

Herramienta para el análisis automático de la accesibilidad web



Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:

Alejandro Mayol Carrión

Tutor/es:

Sergio Luján Mora

Septiembre 2016



Universitat d'Alacant
Universidad de Alicante

Preámbulo

Ya desde pequeño, habiendo nacido en la época digital, he vivido en continuo contacto con la Web, y en ella he crecido, me he formado y me he entretenido. Parecía imposible para un niño formar parte de algo tan grande, pero con Ingeniería Multimedia y mucho trabajo año tras año, he aprendido a hacerme un hueco en este mundo.

Entonces llegó la hora de mi Trabajo de Final de Grado, y con él, la capacidad de hacer algo que siempre había querido: crear una web propia. Además, durante la carrera he adquirido conocimientos de muchos ámbitos que podré aplicar a este trabajo. Con la propuesta de Sergio Luján de crear un proyecto basado en la Web empezó este proyecto.

Durante este Trabajo de Fin de Grado se han aplicado los conocimientos aprendidos durante toda la carrera, y más concretamente en las asignaturas Usabilidad y Accesibilidad, Análisis y Especificación de Sistemas Multimedia, Programación Hipermedia I, Programación Hipermedia II y Diseño de Sistemas Multimedia.

Este proyecto tiene como objetivo el análisis del estado actual del análisis de la accesibilidad web a través de validadores automáticos, la creación de un prototipo funcional y, en general, la adquisición de conocimientos sobre la accesibilidad web y cómo aplicarla.

Agradecimientos

A la Universidad de Alicante y al Grado en Ingeniería Multimedia, que me han dado la oportunidad de formarme en lo que me apasiona a lo largo de cuatro años de retos, proyectos y superación personal.

A mi profesor y tutor, Sergio Luján Mora, por proponer un proyecto tan interesante, útil y de actualidad, y por su ayuda a lo largo del desarrollo del mismo.

A todos mis compañeros de clase, por ser un apoyo inestimable a lo largo de estos años de formación y amigos que dejan marca.

A todos los trabajadores del *World Wide Web Consortium* (W3C) por el inestimable trabajo que realizan para que todos los desarrolladores web tengamos más facilidades a la hora de trabajar y por las pautas de accesibilidad web que elaboran.

Sin todos ellos este trabajo no podría haber sido posible.

The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.

Tim Berners-Lee

Fundador y director del *World Wide Web Consortium*

Índice general

1. Introducción.....	1
1.1. Contexto	1
1.1.2. La accesibilidad web	2
1.1.3. Medir la accesibilidad web.....	3
1.1.4. Analizadores de la accesibilidad web.....	5
1.2. Enfoque principal	6
1.3. Defensa del problema.....	6
2. Marco teórico.....	8
2.1. Alcance.....	8
2.2. Estado de la cuestión.....	8
3. Objetivos	10
4. Metodología	11
4.1 Planificación.....	11
4.2 Análisis y especificación de requisitos.....	13
5. Antecedentes	17
5.1. AccessLint.....	17
5.2. AChecker.....	18
5.3. Examiner.....	21
5.4. TAW.....	22
5.5. Tenon.io	24
5.6. WAVE.....	27
5.7. Conclusiones	28
6. Desarrollo.....	30
6.1. Aprendizaje de la accesibilidad web	30
6.2. El análisis automático.....	30
6.3. Implementación de las técnicas en PhantomJS	33

6.3.1. Proceso de implementación.....	34
6.3.2. Resultado de la implementación.....	36
6.4. Creación del sitio web AccesTitan.....	37
6.4.1. Tecnologías	37
6.4.2. Diseño	38
6.4.3. Página principal.....	38
6.4.4. Página de resultados	39
6.4.5. Otras páginas	40
6.4.6. Flujo de ejecución	40
7. Resultados	42
7.1. Lista de técnicas analizables	42
7.2. Implementación de las técnicas.....	42
7.3. Sitio web AccesTitan	42
7.4. Ejecución del prototipo	47
8. Conclusiones	49
8.1. Mejoras y ampliaciones.....	49
8.2. Modelo de negocio	50
9. Bibliografía	51
A. Anexos	52
A.1. Aprendiendo sobre accesibilidad web.....	52
A.2. El formato JSON	53
A.3. Tipos de discapacidades	54
A.3.1. Discapacidad visual.....	54
A.3.2. Discapacidad auditiva.....	54
A.3.3. Discapacidad física o motriz	55
A.3.4. Discapacidad del habla.....	56
A.3.5. Discapacidad cognitiva.....	56
A.3.6. Otros tipos de discapacidades.....	57
A.4. Principios de la accesibilidad web.....	57

A.5. Legislación española sobre accesibilidad web	58
A.6. El futuro de la accesibilidad web.....	59
A.7. Listado de técnicas WCAG 2.0	60
A.7.1. Conjunto de técnicas evaluadas	61
A.7.2. Tabla de técnicas implementadas en AccesTitan	103

Índice de figuras

Figura 1: Resultado de un análisis con AChecker	5
Figura 2: Diagrama de desarrollo del TFG	13
Figura 3: Página principal de AccessLint	17
Figura 4: Lista de resultados de AccessLint.....	18
Figura 5: Opciones del análisis con AChecker	19
Figura 6: Resultado del análisis con AChecker	19
Figura 7: Resultado en formato PDF con AChecker.....	20
Figura 8: Directriz personalizada con AChecker	20
Figura 9: Página principal de Examiantor.....	21
Figura 10: Página de resultados de Examiner.....	22
Figura 11: Interfaz de análisis de TAW	23
Figura 12: Lista de resultados de TAW	23
Figura 13: Lista de resultados en detalle de TAW	24
Figura 14: Página principal de Tenon	25
Figura 15: Precios de las suscripciones a Tenon.....	25
Figura 16: Página de resultados de Tenon (1).....	26
Figura 17: Página de resultados de Tenon (2).....	26
Figura 18: Página principal de WAVE	27
Figura 19: Nomenclatura de símbolos y pruebas en WAVE	27
Figura 20: Página de resultados con WAVE.....	28
Figura 21: Ejemplo de técnica descrita en las WCAG 2.0.....	32
Figura 22: Ejemplo de prueba sencilla en lenguaje JavaScript.....	34
Figura 23: Ejemplo de prueba sencilla ejecutada en PhantomJS	35
Figura 24: Test y resultados esperados proporcionados por las WCAG 2.0.....	35
Figura 25: Ejemplo de función JavaScript que utiliza PhantomJS	36
Figura 26: Flujo del análisis	41
Figura 27: Página principal de AccesTitan	43
Figura 28: Página de información sobre AccesTitan	43
Figura 29: Resultado de un análisis con AccesTitan (1).....	44
Figura 30: Resultado de un análisis con AccesTitan (2).....	44
Figura 31: Resultado de un análisis con AccesTitan (3).....	45
Figura 32: Página incorrecta y mensaje de error.....	45
Figura 33: Diseño adaptativo de la web (1)	46
Figura 34: Diseño adaptativo de la web (2)	46

Figura 35: Paned de control de XAMPP	48
Figura 36: Diploma del curso.....	52
Figura 37: Ejemplo de formato JSON.....	53
Figura 38: Teclado adaptado para personas con discapacidad motriz	55
Figura 39: Programa WWAAC Web Broeser.....	56

Índice de tablas

Tabla 1: Requisito RE01	14
Tabla 2: Requisito RE02	14
Tabla 3: Requisito RE03	15
Tabla 4: Requisito RE04	15
Tabla 5: Requisito RE05	15
Tabla 6: Requisito RE06	15
Tabla 7: Requisito RE07	16
Tabla 8: Requisito RE08	16

1. Introducción

1.1. Contexto

Cuando en 1989 surgió el proyecto de la *World Wide Web* de las manos de Tim Berners-Lee mientras trabajaba en el CERN, nadie se esperaba la repercusión y el alcance que llegaría a tener en la vida de todas las personas del mundo. Fue el principio de una gran revolución digital que llega hasta nuestros días y no tiene intención de detenerse.

La Web ha mejorado nuestra calidad de vida en muchos aspectos, permitiéndonos realizar todo tipo de acciones a través de la red, como transacciones bancarias, compras o simplemente como forma de ocio. A diario utilizamos la red para comunicarnos con otras personas o mantenernos informados. Son acciones que ya todos hemos interiorizado como normales, habituales e, incluso, necesarias.

Todas estas ventajas y placeres, por desgracia, no están al alcance de todos los usuarios: hay un porcentaje de personas que, debido a algún tipo de discapacidad o enfermedad, tienen dificultades para navegar por la red sin barreras y de manera cómoda.

Para ellos, realizar acciones que se podrían considerar básicas y sencillas puede resultar una tarea complicada e incluso imposible.

Esto es debido a un conjunto de factores:

- La persona sufre algún tipo de discapacidad, motora o psicológica.
- No se disponen de máquinas, accesorios o software necesario para ayudar o facilitar la navegación por la web.
- La página web visitada no es accesible por una serie de motivos y dificulta la navegación e interacción en ella debido a un mal diseño e implementación.
- Se está accediendo a una página web desde un dispositivo antiguo o bajo condiciones ambientales adversas.

Por ello, desde hace años, existen, para cada uno de los puntos anteriores, una serie de soluciones que se pueden aplicar. Son las siguientes:

- Asociaciones de afectados donde se ayuda al afectado a aprender a navegar por la web.
- Empresas que desarrollan software, hardware y accesorios que ayudan enormemente a los afectados.

- Normas y directrices para crear páginas webs accesibles y que se complementan con los elementos del punto anterior para lograr un acceso total a todas las opciones que ofrece la web, sin impedimentos físicos o psicológicos.
- Nuevas formas de desarrollo web que permiten la compatibilidad con las versiones más antiguas.

El tema de la accesibilidad web siempre será actual y necesario, pues continuamente se elaboran páginas webs y no todas siguen las directrices que aseguran que sean accesibles para todos los grupos de usuarios posibles.

Este Trabajo de Fin de Grado tiene como finalidad elaborar un sitio web que permita al usuario realizar un análisis automático de la accesibilidad cualquier otra web y ver los fallos de esta.

1.1.2. La accesibilidad web

Durante el punto anterior se ha mencionado repetidamente el término *accesibilidad web*, pero ¿qué significa exactamente?

Según el organismo *World Wide Web Consortium (W3C)*, se define de la siguiente manera:

La accesibilidad web significa que personas con algún tipo de discapacidad van a poder hacer uso de la Web. En concreto, al hablar de accesibilidad web se está haciendo referencia a un diseño web que va a permitir que estas personas puedan percibir, entender, navegar e interactuar con la Web, aportando a su vez contenidos. La accesibilidad web también beneficia a otras personas, incluyendo personas de edad avanzada que han visto mermadas sus habilidades a consecuencia de la edad.¹

Así pues, la accesibilidad web es una característica de cualquier página web. Podemos decir que una página es más o menos accesible en función del número de colectivos a los que facilite la navegación debido a un buen diseño o arquitectura. Es por ello que existen unas pautas que nos indican cómo comprobar si una página es accesible o no, y se dividen en distintos principios que engloban la totalidad de dificultades posibles que una persona puede encontrarse al navegar por un sitio web.

Conseguir que un sitio web sea accesible no solo beneficia a las personas con alguna discapacidad que vayan a poder usarla: beneficia a todos los usuarios potenciales e incluso al propietario de

¹ Referencia: <https://www.w3.org/standards/webdesign/accessibility>

dicho sitio web, pues una página accesible implica un correcto diseño e implementación de la misma. Que una página web accesible solo beneficia a las personas con discapacidad es un mito.²

Cuando una página web es accesible, significa que su uso y visualización se adapta a todos los dispositivos posibles y a situaciones adversas del entorno del usuario, como una mala iluminación o una conexión a internet lenta.

Además, los motores de búsqueda actuales se basan en ciertas normas que premian una estructuración correcta de las partes y contenidos de una web. Esto se puede lograr siguiendo las pautas y directrices de accesibilidad actuales. Por tanto, una web accesible también tendrá un mejor posicionamiento en los motores de búsqueda, es decir, un mejor *Search Engine Optimization (SEO)*.

Asimismo, otro motivo por el que lograr que un sitio web sea accesible es que, en la actualidad, en muchos países es obligatorio. En el caso concreto de España, en la legislación se establece que todas las administraciones públicas y ciertas empresas, en función de su tamaño y sector en el que trabajen, deben ofrecer sitios webs accesibles. En el apartado [A.5. Legislación española sobre accesibilidad web](#) se puede encontrar más información.

En resumen, la accesibilidad de una web supone una serie de ventajas que suplen los siguientes problemas:

- Problemas de oído, visión, movilidad.
- Dificultades de lectura o comprensión cognitiva.
- Imposibilidad de utilización del teclado o el ratón
- Lector de sólo texto, pantalla pequeña, conexión lenta o mala iluminación.

La accesibilidad mejora el acceso a la Web en general, no es de interés únicamente para personas con discapacidad. En el apartado [A.3. Tipos de discapacidades](#) del Anexo se explica con más profundidad las discapacidades existentes, cómo afectan a la navegación en un sitio web y cómo las se pueden suplir.

1.1.3. Medir la accesibilidad web

En el año 1999 surgió la primera recomendación oficial del W3C sobre accesibilidad web en forma de **directrices** sobre buenas prácticas de desarrollo web. Son las *Web Content Accessibility Guidelines (WCAG 1.0)*³.

² Referencia: <http://accesibilidadweb.dlsi.ua.es/?menu=mitos>

³ WCAG 1.0: <https://www.w3.org/TR/WCAG10/>

En este primer documento sobre cómo lograr accesibilidad en un sitio web, se recogen catorce directrices, cada una de las cuales describe un principio básico del diseño accesible de una web. Dentro de cada directriz se estipulan una serie de puntos a cumplir para lograr distintos grados de accesibilidad dentro de una página.

A partir del año 2008, estas directrices se actualizaron. A este nuevo documento actualizado, y que es el referente actual en cuanto a normas sobre la accesibilidad web, se le llamó **Web Content Accessibility Guidelines 2.0 (WCAG 2.0)**⁴. Se basa en cuatro principios:

- **Perceptible**: el contenido debe poder ser accedido a través de alguno de los sentidos humanos.
- **Operable**: se debe poder acceder al contenido mediante algún dispositivo, como un teclado
- **Comprensible**: cualquier persona debe ser capaz de entender el contenido de la página.
- **Robusto**: el contenido debe poder ser reproducido tanto con tecnologías actuales como con algunas más antiguas.

Las reglas y requisitos expuestos en él son los que se implementan en las páginas web que analizan automáticamente la accesibilidad web de cualquier sitio, y sobre las que se trabajará en este proyecto. Se puede encontrar más información sobre estos principios en el apartado [A.4. Principios de la accesibilidad web](#) de este documento.

Así pues, cada uno de los puntos a analizar tiene un nivel de *prioridad* que indica su impacto en la accesibilidad:

- **Nivel A o Simple A**: cualquier página que pretenda ser accesible para la mayoría de usuarios deberá cumplir todos los puntos cuyo nivel sea A. Por lo tanto, los desarrolladores web deben cumplir con todos estos requerimientos para poder considerar su página como accesible. Si bien un nivel A no garantiza que el sitio sea accesible para todos, es un paso para lograr la accesibilidad total de la web.
- **Nivel AA o Doble A**: son requerimientos más exigentes que los de nivel A, y los desarrolladores web deberían cumplirlos para que un mayor número de colectivos sea capaz de utilizar el sitio web.
- **Nivel AAA o Triple A**: son los requerimientos más exigentes, y los desarrolladores web pueden cumplirlos para conseguir que su sitio sea accesible para cualquier usuario o colectivo, sin importar el problema, pues el sitio web creado estará preparado para ello.

⁴ WCAG 2.0: <https://www.w3.org/TR/WCAG20/>

Cuando un sitio web cumple estos puntos y logra uno de estos niveles de accesibilidad web, junto con otros requisitos necesarios, se dice que el sitio es conforme con las WCAG 2.0 en ese nivel.

1.1.4. Analizadores de la accesibilidad web

Hemos visto la importancia de que un sitio web sea accesible y las ventajas que ello reporta, así como las normas y pautas existentes para hacer que un sitio web se adecúe al estándar y sea accesible para el mayor número de colectivos posibles.

Conseguir que un sitio web sea accesible se puede conseguir durante la **planificación** del proyecto, durante el **desarrollo** del mismo o bien una vez terminada la web.

Por razones económicas, lo más recomendable es trabajar en la accesibilidad durante el desarrollo del proyecto. Para ello, sería recomendable contar en el equipo de desarrollo con un **experto** en accesibilidad web que sea capaz de guiar a los desarrolladores y encontrar fallos, así como prevenir su aparición, y explicarles maneras más adecuadas de hacer la página. Por desgracia, esto no es lo habitual en un desarrollo web.

Por ello, existen multitud de **herramientas en línea**, gratuitas y de pago, que permiten encontrar fallos en la accesibilidad de una web mediante la aplicación de pruebas basadas en las directrices de las WCAG 2.0 de forma automatizada y que presentan al usuario un resultado final orientativo.

Estas páginas suelen presentar dos partes bien diferenciadas: una página principal donde se introduce la dirección de otra web a analizar y una página con los resultados del análisis, como se aprecia en la siguiente figura.

The screenshot shows the AChecker Accessibility Review interface. At the top, there's a header bar with the title 'Accessibility Review' and a sub-header 'Accessibility Review (Guidelines: WCAG 2.0 (Level AA))'. Below the header, there are tabs for 'Known Problems (2)', 'Likely Problems (1)', 'Potential Problems (398)', 'HTML Validation', and 'CSS Validation'. The main content area displays a single error message: '1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.' This message is preceded by 'Success Criteria 1.4.4 Resize text (AA)'. Below the message, it says 'Check 117: *(italic) element used.*' and provides a 'Repair' instruction: 'Replace your *i elements with em or strong.'*

Figura 1: Resultado de un análisis con AChecker

Fuente: <http://achecker.ca>

Estas herramientas suponen una gran ayuda para los **desarrolladores web**, pues en algunos casos permiten escoger el nivel de profundidad del análisis (niveles A, AA o AAA), y dan la opción de evaluar una página entera o incluso un fragmento de código HTML o un archivo de texto. Además, en la sección de resultados se suele indicar el fragmento de código donde se localiza un error en concreto y se remite a la documentación oficial del W3C donde se explica cómo subsanarlo.

Estas herramientas de revisión se pueden presentar tanto como software descargable e instalable en el ordenador como servicio alojado en un sitio web desde donde realizar el análisis en línea en su totalidad.

1.2. Enfoque principal

Una vez explicados los conceptos anteriores, ha quedado patente la importancia y necesidad de que cualquier sitio web sea lo más accesible posible.

El **contexto** del proyecto es la **accesibilidad web** y el objetivo general del proyecto es, pues, crear una herramienta en línea que ayude a analizar el grado de accesibilidad de cualquier página web introducida y ofrezca al usuario un informe con los resultados.

Con una **herramienta** que evalúa automáticamente la accesibilidad de una página web, es más sencillo encontrar y resolver problemas relacionados con esta, para así facilitar su uso por parte de cualquier colectivo.

El **enfoque principal** del problema es aplicar las directrices de accesibilidad actuales publicadas por el W3C para crear una aplicación que ayude a encontrar y corregir problemas de accesibilidad en páginas web.

1.3. Defensa del problema

La necesidad de que existan plataformas que ayuden a detectar problemas de accesibilidad ha quedado patente en los puntos anteriores.

Por una parte, una página que logra ser accesible puede ser consultada por cualquier persona que tenga algún tipo de **discapacidad**, física o intelectual.

Por otra parte, el diseño de una web accesible facilita su uso para personas con problemas de diversa índole, como **problemas** derivados del **entorno** (baja iluminación, ambiente ruidoso, etc.) o usuarios **inexpertos** en el uso de la web o de dispositivos electrónicos.

Es por esto que es necesario hacer las páginas web lo más accesibles posible, porque todos nos **beneficiamos** de ello, aunque en un primer momento no lo parezca.

Se ha visto que es imposible evaluar la accesibilidad web de manera totalmente automática, y que siempre es necesaria la revisión de una persona **experta** en el ámbito. Sin embargo, cuantas más pruebas automáticas supere la página, más sencillo será acotar sus **problemas** de accesibilidad. Por ello, es beneficioso la existencia de diversas plataformas de análisis.

Asimismo, desarrollando un proyecto de este calibre se consigue una visión más cercana a los problemas de muchas personas con alguna discapacidad que usan la web, así como conocimientos relativos al análisis de accesibilidad web, por no mencionar la mejora en las habilidades de desarrollo web.

