 public Document  document = null;
 public Element body = null;
 public void main(HttpMessage httpMessage){

  if (httpMessage.isUrl("^http\:\/\/webexperimental\.csd\.com\.es\/webexperimental\d\/es\/.*")) {
            String HTMLBODY = httpMessage.getBody();
            document = Jsoup.parse(HTMLBODY);
            body = document.body();

            setInclusiteForm();
            setLabelToFieldsRegistro();

            httpMessage.setBody(document.outerHtml());
  }
 }

private void setInclusiteForm(){
  HashMap<String,String> attsMap = new HashMap<String,String>();
  attsMap.put("aria-label","B&uacute;squeda");
  Element form = setToElement("form","div#buscador-google form",body,attsMap);
  if(form != null){
            attsMap.put("aria-label","Termino de búsqueda:");
            setToElement("input","input[type=text]",form,attsMap);
            attsMap.put("aria-label","Enviar");
            setToElement("button","input[type=submit]",form,attsMap);
  }

  attsMap.put("aria-label","Login");
  form = setToElement("form","div#formulario_registro form",body,attsMap);
  if(form != null){
            attsMap.put("aria-label","Introduce tu ID:");
            setToElement("input","input",form,attsMap);
            attsMap.put("aria-label","Contraseña:");
            setToElement("password","input[type=password]",form,attsMap);
            attsMap.put("aria-label","Enviar");
            setToElement("button","input[type=submit]",form,attsMap);
  }

  attsMap.put("aria-label","Inserta tu comentario");
  form  = setToElement("form","form#formulario-comentarios",body,attsMap);
  if(form != null){
            attsMap.put("aria-label","Comentario");
            setToElement("textbox","textarea",form,attsMap);
            attsMap.put("aria-label","Enviar");
```

```
            setToElement("button","input[type=submit]",form,attsMap);
    }

    attsMap.put("aria-label","Registro");
    form = setToElement("form","form#form-registro",body,attsMap);
    if(form != null){
            setToElements("input","input[type=text]",form,null);
            setToElements("password","input[type=password]",form,null);
            setToElements("radio","input[type=radio]",form,null);
            setToElement("button","input[type=button]",form,null);
            attsMap.put("aria-label",form.select(">span").text());
            setToElement("checkboxgroup","div#activar",form,attsMap);
            setToElement("checkbox","input[type=checkbox]",form,attsMap);
            attsMap.put("aria-label","Fecha de Nacimiento,");
            setToElements("combobox","select",form,attsMap);
    }

}

private void setLabelToFieldsRegistro(){
    Element form = getElement(body,"form#form-registro");
    if(form != null){
            Elements inputs = getElements(form,"input[type=text],input[type=password]");
            for(Element input : inputs){
                    input.attr("aria-label",input.parent().text());
            }

            Element button = getElement(form,"input[type=button]");
            if(button != null){
                    button.attr("aria-label",form.select(">p").get(1).text());
            }

            Element checki = getElement(form,">input[type=checkbox]");
            if(checki!= null){
                    checki.attr("aria-label",form.select(">p").get(0).text()+ ", " +checki.attr("aria-label"));
            }

            Elements sels = getElements(form,"select");
            for(Element sel : sels){
                    sel.attr("aria-label",sel.attr("aria-label")+ " " + sel.select("option").first().text());
            }

            Element checkgroup = getElement(form,"div#activar");
            if(checkgroup != null){
                    checkgroup.attr("aria-label",checkgroup.select(">p").text());
                    Elements checks = getElements(checkgroup,"input[type=checkbox]");
                    for(Element check : checks){
                            check.attr("aria-label",check.attr("name"));
                            check.attr("role","checkbox");
                    }
            }

            Element radiogroup = getElement(form,"div.form-section:eq(2)");
            radiogroup.attr("role","radiogroup");
            radiogroup.attr("aria-label","Sexo:");
            Elements radios =  radiogroup.select("input[type=radio]");
            for(Element radio : radios){
                    radio.attr("aria-label",radio.nextElementSibling().text());
```

```
            }
  }
}

private void setInclusiteToContent(Elements elems,String roleArea,String titulo,String queryTitle,String
offsetx, String width,String offsety, String height,String ariaOwns,boolean isDefault){
  if(elems.size()>0){
            for(int i=0;i<elems.size();i++){
                        Element elem = elems.get(i);
                        String label = titulo;
                        if(!queryTitle.equals("")){
                                    label = titulo + " " + body.select(queryTitle).get(i).text();
                        }
                        elem.attr("aria-label",label);
                        elem.attr("role",roleArea);
                        if(isDefault)elem.attr("inclu-default","true");
                        setOffset(elem,offsetx,width,offsety,height);
            }
  }
}


private void setOffset(Element elem,String offsetx, String width,String offsety, String height){
  if(offsetx.length() >0)elem.attr("inclu-offsetx",offsetx);
  if(offsety.length() >0)elem.attr("inclu-offsety",offsety);
  if(width.length() >0)elem.attr("inclu-width",width);
  if(height.length() >0)elem.attr("inclu-height",height);
}

private void setToElements(String role, String navName, Element elementToFind,
HashMap<String,String> attsMap) {
  Elements elements = getElements(elementToFind, navName);
  for(Element element : elements)
  {
            setToElement(role, element, attsMap);
  }
}

private Element setToElement(String role, String navName, Element ElementToFind,
HashMap<String,String> attsMap) {
  Element element = getElement(ElementToFind, navName);
  return setToElement(role, element, attsMap);
}

private Element setToElement(String role, Element element, HashMap<String,String> attsMap) {
  if (element != null && role != null && !role.isEmpty()){
            element.attr("role", role);
            if(attsMap != null) {
                        for(String key : attsMap.keySet())
                        {
                                    String value = attsMap.get(key);
                                    if (value != null && !value.isEmpty()) element.attr(key, value);
                        }
            }
  }
  return element;
}
```

```
private Element setToElement(String role, String navName, HashMap<String,String> attsMap) {
  Element element = getElement(null, navName);
  return setToElement(role, element, attsMap);
}

private Element getElement(Element ElementToFind, String navName) {
  Elements elems = getElements(ElementToFind, navName);
  if (elems == null || elems.size() < 0) { return null; }
  return elems.first();
}

private Elements getElements(Element ElementToFind, String navName) {
  if (navName == null || navName.isEmpty()) { return null; }
  if (ElementToFind != null) {
            return ElementToFind.select(navName);
  }
  return document.select(navName);
}
}
```

ANEXO 7. Regla WebexModelService

```
import org.inclusite.htmlparser.HttpMessage;

public class WebexModelService{

 final String CHECKGROUP = "div#activar:not([style*=none])";

 public void main(HttpMessage httpMessage){

   if (httpMessage.isUrl("^http\:\/\/webexperimental\.csd\.com\.es\/webexperimental\d\/es\/.*")) {
            String JSON = "[{"+
                     "\"query\": \"" + CHECKGROUP  + "\", "+
                     "\"atts\": {"+
                              "\"role\": \"checkboxgroup\""+
                     "}"+
            "}]";

            httpMessage.setBody(JSON);
   }
 }
}
```