UNIVERSIDAD EAFIT

# UNA ESTRATEGIA PARA FACILITAR LA IMPLEMENTACIÓN DE LA ACCESIBILIDAD WEB EN DOCUMENTOS HTML, CONFORME A LAS WCAG 2.2.

VANESSA MARELY ARISTIZABAL ANGEL

Trabajo de grado

Asesores

Daniel Correa Botero

Paola Andrea Vallejo

UNIVERSIDAD EAFIT
ESCUELA DE CIENCIAS APLICADAS E INGENIERÍA
MAESTRÍA EN INGENIERÍA
MEDELLÍN
2025

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

***Resumen***

La accesibilidad web es fundamental para garantizar el acceso a los contenidos digitales, especialmente para personas con discapacidad. Aunque existen estándares internacionales como las *Web Content Accessibility Guidelines* (*WCAG*), sigue siendo poco común que los desarrolladores las implementen correctamente. Diversos estudios muestran que una parte importante de los desarrolladores no integra la accesibilidad de manera consistente en sus proyectos, lo que mantiene barreras en la web.

Este trabajo propone una estrategia que facilita la implementación de la accesibilidad web en documentos *HTML*, de acuerdo con las *WCAG* 2.2. La propuesta identifica y relaciona etiquetas y atributos *HTML* que permiten cumplir con los criterios de éxito de accesibilidad, organizando esta información de manera estructurada para guiar a los desarrolladores en su correcta aplicación.

Una de las principales ventajas de esta estrategia es que permite validar las etiquetas del documento *HTML* durante el desarrollo, mediante una extensión para el editor *Visual Studio Code*, que el desarrollador puede activar cuando lo necesite para facilitar la detección de errores de accesibilidad.

***Palabras clave: Accesibilidad web, Documento HTML, Normas de accesibilidad, Desarrollo web accesible, WCAG 2.2, W3C/WAI***

***Abstract***

Web accessibility is essential to ensure access to digital content, especially for people with disabilities. Although international standards such as the Web Content Accessibility Guidelines (*WCAG*) exist, it is still uncommon for developers to implement them correctly. Various studies show that a significant number of developers do not consistently integrate accessibility into their projects, which continues to create barriers on the web.

This work proposes a strategy to facilitate the implementation of web accessibility in *HTML* documents, aligned with *WCAG* 2.2. The strategy identifies and maps *HTML* tags and attributes to the corresponding accessibility success criteria, organizing this information in a structured way to guide developers in proper implementation.

A key benefit of this strategy is its capacity to enable validation of *HTML* elements during development, facilitated by an extension for the *Visual Studio Code* editor. This tool can be activated by the developer as needed, significantly aiding in the detection of accessibility errors.

***Keywords: Web accessibility, HTML document, Accessibility standards, Accessible web development, WCAG 2.2, W3C/WAI***

# 1. INTRODUCCIÓN

La accesibilidad web constituye un componente esencial en el desarrollo de aplicaciones web, al permitir que personas con diferentes tipos de discapacidad puedan navegar, interactuar y acceder a los recursos disponibles en línea. De acuerdo con el Consorcio *World Wide Web* (*W3C*), la accesibilidad también beneficia a personas mayores, usuarios con limitaciones temporales o situacionales, y a quienes acceden desde dispositivos con conexiones inestables o de baja velocidad [1]. Las Pautas de Accesibilidad para el Contenido Web (*WCAG*) 2.2 se han consolidado como un estándar internacional que orienta el desarrollo de contenido digital inclusivo [2]. Sin embargo, persiste una brecha significativa entre la normativa técnica y su aplicación práctica, lo que motiva investigaciones continuas y la propuesta de nuevas formas de trabajo, incluso a nivel de administraciones públicas nacionales, para abordar estos desafíos [20].

El informe *WebAIM Million Report* 2025 [3] identificó que el 94.8 % de los sitios web analizados presentan al menos un error de accesibilidad, siendo los más comunes: contraste insuficiente entre texto y fondo (79.1 %), imágenes sin texto alternativo (55.5 %) y formularios sin etiquetas adecuadas (48.2 %) [3]. Este patrón también ha sido documentado en contextos institucionales, como portales gubernamentales y educativos, tanto en Europa como en América Latina [22], [32].

A pesar de la existencia de versiones actualizadas como las *WCAG* 2.2 [2], su adopción sigue siendo limitada. Según *Waltner* [31] y *York* [17], apenas el 26 % de los ingenieros consideran siempre la accesibilidad en su código. Además, un informe de 2023 reveló que solo el 21 % de los sitios europeos cumplen plenamente con los estándares actuales de accesibilidad. Por su parte, *Abuaddous et al.* [4] clasifican los principales retos en tres categorías: la interpretación de las *WCAG* 2.2, los desafíos técnicos en su implementación durante el diseño, y la falta de mecanismos eficaces de evaluación. Estas dificultades suelen acentuarse en contextos donde la accesibilidad solo se contempla ante exigencias legales, lo que deriva en prácticas poco sostenibles según *Halpin* [5], [21]. En el caso de América Latina, *Pinedo et al.* [32] evidenciaron que solo uno de los sitios evaluados de instituciones públicas y educativas cumplía con el nivel AAA de accesibilidad, lo que subraya la urgencia de promover una cultura de desarrollo inclusiva, con mayores competencias técnicas, en la región.

Ante este panorama, el presente trabajo propone una estrategia para facilitar la implementación de la accesibilidad web en documentos *HTML*, en conformidad a las *WCAG* 2.2. Estas están organizadas en

principios, que contienen unas pautas, cada una de las cuales se acompaña de criterios de éxito, que permiten verificar su cumplimiento y para evaluar contienen varios niveles de conformidad de la accesibilidad web [9]. La estrategia propuesta en este trabajo, se fundamenta en la identificación de etiquetas clave del código *HTML*, su relación con los criterios de éxito de las *WCAG* 2.2 y la aplicación de técnicas suficientes. Como resultado de este análisis, se ha elaborado una estrategia que organiza, de manera estructurada, la correspondencia entre los criterios de éxito aplicables, las etiquetas *HTML* involucradas y las técnicas suficientes recomendadas. Esta surge como parte de la propuesta desarrollada en este trabajo y tiene como objetivo servir de apoyo práctico para los desarrolladores, explicando cómo implementar cada criterio directamente en el código *HTML*. De este modo, facilita la revisión y mejora del código durante su creación o edición, antes de que el sitio web sea publicado.

Finalmente, la estructura de este documento se organiza de la siguiente manera: La Sección 2 "Planteamiento del problema y justificación", se aborda el planteamiento del problema y la justificación, así como la pregunta de investigación. En la Sección 3 "Objetivos", contiene el objetivo general y los objetivos específicos. La Sección 4 "Marco Teórico" desarrolla el marco teórico, con énfasis en los principios de las *WCAG* 2.2 y su jerarquía. La Sección 5 "Metodología", expone la metodología aplicada para el desarrollo de este trabajo. La Sección 6 "Diseño y desarrollo de la propuesta", presenta la estrategia para facilitar la implementación de la accesibilidad web. La Sección 7 "Aplicación técnica de la propuesta" presenta la implementación de la propuesta como una extensión de *Visual Studio Code* y la evaluación de la efectividad de la estrategia mediante su aplicación en documentos *HTML* de prueba y su comparación con otras herramientas de validación. La Sección 8 "Resultados" presenta los resultados obtenidos del análisis de la evaluación descrita en la sección anterior, y la Sección 9 "Conclusiones" expone las conclusiones del trabajo.

## 2. PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN

La accesibilidad web es esencial para desarrollar plataformas digitales permitiendo que todas las personas, incluyendo a aquellos con alguna discapacidad, encuentren facilidades para acceder y hacer uso del contenido en línea. No obstante, su implementación sigue enfrentando barreras importantes, tanto a nivel técnico para los desarrolladores, como en las organizaciones que producen contenidos digitales. Uno de los principales problemas identificados en la literatura es la distancia que existe entre lo que establecen las *WCAG* 2.2 y cómo se aplican en la práctica al crear sitios web [22], [26]. Aunque las *WCAG* 2.2 existen desde hace más de veinte años, varios estudios han demostrado que todavía no son adoptadas de manera consistente por instituciones públicas y educativas, especialmente en América Latina [23], [25], [27].

El desconocimiento sobre accesibilidad web, la escasa formación específica en el uso adecuado del código *HTML* y la falta de integración de las *WCAG* 2.2 han sido identificados como factores limitantes [18], [28]. *Abuaddous et al.* [4] explican que los desafíos se pueden agrupar en tres grandes categorías: 1) problemas para entender las *WCAG* 2.2, porque algunas de sus indicaciones son confusas o difíciles de interpretar; 2) desafíos que surgen al crear sitios web, por la falta de conocimientos o recursos; y 3) problemas en la fase de evaluación, porque las herramientas automáticas y los expertos no siempre coinciden en la identificación de los errores. Por ejemplo, estudios han encontrado que aproximadamente el 50 % de los evaluadores no logran ponerse de acuerdo sobre las inconsistencias de accesibilidad en los sitios web; a ello se suma que el 20 % de los errores reportados son falsos positivos y un 32 % pasan desapercibidos [4]. Comprender los factores que impulsan o dificultan la adopción de la accesibilidad es crucial; estudios como el de *Yesilada et al.* [24] analizan estos elementos, identificando que abarcan desde la concienciación y la legislación hasta la disponibilidad de herramientas y recursos adecuados para los desarrolladores.