2. Marco teórico

2.1. Alcance

Tras haber definido el contexto de este documento y haber expuesto la importancia de la accesibilidad web a día de hoy, a continuación se va a especificar el alcance de este proyecto.

El trabajo consta de tres partes diferenciadas:

- Estudio de las **técnicas WCAG 2.0** y selección de las que sea posible analizar **automáticamente** (sin la supervisión de un experto en accesibilidad web).
- Creación de **un sitio web** cliente donde un usuario pueda introducir la dirección URL de otra web, se analice y se devuelva un resultado.
- **Implementación** en lenguaje JavaScript de las **técnicas WCAG 2.0** previamente escogidas para su análisis mediante un navegador web sin interfaz, PhantomJS.

La intención es elaborar un prototipo funcional que permita al usuario introducir la dirección URL de una página web, analice la página y devuelva un resultado de forma visual, con una nota global orientativa que indicará el grado de accesibilidad de la web analizada y listará cuáles son los errores encontrados para que el usuario que ha realizado el análisis pueda verlos, entenderlos y corregirlos.

2.2. Estado de la cuestión

A día de hoy hay bastantes productos parecidos a este disponibles en Internet. Muchos de ellos llevan años ofreciendo este servicio de análisis, de forma gratuita en la mayoría de casos.

Un ejemplo es **Examinator**, una sencilla web que lleva diez años en continua mejora, adaptando las nuevas normas de accesibilidad publicadas por el W3C. Otro ejemplo es el de **TAW**, que lleva quince años ofreciendo el mismo servicio de análisis y permite la integración de su servicio en un *Content Management System* o **CMS** como Wordpress, Joomla! o Drupal, lo que permite su uso de forma sencilla a la gran cantidad de usuarios de estos sistemas y les ayuda a crear, mediante estos gestores, páginas más accesibles.

También hay ejemplos de **modelos de negocio** basados en esta actividad, como es el caso de **Tenon**, que cuenta con un sistema de suscripción de pago para poder realizar más análisis mensuales utilizando su *Application Programming Interface (API)*.

Hay muchos más casos, como AccessLint, AChecker o WAVE, que serán analizados en el apartado [**5. Antecedentes**](#) de este documento, y constituyen un claro ejemplo del estado actual de los analizadores automáticos de la accesibilidad web y de la importancia que se la da actualmente a la misma.

El factor común de todas estas herramientas es que ofrecen al usuario la posibilidad de **analizar la accesibilidad** de una página web introducida en un campo de texto y devuelven el **resultado** del análisis de forma comprensible y ordenada.

El **objetivo** de este **proyecto** no es competir con estas aplicaciones, sino desarrollar una propia, implementar técnicas de las WCAG 2.0, aprender sobre el **análisis de la accesibilidad web** y la programación web en el proceso y aplicar estos conocimientos en el proyecto y en mi futuro profesional.

3. Objetivos

Gracias a la propuesta del tutor de enfrentarme al **desarrollo** de un **producto** real sobre accesibilidad web, durante este tiempo he aprendido sobre nuevas técnicas de accesibilidad, tecnologías novedosas para el desarrollo web y metodologías de trabajo. Por ello, este Trabajo de Fin de Grado me ha permitido poner en práctica y ampliar los **conocimientos** adquiridos durante el grado en Ingeniería Multimedia.

Con **AccesTitan**, que es el nombre del sitio web del proyecto, se pretende desarrollar un sitio web que permita, mediante la introducción de una URL de otra web en un campo de texto, analizar el grado de accesibilidad de la misma, y ofrecer al usuario un resultado del análisis de manera comprensible y útil.

Los **objetivos principales** del proyecto son los siguientes:

- Elaborar una lista con las técnicas de accesibilidad descritas en la especificación actual del W3C (las WCAG 2.0) para determinar las que es posible implementar mediante el uso del software PhantomJS y permiten analizar una página web de manera automática, sin la intervención de una persona.
- Definir los requisitos del sitio web AccesTitan que se va a desarrollar.
- Implementar las funciones escogidas previamente en el lenguaje JavaScript y que se emplearán para analizar la accesibilidad de una pagina web.
- Diseñar e implementar una arquitectura del sistema capaz de:
 - Proporcionar al usuario una navegación accesible mediante el diseño adecuado de la página, siguiendo las pautas de accesibilidad generales.
 - Permitir la introducción de una URI a analizar.
 - Analizar la página enviada al servidor mediante las funciones implementadas.
 - Devolver un resultado del análisis y mostrarlo en la web de manera clara y concisa.

4. Metodología

Dado que este Trabajo de Fin de Grado tiene como finalidad aprender más sobre la accesibilidad en la Web y sobre la programación web en general, se ha empleado una **metodología incremental**, donde antes de empezar con una fase era necesario acabar con la anterior y se hacen prototipos funcionales rápidamente, se prueban y se siguen mejorando. Es un proceso similar al empleado en la metodología SCRUM⁵. Sin embargo, al no haber un equipo de trabajo, es decir, al ser un proyecto individual, no se ha podido aplicar ninguna metodología aprendida durante el grado, pues todas están enfocadas a grupos de trabajo formados por varios miembros.

Tras la finalización de la primera fase de **documentación**, se tuvieron los conocimientos necesarios para especificar los **requisitos** funcionales que debería incorporar el proyecto, y una vez definidos, se pudo pasar al **desarrollo** del sitio web y la **implementación** de las técnicas descritas en las WCAG 2.0.

4.1 Planificación

El trabajo se ha estructurado en las siguientes fases:

Fase 1: Documentación.

Duración: 3 meses.

Objetivos:

- Realizar el curso [Aprende Accesibilidad Web paso a paso](#)⁶ para afianzar los conocimientos sobre el tema.
- Probar páginas de análisis automáticos similares para entender su funcionamiento.
- Documentarme con artículos científicos y de actualidad sobre accesibilidad web.
- Leer la documentación sobre el estado actual de la [especificación WCAG 2.0](#).⁷

⁵ Información sobre SCRUM: <http://scrummethodology.com/>

⁶ Curso: <https://www.udemy.com/aprende-accesibilidad-web-paso-a-paso/>

⁷ Especificación oficial: <https://www.w3.org/WAI/WCAG20/quickref/>

Fase 2: Especificación.

Duración: 1 mes.

Objetivos:

- Definir los objetivos del proyecto
- Realizar una tabla con las técnicas de accesibilidad que es posible analizar, las que requieren revisión manual y las que no se pueden analizar automáticamente.
- Definir el diseño visual del sitio web, así como sus secciones y distribución.

Fase 3: Desarrollo.

Duración: 2 meses.

Objetivos:

- Crear un sitio web completo, sin utilizar herramientas o plantillas externas, que permita al usuario realizar un análisis automatizado de la accesibilidad de la web introducida, y recibiendo como resultado una nota orientativa y una lista con las pruebas superadas, fallidas y aquellas que se deben revisar manualmente.
- Crear una librería de funciones JavaScript que se puedan ejecutar con el navegador sin interfaz PhantomJS y que nos devuelvan un resultado en formato JSON. Incluyen comprobaciones sobre técnicas relacionadas con el lenguaje HTML, sobre el diseño de la página y sobre errores comunes.

Fase 4: Conclusión.

Duración: 1 mes.

Objetivos:

- Depuración del sitio web, que incluye mejora en el diseño y en el código fuente.
- Elaboración de la memoria del proyecto, que recoge y explica todas las fases del proyecto, desde su planteamiento hasta su consecución total, así como toda la información relacionada que se ha recopilado.
- Alojar el proyecto web en la página GitHub para que el código fuente sea más fácilmente accesible por la comunidad.

En la Figura 2 se muestra un gráfico sobre el progreso del desarrollo a lo largo del tiempo.

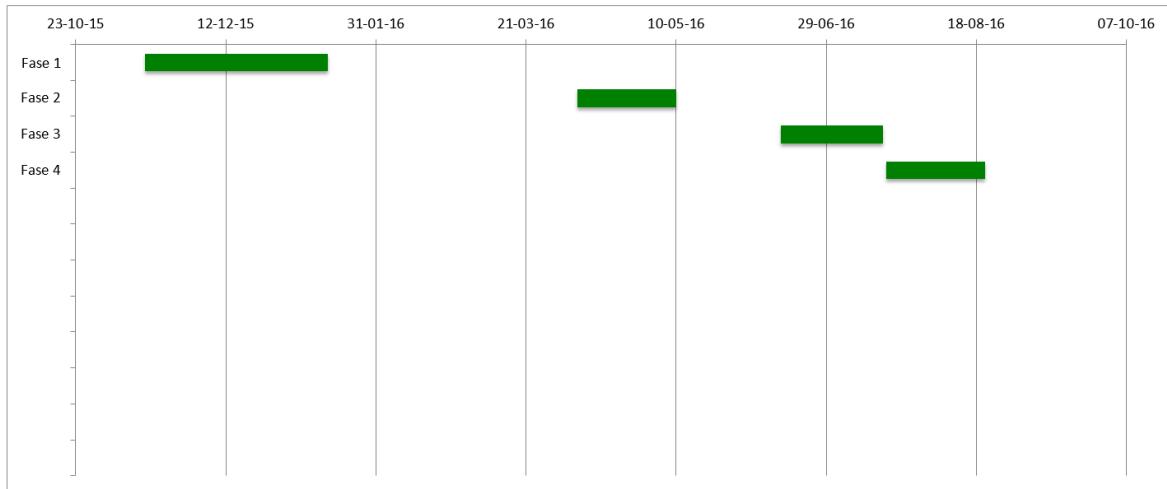


Figura 2: Diagrama de desarrollo del TFG

4.2 Análisis y especificación de requisitos

Antes de empezar con el diseño de cualquier proyecto, es necesario definir sus **funcionalidades**, establecer los **servicios** que va a brindar y sus restricciones. En este apartado se va a hablar de los requisitos funcionales.

Se detallan las funcionalidades que debe tener el prototipo a desarrollar:

Sitio web

- Debe cumplir con el nivel A de las WCAG 2.0 para ser accesible.
- Cualquier usuario debe poder utilizarla sin problemas.
- Debe tener un campo de texto donde introducir una dirección web (una URI, por ejemplo, <http://www.ua.es/>).
- Debe proporcionar de forma sencilla el resultado del análisis, mostrando una nota final orientativa y una lista con las pruebas de accesibilidad superadas, los fallos encontrados y los elementos que deben revisarse de manera manual porque un el análisis automático no ha sido concluyente.

Servidor

- Se debe instalar en un servidor el software PhantomJS para trabajar con él, debido a las facilidades que ofrece para cargar y trabajar con el contenido de una página web, así como

por su documentación, que es extensa, detallada y facilita comenzar a trabajar con él.

Además, es el navegador sin interfaz más utilizado actualmente, por lo que encontrar ayuda o recursos es más fácil.

- Cuando la página web envíe una petición de análisis, el servidor debe ejecutar una serie de funciones JavaScript creadas que comprobarán cada una de las reglas de las WCAG 2.0 que se han considerado analizables en el estudio previo.
- Se debe devolver un fichero JSON con el resultado del análisis para que el sitio web pueda crear una página de resultado utilizando el lenguaje PHP.

Se procede a usar tablas para describir cada **requisito funcional** del proyecto, junto con los siguientes atributos:

- **Identificador:** se usa para identificar cada requisito funcional de forma única. Se van a usar las siglas RE y un número único.
- **Nombre:** título del requisito.
- **Importancia:** puede ser baja, media o alta y designa la prioridad de implementación dentro del proyecto.
- **Descripción:** descripción detallada del requisito.

Identificación	RE01
Nombre	Sitio web completo
Importancia	Alta
Descripción	Se debe crear un sitio web completo y robusto.

Tabla 1: Requisito RE01

Identificación	RE02
Nombre	Formulario de envío
Importancia	Alta
Descripción	En la página debe hacer un formulario de texto en el que el usuario pueda introducir la URL de una web y pulsar un botón que comenzará el análisis.

Tabla 2: Requisito RE02

Identificación	RE03
Nombre	Página de información
Importancia	Baja
Descripción	El sitio web ha de tener un apartado donde se informe sobre él, su uso y finalidad.

Tabla 3: Requisito RE03

Identificación	RE04
Nombre	Página accesible
Importancia	Alta
Descripción	La totalidad del sitio web ha de ser accesible y cumplir con las directrices actuales de accesibilidad.

Tabla 4: Requisito RE04

Identificación	RE05
Nombre	Nota del análisis
Importancia	Alta
Descripción	Al realizarse el análisis, se debe mostrar una nota orientativa, que podrá ir desde cero a diez, e indicará el grado de accesibilidad de la página analizada. Para decidir la nota se tendrán en cuenta las pruebas superadas y fallidas.

Tabla 5: Requisito RE05

Identificación	RE06
Nombre	Lista de pruebas
Importancia	Alta
Descripción	Tras realizar el análisis, se debe mostrar un listado con todas las pruebas aplicadas durante el mismo, y se debe dividir en cuatro apartados: pruebas superadas, pruebas fallidas, pruebas que requieren de comprobación manual y pruebas no aplicables.

Tabla 6: Requisito RE06

Identificación	RE07
Nombre	Utilizar PhantomJS
Importancia	Alta
Descripción	Se debe utilizar el navegador sin interfaz PhantomJS para ejecutar las funciones que determinen la accesibilidad de la web, ya que este programa permite acceder al <i>Document Object Model</i> (DOM) de la página web.

Tabla 7: Requisito RE07

Identificación	RE08
Nombre	Pruebas WCAG 2.0
Importancia	Alta
Descripción	Las comprobaciones que se apliquen a la página analizada deben ser las escogidas entre las especificadas en el listado de las WCAG 2.0.

Tabla 8: Requisito RE08

5. Antecedentes

Para hablar sobre análisis automático de la accesibilidad web es necesario conocer los ejemplos actuales de referencia y ponernos en contexto.

En la actualidad existen diversas **webs especializadas** en el análisis de la accesibilidad web. Dado que el primer documento del W3C que trataba este tema de manera oficial se remonta al 1999, hace más de una década que empezaron a surgir páginas web que facilitan el análisis a los expertos en accesibilidad.

Sin embargo, no olvidemos que, en ningún caso, una web de este tipo puede reemplazar por completo el trabajo de una persona especializada en accesibilidad. Por lo tanto, el resultado de un análisis automático siempre deberá ser respaldado y **verificado** por un **experto** y no debe tomarse como definitivo en ningún caso, pues llevaría a errores.

A continuación, y para dar una visión más amplia sobre el trabajo que realiza un sitio web de este tipo, se va a analizar la forma de **funcionamiento** de una serie de páginas de referencia para ver cómo se presenta al usuario la información de su uso, su interfaz y cómo ofrece los resultados del análisis.

5.1. AccessLint

Un sitio web muy sencillo, que reduce al mínimo la presentación, como se aprecia en la Figura 3, y el análisis realizado. Es la más fácil de usar junto con Examinator. Simplemente permite analizar una página mediante su URL.

AccessLint

Keep accessibility issues in check

URL to Test

Example: [W3C's demonstration of an inaccessible site](#)

Analyze

Figura 3: Página principal de AccessLint

Fuente: <https://accesslint.com/>

Presenta los resultados de forma muy escueta, ilustrados en la Figura 4, haciendo que sea recomendable solo para análisis poco profundos y rápidos.

Needs review

Text elements should have a reasonable contrast ratio. (2 issues)

Avoid positive integer values for tabIndex. (3 issues)

Looks good

Elements should use supported ARIA roles, states and properties.

Controls and media elements should have labels.

Focusable elements should be visible and unobscured.

The web page should have the content's human language indicated in the markup.

Images should have an alt attribute.

Figura 4: Lista de resultados de AccessLint

Fuente: <https://accesslint.com/>

5.2. AChecker

AChecker es otro ejemplo de plataforma gratuita que ofrece una gran libertad al usuario para escoger el tipo de análisis a realizar, ya sea de una web, un archivo HTML o un fragmento de código, así como el conjunto de directrices de accesibilidad a aplicar durante el análisis, como se puede ver en la siguiente figura.

Check Accessibility By:

Web Page URL **HTML File Upload** **Paste HTML Markup**

Address:

Options

Enable HTML Validator Enable CSS Validator Show Source

Guidelines to Check Against

BITV 1.0 (Level 2) Section 508 Stanca Act
 WCAG 1.0 (Level A) WCAG 1.0 (Level AA) WCAG 1.0 (Level AAA)
 WCAG 2.0 (Level A) WCAG 2.0 (Level AA) WCAG 2.0 (Level AAA)

Report Format

View by Guideline View by Line Number

Figura 5: Opciones del análisis con AChecker

Fuente: <https://http://achecker.ca/>

En el apartado de **resultados** nos encontramos con una forma de presentarlos similar a lo visto en Tenon y Examinator: una **lista** de los errores encontrados y problemas potenciales, así como su localización dentro del código fuente de la página, como se ilustra en la Figura 6. Otra opción interesante es la capacidad de descargar los **resultados** en varios formatos, y, además, escoger qué parte del análisis exportar. En la Figura 7 se muestra un ejemplo de análisis exportado a formato PDF.

Accessibility Review

Accessibility Review (Guidelines: WCAG 2.0 (Level AA))

Export Format: Report to Export:

Known Problems(2) **Likely Problems (1)** **Potential Problems (398)** **HTML Validation** **CSS Validation**

1.1 Text Alternatives: Provide text alternatives for any non-text content

Success Criteria **1.1.1 Non-text Content (A)**

Check 3: **Image Alt text may be too long.**

Question: Is this Alt text as short as possible?
 PASS: Alt text is as short as possible.
 FAIL: Alt text is not as short as possible.

Pass? select/unselect all

▲ **Line 185, Column 84:**

Tuesday August 30, 2016 10:32:36

Source URL: <http://www.w3c.es/>

Source Title: World Wide Web Consortium (W3C) - España

Accessibility Review (Guidelines: WCAG 2.0 (Level AA))**Report on known problems (2 found):****1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.****Success Criteria 1.4.4 Resize text (AA)****Check 117: i (italic) element used.**

Repair: Replace your i elements with em or strong.

✖ Line 208, Column 148:

<i>smart cities</i>

✖ Line 208, Column 176:

<i>smart home</i>

*Figura 7: Resultado en formato PDF con AChecker**Fuente: <https://http://achecker.ca/>*

Entre sus características destaca la opción de **registrarse** en el sitio web, lo que nos permite, entre otras cosas, algo tan interesante como crear nuestras propias directrices de accesibilidad. Si bien podemos analizar una página empleando las pautas de las WCAG 1.0 o WCAG 2.0, también podemos crear nuestro propio conjunto de pruebas que se aplicarán en el análisis, de modo que podemos comprobar solo los aspectos que queramos dentro de página web, de manera cómoda y fácil. En la Figura 8 se muestra un ejemplo de pruebas seleccionadas para probar esta opción.

Create/Edit Guideline

* indicates required fields.

Title	Reglas de AccesITan
Abbreviation	RAcc
Long Name	Conjunto de comprobaciones
Published Date (yyyy-mm-dd)	2016-08-31
URL	
Status	<input type="radio"/> Disabled <input checked="" type="radio"/> Enabled

Checks 5+

Element	Error Type	Description	Check ID
a	Known	Link sets must be grouped.	155
a	Known	Anchor element must have a <code>title</code> attribute.	190
a	Known	Visited link text colour must contrast sufficiently with its background colour.	302
a	Known	Active link text colour must provide high contrast with its background colour.	308
body	Potential	Acronyms must be marked with <code>acronym</code> element.	99

*Figura 8: Directriz personalizada con AChecker**Fuente: <https://http://achecker.ca/>*

Debido a esta característica, AChecker es una de las opciones más interesantes y potentes en el ámbito del análisis automático de la accesibilidad web.

5.3. Examinator

Esta página presenta un diseño muy sencillo, para que el usuario sepa fácilmente qué puede hacer, es decir, es muy **usable**, y las distintas partes que componen la página están claramente diferenciadas, por lo que es sencillo navegar a través de ella, pues presenta tres secciones importantes: campo de texto donde introducir la web a analizar, información sobre la página y su uso y ejemplos de otros análisis realizados. En la Figura 9 se aprecian estas partes claramente diferenciadas.

Una característica importante que no todas las webs de este tipo implementan es que, además de poder analizar una página web al completo, también permite inspeccionar un archivo o fragmento de código HTML en busca de fallos. Esto ayuda a detectar errores de accesibilidad web en una fase temprana del desarrollo.