Un problema concreto es que los desarrolladores no siempre saben cómo se relacionan las etiquetas y atributos del *HTML* (las partes del código que estructuran la página) y las especificaciones propuestas en las *WCAG* 2.2 [2]. Esto hace difícil que puedan comprobar, mientras escriben su código, si realmente están cumpliendo con esos requisitos [2], [19]. Además, la mayoría de las herramientas actuales están pensadas para revisar la accesibilidad una vez que el sitio web ya está terminado, pero no ayudan a los desarrolladores durante la creación o edición de la página. Algunos estudios recientes han explorado formas de resolver este problema. Por ejemplo, el trabajo *From Code to Compliance* [33],

muestra cómo los modelos de lenguaje como *ChatGPT* [37] pueden ayudar a escribir y revisar código accesible, ofreciendo recomendaciones que explican cómo corregir errores según las *WCAG* 2.2. *Calì et al.* [34] desarrollaron en 2024 una extensión para *Visual Studio Code* [36] (un editor de código muy utilizado) que permite validar automáticamente ciertas reglas de accesibilidad mientras se trabaja con *HTML*, ayudando a los desarrolladores a detectar problemas mientras editan el código

A diferencia de estas herramientas, que se centran en realizar validaciones automáticas, la estrategia presentada en este trabajo estructura la relación entre los criterios de éxito de las *WCAG* 2.2 y las etiquetas y atributos del *HTML* que ayudan a cumplirlos. De esta manera, no solo permite la verificación, sino que también ayuda al desarrollador a aprender y aplicar buenas prácticas de accesibilidad web.

Este trabajo se centra específicamente en los documentos *HTML*, entendidos como los archivos que contienen la estructura básica de las páginas web utilizando etiquetas (como por ejemplo: la etiqueta <*img*> para imágenes o <*a*> para enlaces) y atributos (como *alt* o *title*). La estrategia propuesta busca ayudar a los desarrolladores a aplicar las *WCAG* 2.2 directamente sobre este código, mientras están construyendo o editando sus páginas web, y no solo al final, cuando ya se ha terminado el sitio y se realizan auditorías. Al facilitar la aplicación de las *WCAG 2.2*, se espera que este trabajo impulse el desarrollo de sitios web que se alineen con los estándares internacionales y sean accesibles para todas las personas.

En este trabajo se aborda la siguiente pregunta de investigación:

¿Cómo facilitar la implementación de la accesibilidad web en documentos *HTML,* conforme a las *WCAG* 2.2?

# 3. OBJETIVOS

## 3.1 OBJETIVO GENERAL

Definir una estrategia que facilite la implementación de la accesibilidad web en documentos *HTML*, conforme a los criterios de éxito establecidos por las *WCAG* 2.2.

## 3.2 OBJETIVOS ESPECÍFICOS

- Identificar las barreras técnicas y los errores comunes de accesibilidad que obstaculizan la implementación de las *WCAG 2.2* en documentos *HTML*.
- Seleccionar los criterios de éxito definidos por las *WCAG* 2.2, directamente aplicables a las etiquetas y atributos de los documentos *HTML*.
- Diseñar una estrategia que mapee los criterios de éxito de las *WCAG* 2.2 con el uso de etiquetas y atributos *HTML*, como reglas para facilitar la aplicación de la accesibilidad web.
- Implementar la estrategia, como una extensión para el editor de *Visual Studio Code*, que permite revisar las reglas diseñadas de la estrategia durante la edición de los documentos *HTML* para detectar errores.
- Probar la extensión implementada para detectar errores de accesibilidad en una muestra de documentos *HTML*, comparando su desempeño con herramientas de referencia y enfoques basados en Inteligencia Artificial.

# 4. MARCO TEÓRICO

El presente marco teórico establece los fundamentos conceptuales, normativos y prácticos de la accesibilidad web, abordando las *WCAG* 2.2, su relación con el desarrollo web y las estrategias para su evaluación. La accesibilidad web representa una condición esencial para que todas las personas, incluidas aquellas con discapacidades, puedan interactuar con tecnologías digitales sin barreras [1].

## 4.1 ACCESIBILIDAD WEB

La accesibilidad web se define como la capacidad de los sitios web, herramientas y tecnologías digitales para ser utilizados por todas las personas, incluidas aquellas con discapacidades [1]. Este concepto incluye la creación de contenido que pueda ser percibido, comprendido, navegado e interactuado por una diversidad de usuarios sin barreras. Según el Consorcio *World Wide Web*, la accesibilidad web no solo beneficia a personas con discapacidades permanentes, sino también a personas mayores, individuos con limitaciones temporales y usuarios en entornos con conectividad limitada [3].

En el contexto iberoamericano, estudios como el de *Serrano Mascaraque et al.* [22] y *Campoverde Molina* [25] han señalado que, aunque existen esfuerzos normativos y técnicos en marcha, la aplicación práctica de la accesibilidad web en instituciones públicas y educativas sigue siendo limitada. Esta situación evidencia la necesidad de fortalecer los marcos de referencia que orientan a los desarrolladores en el cumplimiento de criterios tanto a nivel técnico como semántico.

Este trabajo se centra en la accesibilidad web como concepto primordial, y con base en ello, plantea una estrategia para facilitar la implementación de este principio en documentos *HTML*. La definición sirve como marco de referencia para identificar barreras y proponer soluciones técnicas que respondan a estas necesidades.

## 4.2 PAUTAS DE ACCESIBILIDAD PARA EL CONTENIDO WEB (*WCAG* 2.2)

Desarrolladas por el *W3C*, las *WCAG* 2.2, constituyen un estándar internacional [2] que orienta la creación de contenido web inclusivo, asegurando su accesibilidad a individuos con diversas discapacidades. Las *WCAG* 2.2 se componen de cuatro principios fundamentales: perceptible, operable, comprensible y robusto (principios *P.O.U.R.*) [2], [4].

Las *WCAG* 2.2 son relevantes en el desarrollo de este trabajo, ya que forman la base de la creación de esta estrategia, cuyo objetivo es facilitar la implementación de la accesibilidad web en los documentos *HTML*. El conocimiento de las pautas permite establecer una relación entre las etiquetas y atributos del *HTML* y los criterios de éxito de accesibilidad.

La Figura 1 muestra la organización de las *WCAG* 2.2 en cuatro principios. Cada uno agrupa un conjunto de pautas, que suman 13 en total. A su vez, estas contienen uno o más criterios de éxito, alcanzando 86. Cada criterio está asociado a un nivel de conformidad (A, AA o AAA) y cuenta con más de 250 técnicas suficientes para guiar su correcta implementación.

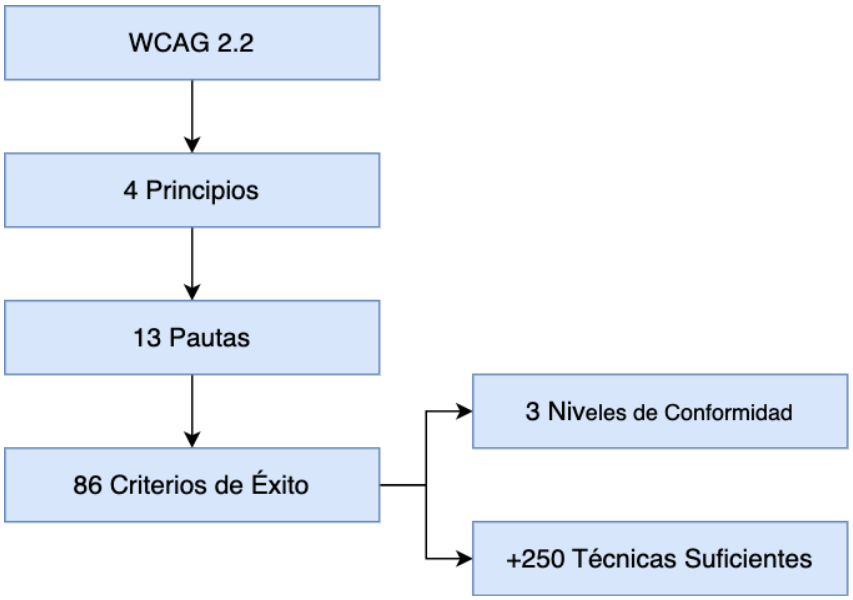


*Figura 1. Elementos fundamentales de las WCAG 2.2*

La Figura 2 ofrece un ejemplo de la estructura jerárquica de las *WCAG* 2.2, mostrando la relación entre un principio, su pauta correspondiente, el criterio de éxito, el nivel de conformidad y una técnica suficiente. Específicamente, muestra el principio **1 Perceptible**, del cual se deriva la pauta **1.1 Alternativas de texto**. A su vez, esta pauta se concreta en el criterio de éxito: **1.1.1 Contenido no textual**. La figura también indica que este criterio de éxito corresponde a un nivel de conformidad **Nivel A** y presenta un ejemplo de técnica suficiente para su cumplimiento: la técnica **H37**, que consiste en "**Usar el atributo *alt* en el elemento *<img>***". Este desglose visualiza cómo cada nivel de las *WCAG* 2.2, desde los principios hasta las técnicas suficientes, se interconectan para guiar la creación de contenido web accesible.

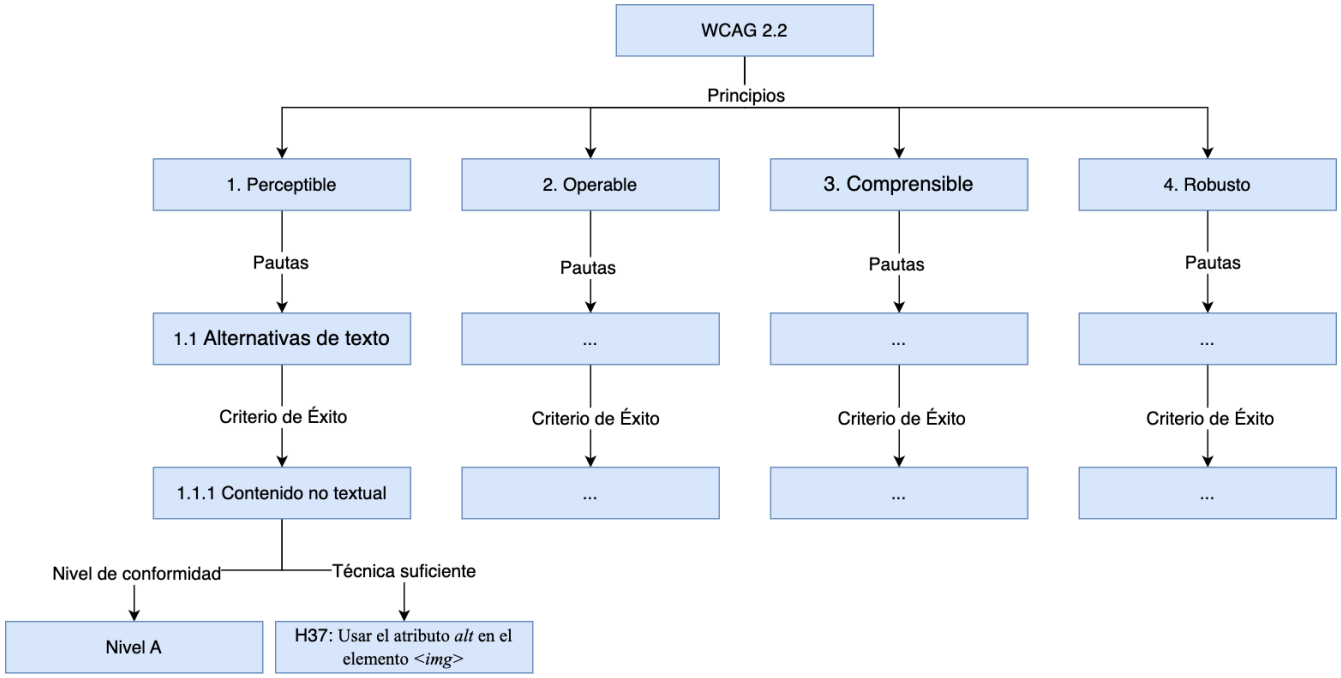


*Figura 2. Ejemplo de relación de los elementos fundamentales de las WCAG 2.2*

### 4.2.1 PRINCIPIOS DE LAS *WCAG* 2.2

Toda información accesible debe cumplir con los siguientes cuatro principios, conocidos por el acrónimo *P.O.U.R.* (por sus siglas en inglés):

1. **Perceptible:** La información y los componentes de la interfaz deben ser presentados de forma que puedan ser percibidos por los usuarios. Esto incluye alternativas textuales, adaptabilidad y distinción visual [6], [7].

2. **Operable:** La navegación debe ser accesible a través de diferentes métodos, permitiendo el uso del teclado y evitando contenidos que puedan causar efectos adversos [6], [7].
3. **Comprensible:** El contenido debe ser claro, legible y predecible, con mecanismos de ayuda y corrección de errores [6], [7].
4. **Robusto:** El contenido debe ser interpretable por tecnologías de asistencia y compatible con futuras herramientas [6], [7].

### 4.2.2 PAUTAS DENTRO DE CADA PRINCIPIO

Dentro de las WCAG 2.2, se identifican principios que, en conjunto, comprenden 13 pautas [9]. Estas pautas sirven para fijar las metas esenciales destinadas a lograr que el contenido sea más accesible para personas con distintas discapacidades [6], [7]. A continuación, se presenta la relación entre los principios y sus pautas:

1. **Perceptible**
   - 1.1. **Alternativas de texto:** Consisten en un equivalente textual para todo contenido no textual, facilitando su adaptación a formatos requeridos por los usuarios (como letra grande, braille, voz o símbolos) [8]. Estas alternativas permiten que los usuarios que no pueden ver o interactuar con el contenido no textual comprendan su propósito y contexto [8].
   - 1.2. **Medios basados en el tiempo**: El texto alternativo, las transcripciones y los subtítulos deben sincronizarse con los medios interactivos [8].
   - 1.3. **Adaptable**: Todo el contenido debe poder ajustarse sin perder información cuando se cambia el tamaño de visualización de un sitio web [8].
   - 1.4. **Distinguible**: El contenido debe tener un formato con el espaciado y el tamaño de texto adecuados. Esto incluye un fuerte contraste del texto en primer plano con el color y las imágenes de fondo [8].

2. **Operable**
   - 2.1. **Accesible mediante teclado**: Los sitios web deben poder operarse completa y fácilmente mediante la navegación solo con teclado. Esto significa que los sitios web no deben interrumpir las funciones típicas de las pulsaciones de teclas o los comandos de acceso directo [8].

2.2. **Tiempo suficiente**: Se debe programar un tiempo suficiente para todos los medios basados en el tiempo y el acceso sin perder la oportunidad de leer o ingresar las respuestas respectivas [8].

2.3. **Convulsiones y reacciones físicas**: Evite la creación de contenido susceptible de inducir ataques o respuestas físicas [8].

2.4. **Navegable**: Un sitio web debe ofrecer a los usuarios formas sencillas y lógicas de descubrir y encontrar contenido. Los menús de navegación y los menús desplegables deben tener sentido y ser fáciles de usar. Se deben usar etiquetas de encabezado y subtítulo para dividir los cuadros de texto largos [8].

2.5. **Modalidades de entrada**: Los usuarios deben tener una experiencia similar en el sitio web incluso con diferentes opciones de entrada más allá de un teclado y un mouse. Esto incluye controles de interruptores, rastreadores oculares y comandos de voz [8].

3. **Comprensible**

3.1. **Legible:** Todo lo escrito debe ser claro y comprensible, evitando la jerga, las palabras inusuales y teniendo acrónimos limitados. Cuando se requiere un lenguaje desconocido, se debe calificar con definiciones y la frase completa sin abreviar [8].

3.2. **Predecible:** Los sitios web y el contenido deben diseñarse para funcionar como las personas esperan, con una orientación de desplazamiento vertical y todos los enlaces de navegación incluidos en los márgenes del encabezado y el pie de página [8].

3.3. **Entrada de datos asistida:** Es fundamental que los usuarios empleen la información precargada por el navegador, lo que les permite minimizar equivocaciones y asegurar la exactitud al diligenciar campos de inicio de sesión y formularios de alta [8].

4. **Robusto**

4.1. **Compatible:** Los sitios web deben esforzarse por ser compatibles con los agentes de usuario actuales y futuros, incluidas las tecnologías asistidas, para brindar la mejor experiencia de usuario para todos los usuarios [8].

### 4.2.3 CRITERIOS DE ÉXITO EN LAS PAUTAS

Continuando con la estructura jerárquica de las *WCAG* 2.2, cada una de las 13 pautas mencionadas anteriormente, se desglosa en criterios de éxito específicos y verificables. Estos son elementos esenciales de las *WCAG* 2.2, que posibilitan una evaluación objetiva del cumplimiento de la accesibilidad en el contenido web [9].

En total son 86 criterios de éxito. Cada uno de estos se presenta como una proposición verificable (verdadera o falsa) al ser aplicada a un contenido web específico, lo que permite una valoración imparcial de su accesibilidad [6]. La Tabla 1 ofrece un listado detallado de estos 86 criterios de éxito, organizados según la pauta y el principio al que pertenecen [2]; incluyendo adicionalmente el nivel de conformidad que será mencionado en la subsección 4.2.3.

Tabla 1. Estructura Jerárquica de las *WCAG* 2.2

| **Principio** | **Pauta** | **Criterio de Éxito** | **Nivel de Conformidad** |
|---|---|---|---|
| 1. Perceptible | 1.1 Alternativas de texto | 1.1.1 Contenido no textual | A |
| 1. Perceptible | 1.2 Medios basados en el tiempo | 1.2.1 Solo audio y solo vídeo (grabado) | A |
| 1. Perceptible | | 1.2.2 Subtítulos (grabados) | A |
| 1. Perceptible | | 1.2.3 Audiodescripción o medio alternativo (grabado) | A |
| 1. Perceptible | | 1.2.4 Subtítulos (en directo) | AA |
| 1. Perceptible | | 1.2.5 Audio descripción (grabado) | AA |
| 1. Perceptible | | 1.2.6 Lenguaje de señas (pregrabado) | AAA |
| 1. Perceptible | | 1.2.7 Descripción de audio extendida (pregrabada) | AAA |
| 1. Perceptible | | 1.2.8 Medios alternativos (pregrabados) | AAA |
| 1. Perceptible | | 1.2.9 Solo audio (en vivo) | AAA |

| | | | |
|---|---|---|---|
| 1. Perceptible | 1.3 Adaptable | 1.3.1 Información y relaciones | A |
| 1. Perceptible | | 1.3.2 Secuencia significativa | A |
| 1. Perceptible | | 1.3.3 Características sensoriales | A |
| 1. Perceptible | | 1.3.4 Orientación | AA |
| 1. Perceptible | | 1.3.5 Identificación del propósito de entrada | AA |
| 1. Perceptible | | 1.3.6 Identificación del propósito | AAA |
| 1. Perceptible | 1.4 Distinguible | 1.4.1 Uso del color | A |
| 1. Perceptible | | 1.4.2 Control del audio | A |
| 1. Perceptible | | 1.4.3 Contraste (mínimo) | AA |
| 1. Perceptible | | 1.4.4 Cambio de tamaño del texto | AA |
| 1. Perceptible | | 1.4.5 Imágenes de texto | AA |
| 1. Perceptible | | 1.4.6 Contraste (mejorado) | AAA |
| 1. Perceptible | | 1.4.7 Sonido de fondo bajo o ausente | AAA |
| 1. Perceptible | | 1.4.8 Presentación visual | AAA |
| 1. Perceptible | | 1.4.9 Imágenes de texto (sin excepciones) | AAA |
| 1. Perceptible | | 1.4.10 Reajuste | AA |
| 1. Perceptible | | 1.4.11 Contraste no textual | AA |
| 1. Perceptible | | 1.4.12 Espaciado del texto | AA |
| 1. Perceptible | | 1.4.13 Contenido en *hover* o *focus* | AA |