The screenshot shows the main interface of the Examiantor web application. At the top, there is a header bar with the title "Evaluación de la accesibilidad web". Below the header, there are three tabs: "Página web", "Archivo", and "Código". A text input field labeled "Indique el URI de la página" is present, with a placeholder "http://". Below the input field is a button labeled "Aceptar". In the bottom right corner of the main area, there is a sidebar titled "Ejemplos" which lists several URLs and their scores, such as "(Sin título) (6)" and "... Colegio Secundario N° 22... (5.9)".

Figura 9: Página principal de Examiantor

Figura: http://examinator.ws/

Otra de las grandes virtudes de Examinator es la forma en la que presenta el resultado del análisis: muestra una nota sobre 10, de modo que es muy intuitivo hacerse una idea del grado de accesibilidad de la web analizada. En la siguiente figura se muestra un análisis realizado a la página del W3C.

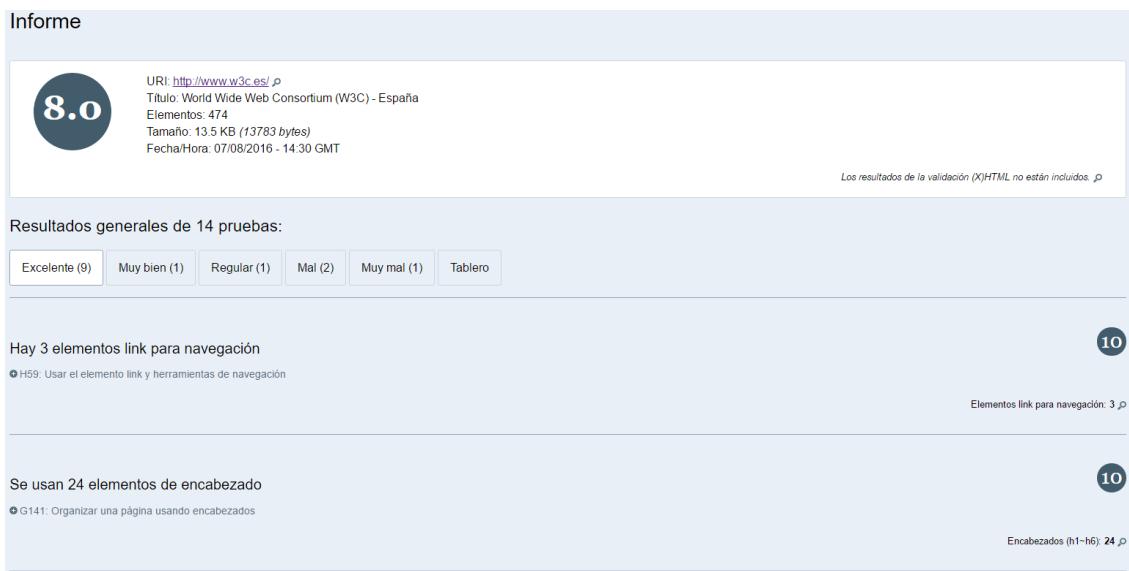


Figura 10: Página de resultados de Examinator

Figura: <http://examiner.ws/>

También se muestran diversas tablas, con la lista de técnicas analizadas y si se han superado o no. Para calcular esta nota, pondera cada test realizado con una puntuación individual y calcula aplica la media. De manera rápida y **visual**, el usuario obtiene mucha información, aunque no realiza un análisis tan profundo como otras webs de este tipo.

5.4. TAW

Esta página permite introducir en un campo de texto la URL del sitio que se desea analizar, y como característica importante destaca el hecho de que permite seleccionar el nivel de análisis a realizar (como se ha comentado anteriormente, los niveles A indican el grado de accesibilidad de una web, siendo AAA el máximo). También permite analizar las distintas **tecnologías** que presenta una página web en el lado del cliente -HTML, CSS y JavaScript-, como se puede apreciar en la Figura 11.

Seleccione la normativa sobre la que desea analizar su web, introduzca la URL de la página y el nivel de análisis que deseé validar.

contacte con nosotros'."/>

Figura 11: Interfaz de análisis de TAW

Fuente: <http://www.tawdis.net/>

Una vez que introducimos una URL y la analizamos, obtenemos los resultados en forma de **listas**. Nótese que en este resultado, mostrado en la Figura 12, hay tres columnas: **problemas** identificados, **advertencias** sobre posibles problemas cuyo análisis automático no es concluyente y comprobaciones que deben hacerse siempre de forma **manual** para asegurar que se realizan bien, pues no es posible su análisis automático.



Figura 12: Lista de resultados de TAW

Fuente: <http://www.tawdis.net/>

Tanto las advertencias como los problemas no verificados deben ser revisados por un experto en accesibilidad, pues no es suficiente con el análisis automático para sentenciar que la prueba ha fallado.

Si lo deseamos, podemos acceder a un informe mucho más detallado, ilustrado en la Figura 13, en el que se indica qué directrices del WCAG 2.0 se han analizado, así como la parte del documento HTML donde se encuentra el fallo. Es una lista muy completa y de gran utilidad.

Perceptible	Operable	Comprensible	Robusto					
Tipología				Comprobación	Técnicas	Resultado	Incidencias	Números de Líneas
<i>La información y los componentes de la interfaz de usuario deben ser presentados a los usuarios de modo que puedan percibirlos.</i>								
1.1.1 - Contenido no textual								
Imágenes Imágenes que pueden requerir descripción larga ?								
				[H45]		1	13	178, 179, 210, 219, 220, 229, 230, 235, ...
Formularios Opciones de los menús de selección agrupadas ?								
				[H85]		1	188	
1.3.2 - Secuencia con significado								
Presentación	Posicionamiento de elementos mediante flotado ? (http://www.sidar.org/css/sidar1.css)			[C22]		1	6	124, 159, 416, 423, 684, 1029
	Posicionamiento de elementos de forma absoluta ? (http://www.sidar.org/css/sidar1.css)			[C22]		4	36, 53, 86, 1074	
	Posicionamiento de elementos de forma absoluta ? (http://www.sidar.org/jquery/jquery+aria/jquery-ui-min.jquery-ui.min.css)			[C22]		1	5	
1.3.3 - Características sensoriales								
Presentación	Características sensoriales ?			[G96]		1	1	
1.4.1 - Uso del color								
Presentación	Información mediante color ?			[G14 G122 G182 G183]		1	1	
1.4.3 - Contraste (Mínimo)								
Presentación	Contraste ?			[G18 G148 G174]		1	1	
	Contraste para fuentes grandes ?			[G145 G148 G174]		1	1	
1.4.5 - Imágenes de texto								
Imágenes	Imágenes susceptibles de ser sustituidas por marcado ?			[C22 C30 G140]		1	1	
1.4.6 - Contraste (Mejorado)								
Presentación	Contraste mejorado ?			[G18 G148 G174]		1	1	
	Contraste mejorado para fuentes grandes ?			[G145 G148 G174]		1	1	
1.4.8 - Presentación visual								
Presentación	Colores de fondo y primer plano en bloques de texto ?			[C23 C25 G156 G148 G175]		1	1	
	Ancho en caracteres de bloques de texto ?			[H87 C20]		1	1	
	Alineación de bloques de texto ?			[C19 G172 G169]		1	1	

Figura 13: Lista de resultados en detalle de TAW

Fuente: <http://www.tawdis.net/>

5.5. Tenon.io

Es la herramienta más completa que existe actualmente de análisis automático, y un claro ejemplo de modelo de **negocio** para aplicaciones de este tipo.

Es la que más técnicas WCAG 2.0 analiza y permite, además, tener un panel de control en el que se supervisan los errores de accesibilidad y su impacto en la página, así como indicaciones sobre cuáles requieren una mayor prioridad de corrección. En la Figura 14 se ven los apartados más interesantes de su página principal.

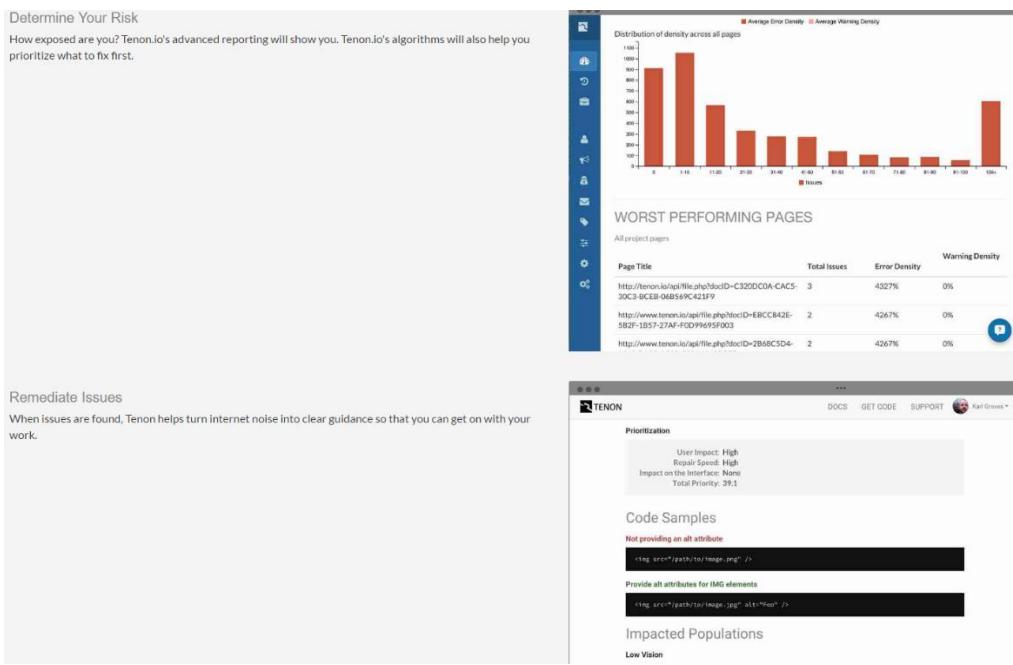


Figura 14: Página principal de Tenon

Fuente: <https://tenon.io/>

En sus distintas versiones de pago nos proporciona la capacidad de realizar más análisis mensuales utilizando su API. Esto es recomendable para, por ejemplo, empresas que se dedican al desarrollo web y quieren hacerlas más accesibles y requieren de análisis continuos a medida que desarrollan su sitio web. Como vemos en la Figura 15, hay diversas cuotas, que permiten un número variable de usos de la API para realizar análisis.

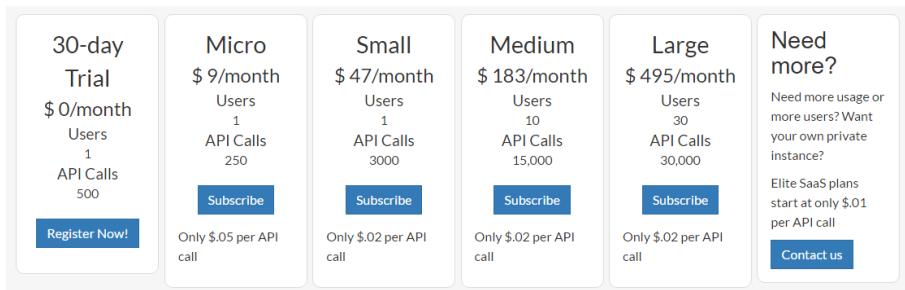


Figura 15: Precios de las suscripciones a Tenon

Fuente: <https://tenon.io/>

En cuanto a la forma de presentar el **resultado** del análisis, combina una gráfica muy visual y una lista con todos los errores encontrados, indicando la línea del código fuente de la página donde se encuentra el problema. Podemos ver estos resultados en las figuras 16 y 17. Permite descargar el resultado del análisis en formato CVS, es decir, en formato de tabla.

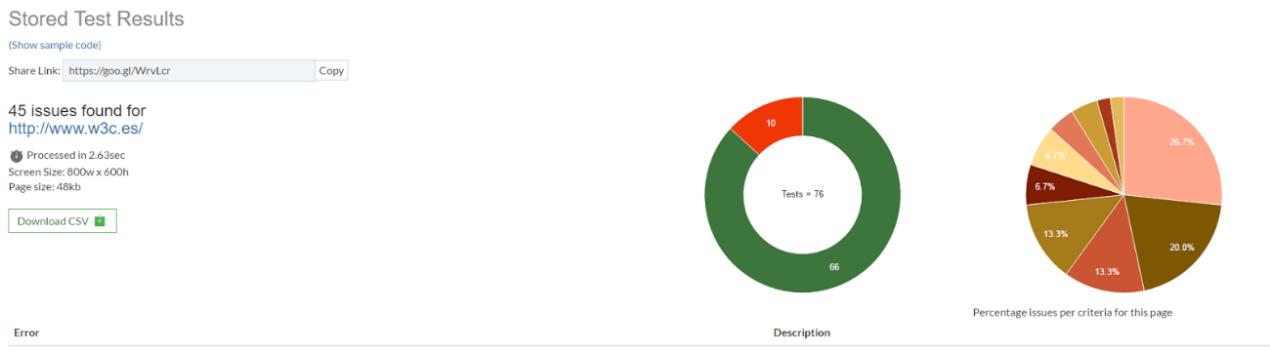


Figura 16: Página de resultados de Tenon (1)

Fuente: <https://tenon.io/>

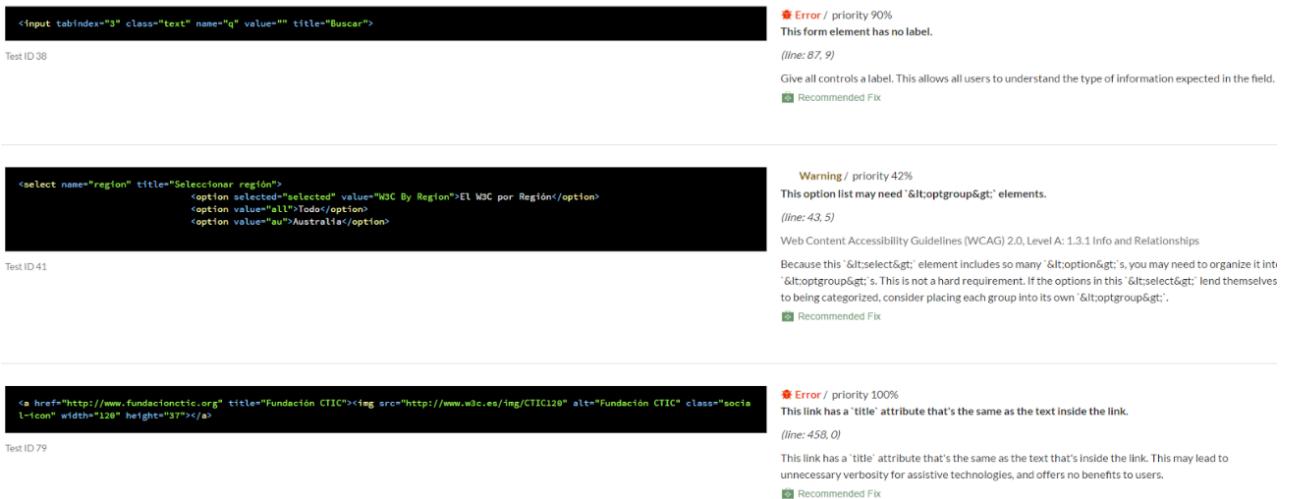


Figura 17: Página de resultados de Tenon (2)

Fuente: <https://tenon.io/>

De esta forma el usuario puede localizar fácilmente los errores y corregirlos de acuerdo a la especificación actual. Por último, cuentan con un blog donde escriben sobre artículos relacionados con la accesibilidad web y su análisis automático, así como algunos tutoriales sobre el uso de su API.

5.6. WAVE

Una herramienta más difícil de utilizar, y claramente enfocada a **expertos** en análisis de la accesibilidad web. Pese a que su página principal es la más simple de todas (véase la Figura 18), lo que llama la atención es la manera de presentar los resultados al usuario.



Figura 18: Página principal de WAVE

Fuente: <http://wave.webaim.org/>

WAVE carga la página web objeto de análisis y sobre ella marca los errores de accesibilidad con una nomenclatura propia de **símbolos** que, al pinchar sobre ellos, nos informan del error en cuestión. En las dos siguientes figuras se muestran sendos ejemplos.

The screenshot displays the "Summary" panel of the WAVE interface. At the top, it says "WAVE has detected the following:" followed by a list of counts: 27 Errors (red), 9 Alerts (yellow), 2 Features (green), 3 Structural Elements (blue), 2 HTML5 and ARIA (purple), and 10 Contrast Errors (black). On the left, there is a vertical sidebar with icons for each category: a clipboard for Errors, a magnifying glass for Alerts, a green circle for Features, a blue square for Structural Elements, a purple square for HTML5 and ARIA, and a black square for Contrast Errors. To the right of the sidebar, under "Panel Options", are three items: "DETAILS: A listing of all the WAVE icons in your page.", "DOCUMENTATION: Explanation of the WAVE icons and how you can make your page more accessible.", and "OUTLINE: The heading structure of the web page.".

Figura 19: Nomenclatura de símbolos y pruebas en WAVE

Fuente: <http://wave.webaim.org/>

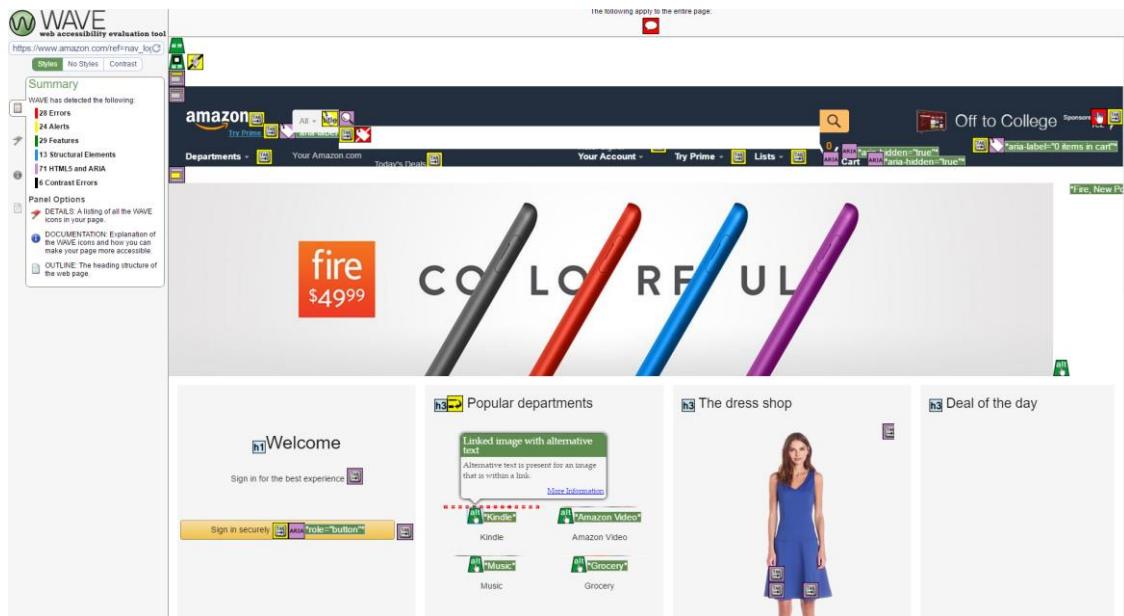


Figura 20: Página de resultados con WAVE

Fuente: <http://wave.webaim.org/>

Esta forma de mostrar los resultados es extremadamente útil, pues al ser **visual** y no en forma de líneas de código HTML, el usuario puede entender mejor el fallo y su localización dentro del contexto visual de la página.

Otra característica muy interesante y útil es que ofrece la posibilidad de visualizar la página sin estilos (sin cargar las reglas de los documentos CSS) para poder visualizar la página tal y como la vería un software de lectura de pantalla, por ejemplo, y así hacerse una idea de cómo una persona ciega navegaría por el sitio. Por último, también permite analizar el contraste de color en la página, para determinar si el texto es legible y supera los niveles AA y AAA.

Por todo esto, AChecker es, junto a Tenon, una de las herramientas de análisis automático más potentes y que todo experto en accesibilidad web debería emplear como complemento.

5.7. Conclusiones

Como hemos podido ver, existen diversas páginas que cumplen el cometido de analizar la accesibilidad web, y cada una presenta unas características y funcionalidades propias. Es, por tanto, altamente recomendable que al realizar un **análisis** se emplee más de una herramienta de análisis automático, pues los resultados pueden variar ligeramente de un sitio a otro e incluso podemos obtener más información o entender mejor el fallo al comparar resultados.