| | | | |
|---|---|---|---|
| 2. Operable | 2.1 Accesible por teclado | 2.1.1 Teclado | A |
| 2. Operable | | 2.1.2 Sin trampas para el foco del teclado | A |
| 2. Operable | | 2.1.3 Teclado (sin excepciones) | AAA |
| 2. Operable | | 2.1.4 Atajos de teclado | A |
| 2. Operable | 2.2 Tiempo suficiente | 2.2.1 Tiempo ajustable | A |
| 2. Operable | | 2.2.2 Poner en pausa, detener, ocultar | A |
| 2. Operable | | 2.2.3 Sin tiempo | AAA |
| 2. Operable | | 2.2.4 Interrupciones | AAA |
| 2. Operable | | 2.2.5 Re-autentificación | AAA |
| 2. Operable | | 2.2.6 Timeouts | AAA |
| 2. Operable | 2.3 Convulsiones y reacciones físicas | 2.3.1 Umbral de tres destellos o menos | A |
| 2. Operable | | 2.3.2 Tres destellos | AAA |
| 2. Operable | | 2.3.3 Animación de interacciones | AAA |
| 2. Operable | 2.4 Navegable | 2.4.1 Evitar bloques | A |
| 2. Operable | | 2.4.2 Titulado de páginas | A |
| 2. Operable | | 2.4.3 Orden del foco | A |
| 2. Operable | | 2.4.4 Propósito de los enlaces (en contexto) | A |
| 2. Operable | | 2.4.5 Múltiples vías | AA |
| 2. Operable | | 2.4.6 Encabezados y etiquetas | AA |
| 2. Operable | | 2.4.7 Foco visible | AA |
| 2. Operable | | 2.4.8 Ubicación | AAA |
| 2. Operable | | 2.4.9 Propósito de los enlaces (solo enlaces) | AAA |
| 2. Operable | | 2.4.10 Encabezados de sección | AAA |

| | | | |
|---|---|---|---|
| 2. Operable | 2.5 Modalidades de entrada | 2.5.1 Gestos del puntero | A |
| 2. Operable | | 2.5.2 Cancelación del puntero | A |
| 2. Operable | | 2.5.3 Etiqueta en el nombre | A |
| 2. Operable | | 2.5.4 Actuación por movimiento | A |
| 2. Operable | | 2.5.5 Tamaño del objetivo | AAA |
| 2. Operable | | 2.5.6 Mecanismos de entrada concurrentes | AAA |
| 3. Comprensible | 3.1 Legible | 3.1.1 Idioma de la página | A |
| 3. Comprensible | | 3.1.2 Idioma de las partes | AA |
| 3. Comprensible | | 3.1.3 Palabras inusuales | AAA |
| 3. Comprensible | | 3.1.4 Abreviaturas | AAA |
| 3. Comprensible | | 3.1.5 Nivel de lectura | AAA |
| 3. Comprensible | | 3.1.6 Pronunciación | AAA |
| 3. Comprensible | 3.2 Predecible | 3.2.1 Al recibir el foco | A |
| 3. Comprensible | | 3.2.2 Al recibir entradas | A |
| 3. Comprensible | | 3.2.3 Navegación coherente | AA |
| 3. Comprensible | | 3.2.4 Identificación coherente | AA |
| 3. Comprensible | | 3.2.5 Cambios a petición | AAA |
| 3. Comprensible | 3.3 Entrada de datos asistida | 3.3.1 Identificación de errores | A |
| 3. Comprensible | | 3.3.2 Etiquetas o instrucciones | A |
| 3. Comprensible | | 3.3.3 Sugerencias ante errores | AA |
| 3. Comprensible | | 3.3.4 Prevención de errores (legales, financieros, datos) | AA |
| 3. Comprensible | | 3.3.5 Ayuda | AAA |
| 3. Comprensible | | 3.3.6 Prevención de errores (todos) | AAA |
| 4. Robusto | 4.1 Compatible | 4.1.1 Procesamiento | A |
| 4. Robusto | | 4.1.2 Nombre, función, valor | A |
| 4. Robusto | | 4.1.3 Mensajes de estado | AA |

Los criterios de éxito están diseñados para ser neutrales, comprobables y aplicables a diferentes tipos de contenido web. Su estructura permite a los desarrolladores verificar de manera objetiva si su implementación cumple con los requisitos establecidos, ya sea mediante pruebas automatizadas o evaluación humana.

La relación entre los criterios de éxito y las etiquetas *HTML* es particularmente relevante para este trabajo, ya que muchos de estos criterios de éxito pueden implementarse directamente a través de elementos y atributos *HTML* específicos, como textos alternativos para imágenes, etiquetas para campos de formulario, o estructura semántica adecuada mediante encabezados y *landmarks* [2].

Los criterios de éxito se clasifican según los niveles de conformidad (A, AA, AAA) [2], que de acuerdo al criterio de éxito se pueden ver en la Tabla 1. Además, las *WCAG* 2.2 proveen un conjunto de técnicas para guiar su implementación, aspectos que se describen en la subsección (4.2.3.2).

#### 4.2.3.1 NIVELES DE CONFORMIDAD

Conforme a las *WCAG* 2.2 [30], definen los criterios de éxito, los cuales se organizan en los niveles de conformidad A, AA y AAA [2]. La Tabla 1 proporciona una descripción de cada criterio de éxito dentro de esta categorización.

- **Nivel A**: Adecuado como punto de partida mínimo.
- **Nivel AA**: Recomendado para sitios gubernamentales, educativos y de organizaciones públicas.
- **Nivel AAA**: Aplicable cuando la accesibilidad máxima es una prioridad, aunque no es obligatorio en la mayoría de los casos.

#### 4.2.3.2 APLICACIÓN DE TÉCNICAS PARA LOS CRITERIOS DE ÉXITO

Las *WCAG* 2.2 incluyen, además, una serie de técnicas, las cuales detallan el método para satisfacer cada criterio:

- **Técnicas suficientes:** Son formas fiables de cumplir los criterios de éxito [13]. Si utiliza las técnicas suficientes para un criterio determinado correctamente y es compatible con la accesibilidad para sus usuarios, puede estar seguro de que cumple con el criterio de éxito [13], [14].
- **Técnicas de asesoramiento:** Son formas sugeridas de mejorar la accesibilidad [13].

- **Fallos:** Los errores son elementos que generan barreras de accesibilidad y no cumplen con criterios de éxito específicos [13].

### 4.2.4 DESAFÍOS EN LA IMPLEMENTACIÓN DE LAS *WCAG* 2.2

A pesar de la existencia de estas pautas, la implementación efectiva de la accesibilidad web sigue siendo un reto. Según el *WebAIM Million Report* 2025 [3], el 94.8 % de las páginas web analizadas presentaron al menos un error de accesibilidad, evidenciando la falta de cumplimiento generalizado de las *WCAG 2.2* [3]. Los problemas de accesibilidad más comunes que se identifican son: imágenes sin texto alternativo (55.5 %), contraste de color insuficiente (79.1 %), ausencia de etiquetas en formularios (48.2 %) y estructuras semánticas deficientes.

Uno de los principales obstáculos identificados en la literatura es la falta de formación en accesibilidad entre los desarrolladores web. De acuerdo con *Abuaddous et al.* [4], muchos profesionales del desarrollo desconocen la existencia o aplicación de las *WCAG* 2.2, lo que contribuye a la baja tasa de adopción de prácticas accesibles en el desarrollo de software. Además, *Halpin* [5] menciona que muchas empresas priorizan la accesibilidad cuando están sujetas a regulaciones legales, lo que limita la implementación proactiva de estas prácticas.

Otro desafío importante es la ausencia de herramientas que permitan a los desarrolladores comprobar la accesibilidad de forma inmediata. Muchas herramientas actuales están orientadas a auditorías post-desarrollo en lugar de proporcionar retroalimentación inmediata dentro del entorno de desarrollo [10].

## 4.3 NORMAS INTERNACIONALES DE ACCESIBILIDAD WEB.

Además de las *WCAG* 2.2 desarrolladas por el *W3C* [12], existen otras especificaciones y estándares cuyo propósito es promover la accesibilidad en los medios digitales. Estas normas complementan las *WCAG 2.2* al proporcionar un marco legal o regulatorio en diferentes regiones del mundo, facilitando su adopción en contextos específicos como la administración pública, la educación o el diseño de servicios digitales universales.

Estas normas refuerzan la pertinencia de este trabajo en el contexto internacional, mostrando que el enfoque propuesto puede adaptarse a múltiples marcos regionales. Entre las cuales encontramos:

- ***ISO/IEC* 40500:2012:** Es un estándar internacional que ofrece numerosas recomendaciones para mejorar la accesibilidad del contenido web. La implementación de estas pautas asegura la accesibilidad del contenido para un espectro más amplio de usuarios con discapacidades. Esto abarca a quienes presentan ceguera, incluyendo condiciones como baja visión, sordera, pérdida auditiva, problemas de aprendizaje, restricciones cognitivas, movilidad reducida, dificultades del habla, fotosensibilidad, o sus combinaciones. Asimismo, la aplicación de estas guías tiende a mejorar la facilidad de uso del contenido web para el público en general [41].
- **EN 301 549:** Esta norma europea define los requisitos de accesibilidad aplicables a los productos y servicios de Tecnologías de la Información y la Comunicación (*TIC*). Dicha normativa se fundamenta en gran medida en las *WCAG* 2.1 y su cumplimiento es obligatorio para las plataformas web y aplicaciones móviles de las entidades del sector público en la Unión Europea [42].
- **SECCIÓN 508 DEL *REHABILITATION ACT*:** En Estados Unidos, la Sección 508 del *Rehabilitation Act* demanda que las plataformas web y tecnologías electrónicas de la administración federal resulten accesibles. Dicha normativa guarda congruencia con las WCAG 2.0, lo cual asegura que el acceso a los contenidos digitales esté garantizado para todos los usuarios, sin restricciones [43].