Además, estas páginas deben estar en constante **desarrollo**, pues el W3C sigue trabajando en las reglas de accesibilidad WCAG 2.0 y actualizándolas periódicamente, por lo que siempre es necesaria la **actualización** de dichas páginas y de sus métodos de análisis.

6. Desarrollo

Durante los meses que ha durado el desarrollo del **proyecto**, se ha trabajado en diversas partes que, al ponerse en común, dotan de sentido al proyecto y permiten que se complete satisfactoriamente. A continuación se explican estas partes, de manera cronológica e incremental, pues para poder trabajar sobre una, era necesario haber completado la anterior.

Además, de pondrán ejemplos e imágenes del desarrollo para una mayor comprensión de las diversas partes del mismo.

6.1. Aprendizaje de la accesibilidad web

La primera fase del desarrollo de un proyecto de esta envergadura debe ser **aprender** sobre el tema a tratar, formarse sobre la materia y consultar multitud de artículos, cursos y noticias sobre el mismo.

Para una primera toma de contacto con la accesibilidad web, y apoyándome en los conocimientos adquiridos en diversas asignaturas a lo largo de la carrera, realicé el curso *Aprende accesibilidad web paso a paso*⁸, donde, a lo largo de más de cien clases teóricas y exámenes, mejoré mis conocimientos sobre la accesibilidad, la gente que se beneficia de ella, cómo lograr que un sitio web lo sea y cómo desarrollar mejores páginas siguiendo los principios de accesibilidad actuales. En el apartado [A.1. Aprendiendo sobre accesibilidad web](#) del Anexo hay más información al respecto.

Una vez había adquirido los conocimientos básicos necesarios sobre accesibilidad web, pude comenzar a definir el proyecto de sitio web mejor.

6.2. El análisis automático

Una parte fundamental del proyecto es poder analizar una página web aplicando una serie de **funciones** programadas y, en función del resultado, dar un veredicto aproximado sobre su grado de accesibilidad.

Para poder hacer esto, primero es necesario definir qué apartados o elementos se pueden analizar de manera automática y cuáles no dentro de una página web, aplicando las **directrices** de

⁸ Enlace al curso: <https://www.udemy.com/aprende-accesibilidad-web-paso-a-paso/>

accesibilidad recogidas en las WCAG 2.0. Como se ha mencionado en apartados anteriores, siempre será necesaria la revisión de un experto en accesibilidad, pero un análisis automático preliminar es muy útil en un desarrollo web.

Dentro de las directrices actuales sobre accesibilidad en la Web, las WCAG 2.0, encontramos diversos apartados o secciones sobre qué se puede analizar en un sitio web:

- Técnicas generales.
- Técnicas HTML y XHTML.
- Técnicas CSS.
- Técnicas de *scripting* en el cliente.
- Técnicas de *scripting* en el servidor.
- Técnicas SMIL.
- Técnicas de texto plano.
- Técnicas ARIA.
- Técnicas Flash.
- Técnicas Silverlight.
- Técnicas para PDF.
- Fallos comunes.

Cada uno de estos **apartados** define una serie de **técnicas**, con una descripción de cada una, ejemplos que pueden contener fragmentos de código, información externa, técnicas relacionadas y una serie de test que se pueden aplicar y, en función del resultado de los mismos, se puede decir que la página web cumple con dicha técnica o no, es decir, si es **accesible** en ese aspecto.

Para cada una de los grupos listados anteriormente existe una pequeña **nomenclatura** para distinguir sus técnicas de manera sencilla: se nombran con la primera letra del grupo y un número indicativo de la regla. Así pues, en el siguiente ejemplo vemos que la técnica se llama *G4*. Esto significa que pertenece al conjunto de *técnicas generales* y es la cuarta. Por lo tanto, una regla perteneciente a las técnicas sobre HTML es *H36*, o una sobre *fallos comunes*, la *F25*.

En la siguiente figura se puede apreciar un **ejemplo** de una de estas técnicas. Más concretamente, correspondiente a las *técnicas generales*.

G4: Allowing the content to be paused and restarted from where it was paused

Applicability

Any technology that includes moving or scrolling content.

This technique relates to:

- [Success Criterion 2.2.1 \(Timing Adjustable\)](#)
 - [How to Meet 2.2.1 \(Timing Adjustable\)](#)
 - [Understanding Success Criterion 2.2.1 \(Timing Adjustable\)](#)
- [Success Criterion 2.2.2 \(Pause, Stop, Hide\)](#)
 - [How to Meet 2.2.2 \(Pause, Stop, Hide\)](#)
 - [Understanding Success Criterion 2.2.2 \(Pause, Stop, Hide\)](#)

Description

The objective of this technique is to provide a way to pause movement or scrolling of content. If the user needs to pause the movement, to reduce distraction or to have time to read it, they can do so, and then restart it as needed. This mechanism can be provided either through interactive controls that conform to WCAG or through keyboard shortcuts. If keyboard shortcuts are used, they are documented.

Examples

- A site contains a scrolling news banner at the top of the page. Users who need more time to read it can press the Escape key to pause the scrolling. Pressing Escape again restarts it.
- A Web page contains a link labeled "How to tie a shoe" which links to a Flash animation. Text immediately preceding the link informs the user that pressing the spacebar will pause the animation and restart it again.

Related Techniques

- [G75: Providing a mechanism to postpone any updating of content](#)
- [G76: Providing a mechanism to request an update of the content instead of updating automatically](#)
- [G186: Using a control in the Web page that stops moving, blinking, or auto-updating content](#)
- [SCR33: Using script to scroll content, and providing a mechanism to pause it](#)

Tests

Procedure

On a page with moving or scrolling content,

1. Use the mechanism provided in the Web page or by the user agent to pause the moving or scrolling content.

Figura 21: Ejemplo de técnica descrita en las WCAG 2.0

Fuente: <https://www.w3.org/TR/WCAG20/>

Hay una gran cantidad de técnicas que es necesario **aplicar** para evaluar la accesibilidad de una web. Para este proyecto se ha hecho una **selección** de los grupos listados anteriormente. Debido a la necesidad de usar el software PhantomJS para poder evaluar una página de manera automática, no todos los grupos podían ser analizados, pues, en la mayoría de los casos, requieren de comprobaciones manuales o no podían comprobarse con PhantomJS. Por tanto, se escogieron tres grupos: *técnicas HTML, técnicas generales y fallos comunes*. Dentro de estos tres grupos, a su vez, encontramos decenas de técnicas aplicables, y es necesario **dividirlas**, según se puedan analizar de manera automática o requieran de revisión manual.

Para ello, se ha creado una hoja de cálculo que recoge todas las técnicas de estos grupos y las clasifica en diversos apartados. En el apartado [A.7. Listado de técnicas WACG 2.0](#) implementadas del Anexo se encuentra la lista completa explicada detalladamente, que se ha elaborado con el propósito de organizar el trabajo a implementar.

Una vez se tiene la lista hecha con el conjunto de técnicas que sí se pueden analizar automáticamente mediante código, utilizando PhantomJS, es el momento de empezar a hacer pequeñas pruebas para tomar contacto con este software.

6.3. Implementación de las técnicas en PhantomJS

A lo largo de este documento se ha hecho mención en repetidas ocasiones al software PhantomJS, pero ¿qué es exactamente?

Según sus desarrolladores, es “*un navegador web sin interfaz programable con una API JavaScript. Soporta de forma nativa varios estándares: DOM, selectores CSS, JSON, Canvas y SVG*”⁹.

Esto quiere decir que es una potente herramienta, pues permite **cargar** una página web con todos los elementos que la componen, **acceder** a su contenido, modificarlo, crear **archivos** de texto e imágenes, y todo ello mediante líneas de comando. Además, usa JavaScript, un lenguaje muy conocido y extendido y flexible a la hora de programar con él.

Para este proyecto, este software es muy útil, ya que, una vez el usuario introduzca la URL de la página web que desea analizar, esta es abierta por PhantomJS y, mediante las funciones

⁹ Fuente: <http://phantomjs.org/>

implementadas para el análisis de la página, se accede a su **DOM**, es decir, a los elementos que componen la página, y así es posible evaluarla.

6.3.1. Proceso de implementación

Desde la página oficial de PhantomJS se puede acceder a su **documentación**, donde se explica detalladamente las posibilidades que ofrece. La más interesante es la de cargar una página web a partir de su URL. Esto es la base del análisis automático del proyecto.

Para tener una primera toma de contacto con el programa, se realizaron diversas **pruebas** pequeñas para entender mejor el funcionamiento del mismo.

Como ejemplo, en las dos siguientes figuras se puede ver que, en unas pocas líneas de código, se puede tener lista una prueba sencilla que abre una URL especificada, comprueba si esta es valida y, en caso afirmativo, devuelve el título de la misma, además de imprimirlo y contar el número de caracteres que contiene. Esto será la base para posteriores **análisis** más complejos, pero que se basan en el mismo principio de acceder al **contenido** de la página que ha abierto PhantomJS y **evaluarlo** mediante JavaScript.

```
var page = require( 'webpage' ).create();
page.open( 'http://www.ua.es/' , function( status ) {
    if ( status === "success" ) {
        var G141 = page.evaluate(function() {
            return document.title;
        });
    }
    console.log("Titulo de la pagina : "+G141);
    console.log("Longitud del titulo : "+G141.length+" caracteres");
    phantom.exit();
} );
```

Figura 22: Ejemplo de prueba sencilla en lenguaje JavaScript

```

Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Alejandro>cd Desktop

C:\Users\Alejandro\Desktop>phantomjs prueba.js
Titulo de la pagina : Universidad de Alicante
Longitud del titulo : 23 caracteres

C:\Users\Alejandro\Desktop>

```

Figura 23: Ejemplo de prueba sencilla ejecutada en PhantomJS

Una vez realizadas varias pruebas, era el momento de implementar las técnicas de accesibilidad escogidas y que hay en la lista anteriormente mencionada.

El **proceso de trabajo** es el siguiente:

- Se consultan los test que se proponen en las WCAG 2.0, donde se explican qué **comprobaciones** se pueden aplicar a cada prueba y el **resultado** necesario para que se cumpla, como se aprecia en la siguiente figura.

Tests

Procedure

For each `a` applying this technique:

1. Check that every `img` element contained within the `a` element has a null value set for its `alt` attribute.
2. Check that the `a` element contains an `img` element that has either a null `alt` attribute value or a value that supplements the link text and describes the image

Expected Results

- All checks above are true.

If this is a sufficient technique for a success criterion, failing this test procedure does not necessarily mean that the success criterion has not been satisfied in some other way, only that this technique has not been successfully implemented and can not be used to claim conformance.

Figura 24: Test y resultados esperados proporcionados por las WCAG 2.0

Fuente: <https://www.w3.org/TR/WCAG20/>

- Para cada técnica analizable, se crea una **función** JavaScript diferente dentro del archivo principal que contiene todas las comprobaciones implementadas y es ejecutado por PhantomJS. Las funciones devuelven un número en función del resultado de la prueba que

será interpretado por el lenguaje PHP a la hora de crear y mostrar la página de resultados del análisis en AccesTitan. En la siguiente figura se muestra un ejemplo de función.

```
function testH37 (document) {
    var im = document.querySelectorAll("img");
    if(im.length > 0){
        for(var i = 0; i < im.length; i++){
            if(im[i].getAttribute("alt") == null){
                return 2; //prueba fallida
            }
        }
        return 1; //prueba superada
    }
    return 0; //prueba no aplicable
}
```

Figura 25: Ejemplo de función JavaScript que utiliza PhantomJS

- Cuando la función se ha implementado, se prueba introduciendo la URL de varias webs y comprobando que el resultado devuelto por la función es el adecuado en cada caso. Por lo tanto, cada función implementada se ha probado en varios escenarios y se ha ajustado lo máximo posible para evitar errores.
-

6.3.2. Resultado de la implementación

El resultado del proceso de trabajo anteriormente descrito es un **archivo JavaScript** que el software PhantomJS es capaz de ejecutar. Este archivo está compuesto por tres partes bien diferenciadas e importantes:

- **Conjunto de funciones:** el conjunto de funciones implementadas para comprobar las técnicas de accesibilidad se encuentran agrupadas para que sea más fácil trabajar con ellas. Cada una se llama igual que la técnica que analiza, por lo que la prueba *testH25* hace referencia a la técnica para HTML H25: *Ofrecer un título usando el elemento 'title'*. De esta forma, es fácil hacer cambios en el documento y acotar los fallos. Además, cada una de estas funciones devuelve un número identificativo del resultado de la misma: un cero indica que la técnica no es aplicable, un uno indica que se ha superado la prueba, un dos indica que se ha fallado la prueba y un tres indica que se requiere una comprobación manual para la misma.
- **Elemento JSON:** el conjunto de pruebas implementadas hace referencia a diversas técnicas de accesibilidad web definidas en las WCAG 2.0. Cada prueba contiene un título descriptivo de la misma, un enlace a la documentación oficial y el resultado del análisis asociado. Estos datos se almacenan en un archivo JSON para, posteriormente, mostrarlos

en la sección de resultados de AccesTitan. En el apartado [A.2. El formato JSON](#) del anexo hay más información sobre el mismo.

- **Bucle de análisis:** es necesario ejecutar todas las funciones creadas para que nos devuelvan un resultado y así contar con el resultado total del análisis. Una vez ejecutadas todas estas pruebas, se genera un archivo JSON y se finaliza el proceso de trabajo con PhantomJS.

Así pues, lo que se obtiene a partir de ejecutar este archivo JavaScript es una salida en forma de archivo JSON, que utilizaremos para crear la página de resultados de AccesTitan.

6.4. Creación del sitio web AccesTitan

Una de las partes básicas del proyecto es la creación de un sitio web que permita al usuario analizar la página web deseada y ver el resultado del análisis.

El sitio debe ser sencilla de usar para cualquier usuario, intuitivo y debe satisfacer el propósito principal de poder analizar una web introduciendo su URL.

6.4.1. Tecnologías

Para la creación del sitio web se ha trabajado empleando las siguientes tecnologías:

- **HTML 5:** siglas de *Hypertext Markup Language*. Es el lenguaje estándar para el desarrollo de páginas web y se basa en el uso de **etiquetas**. Con este lenguaje se ha escrito el cuerpo de la página y su **contenido**.
- **CSS:** siglas de *Cascading Style Sheets*. Es un lenguaje que permite definir y crear la presentación visual de un documento escrito en HTML. Con él se le ha dado **estilo** a la web y se ha conseguido lograr un diseño adaptativo o **responsive**.
- **JavaScript:** es un lenguaje de programación enfocado a páginas web, orientado a **objetos** y débilmente tipado. Mediante su uso, se consiguen ciertas tareas dinámicas que no pueden ser definidas de manera estática en el código fuente de la página.
- **PHP:** lenguaje de programación de la parte del **servidor**. Se ha utilizado para ejecutar el comando de PhantomJS que inicia el **análisis**, abrir el archivo JSON generado, leerlo correctamente e imprimir estos resultados como código HTML incrustado para que el navegador los interprete.

En la actualidad, existen multitud de **frameworks** o plantillas predefinidas para crear multitud de tipos de páginas web de forma rápida y personalizable. Cabe destacar que para desarrollar

AccesTitan se ha optado por no emplear ninguna de estas herramientas y, en su lugar, crear la página desde cero, escribiendo todo el código y contenido de ella, para así tener control total sobre todos los aspectos de la misma.

Esto se debe a una sencilla razón: si se va a desarrollar una página que analice la accesibilidad web, lo mejor es “predicar con el ejemplo”, es decir, que nuestra web sea completamente **accesible**. Por desgracia, muchas plantillas no tienen en cuenta la accesibilidad web o ciertos aspectos de ella, por lo que lo más recomendable es crear la página completamente a mano. De esta forma, podemos controlar que los **elementos** en ella presentes sean accesibles y se cumplan las directrices de las WCAG 2.0 aplicables.

6.4.2. Diseño

El diseño del sitio web está inspirado en el de la página Examinator, pues es sencillo y facilita al usuario una navegación clara e intuitiva a través de todas las secciones que componen la misma. Para lograr un diseño de este estilo también se ha realizado una serie de bocetos previos para visualizar mejor la distribución de los elementos, su tamaño, etc.

Una vez hechos los bocetos previos, y durante el desarrollo del sitio web, se fueron probando diversas combinaciones de colores y disposiciones de los elementos importantes. Al final se consiguió una combinación que tiene suficiente contraste entre el fondo y los elementos y no dificulta la lectura del texto en ningún momento.

Cabe destacar que todos los elementos de la página tienen un tamaño definido en porcentaje y no en píxeles, lo que favorece enormemente al diseño **responsive** o **adaptativo** de la página. Esto quiere decir que esta se adapta de forma correcta a los distintos tamaños de pantalla que pueden presentar los dispositivos en los que va a ser visualizada.

6.4.3. Página principal

La página principal de AccesTitan es la primera que el usuario ve cuando accede a la página, por lo que debe ofrecer la máxima cantidad de información de forma breve. Como se desea que la web sea fácil de usar y el usuario final solo deba introducir una URL para realizar un análisis, la página principal debe ser la que ofrezca al usuario la información de cómo se realiza un análisis y la que permita que lo lleve a cabo.

Por ello, en esta página hay un campo de texto donde introducir dicha URL y un botón de envío que comienza el análisis. Estos elementos son los más importantes, por lo que están destacados y centrados para la vista.

Además, desde esta página principal tenemos acceso al resto de páginas de la web, que son la de resultados del análisis y la de información sobre AccesTitan.

6.4.4. Página de resultados

La otra gran parte de la web AccesTitan es la página de resultados. Una vez que el usuario ha introducido una URL en el campo de texto de la página principal y lo envía para el análisis, será redirigido a la página de resultados una vez que el análisis haya concluido y se haya recibido del servidor los datos del mismo.

En esta página es fundamental que el usuario entienda fácilmente la información ofrecida, por lo que consta de dos partes:

- **Nota y datos del análisis:** se muestra una nota **orientativa** sobre el grado de accesibilidad de la página web analizada, fruto de aplicar las funciones definidas para su evaluación y del resultado obtenido en las mismas. Además, se muestra la URL que se ha analizado, por si hubiera habido algún error con lo introducido, así como la fecha en la que se ha realizado el análisis.
- **Resultados del análisis:** esta parte es la más interesante para el usuario, pues muestra los errores detectados en la web analizada e incluye un enlace a la documentación oficial que explica cómo solucionarlos. En ella, se muestra una lista con las pruebas realizadas. A su vez, esta lista se divide en cuatro partes:
 - **Pruebas superadas:** cuando una página implementa correctamente una técnica de accesibilidad, también es importante destacarlo, por lo que aquí se listan las pruebas que ha superado satisfactoriamente la página.
 - **Pruebas no superadas:** los fallos de accesibilidad de la web que se han podido analizar de manera automática. Viendo estos fallos, el usuario puede comprender mejor la forma de subsanarlos.
 - **Revisión manual:** como se ha explicado en apartados anteriores, no todas las técnicas para la accesibilidad web pueden ser comprobadas de manera automática. Se puede dar el caso de que se aplique una comprobación y se obtenga un falso positivo, o un falso negativo. Por ello, estas técnicas no pueden ser revisadas mediante comprobaciones automáticas y han de ser revisadas por un experto en

accesibilidad web. Aun así, es importante que el usuario pueda verlas para obtener más información sobre posibles fallos.

- **Pruebas no aplicables:** no todas las pruebas implementadas pueden aplicarse a una página durante el análisis de la misma. Por ejemplo, si una prueba analiza los formularios pero la página no contiene ninguno, esta prueba no es aplicable. Se lista en la página de resultados porque es recomendable ofrecer la mayor cantidad posible de información al usuario.

Además, es posible seleccionar qué lista queremos mostrar en cada momento mediante los botones implementados justo antes de la misma, que ocultan o muestran los distintos bloques en los que se muestran los resultados.

6.4.5. Otras páginas

En cualquier página web es importante ofrecer la máxima información posible sobre la misma y sobre su contenido. AccesTitan tiene una página dedicada a dar información sobre ella, y desde la que se puede acceder a la página del proyecto en GitHub, un repositorio en línea desde donde cualquiera puede realizar contribuciones al proyecto.

6.4.6. Flujo de ejecución

En la Figura 26 se muestra el proceso que se sigue cada vez que se realiza un análisis en AccesTitan, es decir, todo lo que sucede desde que el usuario introduce una URL y pulsa el botón *Analizar* hasta que obtiene los resultados.

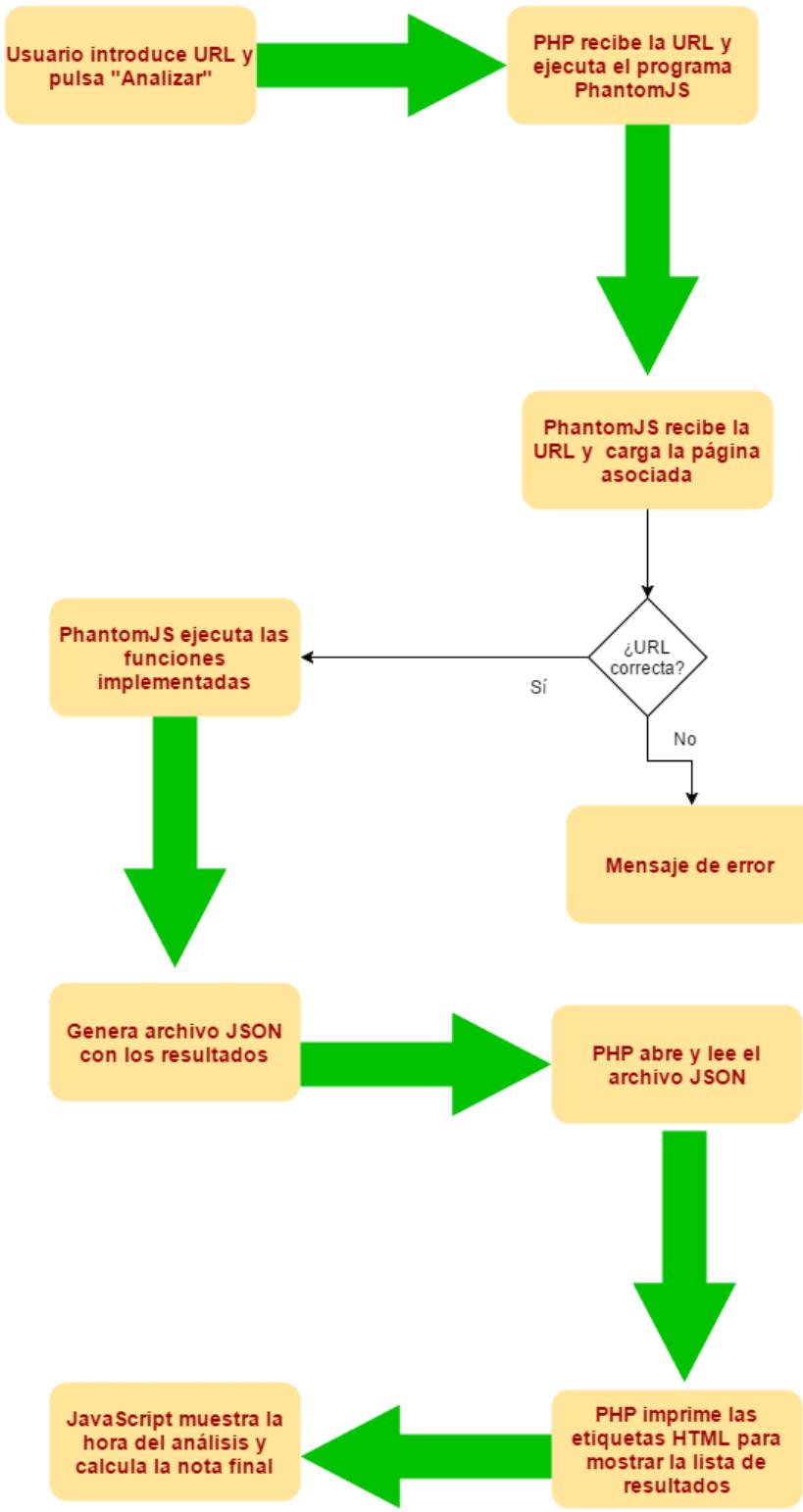


Figura 26: Flujo del análisis

7. Resultados

Durante el desarrollo de este proyecto se ha diseñado un sitio web que cumple con los requisitos establecidos en el [apartado 4.2](#). Además, se han analizado una gran cantidad de técnicas del WCAG 2.0 y se han adquirido conocimientos generales sobre accesibilidad web y cómo conseguirla.

En los siguientes apartados se muestra el resultado del trabajo en las distintas partes del proyecto.

7.1. Lista de técnicas analizables

En el apartado [A.7. Listado de técnicas WCAG 2.0](#) se puede consultar la lista en su totalidad. Cabe recordar que han sido extraídas directamente de la [documentación oficial¹⁰](#) del W3C.

7.2. Implementación de las técnicas

Se ha logrado crear un archivo JavaScript ejecutable mediante el software PhantomJS que permite analizar la accesibilidad web mediante la aplicación de sucesivas pruebas sobre la página analizada. El archivo generado a raíz de este análisis es usado por la web AccesTitan para mostrar el resultado al usuario. En el apartado [A.7.2. Tabla de técnicas implementadas en AccesTitan](#) se pueden consultar el conjunto de técnicas implementadas.

7.3. Sitio web AccesTitan

Se ha conseguido un prototipo totalmente funcional que cumple con los requisitos especificados. El prototipo es capaz de realizar análisis automáticos de la accesibilidad de una página web, mostrar una nota orientativa y listar las pruebas superadas y fallidas, así como las que necesitan de revisión manual y las pruebas que no se han aplicado.

Además, ofrece un mensaje de error cuando la URL introducida no es válida o no ha podido analizarse, de modo que el usuario conoce en cada momento qué está pasando.

En las siguientes figuras se muestran las distintas partes que componen la web, así como varios ejemplos de análisis y el diseño adaptativo que presenta la web cuando se muestra en pantallas pequeñas o dispositivos móviles.

¹⁰ Documentación oficial: <https://www.w3.org/TR/WCAG20-TECHS/Overview.html>

AccesTitan

Estás en: [Inicio](#)

Análisis de la Accesibilidad Web

Introduce la URL de la web

Analizar

¿Qué es AccesTitan?

Es una página que permite analizar la Accesibilidad Web de cualquier otro sitio web. Es parte de un Trabajo de Fin de Grado del grado en Ingeniería Multimedia de la Universidad de Alicante.

¿Cómo funciona?

Introduce la URL del sitio que quieras analizar y pulsa el botón de "Analizar". Una vez finalice el análisis, el resultado será mostrado en la misma página. Una nota indicará el grado de accesibilidad de una página, siendo 10 el máximo posible. Si quieres más información, visita nuestra [página de información](#).

Figura 27: Página principal de AccesTitan

AccesTitan

Estás en: [Inicio](#) > [Información](#)

Información sobre AccesTitan

El código fuente de AccesTitan

Si estás interesado en el desarrollo web yquieres conocer cómo se ha creado esta página, puedes ver su código fuente y todos los recursos utilizados en ella desde GitHub, a través del siguiente enlace: [Código de AccesTitan en GitHub](#)

Sobre AccesTitan

Con AccesTitan pueden analizar de manera automática la accesibilidad de cualquier web, pues se trata de una herramienta automatizada y que aplica las directrices de las directrices actuales sobre accesibilidad web, las **WCAG 2.0**. El resultado del análisis que muestra AccesTitan sobre el grado de accesibilidad de la web evaluada es **orientativo** y en ningún caso debe tomarse como definitivo, siendo necesaria la revisión de un experto en accesibilidad web.

¿Cómo funciona?

Introduce la URL del sitio que quieras analizar y pulsa el botón de "Analizar". Una vez finalice el análisis, el resultado será mostrado en una nueva página. Este incluye una nota final orientativa, que puede oscilar entre 0 y 10, que es el máximo posible. Además, se mostrarán cuatro listas, correspondientes a las pruebas realizadas:

- Pruebas superadas: aquellas directrices de accesibilidad que cumple la página.
- Pruebas no superadas: las directrices de accesibilidad que no cumple la página y es necesario corregir.
- Comprobación manual: algunas directrices de accesibilidad no se han podido analizar automáticamente y es necesario que sean revisadas por un experto.
- No aplicables: las directrices que no se aplican a la página que se ha analizado, se listan con fin meramente informativo.

Las comprobaciones automáticas se han realizado empleando el software [PhantomJS](#) y una serie de funciones implementadas en JavaScript.

Fallo al analizar

En ocasiones, es posible que la página web que se ha querido analizar no ha podido ser abierta por el programa, por lo que el análisis no ha podido llevarse a cabo. En esta situación, le recomendamos que vuelva a intentar realizar el análisis desde la [página principal](#).

Figura 28: Página de información sobre AccesTitan

En la Figura 29 se puede apreciar el resultado de realizar un análisis a una página comercial. Se indica el número de pruebas realizadas sobre esta página. Cada prueba enlaza a la documentación oficial de la misma.

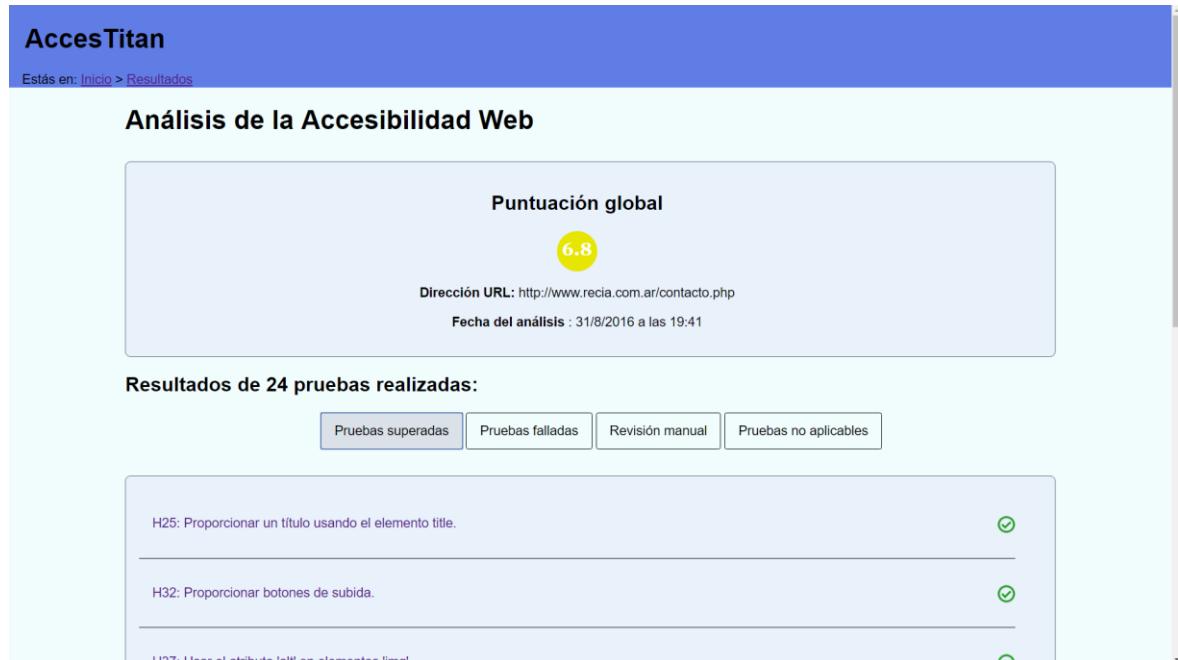


Figura 29: Resultado de un análisis con AccesTitan (1)

Podemos cambiar de bloque para ver las pruebas fallidas, y se resalta el bloque actual para una mejor comprensión del usuario.



Figura 30: Resultado de un análisis con AccesTitan (2)

En esta Figura 31 se muestra un análisis sobre otra página. La nota obtenida ha sido menor que en el ejemplo de la Figura 30, por lo que el color de la nota será rojo, indicador de que esta página *suspende* en accesibilidad.

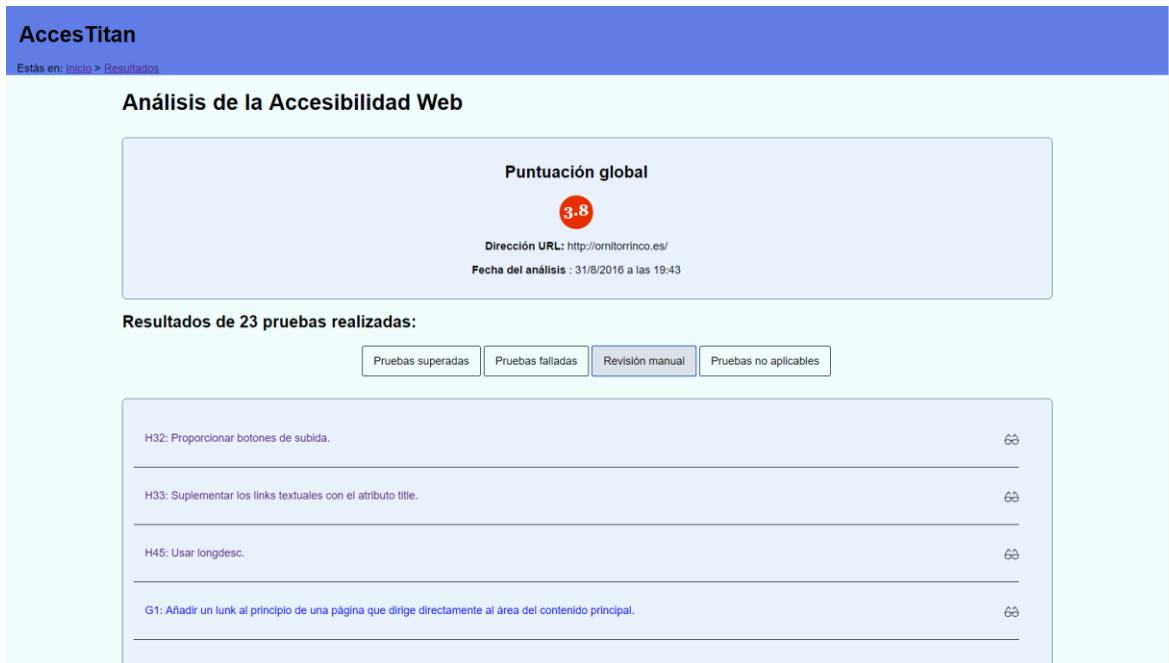


Figura 31: Resultado de un análisis con AccesTitan (3)

Si introducimos una URL incorrecta o que PhantomJS no puede abrir, se mostrará el mensaje de error de la Figura 32 al usuario.

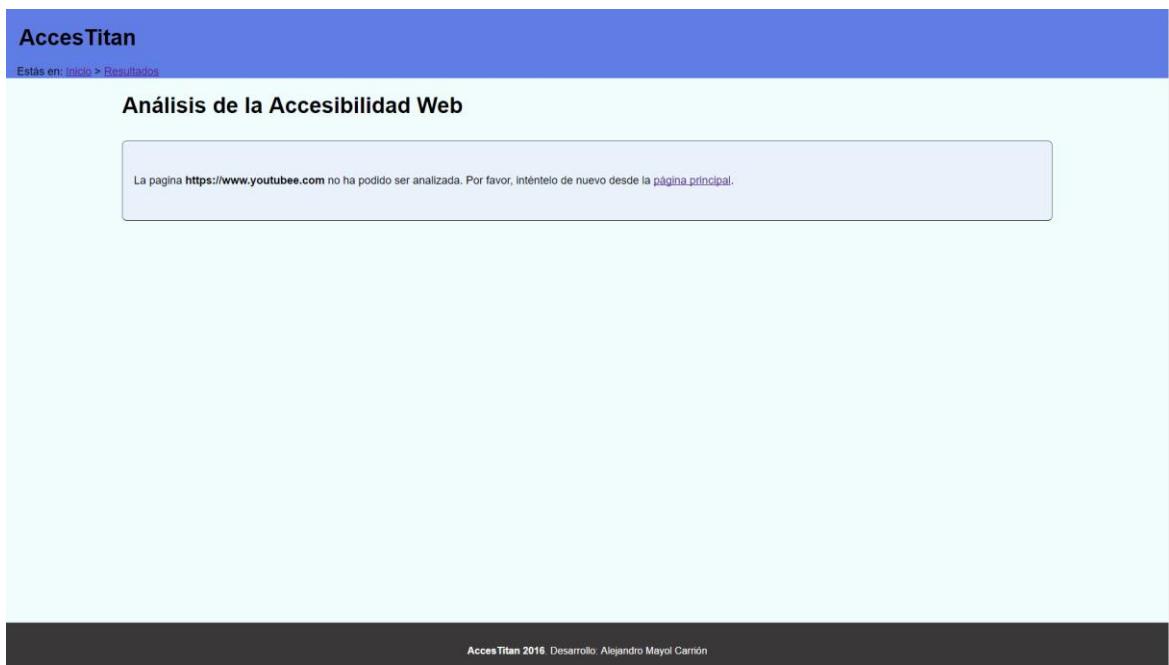


Figura 32: Página incorrecta y mensaje de error

Las figuras 33 y 34 muestran el diseño que presenta AccesTitan cuando se abre en un navegador con pantalla reducida, como puede ser un móvil, tableta o simplemente al redimensionar la ventana del navegador en un ordenador.

The screenshot shows the main page of the AccesTitan website. The header is blue with the text "AccesTitan". Below it, a breadcrumb navigation says "Estás en: Inicio". The main title is "Análisis de la Accesibilidad Web". A central box contains the instruction "Introduce la URL de la web" with a text input field containing "http://". Below the input field is a green button labeled "Analizar". To the right of this box, there is a section titled "¿Qué es AccesTitan?" which describes the site as a tool for analyzing web accessibility. Another section titled "¿Cómo funciona?" provides instructions on how to use the service.

Figura 33: Diseño adaptativo de la web (1)

This screenshot shows the results page after an analysis has been performed. The header is blue with the text "AccesTitan" and the breadcrumb "Estás en: Inicio > Resultados". The main title is "Análisis de la Accesibilidad Web". A central box displays the "Puntuación global" (Global Score) as "6.8" inside a yellow circle. Below the score, the "Dirección URL:" is listed as "http://selingerbarbacoas.com/" and the "Fecha del análisis :" is "31/8/2016 a las 19:49". At the bottom of this box, the text "Resultados de 25 pruebas realizadas:" is visible. A button at the bottom of the box says "Pruebas superadas".

Figura 34: Diseño adaptativo de la web (2)

El resultado obtenido ha sido fruto del uso combinado de las diversas tecnologías empleadas y la aplicación de una metodología de trabajo incremental. Se puede considerar que el sitio web creado es moderno y está dentro de los estándares actuales.

El código fuente del proyecto se ha alojado en **GitHub** para que cualquier persona pueda acceder a él, consultarla, ampliarlo o usarlo, y es accesible desde este enlace:

<https://github.com/AlexMayol/AccessTitan>

7.4. Ejecución del prototipo

En este apartado se va a explicar cómo se puede utilizar el proyecto **AccessTitan** en un ordenador personal.

1. Debemos instalar en nuestro equipo el programa [XAMPP¹¹](#), que nos permite tener un servidor web de manera local para poder hacer pruebas en él.
2. Una vez instalado XAMPP, se debe copiar y pegar la carpeta *AccessTitan* dentro del directorio *xampp/htdocs*. Esta carpeta se ha proporcionado en la entrega, pero también puede [descargarse desde GitHub¹²](#) en formato Zip.
3. Ahora debemos iniciar XAMPP. Para ello, basta con ejecutar el archivo *xampp-control.exe* que se encuentra dentro de la carpeta del programa. Se abrirá un pequeño cuadro de mandos y debemos iniciar el módulo Apache que incluye XAMPP, como se aprecia en la Figura 35
4. Por último, basta con escribir la dirección <http://localhost/AccessTitan/index.html> en cualquier navegador web.
5. La página ya está lista para usarse. Se puede navegar por ella o introducir una URL y pulsar el botón *analizar* para comenzar un análisis. Tras unos segundos, seremos redirigidos a la página de resultados.