## 4.4 HERRAMIENTAS Y MÉTODOS

El proceso de evaluar la accesibilidad web es esencial para detectar y eliminar barreras que puedan impedir el acceso equitativo a la información. Existen diversas herramientas y métodos que facilitan esta evaluación, incluyendo extensiones para editores de código que permiten a los desarrolladores integrar prácticas de accesibilidad durante el proceso de desarrollo [11]. Estas herramientas y su integración influyen en este documento al demostrar que los procesos de verificación pueden incorporarse, ayudando a los desarrolladores a adoptar prácticas accesibles de manera eficiente.

### 4.4.1 HERRAMIENTAS USADAS PARA EVALUAR ACCESIBILIDAD

- ***WAVE* (*Web Accessibility Evaluation Tool*):** Una herramienta que realiza diagnósticos visuales en las páginas web, permitiendo detectar inconsistencias y brindar propuestas para su perfeccionamiento [38].
- ***Lighthouse*:** Desarrollada por *Google*, es una herramienta de código abierto integrada en las herramientas de desarrollo de *Chrome* que evalúa la calidad de una página web, incluida su accesibilidad [39].
- ***axe*:** Es una familia de herramientas de *Deque Systems* centrada en los desarrolladores que permite realizar pruebas de accesibilidad. Su extensión de navegador, *axe DevTools* [40], es compatible con *Firefox* y *Chrome*, facilitando la detección de problemas de accesibilidad de manera eficiente.

### 4.4.2 EXTENSIONES PARA EDITORES DE CÓDIGO

Integrar herramientas de evaluación de accesibilidad directamente en los editores de código permite a los desarrolladores identificar y solucionar problemas en tiempo real, mejorando la eficiencia y eficacia del proceso de desarrollo. Algunas extensiones destacadas incluyen:

- ***axe Accessibility Linter*:** Es una extensión para *Visual Studio Code* que analiza el código en busca de problemas de accesibilidad según las reglas de *axe*, proporcionando retroalimentación inmediata al desarrollador [35].
- ***Accessibility Insights for Web*:** Disponible como extensión para navegadores y como herramienta independiente, esta herramienta ayuda a los desarrolladores a encontrar y solucionar problemas de accesibilidad en sitios web y aplicaciones web [45].

## 4.5 *W3C*, *HTML* Y EL ROL DEL MARCADO SEMÁNTICO EN LA ACCESIBILIDAD

El *World Wide Web Consortium* (*W3C*) [12], es la entidad encargada de desarrollar los estándares fundamentales para la web, incluyendo aquellos relacionados con la accesibilidad digital [1], [2]. A través de la Iniciativa de Accesibilidad Web (*WAI*), este organismo promueve las *WCAG* 2.2 como una guía integral para asegurar que los contenidos web puedan ser percibidos, operados, comprendidos y procesados correctamente por todos los usuarios, incluyendo personas con discapacidades [2], [6].

Uno de los pilares para garantizar esta accesibilidad es el correcto uso del lenguaje de marcado *HTML* (*HyperText Markup Language*) [16]. Este lenguaje proporciona la estructura semántica de las aplicaciones web, entre ellas las páginas web, permitiendo definir elementos como encabezados, listas, tablas, formularios e imágenes, los cuales juegan un rol clave en la interpretación de los contenidos por parte de tecnologías de asistencia [3].

Por su parte, las aplicaciones web (páginas de contenido estático, como sitios web y dinámicos) son sistemas interactivos que operan a través de un navegador y cuya interfaz se construye fundamentalmente con tecnologías del lado del cliente como *HTML*, *CSS* y *JavaScript*. Estas aplicaciones deben ser diseñadas desde su arquitectura básica con criterios de accesibilidad que permitan la navegación y comprensión por parte de todo tipo de usuarios [4], [19].

En este contexto, comprender la relación entre los lineamientos del *W3C*, el código *HTML* y la construcción de sitios web para lograr una aplicación efectiva de las *WCAG* 2.2. Esta relación técnica constituye la base del presente trabajo, que busca fortalecer la accesibilidad web en su implementación, desde el momento en que los desarrolladores comienzan a escribir código en documentos *HTML*.

# 5. METODOLOGÍA

Para la realización de este trabajo, se sigue la metodología *Design Science Research* (*DSR*), propuesta por *Peffers et al*. [29], que proporciona un marco estructurado para la creación y evaluación de artefactos orientados a resolver problemas específicos. Esta metodología permite abordar el diseño de la estrategia para facilitar la aplicación de la accesibilidad web en documentos *HTML,* en conformidad con las *WCAG* 2.2.

La metodología consta de seis etapas, representadas visualmente en la Figura 3, que se basa en el modelo de *Design Science Research* [29]. Estas etapas se describen a continuación:

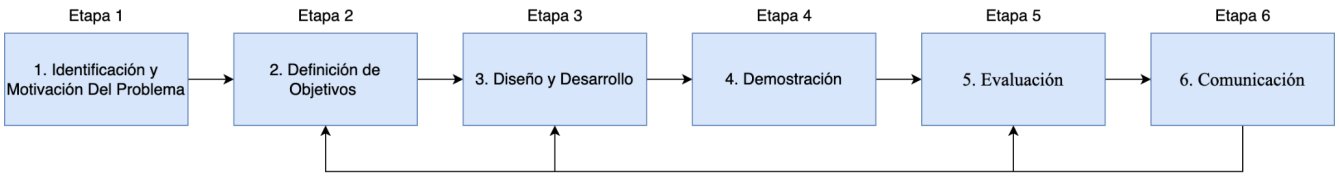


*Figura 3. Design Science Research*

## 5.1. IDENTIFICACIÓN Y MOTIVACIÓN DEL PROBLEMA

Esta etapa tiene como objetivo detectar y analizar problemas prácticos en el mundo real que requieren soluciones innovadoras. En este proceso, se busca comprender la naturaleza del problema, su relevancia y el impacto potencial de una solución, lo que justifica la necesidad de un enfoque sistemático y científico para abordarlo.

En la Sección 2 “Planteamiento del problema y justificación”, se menciona la brecha existente entre las *WCAG* 2.2 y su implementación en la creación de sitios web, evidenciando la necesidad de una estrategia que facilite la implementación de la accesibilidad web en documentos *HTML*.

## 5.2 DEFINICIÓN DE OBJETIVOS

Esta etapa se enfoca en establecer metas claras y alcanzables que guiarán la solución propuesta. Estos objetivos deben ser específicos y medibles, para asegurar que la investigación aborde de manera efectiva el problema identificado.

La definición de los objetivos, tanto generales como específicos para este trabajo, se detalla en la Sección 3 “Objetivos”.

## 5.3 DISEÑO Y DESARROLLO

Se centra en la creación de una solución innovadora basada en los objetivos definidos, transformando el conocimiento teórico en un artefacto funcional. En esta etapa, se desarrollan y refinan los componentes clave de la solución, asegurando su alineación con los problemas identificados y los objetivos de la investigación.

Esta etapa abarca el diseño y desarrollo de la estrategia, con el objetivo de simplificar la integración de la accesibilidad web en el *HTML*. El proceso incluye:

- Análisis de las *WCAG* 2.2 para identificar los criterios de éxito.
- Asociación de las etiquetas y atributos *HTML* con cada criterio de éxito.
- Elaboración de una tabla que vincule los elementos *HTML* y criterios de éxito de las *WCAG* 2.2.
- Desarrollo de una extensión de *Visual Studio Code* para analizar y validar las etiquetas *HTML*.

Es importante señalar que, a lo largo del desarrollo del trabajo, algunos de los objetivos inicialmente definidos se fueron refinando de acuerdo a la evolución del trabajo. La selección de los criterios de éxito aplicables a las etiquetas y atributos *HTML* fue fundamental para delimitar el alcance y definir la naturaleza de las pruebas a ejecutar. Asimismo, la construcción de la tabla de reglas y la implementación práctica de la estrategia como extensión en el editor de *Visual Studio Code* contribuyó a la redefinición de los objetivos específicos, manteniendo siempre la coherencia con el objetivo general del trabajo.

El diseño se fundamenta en los hallazgos de *Abuaddous et al.* [4], sobre los desafíos en la implementación de la accesibilidad web, particularmente aquellos relacionados con la interpretación de las *WCAG* 2.2, el diseño e implementación, y la fase de evaluación. Esta etapa se detalla en la Sección 6 “Diseño y desarrollo de la propuesta” y la Sección 7 “Aplicación técnica de la propuesta”.

## 5.4 DEMOSTRACIÓN

Implica validar la efectividad de la solución propuesta mediante su implementación en un entorno real o simulado. En esta etapa, se prueba el artefacto desarrollado para garantizar que cumpla con los objetivos establecidos y resuelva el problema identificado.

En este trabajo, la etapa de demostración se centró en la aplicación práctica de la estrategia desarrollada sobre documentos *HTML*, los cuales fueron específicamente creados para verificar su capacidad de detección de errores. Este proceso se hizo en dos actividades principales:

- Selección de una muestra de 10 documentos *HTML* de prueba, con errores comunes de accesibilidad, basados en los hallazgos del *WebAIM Million Report* 2025 [3].
- Aplicación de la estrategia propuesta en la extensión de *Visual Studio Code*, para analizar los documentos de prueba e identificar los errores de accesibilidad presentes en ellos.

El diseño de la estrategia que se demuestra en esta etapa se detalla en la Sección 6 “Diseño y desarrollo de la propuesta”. La implementación técnica de esta estrategia en la extensión de *Visual Studio Code* y el proceso detallado de su aplicación sobre los documentos de prueba para esta demostración se exponen en la Sección 7 “Aplicación técnica de la propuesta”.

## 5.5 EVALUACIÓN

Se centra en medir el rendimiento y la efectividad de la solución propuesta frente a los objetivos establecidos. A través de pruebas y análisis detallados, se determina si el artefacto cumple con los requisitos y resuelve el problema de manera adecuada.

En este trabajo, la evaluación de la estrategia propuesta se llevó a cabo mediante un análisis comparativo de su capacidad para detectar errores de accesibilidad. Para ello, se contrastó el desempeño de la extensión de *Visual Studio Code*, con otras herramientas.