¹¹ Página de XAMPP: <https://www.apachefriends.org/es/index.html>

¹² Repositorio en GitHub: <https://github.com/AlexMayol/AccessTitan>

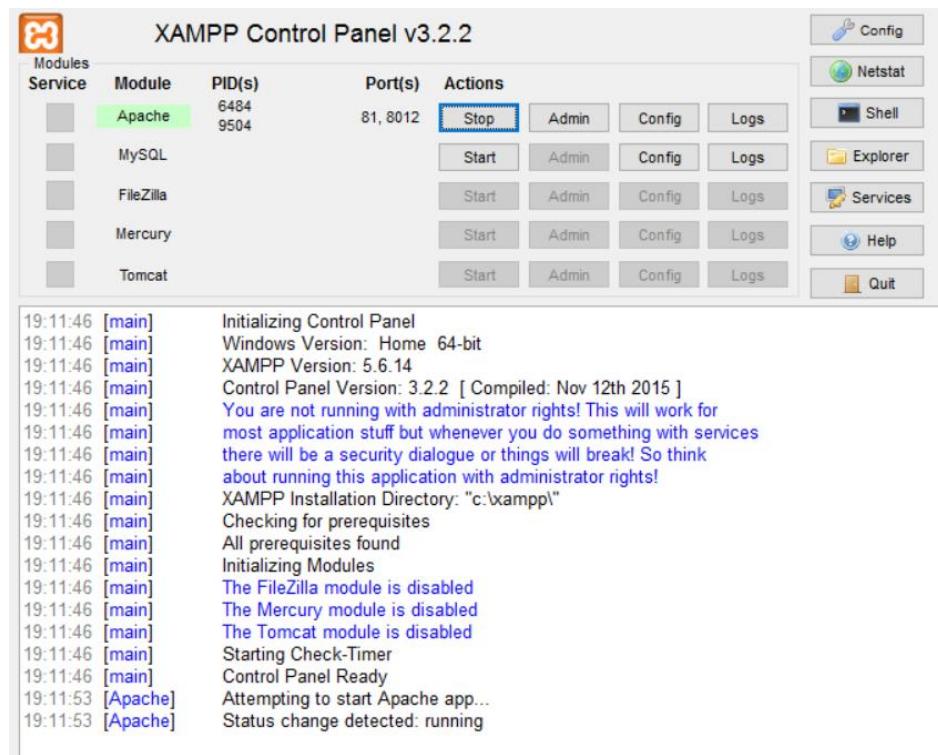


Figura 35: Paned de control de XAMPP

8. Conclusiones

A lo largo de la elaboración de este proyecto se ha tratado el tema de la **accesibilidad web** de una manera amplia y profunda, investigando sobre los distintos tipos de discapacidades y su impacto a la hora de usar Internet, las tecnologías de apoyo existentes, las pautas y consejos para lograr que una página sea accesible y las aplicaciones existentes para comprobarlo.

Se ha visto que la accesibilidad web no es solo una necesidad, sino un tema de **actualidad**, sobre el que se escriben decenas de artículos al día, se desarrollan nuevas aplicaciones continuamente y que recibe apoyo gubernamental debido a su importancia y al impacto positivo que tiene en la población.

8.1. Mejoras y ampliaciones

Una mejora evidente del proyecto es alojarlo en un servidor y que cualquier persona pueda acceder al sitio web y realizar un análisis de la accesibilidad de cualquier página web. Con esto, el proyecto adquiriría **visibilidad** global y sería de gran utilidad para aquellos que realicen análisis automáticos y comparén los resultados de diversas herramientas en línea.

Una posible ampliación sería refinar y aumentar el número de comprobaciones que hace el código escrito y convertirlo en una **API** accesible de manera sencilla por cualquiera que quiera realizar un test. Bastaría con adaptar el proyecto de forma que fuese más entendible para cualquiera y fácilmente escalable.

Otra ampliación factible es ofrecer la página en distintos idiomas, es decir, que sea, multilingüe. Esto permitiría que un mayor número de usuarios utilizasen la web.

Una opción interesante y que los analizadores descritos en el apartado [5. Antecedentes](#) no incorporan es la posibilidad de analizar todo un sitio web a partir de una URL, de modo que el análisis se realizaría sobre la página seleccionada y, a partir de los enlaces que contenga, iría extendiendo el análisis al resto de páginas de la web, en caso de que las haya. Así se consigue un resultado mucho más real y completo sobre el grado de accesibilidad de la web.

Además, otra característica interesante sería habilitar un sistema de **usuarios**, de forma que una persona se pueda registrar en AccesTitan y pueda tener en su cuenta un **historial** de los análisis realizados sobre cualquier página. Además, esto permitiría poder comparar varios análisis hechos sobre una misma URL a lo largo del tiempo para comprobar cómo **evoluciona** su accesibilidad.

Por último, se puede implementar en la página de resultados un botón para **compartir** el análisis realizado en **redes sociales**, como Twitter o Facebook, de modo que el resultado adquiera más visibilidad y un mayor público sea consciente y conozca la accesibilidad web.

8.2. Modelo de negocio

Existen varias formas viables de monetizar este proyecto u otro basado en él y de mayor calibre.

La primera es realizar una API y cobrar por el acceso a ella, como hace la web [Tenon](#)¹³. Se puede dar de forma gratuita un número de usos o llamadas a la API, pero luego sería necesario un tipo de **suscripción** para seguir usando el analizador. Debido a que existen multitud de páginas que ofrecen servicios idénticos de manera gratuita, este modelo de negocio es arriesgado.

Otra forma posible, y que no he visto implementada en ninguna web hasta la fecha, es ofrecer la posibilidad de contactar con un empleado de la web vía *chat* para que solucione las posibles dudas del cliente sobre el análisis que acaba de realizar. Si el usuario realiza un pago mensual por el servicio, podría acceder tanto a la API de análisis como a la **consulta con un experto** en accesibilidad web de manera ilimitada. Esto sería parecido a contar con un experto en accesibilidad dentro de la empresa, pero a un precio mucho menor.

Cabe destacar algo importante: Internet es un recurso libre, y acceder a él debería ser un derecho. Por ello, la accesibilidad web también es algo a lo que todos tenemos derecho y de la que todos nos beneficiamos. Ofrecer un servicio como este de manera gratuita es una opción muy adecuada, así que una forma justa de monetización sería implementar un sistema de **donaciones** para el desarrollador o incluso un modelo de **micromecenazgo** (*crowdfunding*) como el que ofrece [Patreon](#)¹⁴, de forma que cualquier persona que lo deseara pudiera donar para que el proyecto siguiera creciendo. Esta forma de financiación es cada vez más popular en la actualidad, sobre todo en proyectos *software* o audiovisuales.

¹³ Tenon: <https://tenon.io/>

¹⁴ Patreon: <https://www.patreon.com/>

9. Bibliografía

- Brajnik, G. (2004). *Using automatic tools in accessibility and usability*. Recuperado de:
https://users.dimi.uniud.it/~giorgio.brajnik/papers/qa_processes.pdf
- Brajnik, G. (2008a). *Towards a sustainable web accessibility*. Recuperado de:
<https://users.dimi.uniud.it/~giorgio.brajnik/papers/york08.pdf>
- Brajnik, G. (2008b). *A comparative test of web accessibility evaluation methods*. Recuperado de:
<https://users.dimi.uniud.it/~giorgio.brajnik/papers/bw-study-assets08.pdf>
- Carreras, O. (2012) *Consejos avanzados de accesibilidad web*. Recuperado de
<http://olgacarreras.blogspot.com.es/2012/01/consejos-avanzados-de-accesibilidad-web.html>
- Luján, S. (2012) *Accesibilidad web: Discapacidades*. Recuperado de
<http://accesibilidadweb.dlsi.ua.es/?menu=discapacidad>
- Roy, E. 2016. *When we design for disability, we all benefit*. Recuperado de:
<https://www.youtube.com/watch?v=g2m97gPI70I>
- Vanderheiden, G. (2000) *Fundamental Principles and Priority Setting for Universal Usability*. Recuperado de http://trace.wisc.edu/docs/fundamental_princ_and_priority_acmcuu2000/
- W3C (2005). *Introducción a la Accesibilidad Web*. Recuperado de
<http://www.w3c.es/Traducciones/es/WAI/intro/accessibility>
- W3C (2005) *How People with disabilities use the web*. Recuperado de
<https://www.w3.org/WAI/EO/Drafts/PWD-Use-Web>
- W3C, (2008) Conformidad con las WCAG 2.0. Recuperado de
<http://www.sidar.org/traducciones/wcag20/es/#conformance-reqs>
- W3C (2008). *Web Content Accessibility Guidelines (WCAG) 2.0*. Recuperado de
<https://www.w3.org/TR/WCAG20/>
- W3C (2016). *How to meet WCAG 2.0*. Recuperado de
<https://www.w3.org/WAI/WCAG20/quickref/>

A. Anexos

A.1. Aprendiendo sobre accesibilidad web

Antes de embarcarse en un proyecto de este calibre, lo primero es informarse en profundidad sobre el tema. Por ello, dediqué los primeros meses del proyecto a formarme, repasar conceptos aprendidos en diversas asignaturas y realizar el curso [Aprende Accesibilidad Web paso a paso.](https://www.udemy.com/aprende-accesibilidad-web-paso-a-paso/)¹⁵

Este curso es impartido por diversos profesionales en el desarrollo de aplicaciones web y de la accesibilidad. Está formado por ciento cuarenta y tres elementos, entre los que hay lecciones textuales, vídeos explicativos, recursos para ampliar conocimientos y pequeños exámenes.

Tras completar el curso, de una duración aproximada de treinta horas, recibí un diploma acreditativo del mismo.



Figura 36: Diploma del curso

¹⁵ Curso: <https://www.udemy.com/aprende-accesibilidad-web-paso-a-paso/>

A.2. El formato JSON

En este proyecto se ha utilizado este formato de intercambio de datos debido a lo útil que supone su utilización con los lenguajes JavaScript y PHP, aunque en la actualidad puede ser leído por prácticamente cualquier lenguaje de programación. Este formato se basa en el uso del denominado par **nombre/valor**. En base al almacenamiento de los datos en variables con un nombre, se pueden construir *arrays* (conjuntos de valores) y tratarlos automáticamente. En la siguiente figura se muestra el archivo generado por AccesTitan tras el análisis de una página, y los valores que almacena en cada posición del Array.

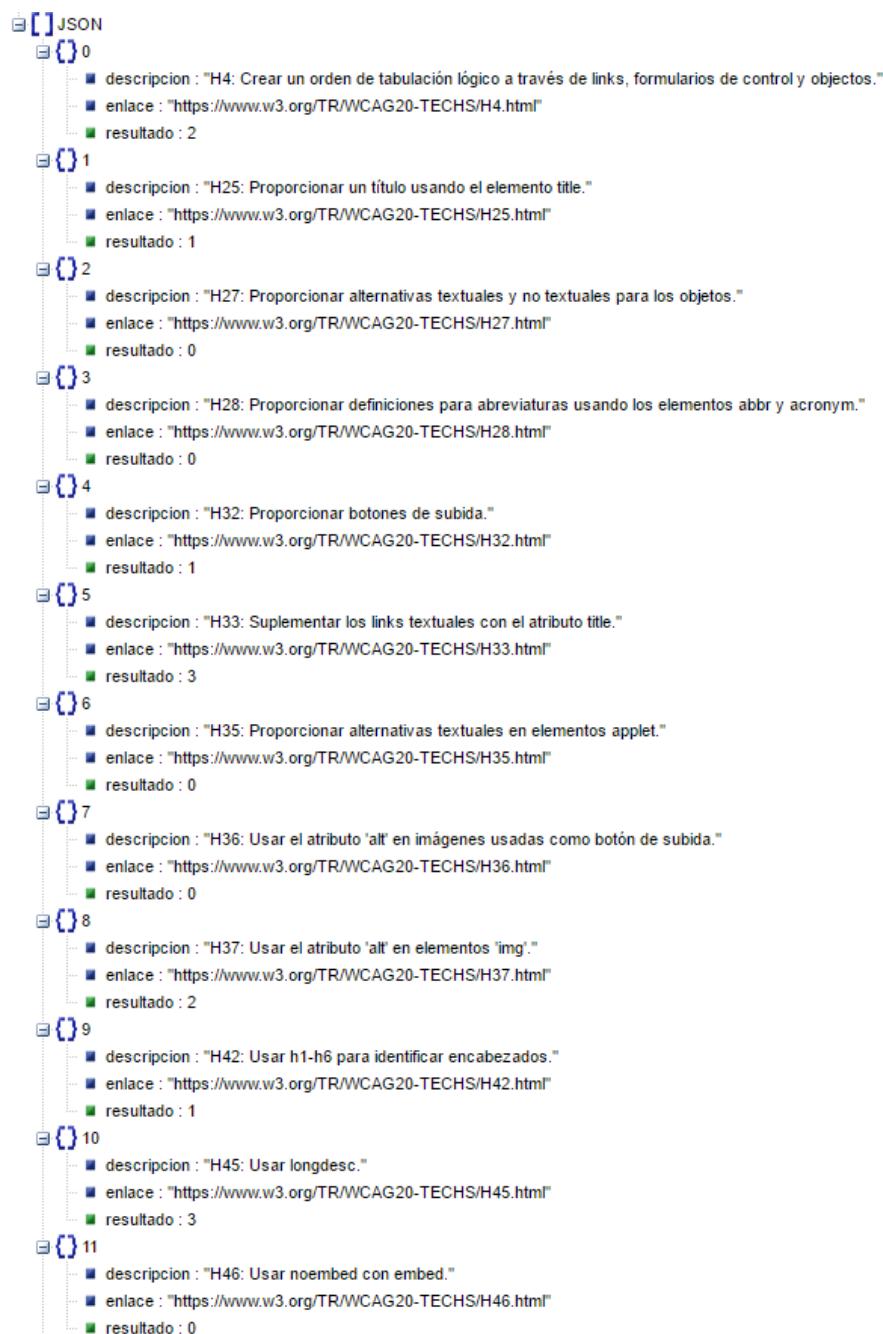


Figura 37: Ejemplo de formato JSON

A.3. Tipos de discapacidades

La accesibilidad web ayuda enormemente a que las personas que sufren algún tipo de **discapacidad** puedan **navegar** cómodamente por la red. A continuación se describen los **tipos** de discapacidad, cómo **afecta** en la navegación web dichas discapacidades y cómo las personas que las sufren consiguen navegar gracias a distintos **productos de apoyo**.

A.3.1. Discapacidad visual

Puede tratarse de ceguera, visión reducida, o daltonismo. Las tres comparten ciertas desventajas para las personas que las padecen a la hora de navegar por la red.

Son el grupo **más afectado** si una web no es lo suficientemente accesible. Cuando en la página hay imágenes sin texto alternativo, el tamaño del texto es muy pequeño o se utilizan combinaciones de colores con poco contraste, podemos decir que la página no es accesible para este grupo.

Por suerte, existen distintos **dispositivos** de apoyo para estas personas:

- Para las personas con ceguera total o visión muy reducida. Cuando no pueden utilizar la pantalla del ordenador, utilizar **lectores de pantalla**, un software que permite al usuario utilizar el sistema operativo y sus distintas aplicaciones mediante un sintetizador de voz que reproduce lo que se visualiza en la pantalla.
 - Uno de los programas más populares de este tipo es [JAWS](#), que es de pago. También hay múltiples ejemplos de softwares gratuitos, como [NVDA](#) o [MexVox](#).
- Para personas con deficiencia visual o visión parcial. Cuando no pueden ver correctamente el contenido de una página web, suelen emplear software o hardware que magnifican la pantalla.
 - Windows incorpora un programa de lupa, aunque otros ejemplos de más calidad son [Dolphin LunarPlus](#) o [The Magnifier](#)

A.3.2. Discapacidad auditiva

A veces es posible que alguna página disponga de algún elemento cuya interacción requiere de sonido. Además, necesitan fijarse más en el texto de la página o sus imágenes. Por ello, webs que tengan elementos sonoros sin **transcripción** textual, que carezcan de imágenes explicativas del contenido o que no emplee un lenguaje claro y simple pueden dificultar el acceso de este grupo de personas al contenido. Esto es debido a que están acostumbradas al lenguaje de signos y no al

escrito/hablado, por lo que pueden no tener la misma facilidad de **compresión** de los elementos mostrados.

Si bien las personas con este tipo de discapacidad no suelen emplear ningún dispositivo de soporte, una web accesible puede ayudar en gran medida a que naveguen sin problemas, debido a que, por ejemplo, siempre habrá alternativas a todos los contenidos.

A.3.3. Discapacidad física o motriz

Cualquier condición que afecte a las manos y brazos, como puede ser parálisis, problemas de articulaciones o dolor físico, dificultan el uso de la Web.

Si una página presenta interfaces que sólo pueden operarse mediante ratón, elementos muy pequeños y que requieran de un control muy preciso o interfaces que requieren de un tiempo de respuesta muy pequeño, puede dificultar el acceso de personas con este tipo de discapacidad. Para solucionarlo, pueden utilizar **productos** de apoyo adaptados como teclados especiales, conmutadores, sistemas de reconocimiento facial, etc.

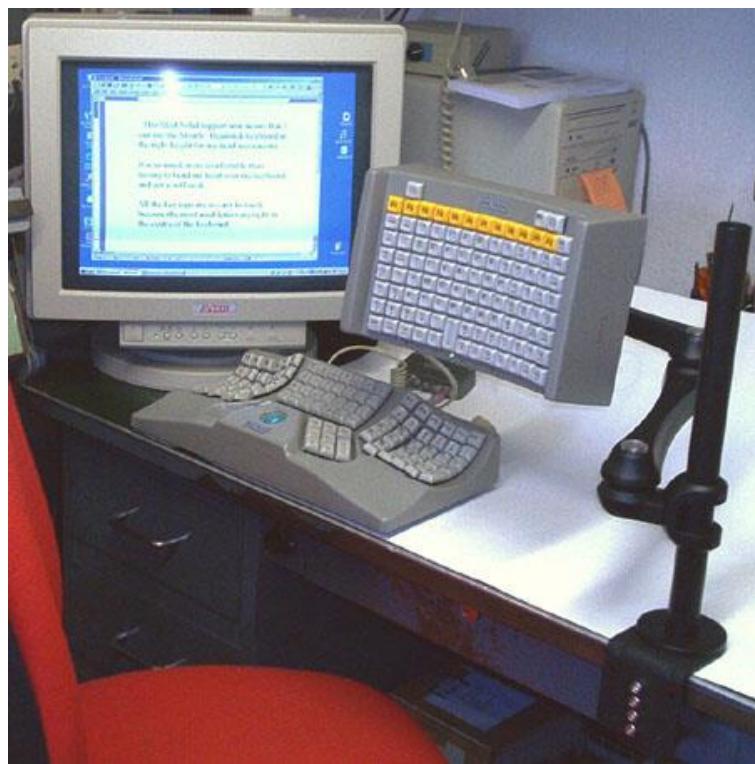


Figura 38: Teclado adaptado para personas con discapacidad motriz

Fuente: <http://www.maltron.com/>

A.3.4. Discapacidad del habla

Puede suponer un problema en alguna página que incluya el reconocimiento de voz. Por suerte, no supone un grave problema en la actualidad, pues todavía no se emplea el reconocimiento de voz en las páginas web.

A.3.5. Discapacidad cognitiva

Problemas como el déficit de atención, discapacidades intelectuales, problemas de memoria o problemas mentales dificultan el uso de la Web.

Los textos largos y **complejos**, la ausencia de imágenes que complementen la información textual o las **inconsistencias** entre distintas páginas de un sitio web son un impedimento para las personas con estas discapacidades.

Para paliar estos problemas, las personas con discapacidad cognitiva suelen utilizar navegadores web accesibles, que les facilitan la navegación. Ejemplo de ello son [WWAAC Web Browser](#).



Figura 39: Programa WWAAC Web Broeser

Fuente: <http://www.wwaac.eu/products/browser.asp>

A.3.6. Otros tipos de discapacidades

Existen más tipos de discapacidades que las mencionadas anteriormente, pero no siempre se tienen en cuenta a la hora de hablar de accesibilidad web o discapacidad. Dos de los ejemplos más claros son los siguientes:

- **Situaciones relacionadas con el envejecimiento:** cuando una persona envejece y pierde facultades físicas, normalmente aparecen problemas relacionados con las discapacidades visual, auditiva y física, en distintos niveles o formas.
- **Discapacidad tecnológica:** una página web puede no estar preparada para visualizarse en sistemas operativos antiguos, navegadores nuevos, mala conexión a Internet o pantalla de visualización pequeña.

A.4. Principios de la accesibilidad web

En el documento de las WCAG 2.0 se especifican los cuatro principios en los que se basan. A continuación se explican con más detalle.

Principio 1: Perceptible - La información y los componentes de la interfaz de usuario deben ser presentados a los usuarios de modo que ellos puedan percibirlos.

- **Pauta 1.1 Alternativas textuales:** Proporcionar alternativas textuales para todo contenido no textual de modo que se pueda convertir a otros formatos que las personas necesiten, tales como textos ampliados, braille, voz, símbolos o en un lenguaje más simple.
- **Pauta 1.2 Medios tempodependientes:** Proporcionar alternativas para los medios tempodependientes.
- **Pauta 1.3 Adaptable:** Crear contenido que pueda presentarse de diferentes formas (por ejemplo, con una disposición más simple) sin perder información o estructura.
- **Pauta 1.4 Distinguible:** Facilitar a los usuarios ver y oír el contenido, incluyendo la separación entre el primer plano y el fondo.

Principio 2: Operable - Los componentes de la interfaz de usuario y la navegación deben ser operables.

- **Pauta 2.1 Accesible por teclado:** Proporcionar acceso a toda la funcionalidad mediante el teclado.
- **Pauta 2.2 Tiempo suficiente:** Proporcionar a los usuarios el tiempo suficiente para leer y usar el contenido.

- **Pauta 2.3 Convulsiones:** No diseñar contenido de un modo que se sepa podría provocar ataques, espasmos o convulsiones.
- **Pauta 2.4 Navegable:** Proporcionar medios para ayudar a los usuarios a navegar, encontrar contenido y determinar dónde se encuentran.

Principio 3: Comprensible - La información y el manejo de la interfaz de usuario deben ser comprensibles.

- **Pauta 3.1 Legible:** Hacer que los contenidos textuales resulten legibles y comprensibles.
- **Pauta 3.2 Predecible:** Hacer que las páginas web aparezcan y operen de manera predecible.
- **Pauta 3.3 Entrada de datos asistida:** Ayudar a los usuarios a evitar y corregir los errores.

Principio 4: Robusto - El contenido debe ser suficientemente robusto como para ser interpretado de forma fiable por una amplia variedad de aplicaciones de usuario, incluyendo las ayudas técnicas.

- **Pauta 4.1 Compatible:** Maximizar la compatibilidad con las aplicaciones de usuario actuales y futuras, incluyendo las ayudas técnicas.

A.5. Legislación española sobre accesibilidad web

Debido a las ventajas que aporta la accesibilidad web a todos los usuarios de un sitio, y para garantizar la igualdad de oportunidades y de trato para las personas con discapacidad, en España, desde el año 2002, se han desarrollado diversas leyes para definir los distintos niveles de accesibilidad.¹⁶

En estas leyes se estipulaba que las administraciones públicas debían adoptar las medidas necesarias para que el contenido mostrado en sus páginas web pueda ser accesible para personas de edad avanzada o con discapacidad. Sin embargo, no se cumplió esta ley en el tiempo estipulado.

Así pues, en el año 2007 se volvió a legislar sobre este aspecto. En el Real Decreto 1494/2007, se establecía que las páginas públicas o con financiación pública, debían conseguir que sus sitios web fueran accesibles a partir del 31 de diciembre de 2008.

Pese a esto, la mayoría de sitios web a los que afectaba esta ley no cumplieron con lo estipulado.

¹⁶ Fuente: <http://accesibilidadweb.dlsi.ua.es/?menu=espanola>

Aun así, en España es obligatorio, por ley, que los sitios web de las administraciones públicas y de empresas privadas con un volumen económico importante o más de cien trabajadores, así como sus redes sociales, sean accesibles¹⁷. El incumplimiento de esta ley puede ser sancionado con hasta multas de un millón de euros, en función de la gravedad de la infracción.¹⁸

Un ejemplo de esta sanción se ha dado en las empresas Iberia y Vueling, que han sido multadas a abonar la cifra de treinta mil euros. En el año 2010 se denunció a Iberia por no ser su sitio web accesible, como establece la legislación vigente.¹⁹

Así pues, vemos que la accesibilidad web también está presente en las leyes de nuestro país, pero también de muchos otros, y respalda y trata de garantizar la igualdad y el acceso a los contenidos de la Web para todos.

A.6. El futuro de la accesibilidad web

En la accesibilidad web influyen muchos factores, como la forma de programar un sitio web, su diseño, las tecnologías empleadas en ella, etc.

La especificación **actual** está recogida en las **WCAG 2.0**, pero ¿cuáles son las líneas **futuras** de la accesibilidad web?

Actualmente, están en desarrollo **directrices** para el correcto uso del lenguaje **HTML5**, cuya versión definitiva se publicó en octubre de 2014. Un correcto uso de este lenguaje puede facilitar enormemente la navegación por una web, es decir, hacerla más accesible.

Desde el siguiente enlace se puede ver el documento que se está elaborando en estos momentos para convertirse en estándar:

[Técnicas para HTML5²⁰](#)

No nos olvidemos de los dispositivos móviles. Que una página sea accesible significa que puede ser utilizada en **cualquier dispositivo** preparado para acceder a Internet sin problemas. Para ello, también se están desarrollando actualmente técnicas para la accesibilidad móvil y revisando las actuales. Puede obtenerse más información desde el siguiente enlace:

¹⁷ Fuente: <http://accesibilidadweb.dlsi.ua.es/?menu=rd1494-2007>

¹⁸ Fuente: <http://accesibilidadweb.dlsi.ua.es/?menu=rd11-2013>

¹⁹ Fuente <http://www.efe.com/efe/espana/sociedad/sanidad-sanciona-a-iberia-con-30-000-euros-por-la-inaccesibilidad-de-su-web/10004-2678063>

²⁰ Técnicas para HTML5: <https://www.w3.org/WAI/GL/wiki/Techniques/HTML5>

A.7. Listado de técnicas WCAG 2.0

Como se ha mencionado en el apartado [6.2. El análisis automático](#) de este documento, antes de empezar a desarrollar el proyecto, se hizo un estudio sobre qué técnicas de accesibilidad web era posible aplicar y analizar de manera automática.

La siguiente tabla recoge esta selección, dividida en cinco columnas:

- **Nombre de la técnica:** nombre que el W3C ha dado a la técnica de accesibilidad (en inglés, para evitar pérdidas o modificaciones del significado) y con un hiperenlace a la misma en la documentación oficial.
- **Analizable:** técnicas que se pueden implementar sin problemas y cuyo resultado será el correcto.
- **Comprobación manual:** técnicas que pueden no ser analizables de manera automática, debido a distintos factores de cada web. Es posible obtener falsos resultados, por eso se indica al usuario que debe realizar una comprobación manual de las mismas.
- **No analizable:** técnicas cuyo cumplimiento no se puede analizar de manera automática, por lo que no han sido implementadas en el código del proyecto.
- **Observaciones:** comentarios del autor sobre la técnica en cuestión y sus características.

Al final de esta tabla se encuentra otra que recoge únicamente las técnicas de accesibilidad finalmente implementadas en el proyecto AccesTitan.

²¹ Borrador de técnicas para móviles: https://www.w3.org/WAI/GL/mobile-a11y-tf/wiki/Draft_Mobile_WCAG_Techniques

A.7.1. Conjunto de técnicas evaluadas

	Analizable	Comprobación manual	No analizable	Observaciones
Reglas Generales				
<u>G1: Adding a link at the top of each page that goes directly to the main content area</u>		x		No siempre es posible comprobar el contenido del link
<u>G4: Allowing the content to be paused and restarted from where it was paused</u>			x	No se puede comprobar el scrolling en la página
<u>G5: Allowing users to complete an activity without any time limit</u>			x	No se detectan acciones con tiempo límite
<u>G8: Providing a movie with extended audio descriptions</u>			x	No es posible evaluar si el contenido del audio es adecuado
<u>G9: Creating captions for live synchronized media</u>			x	

<u>G10: Creating components using a technology that supports the accessibility API features of the platforms on which the user agents will be run to expose the names and roles, allow user-settable properties to be directly set, and provide notification of changes</u>			x	
<u>G11: Creating content that blinks for less than 5 seconds</u>			x	Indetectable por código
<u>G13: Describing what will happen before a change to a form control that causes a change of context to occur is made</u>				Indetectable por código
<u>G14: Ensuring that information conveyed by color differences is also available in text</u>			x	Indetectable por código
<u>G15: Using a tool to ensure that content does not violate the general flash threshold or red flash threshold</u>			x	Indetectable por código

<u>G17: Ensuring that a contrast ratio of at least 7:1 exists between text (and images of text) and background behind the text</u>			x	
<u>G18: Ensuring that a contrast ratio of at least 4.5:1 exists between text (and images of text) and background behind the text</u>			x	
<u>G19: Ensuring that no component of the content flashes more than three times in any 1-second period</u>			x	Indetectable por código
<u>G21: Ensuring that users are not trapped in content</u>			x	Indetectable por código, requiere comprobación humana
<u>G53: Identifying the purpose of a link using link text combined with the text of the enclosing sentence</u>			x	No se puede evaluar un texto objetivamente de forma automática
<u>G54: Including a sign language interpreter in the video stream</u>			x	Indetectable por código
<u>G55: Linking to definitions</u>			x	

<u>G56: Mixing audio files so that non-speech sounds are at least 20 decibels lower than the speech audio content</u>			x	Indetectable por código
<u>G57: Ordering the content in a meaningful sequence</u>			x	Indetectable por código
<u>G58: Placing a link to the alternative for time-based media immediately next to the non-text content</u>			x	Indetectable por código
<u>G59: Placing the interactive elements in an order that follows sequences and relationships within the content</u>			x	
<u>G60: Playing a sound that turns off automatically within three seconds</u>			x	
<u>G61: Presenting repeated components in the same relative order each time they appear</u>			x	
<u>G62: Providing a glossary</u>	x			No existe etiqueta HTML estándar para hacerlo, por lo que cada página puede implementarlo de forma diferente

<u>G63: Providing a site map</u>		x		No existe etiqueta HTML estándar para hacerlo, por lo que cada página puede implementarlo de forma diferente
<u>G64: Providing a Table of Contents</u>			x	No existe etiqueta HTML estándar para hacerlo, por lo que cada página puede implementarlo de forma diferente
<u>G65: Providing a breadcrumb trail</u>		x		No existe etiqueta HTML estándar para hacerlo, por lo que cada página puede implementarlo de forma diferente
<u>G68: Providing a descriptive label that describes the purpose of live audio-only and live video-only content</u>			x	Indetectable por código
<u>G69: Providing an alternative for time based media</u>			x	
<u>G70: Providing a function to search an online dictionary</u>			x	Indetectable por código
<u>G71: Providing a help link on every Web page</u>		x		

<u>G73: Providing a long description in another location with a link to it that is immediately adjacent to the non-text content</u>			x	Indetectable por código
<u>G74: Providing a long description in text near the non-text content, with a reference to the location of the long description in the short description</u>			x	Indetectable por código
<u>G75: Providing a mechanism to postpone any updating of content</u>			x	Indetectable por código
<u>G76: Providing a mechanism to request an update of the content instead of updating automatically</u>			x	Indetectable por código
<u>G78: Providing a second, user-selectable, audio track that includes audio descriptions</u>			x	Indetectable por código
<u>G79: Providing a spoken version of the text</u>			x	Indetectable por código
<u>G80: Providing a submit button to initiate a change of context</u>			x	

<u>G81: Providing a synchronized video of the sign language interpreter that can be displayed in a different viewport or overlaid on the image by the player</u>			x	Indetectable por código
<u>G82: Providing a text alternative that identifies the purpose of the non-text content</u>			x	
<u>G83: Providing text descriptions to identify required fields that were not completed</u>		x		Cada formulario es diferente y no se pueden llenar automáticamente para esta comprobación
<u>G84: Providing a text description when the user provides information that is not in the list of allowed values</u>			x	
<u>G85: Providing a text description when user input falls outside the required format or values</u>			x	Indetectable por código
<u>G86: Providing a text summary that requires reading ability less advanced than the upper secondary education level</u>			x	

<u>G87: Providing closed captions</u>			x	Indetectable por código
<u>G88: Providing descriptive titles for Web pages</u>	x			Solo se puede comprobar si se ofrece un título largo, no si describe bien la página en cuestión
<u>G89: Providing expected data format and example</u>			x	
<u>G90: Providing keyboard-triggered event handlers</u>			x	
<u>G91: Providing link text that describes the purpose of a link</u>			x	No se puede asegurar que se describe correctamente
<u>G92: Providing long description for non-text content that serves the same purpose and presents the same information</u>			x	No se puede asegurar que se describe correctamente
<u>G93: Providing open (always visible) captions</u>			x	Indetectable por código
<u>G94: Providing short text alternative for non-text content that serves the same purpose and presents the same information as the non-text content</u>			x	Indetectable por código

<u>G95: Providing short text alternatives that provide a brief description of the non-text content</u>			x	Indetectable por código
<u>G96: Providing textual identification of items that otherwise rely only on sensory information to be understood</u>			x	Indetectable por código
<u>G97: Providing the abbreviation immediately following the expanded form</u>			x	No se puede saber si una abreviación está junto a su forma expandida
<u>G98: Providing the ability for the user to review and correct answers before submitting</u>			x	Indetectable por código
<u>G99: Providing the ability to recover deleted information</u>			x	Indetectable por código
<u>G100: Providing the accepted name or a descriptive name of the non-text content</u>			x	Indetectable por código
<u>G101: Providing the definition of a word or phrase used in an unusual or restricted way</u>			x	Indetectable por código

<u>G102: Providing the expansion or explanation of an abbreviation</u>		x		Se pueden detectar abreviaturas, pero no la expansión.
<u>G103: Providing visual illustrations, pictures, and symbols to help explain ideas, events, and processes</u>			x	
<u>G105: Saving data so that it can be used after a user re-authenticates</u>			x	No se puede hacer loggin automaticamente en todas las páginas
<u>G107: Using "activate" rather than "focus" as a trigger for changes of context</u>			x	No se puede emular un teclado con este software
<u>G108: Using markup features to expose the name and role, allow user-settable properties to be directly set, and provide notification of changes</u>			x	Indetectable por código
<u>G110: Using an instant client-side redirect</u>			x	Indetectable por código
<u>G111: Using color and pattern</u>			x	Indetectable por código
<u>G112: Using inline definitions</u>			x	Indetectable por código

<u>G115: Using semantic elements to mark up structure</u>		x		
<u>G117: Using text to convey information that is conveyed by variations in presentation of text</u>			x	Indetectable por código
<u>G120: Providing the pronunciation immediately following the word</u>			x	No se puede saber si una palabra necesita pronunciación si no se usa el tag 'lang'
<u>G121: Linking to pronunciations</u>			x	
<u>G122: Including a text cue whenever color cues are used</u>			x	Indetectable por código
<u>G123: Adding a link at the beginning of a block of repeated content to go to the end of the block</u>			x	Indetectable por código
<u>G124: Adding links at the top of the page to each area of the content</u>			x	Cada página puede crear este menú de diferente forma. Requiere de la inspección humana
<u>G125: Providing links to navigate to related Web pages</u>			x	No se puede saber si una página está relacionada con

				otra de forma automática
<u>G126: Providing a list of links to all other Web pages</u>			x	Cada página implementa esto de diferente forma, si es que lo hace
<u>G127: Identifying a Web page's relationship to a larger collection of Web pages</u>			x	Indetectable por código
<u>G128: Indicating current location within navigation bars</u>		x		Solo se puede comprobar el uso de la etiqueta nav
	x			
<u>G131: Providing descriptive labels</u>			x	
<u>G133: Providing a checkbox on the first page of a multipart form that allows users to ask for longer session time limit or no session time limit</u>			x	Indetectable por código
<u>G134: Validating Web pages</u>			x	Requiere de validadores externos

<u>G135: Using the accessibility API features of a technology to expose names and roles, to allow user-settable properties to be directly set, and to provide notification of changes</u>			x	
<u>G136: Providing a link at the beginning of a nonconforming Web page that points to a conforming alternate version</u>			x	Indetectable por código
<u>G138: Using semantic markup whenever color cues are used</u>			x	Indetectable por código
<u>G139: Creating a mechanism that allows users to jump to errors</u>			x	Indetectable por código
<u>G140: Separating information and structure from presentation to enable different presentations</u>			x	No es analizable pues hay muchas formas de dividir el contenido de las webs
<u>G141: Organizing a page using headings</u>	x			El uso de h1-h6 es necesario siempre
<u>G142: Using a technology that has commonly-available user agents that support zoom</u>			x	

<u>G143: Providing a text alternative that describes the purpose of the CAPTCHA</u>			x	Los CAPTCHAS son implementados de diversas formas y es muy difícil analizarlos
<u>G144: Ensuring that the Web Page contains another CAPTCHA serving the same purpose using a different modality</u>			x	Los CAPTCHAS son implementados de diversas formas y es muy difícil analizarlos
<u>G145: Ensuring that a contrast ratio of at least 3:1 exists between text (and images of text) and background behind the text</u>				
<u>G146: Using liquid layout</u>			x	
<u>G148: Not specifying background color, not specifying text color, and not using technology features that change those defaults</u>			x	
<u>G149: Using user interface components that are highlighted by the user agent when they receive focus</u>			x	Indetectable por código

<u>G150: Providing text based alternatives for live audio-only content</u>			x	Indetectable por código
<u>G151: Providing a link to a text transcript of a prepared statement or script if the script is followed</u>			x	Indetectable por código
<u>G152: Setting animated gif images to stop blinking after n cycles (within 5 seconds)</u>			x	Indetectable por código
<u>G153: Making the text easier to read</u>			x	Requiere de la revisión de una persona
<u>G155: Providing a checkbox in addition to a submit button</u>	x			
<u>G156: Using a technology that has commonly-available user agents that can change the foreground and background of blocks of text</u>			x	Indetectable por código
<u>G157: Incorporating a live audio captioning service into a Web page</u>			x	Indetectable por código

<u>G158: Providing an alternative for time-based media for audio-only content</u>			x	Indetectable por código
<u>G159: Providing an alternative for time-based media for video-only content</u>			x	Indetectable por código
<u>G160: Providing sign language versions of information, ideas, and processes that must be understood in order to use the content</u>			x	Indetectable por código
<u>G161: Providing a search function to help users find content</u>		x		Se pueden buscar etiquetas o palabras que hagan la función de búsqueda, pero nada asegura que esté implementado de esa manera. Por ello es necesaria una revisión manual
<u>G162: Positioning labels to maximize predictability of relationships</u>			x	Indetectable por código
<u>G163: Using standard diacritical marks that can be turned off</u>			x	

<u>G164: Providing a stated period of time after submission of the form when the order can be updated or canceled by the user</u>			x	Indetectable por código
<u>G165: Using the default focus indicator for the platform so that high visibility default focus indicators will carry over</u>			x	Indetectable por código
<u>G166: Providing audio that describes the important video content and describing it as such</u>			x	
<u>G167: Using an adjacent button to label the purpose of a field</u>			x	Indetectable por código
<u>G168: Requesting confirmation to continue with selected action</u>			x	Indetectable por código
<u>G169: Aligning text on only one side</u>			x	
<u>G170: Providing a control near the beginning of the Web page that turns off sounds that play automatically</u>			X	

<u>G171: Playing sounds only on user request</u>			X	Indetectable por código
<u>G172: Providing a mechanism to remove full justification of text</u>			X	Indetectable por código
<u>G173: Providing a version of a movie with audio descriptions</u>			X	Indetectable por código
<u>G174: Providing a control with a sufficient contrast ratio that allows users to switch to a presentation that uses sufficient contrast</u>			X	Indetectable por código
<u>G175: Providing a multi color selection tool on the page for foreground and background colors</u>			X	Indetectable por código
<u>G176: Keeping the flashing area small enough</u>			X	Indetectable por código
<u>G177: Providing suggested correction text</u>			X	Indetectable por código
<u>G178: Providing controls on the Web page that allow users to incrementally change the size of all text on the page up to 200 percent</u>		X		

<u>G179: Ensuring that there is no loss of content or functionality when the text resizes and text containers do not resize</u>			X	
<u>G180: Providing the user with a means to set the time limit to 10 times the default time limit</u>			X	
<u>G181: Encoding user data as hidden or encrypted data in a re-authorization page</u>			x	Indetectable por código
<u>G182: Ensuring that additional visual cues are available when text color differences are used to convey information</u>			x	Indetectable por código
<u>G183: Using a contrast ratio of 3:1 with surrounding text and providing additional visual cues on focus for links or controls where color alone is used to identify them</u>			x	

<u>G184: Providing text instructions at the beginning of a form or set of fields that describes the necessary input</u>			x	
<u>G185: Linking to all of the pages on the site from the home page</u>			x	No se puede conocer la totalidad de páginas de un sitio de manera automática
<u>G186: Using a control in the Web page that stops moving, blinking, or auto-updating content</u>			x	Indetectable por código
<u>G187: Using a technology to include blinking content that can be turned off via the user agent</u>			x	
<u>G188: Providing a button on the page to increase line spaces and paragraph spaces</u>			x	
<u>G189: Providing a control near the beginning of the Web page that changes the link text</u>			x	Indetectable por código