Este proceso de evaluación no solo permitió validar el funcionamiento aplicado técnicamente de la extensión de *Visual Studio Code*, sino que también evidenció aspectos que requerían ajustes menores en las reglas de verificación, los mensajes generados y la estructura del código de la extensión. Estos

hallazgos permitieron refinar los objetivos del trabajo, reforzar el proceso iterativo del diseño, tal como lo plantea la metodología *Design Science Research* y encontrar mejoras futuras para seguir fortaleciendo la propuesta.

Esta etapa se describe en la Sección 7 “Aplicación técnica de la propuesta”, donde se explica el proceso de implementación y evaluación de la estrategia. Por su parte, la Sección 8 “Resultados”, presenta los hallazgos y el análisis de dicha evaluación.

## 5.6 COMUNICACIÓN

Se enfoca en compartir los resultados de la investigación de manera clara y comprensible, asegurando que los hallazgos sean accesibles a la comunidad académica y profesional. Esta fase es crucial para difundir el impacto y la relevancia de la solución propuesta.

La etapa final consiste en la documentación y comunicación de los resultados obtenidos. El presente trabajo cumple con este objetivo al presentar la estrategia desarrollada en un ejemplo práctico y los hallazgos derivados de su implementación y evaluación comparándolo con otras herramientas.

Esta etapa se detalla en la Sección 8 “Resultados”.

# 6. DISEÑO Y DESARROLLO DE LA PROPUESTA

Esta sección detalla el proceso seguido para el diseño y desarrollo de la estrategia para facilitar la integración de la accesibilidad web en documentos *HTML*, en conformidad con los criterios de éxito, los cuales se derivan de las pautas y principios de las *WCAG* 2.2, que se pueden ver detalladamente en la Tabla 1.

La concepción de esta estrategia parte de facilitar a los desarrolladores la aplicación de la accesibilidad web según las *WCAG* 2.2, cuando están creando las etiquetas, sus atributos y el contenido de texto, en los documentos *HTML*. Además, aborda los obstáculos identificados por *Abuaddous et al*. [4], como son: la complejidad para interpretar correctamente los estándares, no contar con herramientas que validen el código mientras se escribe y las limitaciones de los mecanismos actuales para evaluar la accesibilidad de manera efectiva.

El diseño de la estrategia se basa en el análisis de los criterios de éxito de las *WCAG* 2.2, las técnicas suficientes para su cumplimiento documentadas por el *W3C*, y los errores de accesibilidad más frecuentes reportados en informes de la industria como el *WebAIM Million Report* 2025. A partir de estos elementos, se estructura el proceso de diseño y desarrollo de la estrategia en tres etapas, las cuales se ilustran en la Figura 4.

- **Identificación de errores frecuentes de accesibilidad:** Análisis de los errores más comunes reportados por el *WebAIM Million Report 2025*, para priorizar los criterios de éxito más relevantes a tener en cuenta desde *HTML*.
- **Selección de criterios de éxito aplicables al *HTML*:** Filtrado de aquellos criterios de éxito que pueden aplicarse directamente mediante etiquetas y atributos del *HTML*.
- **Diseño de la estrategia de verificación *HTML-WCAG*:**
    - Establecimiento de la relación entre etiquetas y atributos de *HTML* con los criterios de éxito, fundamentándose en las técnicas suficientes del *W3C*.
    - Definición de reglas de verificación, conectando los criterios de éxito de las *WCAG* 2.2, con las etiquetas y atributos de los documentos del *HTML*, las técnicas suficientes, el error común asociado y una recomendación práctica para su corrección directamente en el documento *HTML*.

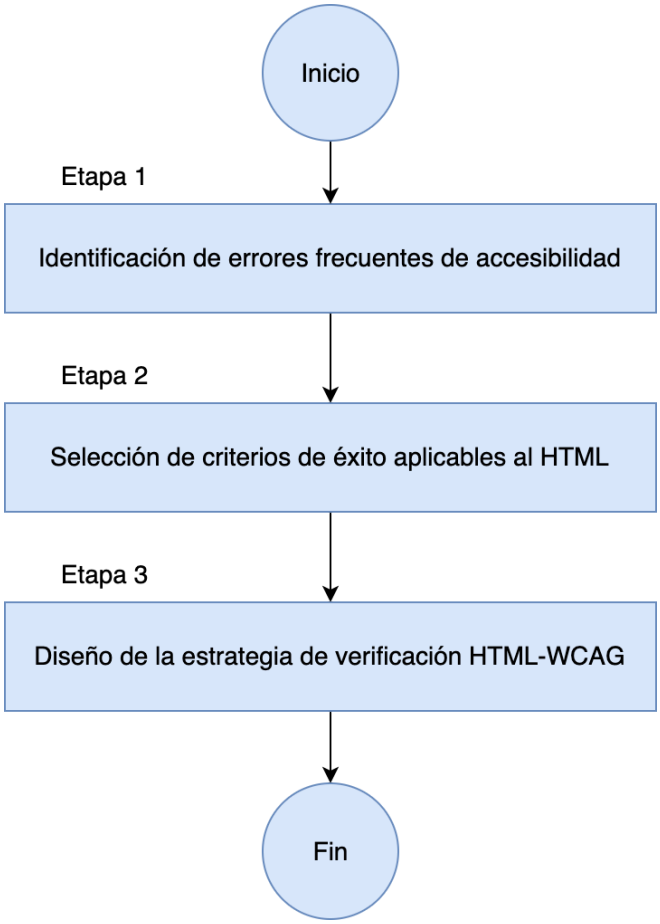


*Figura 4. Etapas del diseño y desarrollo de la propuesta*

Este proceso de tres etapas permitió definir los componentes que se utilizaron para diseñar y desarrollar la estrategia. Facilitando la comprensión de las *WCAG* 2.2 y su aplicación técnica. Se espera que la estrategia sirva como una base reutilizable y adaptable para quienes buscan implementar accesibilidad web de forma efectiva.

## 6.1 IDENTIFICACIÓN DE ERRORES FRECUENTES DE ACCESIBILIDAD

Como punto de partida para el diseño de la estrategia, se realizó una revisión de los errores más comunes identificados en el informe *WebAIM Million Report* 2025 [3], con el propósito de asociarlos a los principios, pautas y criterios de éxito definidos por las *WCAG* 2.2. Estos hallazgos permiten priorizar las validaciones de las etiquetas y atributos del *HTML* necesarias durante el desarrollo en documentos *HTML*. En la Tabla 2 desglosa los errores, mostrando su correspondencia con la pauta, el principio, el criterio de éxito, el nivel de conformidad y el porcentaje de error asociado [3].

Tabla 2. Análisis de errores frecuentes de accesibilidad web

| **Principio**<br>**Pauta** | **Criterio de éxito** | **Nivel de Conformidad** | **Error Común** | **Porcentaje de páginas con el error** |
|---|---|---|---|---|
| 1. Perceptible<br>1.1 Alternativas de texto | 1.1.1 Contenido no textual | A | Imágenes sin texto alternativo | 55.5% |
| 1. Perceptible<br>1.3 Adaptable | 1.3.1 Información y relaciones | A | Falta de etiquetas en formularios (*form labeling*) | 48.2% |
| 1. Perceptible<br>1.4 Distinguible | 1.4.3 Contraste (Mínimo) | AA | Texto de bajo contraste | 79.1% |
| 2. Operable<br>2.4 Navegable | 2.4.1 Evitar bloques | A | Saltos de navegación ausentes o con errores | 10% de los *skip links* estaban rotos |
| | 2.4.4 Propósito de los enlaces (en contexto) | A | Texto de enlace ambiguo | 13.7% |
| | 2.4.6 Encabezados y etiquetas | AA | Niveles de encabezado omitidos o múltiples <*h1*> | 39.0% |
| 3. Comprensible<br>3.1 Legible | 3.1.1 Idioma de la página | A | Falta de idioma del documento | 15.8% |
| 3. Comprensible<br>3.3 Entrada de datos asistida | 3.3.2 Etiquetas o instrucciones | A | Campos de formulario sin etiquetas | 48.2% |
| 4. Robusto<br>4.1 Compatible | 4.1.1 Procesamiento | A | *Doctype* inválido o ausente | 7.6% de páginas sin *doctype HTML5* |
| | 4.1.2 Nombre, función, valor | A | Botones vacíos | 29.6% |
| | 4.1.3 Mensajes de estado | AA | Enlaces vacíos | 45.4% |

## 6.2 IDENTIFICACIÓN DE LOS CRITERIOS DE ÉXITO RELEVANTES DE LA *WCAG* 2.2

El análisis de los errores de accesibilidad web comunes, presentados en la etapa anterior, resaltó la necesidad de que la estrategia propuesta vaya más allá de abordar únicamente las problemáticas más frecuentes. Si bien estos errores representan una parte significativa de las barreras de accesibilidad, su cobertura exclusiva dejaría sin atender otros aspectos igualmente importantes que también pueden ser abordados desde los documentos *HTML*. Por esta razón, en esta etapa se identifican todos aquellos criterios de éxito de las *WCAG* 2.2 que presentan una vinculación directa con los elementos del *HTML*, como etiquetas y atributos.

Para empezar, es fundamental tener conocimiento de todos los criterios de éxito definidos por las *WCAG* 2.2. Estos están organizados en 13 pautas [9] y cuatro principios (principios *P.O.U.R* [6], [7]). Además, cada criterio de éxito, se clasifica por niveles (A, AA [2]). Es importante destacar que, si bien todos los criterios de éxito son importantes para la accesibilidad web, no todos se manifiestan o se pueden verificar exclusivamente a través de las etiquetas y atributos de los documentos *HTML*. Algunos criterios de éxito dependen de tecnologías complementarias como: Las Hojas de Estilo en Cascada (*CSS*), encargadas del estilo visual y el diseño responsivo; *JavaScript (JS)* para interactividad y comportamiento dinámico; contenido multimedia.

Dada la complejidad y especificidad al análisis de cada una de estas tecnologías (*HTML*, *CSS*, *JavaScript*), y con el objetivo de desarrollar una estrategia directamente aplicable a la capa fundamental del contenido en la web, este trabajo se enfoca exclusivamente en los criterios de éxito relacionados con *HTML*. Abordar la accesibilidad en *CSS* y *JavaScript* requeriría estrategias y herramientas de análisis distintas y especializadas para cada contexto, lo cual excede el alcance de este trabajo.

Siguiendo este enfoque centrado en el *HTML*, se realizó una selección específica de los criterios de éxito que sí pueden aplicarse directamente en los documentos *HTML* a sus etiquetas y atributos. Estos criterios de éxito están vinculados al uso correcto de etiquetas, y atributos propios del *HTML*, y son los que forman la base de esta estrategia de implementación. Esta selección se presenta en la Tabla 3 [2].

Tabla 3. Clasificación de los criterios de éxito según su aplicabilidad

| **Criterio de Éxito** | **Etiqueta o atributo del *HTML*** | **Error común** | **Solución en el *HTML*** |
|---|---|---|---|
| 1.1.1 Contenido no textual | *<img>* | Falta de texto alternativo | Agregar atributo *alt* descriptivo. |
| 1.1.1 Contenido no textual | *<a>* | Enlaces con texto poco claro o no descriptivo | Se aconseja que el texto de los enlaces sea descriptivo de su propósito o destino. Adicionalmente, se debe considerar el uso de *aria-label*. |
| 1.1.1 Contenido no textual | *<button>* | Botones carentes de texto o iconos sin *aria-label* | Agregar texto visible o *aria-label.* |
| 1.1.1 Contenido no textual | *<input type="image">* | Falta de atributo *alt* | Agregar *alt* que describa la función del botón. |
| 1.1.1 Contenido no textual | *<svg>* | Elementos SVG sin *aria-label* o *aria-hidden* | Incluir *<title>* y *<desc>* o usar *aria-label* o *aria-hidden="true"* si es decorativo. |
| 1.1.1 Contenido no textual | *<table>* | Ausencia de *<caption>* o uso incorrecto para maquetación y sin descripciones adicionales para contenido complejo. | Usar *<caption>* y etiquetas *<th>* con *scope*.<br><br>Evitar tablas con fines de maquetación sin *role="presentation"*. |
| 1.1.1 Contenido no textual | *<form>* | Campos de formulario sin etiquetas visibles o *aria-label* | Asociar cada campo con una etiqueta *<label>*. |
| 1.1.1 Contenido no textual | *<select>* | Menú desplegable sin descripción clara.<br><br>Opciones sin etiquetas claras | Incluir label asociado a la selección.<br><br>*<label for="country">País:</label><select id="country">...</select>*. |

| | | | |
|---|---|---|---|
| 1.1.1 Contenido no textual | *<textarea>* | Área de texto sin etiqueta visible o atributo *label* o *aria-label*. | Vincular con *<label>*. |
| 1.1.1 Contenido no textual | *<figure>* | Ausencia de *<figcaption>* para contenido visual | Incluir *<figcaption>*. |
| 1.1.1 Contenido no textual | *<picture>* | Ausencia de *alt* en las imágenes fuente | Incluir *alt* en *<img>* dentro de *<picture>*. |
| 1.1.1 Contenido no textual | *<object>* | Ausencia de contenido alternativo | Incluir contenido alternativo dentro del *<object>*. |
| 1.1.1 Contenido no textual | *<video>* | Ausencia de subtítulos o descripción textual.<br><br>Video sin subtítulos, descripción o controles. | Incluir subtítulos y descripción textual. |
| 1.1.1 Contenido no textual | *<audio>* | Audio presentado sin controles ni transcripción. | Proporcionar una transcripción textual. |
| 1.1.1 Contenido no textual | *<iframe>* | Elementos *iframe* sin título o con título poco descriptivo. | Usar atributo *title* para describir el contenido. |
| 1.1.1 Contenido no textual | *<area>* | Falta de atributo *alt* en mapas de imagen | Agregar *alt* descriptivo a cada área. |
| 1.1.1 Contenido no textual | *<canvas>* | *Elementos canvas* sin descripción alternativa | Incluir texto alternativo dentro del *<canvas>*. |

| | | | |
|---|---|---|---|
| 1.1.1 Contenido no textual | <*map*> | Mapas sin explicaciones contextuales asociadas.<br><br>Falta de *alt* en las áreas del mapa | Asegurar que cada <*area*> tenga *alt.* |
| 1.1.1 Contenido no textual | <*embed*> | Contenido embebido sin alternativa textual | Proporcionar descripción textual del contenido. |
| 1.1.1 Contenido no textual | <*track*> | Ausencia de subtítulos en contenido multimedia | Incluir pistas relevantes. |
| 1.1.1 Contenido no textual | <*source*> | Ausencia de subtítulos asociados en medios.<br><br>Contenido multimedia sin subtítulos | Verificar que el <*video*> o <*audio*> relacionado tenga <*track*> para subtítulos. |
| 1.1.1 Contenido no textual | <*applet*> | Ausencia de texto alternativo (obsoleto, pero aún evaluado) | Incluir contenido alternativo. |
| 1.1.1 Contenido no textual | <*progress*> | Ausencia de descripción o nombre accesible | Usar *aria-label* o texto asociado. |
| 1.1.1 Contenido no textual | <*meter*> | Ausencia de descripción o nombre accesible | Usar *aria-label* o texto asociado. |
| 1.1.1 Contenido no textual | <*time*> | Ausencia de contexto para la fecha | Incluir una descripción clara si no es evidente. |
| 1.1.1 Contenido no textual | <*abbr*> | Ausencia de texto descriptivo para acrónimos | Incluir atributo *title* con la expansión. |

| | | | |
|---|---|---|---|
| 1.1.1 Contenido no textual | *<q>* | Ausencia de contexto para citas | Incluir el atributo *cite* si corresponde. |
| 1.1.1 Contenido no textual | *<caption>* | Ausencia de descripción de la tabla | Usar *<caption>* para describir la tabla. |
| 1.1.1 Contenido no textual | *<legend>* | Ausencia de descripción del grupo | Usar *<legend>* en conjunto con *<fieldset>*. |
| 1.1.1 Contenido no textual | *<figcaption>* | Ausencia de descripción del contenido | Incluir *<figcaption>*. |
| 1.1.1 Contenido no textual | *<datalist>* | Ausencia de descripción del propósito | Asociar con *<label>* y el *input* correspondiente. |
| 1.2.1 Solo audio y solo vídeo (grabado) | *<audio>* | Ausencia de transcripción | Proveer transcripción textual. |
| 1.2.1 Solo audio y solo vídeo (grabado) | *<video>* | Sin subtítulos. Sin audiodescripción | Descripción textual para video. |
| 1.2.2 Subtítulos (grabados) | *<video>* | Videos grabados sin subtítulos | Agregar subtítulos en formato compatible con el reproductor. |
| 1.2.2 Subtítulos (grabados) | *<track>* | No usa *<track>* para subtítulos | Usar el elemento *<track>* para subtítulos. |
| 1.2.3 Audiodescripción o medio alternativo (grabado) | *<video>, <track>* | Ausencia de audiodescripción o pistas de áudio alternativas.<br><br>No proporciona enlace a una alternativa textual | Proporcionar una segunda pista de audio o una descripción textual.<br><br>Ubicar un enlace al lado del contenido multimedia que apunte a una versión accesible. |

| | | | |
|---|---|---|---|
| 1.2.3 Audiodescripción o medio alternativo (grabado) | *<object>* | Uso incorrecto del *<object>* sin alternativa | Utilizar el *<body>* del *<object>* para proporcionar la alternativa. |
| 1.2.4 Subtítulos (en directo) | No aplica | No detectable mediante etiquetas *HTML*. | Requiere intervención en multimedia, contenido textual, diseño *CSS*, *Scripts*, o herramientas externas. |
| 1.2.5 Audio descripción (grabado) | *<video>*, *<track>* | El video no tiene audiodescripción o no se proporciona una opción para activar una pista de audiodescripción. | Proveer una versión alternativa del video que incluya audiodescripción embebida o pista de audiodescripción accesible mediante *<track>*. |
| 1.2.6 Lenguaje de señas (pregrabado) | No aplica | No detectable mediante etiquetas *HTML*. | Requiere intervención en multimedia, contenido textual, diseño *CSS*, *Scripts*, o herramientas externas. |
| 1.2.7 Descripción de audio extendida (pregrabada) | No aplica | No detectable mediante etiquetas *HTML*. | Requiere intervención en multimedia, contenido textual, diseño *CSS*, *Scripts*, o herramientas externas. |
| 1.2.8 Medios alternativos (pregrabados) | No aplica | Depende del comportamiento, multimedia o estilos. | No puede resolverse exclusivamente con *HTML*. Requiere *CSS*, *JS*, multimedia o ajustes de diseño externos. |
| 1.2.9 Solo Audio (en vivo) | No aplica | Depende del comportamiento, multimedia o estilos. | No puede resolverse exclusivamente con *HTML*. Requiere *CSS*, *JS*, multimedia o ajustes de diseño externos. |