<u>G190: Providing a link adjacent to or associated with a non-conforming object that links to a conforming alternate version</u>			x	Indetectable por código
<u>G191: Providing a link, button, or other mechanism that reloads the page without any blinking content</u>			x	Indetectable por código
<u>G192: Fully conforming to specifications</u>			x	Requiere de validadores externos y expertos
<u>G193: Providing help by an assistant in the Web page</u>			x	Indetectable por código
<u>G194: Providing spell checking and suggestions for text input</u>			x	Indetectable por código
<u>G195: Using an author-supplied, highly visible focus indicator</u>			x	Indetectable por código
<u>G196: Using a text alternative on one item within a group of images that describes all items in the group</u>			x	No se puede detectar un grupo de forma eficaz

<u>G197: Using labels, names, and text alternatives consistently for content that has the same functionality</u>			x	
<u>G198: Providing a way for the user to turn the time limit off</u>			x	Indetectable por código
<u>G199: Providing success feedback when data is submitted successfully</u>			x	Indetectable por código
Técnicas para HTML				
<u>H2: Combining adjacent image and text links for the same resource</u>			x	No se puede saber el contenido de un link de manera automática
<u>H4: Creating a logical tab order through links, form controls, and objects</u>		x		Se puede controlar que se use tabindex, pero no un orden lógico
<u>H24: Providing text alternatives for the area elements of image maps</u>	x			
<u>H25: Providing a title using the title element</u>	x			

<u>H27: Providing text and non-text alternatives for object</u>	x			
<u>H28: Providing definitions for abbreviations by using the abbr and acronym elements</u>	x			
<u>H30: Providing link text that describes the purpose of a link for anchor elements</u>			x	
<u>H32: Providing submit buttons</u>	x			
<u>H33: Supplementing link text with the title attribute</u>		x		
<u>H34: Using a Unicode right-to-left mark (RLM) or left-to-right mark (LRM) to mix text direction inline</u>			x	
<u>H35: Providing text alternatives on applet elements</u>	x			
<u>H36: Using alt attributes on images used as submit buttons</u>	x			

<u>H37: Using alt attributes on img elements</u>	x			
<u>H39: Using caption elements to associate data table captions with data tables</u>			x	No se puede saber si la tabla es de contenido o de estilo
<u>H40: Using definition lists</u>			x	Cada página puede implementar esto de diferente forma
<u>H42: Using h1-h6 to identify headings</u>	x			
<u>H43: Using id and headers attributes to associate data cells with header cells in data tables</u>			x	
<u>H44: Using label elements to associate text labels with form controls</u>			x	No se puede saber de forma automática si la etiqueta representa información útil o adecuada
<u>H45: Using longdesc</u>	x			
<u>H46: Using noembed with embed</u>	x			
<u>H48: Using ol, ul and dl for lists</u>			x	

<u>H49: Using semantic markup to mark emphasized or special text</u>	x			
<u>H50: Using structural elements to group links</u>			x	No se pueden reconocer grupos de links pues estos se pueden implementar de muchas formas
<u>H51: Using table markup to present tabular information</u>			x	No se puede saber qué información ha de ser presentada así
<u>H53: Using the body of the object element</u>	x			
<u>H54: Using the dfn element to identify the defining instance of a word</u>		x		
<u>H56: Using the dir attribute on an inline element to resolve problems with nested directional runs</u>			x	
<u>H57: Using language attributes on the html element</u>	x			

<u>H58: Using language attributes to identify changes in the human language</u>			x	No se puede saber de forma automática cuándo se ha realizado este cambio
<u>H59: Using the link element and navigation tools</u>	x			
<u>H60: Using the link element to link to a glossary</u>		x		
<u>H62: Using the ruby element</u>		x		
<u>H63: Using the scope attribute to associate header cells and data cells in data tables</u>	x			
<u>H64: Using the title attribute of the frame and iframe elements</u>	x			
<u>H65: Using the title attribute to identify form controls when the label element cannot be used</u>	x			
<u>H67: Using null alt text and no title attribute on img elements for images that AT should ignore</u>	x			

<u>H69: Providing heading elements at the beginning of each section of content</u>			x	No se puede saber qué parte es una sección, ya que no todas las páginas las dividen con la etiqueta section
<u>H70: Using frame elements to group blocks of repeated material</u>		x		Se puede comprobar el uso de frame, pero no de contenido repetido
<u>H71: Providing a description for groups of form controls using fieldset and legend elements</u>			x	
<u>H73: Using the summary attribute of the table element to give an overview of data tables</u>			x	No se puede saber si se da una vista previa o resumen
<u>H74: Ensuring that opening and closing tags are used according to specification</u>			x	
<u>H75: Ensuring that Web pages are well-formed</u>			x	Indetectable mediante código
<u>H76: Using meta refresh to create an instant client-side redirect</u>	x			

<u>H77: Identifying the purpose of a link using link text combined with its enclosing list item</u>			x	No se puede analizar si el contenido del texto refleja el del link
<u>H78: Identifying the purpose of a link using link text combined with its enclosing paragraph</u>			x	No se puede analizar si el contenido del texto refleja el del link
<u>H79: Identifying the purpose of a link using link text combined with its enclosing table cell and associated table headings</u>			x	No se puede analizar si el contenido del texto refleja el del link
<u>H80: Identifying the purpose of a link using link text combined with the preceding heading element</u>			x	No se puede analizar si el contenido del texto refleja el del link
<u>H81: Identifying the purpose of a link in a nested list using link text combined with the parent list item under which the list is nested</u>			x	No se puede analizar si el contenido del texto refleja el del link
<u>H83: Using the target attribute to open a new window on user request and indicating this in link text</u>		x		

<u>H84: Using a button with a select element to perform an action</u>		x		
<u>H85: Using OPTGROUP to group OPTION elements inside a SELECT</u>		x		
<u>H86: Providing text alternatives for ASCII art, emoticons, and leetspeak</u>			x	
<u>H87: Not interfering with the user agent's reflow of text as the viewing window is narrowed</u>			x	
<u>H88: Using HTML according to spec</u>			x	Requiere de validadores externos
<u>H89: Using the title attribute to provide context-sensitive help</u>		x		
<u>H90: Indicating required form controls</u>		x		
<u>H91: Using HTML form controls and links</u>			x	
Técnicas de Fracaso general				
<u>F1: Failure of Success Criterion 1.3.2 due to changing the meaning of content by</u>			x	

<u>positioning information with CSS</u>				
<u>F2: Failure of Success Criterion 1.3.1 due to using changes in text presentation to convey information without using the appropriate markup or text</u>			x	
<u>F3: Failure of Success Criterion 1.1.1 due to using CSS to include images that convey important information</u>			x	
<u>F4: Failure of Success Criterion 2.2.2 due to using text-decoration:blink without a mechanism to stop it in less than five seconds</u>			x	
<u>F7: Failure of Success Criterion 2.2.2 due to an object or applet, such as Java or Flash, that has blinking content without a mechanism to pause the content that blinks for more than five seconds</u>			x	
<u>F8: Failure of Success Criterion 1.2.2 due to captions omitting some dialogue or important sound effects</u>			x	

<u>F9: Failure of Success</u> <u>Criterion 3.2.5 due to changing the context when the user removes focus from a form element</u>			x	
<u>F10: Failure of Success</u> <u>Criterion 2.1.2 and Conformance</u> <u>Requirement 5 due to combining multiple content formats in a way that traps users inside one format type</u>			x	
<u>F12: Failure of Success</u> <u>Criterion 2.2.5 due to having a session time limit without a mechanism for saving user's input and re-establishing that information upon re-authentication</u>			x	
<u>F13: Failure of Success</u> <u>Criterion 1.1.1 and 1.4.1 due to having a text alternative that does not include information that is conveyed by color differences in the image</u>			x	
<u>F14: Failure of Success</u> <u>Criterion 1.3.3 due to identifying content only by its shape or location</u>			x	

<u>F15: Failure of Success Criterion 4.1.2 due to implementing custom controls that do not use an accessibility API for the technology, or do so incompletely</u>			x	
<u>F16: Failure of Success Criterion 2.2.2 due to including scrolling content where movement is not essential to the activity without also including a mechanism to pause and restart the content</u>			x	
<u>F19: Failure of Conformance Requirement 1 due to not providing a method for the user to find the alternative conforming version of a non-conforming Web page</u>			x	
<u>F20: Failure of Success Criterion 1.1.1 and 4.1.2 due to not updating text alternatives when changes to non-text content occur</u>			x	
<u>F22: Failure of Success Criterion 3.2.5 due to opening windows that are not requested by the user</u>			x	

<u>F23: Failure of 1.4.2 due to playing a sound longer than 3 seconds where there is no mechanism to turn it off</u>			x	
<u>F24: Failure of Success Criterion 1.4.3, 1.4.6 and 1.4.8 due to specifying foreground colors without specifying background colors or vice versa</u>			x	
<u>F25: Failure of Success Criterion 2.4.2 due to the title of a Web page not identifying the contents</u>	x			
<u>F26: Failure of Success Criterion 1.3.3 due to using a graphical symbol alone to convey information</u>			x	
<u>F30: Failure of Success Criterion 1.1.1 and 1.2.1 due to using text alternatives that are not alternatives (e.g., filenames or placeholder text)</u>			x	
<u>F31: Failure of Success Criterion 3.2.4 due to using two different labels for the same function on different Web pages within a set of Web pages</u>			x	

<u>F32: Failure of Success</u> <u>Criterion 1.3.2 due to using white space characters to control spacing within a word</u>			x	
<u>F33: Failure of Success</u> <u>Criterion 1.3.1 and 1.3.2 due to using white space characters to create multiple columns in plain text content</u>			x	
<u>F34: Failure of Success</u> <u>Criterion 1.3.1 and 1.3.2 due to using white space characters to format tables in plain text content</u>			x	
<u>F36: Failure of Success</u> <u>Criterion 3.2.2 due to automatically submitting a form and presenting new content without prior warning when the last field in the form is given a value</u>			x	
<u>F37: Failure of Success</u> <u>Criterion 3.2.2 due to launching a new window without prior warning when the selection of a radio button, check box or select list is changed</u>			x	
<u>F38: Failure of Success</u> <u>Criterion 1.1.1 due to</u>		x		

<u>not marking up decorative images in HTML in a way that allows assistive technology to ignore them</u>				
<u>F39: Failure of Success Criterion 1.1.1 due to providing a text alternative that is not null (e.g., alt="spacer" or alt="image") for images that should be ignored by assistive technology</u>			x	
<u>F40: Failure of Success Criterion 2.2.1 and 2.2.4 due to using meta redirect with a time limit</u>			x	
<u>F41: Failure of Success Criterion 2.2.1, 2.2.4, and 3.2.5 due to using meta refresh to reload the page</u>	x			
<u>F42: Failure of Success Criteria 1.3.1, 2.1.1, 2.1.3, or 4.1.2 when emulating links</u>			x	
<u>F43: Failure of Success Criterion 1.3.1 due to using structural markup in a way that does not represent relationships in the content</u>			x	

<u>F44: Failure of Success</u> <u>Criterion 2.4.3 due to using tabindex to create a tab order that does not preserve meaning and operability</u>			x	
<u>F46: Failure of Success</u> <u>Criterion 1.3.1 due to using th elements, caption elements, or non-empty summary attributes in layout tables</u>			x	
<u>F47: Failure of Success</u> <u>Criterion 2.2.2 due to using the blink element</u>	x			
<u>F48: Failure of Success</u> <u>Criterion 1.3.1 due to using the pre element to markup tabular information</u>			x	
<u>F49: Failure of Success</u> <u>Criterion 1.3.2 due to using an HTML layout table that does not make sense when linearized</u>			x	
<u>F50: Failure of Success</u> <u>Criterion 2.2.2 due to a script that causes a blink effect without a mechanism to stop the blinking at 5 seconds or less</u>			x	
<u>F52: Failure of Success</u> <u>Criterion 3.2.1 and 3.2.5 due to opening a</u>			x	

<u>new window as soon as a new page is loaded</u>				
<u>F54: Failure of Success Criterion 2.1.1 due to using only pointing-device-specific event handlers (including gesture) for a function</u>			x	
<u>F55: Failure of Success Criteria 2.1.1, 2.4.7, and 3.2.1 due to using script to remove focus when focus is received</u>			x	
<u>F58: Failure of Success Criterion 2.2.1 due to using server-side techniques to automatically redirect pages after a time-out</u>			x	
<u>F59: Failure of Success Criterion 4.1.2 due to using script to make div or span a user interface control in HTML without providing a role for the control</u>			x	
<u>F60: Failure of Success Criterion 3.2.5 due to launching a new window when a user enters text into an input field</u>			x	
<u>F61: Failure of Success Criterion 3.2.5 due to complete change of main content through</u>			x	

<u>an automatic update</u> <u>that the user cannot</u> <u>disable from within the</u> <u>content</u>				
<u>F63: Failure of Success</u> <u>Criterion 2.4.4 due to</u> <u>providing link context</u> <u>only in content that is</u> <u>not related to the link</u>			x	
<u>F65: Failure of Success</u> <u>Criterion 1.1.1 due to</u> <u>omitting the alt attribute</u> <u>or text alternative on</u> <u>img elements, area</u> <u>elements, and input</u> <u>elements of type</u> <u>"image"</u>			x	
<u>F66: Failure of Success</u> <u>Criterion 3.2.3 due to</u> <u>presenting navigation</u> <u>links in a different</u> <u>relative order on</u> <u>different pages</u>			x	
<u>F67: Failure of Success</u> <u>Criterion 1.1.1 and</u> <u>1.2.1 due to providing</u> <u>long descriptions for</u> <u>non-text content that</u> <u>does not serve the same</u> <u>purpose or does not</u> <u>present the same</u> <u>information</u>			x	
<u>F68: Failure of Success</u> <u>Criterion 4.1.2 due to a</u> <u>user interface control</u> <u>not having a</u>			x	

<u>programmatically determined name</u>				
<u>F69: Failure of Success Criterion 1.4.4 when resizing visually rendered text up to 200 percent causes the text, image or controls to be clipped, truncated or obscured</u>			x	
<u>F70: Failure of Success Criterion 4.1.1 due to incorrect use of start and end tags or attribute markup</u>			x	
<u>F71: Failure of Success Criterion 1.1.1 due to using text look-alikes to represent text without providing a text alternative</u>			x	
<u>F72: Failure of Success Criterion 1.1.1 due to using ASCII art without providing a text alternative</u>			x	
<u>F73: Failure of Success Criterion 1.4.1 due to creating links that are not visually evident without color vision</u>			x	
<u>F74: Failure of Success Criterion 1.2.2 and 1.2.8 due to not labeling a synchronized media</u>			x	

<u>alternative to text as an alternative</u>				
<u>F75: Failure of Success Criterion 1.2.2 by providing synchronized media without captions when the synchronized media presents more information than is presented on the page</u>			x	
<u>F77: Failure of Success Criterion 4.1.1 due to duplicate values of type ID</u>			x	
<u>F78: Failure of Success Criterion 2.4.7 due to styling element outlines and borders in a way that removes or renders non-visible the visual focus indicator</u>			x	
<u>F79: Failure of Success Criterion 4.1.2 due to the focus state of a user interface component not being programmatically determinable or no notification of change of focus state available</u>			x	
<u>F80: Failure of Success Criterion 1.4.4 when text-based form controls do not resize when visually rendered text is resized up to 200%</u>			x	

<u>F81: Failure of Success</u> <u>Criterion 1.4.1 due to identifying required or error fields using color differences only</u>			x	
<u>F82: Failure of Success</u> <u>Criterion 3.3.2 by visually formatting a set of phone number fields but not including a text label</u>			x	
<u>F83: Failure of Success</u> <u>Criterion 1.4.3 and 1.4.6 due to using background images that do not provide sufficient contrast with foreground text (or images of text)</u>			x	
<u>F84: Failure of Success</u> <u>Criterion 2.4.9 due to using a non-specific link such as "click here" or "more" without a mechanism to change the link text to specific text.</u>			x	
<u>F85: Failure of Success</u> <u>Criterion 2.4.3 due to using dialogs or menus that are not adjacent to their trigger control in the sequential navigation order</u>			x	
<u>F86: Failure of Success</u> <u>Criterion 4.1.2 due to</u>			x	

<u>not providing names for each part of a multi-part form field, such as a US telephone number</u>				
<u>F87: Failure of Success Criterion 1.3.1 due to inserting non-decorative content by using :before and :after pseudo-elements and the 'content' property in CSS</u>			x	
<u>F88: Failure of Success Criterion 1.4.8 due to using text that is justified (aligned to both the left and the right margins)</u>			x	
<u>F89: Failure of Success Criteria 2.4.4, 2.4.9 and 4.1.2 due to not providing an accessible name for an image which is the only content in a link</u>			x	
<u>F90: Failure of Success Criterion 1.3.1 for incorrectly associating table headers and content via the headers and id attributes</u>			x	
<u>F91: Failure of Success Criterion 1.3.1 for not correctly marking up table headers</u>			x	

<u>F92: Failure of Success Criterion 1.3.1 due to the use of role presentation on content which conveys semantic information</u>			x	
<u>F93: Failure of Success Criterion 1.4.2 for absence of a way to pause or stop an HTML5 media element that autoplays</u>			x	

A.7.2. Tabla de técnicas implementadas en AccesTitan

	Analizable	Comprobación manual	No analizable	Observaciones
Reglas Generales				
<u>G1: Adding a link at the top of each page that goes directly to the main content area</u>		x		No siempre es posible comprobar el contenido del link
<u>G62: Providing a glossary</u>		x		No existe etiqueta HTML estándar para hacerlo, por lo que cada página puede implementarlo de forma diferente
<u>G63: Providing a site map</u>		x		No existe etiqueta HTML estándar para hacerlo, por lo que cada

				página puede implementarlo de forma diferente
<u>G65: Providing a breadcrumb trail</u>		x		No existe etiqueta HTML estándar para hacerlo, por lo que cada página puede implementarlo de forma diferente
<u>G71: Providing a help link on every Web page</u>		x		
<u>G88: Providing descriptive titles for Web pages</u>	x			Solo se puede comprobar si se ofrece un título largo, no si describe bien la página en cuestión
<u>G102: Providing the expansion or explanation of an abbreviation</u>		x		Se pueden detectar abreviaturas, pero no la expansión.
<u>G115: Using semantic elements to mark up structure</u>		x		
<u>G128: Indicating current location within navigation bars</u>		x		Solo se puede comprobar el uso de la etiqueta nav
<u>G141: Organizing a page using headings</u>	x			El uso de h1-h6 es necesario siempre
<u>G155: Providing a checkbox in addition to a submit button</u>	x			

<u>G161: Providing a search function to help users find content</u>		x		Se pueden buscar etiquetas o palabras que hagan la función de búsqueda, pero nada asegura que esté implementado de esa manera. Por ello es necesaria una revisión manual
Técnicas para HTML				
<u>H4: Creating a logical tab order through links, form controls, and objects</u>		x		Se puede controlar que se use tabindex, pero no un orden lógico
<u>H25: Providing a title using the title element</u>	x			
<u>H27: Providing text and non-text alternatives for object</u>	x			
<u>H28: Providing definitions for abbreviations by using the abbr and acronym elements</u>	x			

<u>H32: Providing submit buttons</u>	x			
<u>H33: Supplementing link text with the title attribute</u>		x		
<u>H35: Providing text alternatives on applet elements</u>	x			
<u>H36: Using alt attributes on images used as submit buttons</u>	x			
<u>H37: Using alt attributes on img elements</u>	x			
<u>H42: Using h1-h6 to identify headings</u>	x			
<u>H45: Using longdesc</u>	x			
<u>H46: Using noembed with embed</u>	x			
<u>H49: Using semantic markup to mark emphasized or special text</u>	x			

<u>H53: Using the body of the object element</u>	x			
<u>H57: Using language attributes on the html element</u>	x			
<u>H64: Using the title attribute of the frame and iframe elements</u>	x			
<u>H67: Using null alt text and no title attribute on img elements for images that AT should ignore</u>	x			
<u>H76: Using meta refresh to create an instant client-side redirect</u>	x			
<u>H85: Using OPTGROUP to group OPTION elements inside a SELECT</u>		x		
Técnicas de Fracaso general				
<u>F25: Failure of Success Criterion 2.4.2 due to the title of a Web page not</u>	x			Relacionada con las técnicas H25 y G88

<u>identifying the contents</u>				
<u>F47: Failure of Success Criterion 2.2.2 due to using the blink element</u>	x			