| | | | |
|---|---|---|---|
| 1.3.1 Información y relaciones | *<table>*, *<fieldset>*, *<label>*, *<caption>*, *<legend>*, *<ul>*, *<ol>*, *<dl>*, *<header>*, *<footer>*, *<main>*, *<nav>*, *<section>*, *<article>* | Ausencia de estructura semántica clara que relacione la información. | Usar correctamente estructuras *HTML* como listas, encabezados, roles, etiquetas y agrupaciones para mostrar relaciones. |
| 1.3.2 Secuencia significativa | * *, ** | Uso de etiquetas *HTML* genéricas sin respetar el orden lógico del contenido. | Utilizar etiquetas semánticas (*<main>, <article>, <section>, <header>*, etc.) que reflejen la secuencia correcta en el *DOM* [15]. |
| 1.3.3 Características sensoriales | *<label>*, *<p>*, ** | Instrucciones como “haz clic en el botón azul” o “ve a la derecha” sin una referencia textual. | Incluir texto descriptivo como haz clic en el botón “Enviar” o usa *aria-label* para asociar propósito. |
| 1.3.4 Orientación | No aplica | Este criterio depende de comportamiento, multimedia o estilos. | El contenido no debe estar restringido a una orientación específica (horizontal o vertical), lo cual se maneja desde el *CSS* y diseño responsive, no *HTML*. |
| 1.3.5 Identificación del propósito de entrada | *<input>* (con atributos *autocomplete*) | Campos de formulario sin atributos que identifiquen su propósito (por ejemplo, *autocomplete=“name”*). | Agregar atributos *autocomplete* apropiados a los campos del formulario para facilitar el llenado automático. |

| | | | |
|---|---|---|---|
| 1.3.6 Identificación del propósito | *<input>*, *<button>*, *<a>* (requiere atributos *ARIA* y *CSS* personalizados) | No se proporciona información semántica suficiente para herramientas de personalización. | Requiere semántica enriquecida mediante atributos *ARIA*, microdatos o *RDFa*, además de *HTML*. No se resuelve solo con *HTML* básico. |
| 1.4.1 Uso del color | Todas (contextual). por ejemplo **, * * con *style="color"* | Uso exclusivo de color para transmitir información (ejemplo: solo rojo para errores) | Acompañar el color con texto, íconos u otras señales accesibles. No es una solución solo *HTML*, requiere diseño visual complementario. |
| 1.4.2 Control del audio | *<audio>*, *<video>* | Audio en reproducción automática por más de 3 segundos sin controles | Incluir controles con *<audio controls>* o evitar la reproducción automática con *autoplay* y *muted* correctamente configurados. |
| 1.4.3 Contraste (mínimo) | Todas (contextual) | Texto con bajo contraste frente al fondo | No se resuelve solo con *HTML*; requiere ajuste de estilos *CSS* para garantizar relación de contraste adecuada. |
| 1.4.4 Cambio de tamaño del texto | Todas (contextual) | Texto que no escala correctamente al aumentar tamaño del navegador | No se resuelve solo con *HTML*; requiere uso adecuado de unidades relativas en *CSS* y diseño adaptable. |
| 1.4.5 Imágenes de texto | *<img>* | Uso de imágenes en lugar de texto real | Evitar *<img>* para representar texto; usar etiquetas textuales reales como **, *<p>* y *<h1>–<h6>*. |
| 1.4.6 Contraste (mejorado) | No aplica | | Este criterio implica un nivel de contraste mínimo de 7:1 para texto normal, lo cual debe evaluarse a través de estilos *CSS* y herramientas de contraste. |

| | | | |
|---|---|---|---|
| 1.4.7 Sonido de fondo bajo o ausente | No aplica | | Este criterio se relaciona con el diseño del audio (mezcla de sonidos), no con etiquetas *HTML*. Aplica más a contenido multimedia. |
| 1.4.8 Presentación visual | No aplica | | Aunque se presenta en *HTML*, este criterio depende del control visual mediante CSS (espaciado, colores, longitud de línea). El *HTML* solo debería estar estructurado adecuadamente para permitir estos ajustes. |
| 1.4.9 Imágenes de texto (sin excepciones) | *<img>* | Uso de imagen con texto decorativo o funcional que no puede adaptarse o traducirse. | Reemplazar imagen por texto *HTML* real. Solo se permiten si son esenciales (ejemplo: logotipos). |
| 1.4.10 Reajuste | Todas | Contenido que no se adapta al redimensionar pantalla | *HTML* debe ser estructurado semánticamente y con contenedores flexibles; requiere también uso adecuado de *CSS.* |
| 1.4.11 Contraste no textual | Todas con íconos, botones, bordes | Ausencia de contraste suficiente en íconos, bordes, indicadores | No se resuelve completamente con *HTML*; requiere diseño y estilos *CSS* con contraste adecuado. |
| 1.4.12 Espaciado del texto | Todas | Espaciado inadecuado entre líneas, párrafos o letras | Debe asegurarse mediante estilos *CSS*. *HTML* debe usar estructuras correctas (*<p>, <li>, <h1>*, etc.). |
| 1.4.13 Contenido en *hover* o *focus* | Interactivas: *<button>, <a>, <input>*, etc. | Información emergente que desaparece al mover el foco | No es solucionable solo desde *HTML*. Requiere *JavaScript* accesible y uso correcto de atributos *ARIA*. |

| | | | |
|---|---|---|---|
| 2.1.1 Teclado | Todas las etiquetas interactivas (*<a>, <button>, <input>*, etc.) | Componentes que solo pueden ser activados con el *mouse*. | Todos los elementos interactivos pueden ser accedidos y activados con el teclado, utilizando atributos como *tabindex* y asegurando la funcionalidad con eventos de teclado. |
| 2.1.2 Sin trampas para el foco del teclado | Elementos con *tabindex* | El *foco* queda atrapado en un componente, sin opción de salir usando el teclado. | Usar *tabindex* de forma responsable y asegurar que el *foco* pueda moverse libremente hacia adelante y hacia atrás. |
| 2.1.3 Teclado (sin excepciones) | No aplica | Depende del comportamiento, multimedia o estilos. | No puede resolverse exclusivamente con *HTML*. Requiere *CSS*, *JS*, multimedia o ajustes de diseño externos. |
| 2.1.4 Atajos de teclado | *<body>* y eventos con atributos como *accesskey* | Conflictos entre atajos del sistema y del sitio web. | Evitar el uso de *accesskey* o proporcionar opciones para desactivarlos o personalizarlos. |
| 2.2.1 Tiempo ajustable | No aplica | Depende del comportamiento, multimedia o estilos. | No puede resolverse exclusivamente con *HTML*. Requiere *CSS*, *JS*, multimedia o ajustes de diseño externos. |
| 2.2.2 Poner en pausa, detener, ocultar | *<video>*, *<audio>*, *<marquee>* | Medios que inician automáticamente sin controles para detener o pausar. | Agregar atributos como *"controls"* o proveer botones de pausa/detención accesibles. |
| 2.2.3 Sin tiempo | No aplica | Contenido con límite de tiempo obligatorio sin opción a desactivarlo. | No solucionable únicamente con *HTML*. Requiere lógica de control por programación para eliminar restricciones de tiempo. |
| 2.2.4 Interrupciones | No aplica | Notificaciones automáticas o redirecciones inesperadas. | Requiere control mediante scripts o configuración del servidor. *HTML* no puede gestionar interrupciones. |

| | | | |
|---|---|---|---|
| 2.2.5 Re-autentificación | No aplica | Pérdida de datos tras re-autenticación. | Requiere gestión del estado y almacenamiento en backend. No se resuelve con *HTML*. |
| 2.2.6 Timeouts | No aplica | Expiración de sesión sin advertencia o sin guardar progreso. | Se debe implementar mediante lógica de backend o *JavaScript*. No es solucionable desde *HTML* puro. |
| 2.3.1 Umbral de tres destellos o menos | No aplica | Este criterio depende de comportamiento, multimedia o estilos. | No puede resolverse exclusivamente con *HTML*. Requiere *CSS*, *JS*, multimedia o ajustes de diseño externos. |
| 2.3.2 Tres destellos | No aplica | Depende del comportamiento, multimedia o estilos. | No puede resolverse exclusivamente con *HTML*. Requiere *CSS*, *JS*, multimedia o ajustes de diseño externos. |
| 2.3.3 Animación de interacciones | No aplica | Depende del comportamiento, multimedia o estilos. | No puede resolverse exclusivamente con *HTML*. Requiere *CSS*, *JS*, multimedia o ajustes de diseño externos. |
| 2.4.1 Evitar bloques | *<a>*, *<nav>*, *<header>*, *<main>*, *<footer>* | Ausencia de enlaces para saltar bloques repetitivos de contenido | Incluir un enlace *"skip to content"* al inicio de la página que apunte al *<main>* u otra sección principal. |
| 2.4.2 Titulado de páginas | *title* | Página sin título o con un título genérico | Agregar un elemento *<title>* claro y específico dentro del *<head>*. |
| 2.4.3 Orden del foco | Todas con atributos *tabindex* | Orden de navegación por teclado ilógico | Asegurar un orden lógico con *tabindex* y estructura semántica clara. |
| 2.4.4 Propósito de los enlaces (en contexto) | *<a>* | Texto de enlace ambiguo como 'click aquí' | Utilizar textos de enlace descriptivos que indiquen el destino o acción. |

| | | | |
|---|---|---|---|
| 2.4.5 Múltiples vías | *<nav>, <a>* | Solo una forma de acceder al contenido | Proporcionar varias vías de navegación como menú, buscador, mapa del sitio |
| 2.4.6 Encabezados y etiquetas | *<h1>-<h6>, <label>* | Encabezados ausentes, desordenados o etiquetas poco claras | Usar *<h1>* a *<h6>* de forma jerárquica; *<label>* descriptivo para formularios. |
| 2.4.7 Foco visible | Todos los elementos interactivos | Elementos que no muestran *foco* al navegar con teclado | Asegurar que el estilo *:focus* esté visible para todos los elementos navegables. |
| 2.4.8 Ubicación | No aplica | Depende del comportamiento, multimedia o estilos. | No puede resolverse exclusivamente con *HTML*. Requiere *CSS*, *JS*, multimedia o ajustes de diseño externos. |
| 2.4.9 Propósito de los enlaces (solo enlaces) | *<a>* | Enlaces sin contexto comprensible fuera de su entorno | Proveer contexto completo dentro del texto del enlace o usar *aria-label.* |
| 2.4.10 Encabezados de sección | *<h2>-<h6>* | Contenido largo sin subtítulos o divisiones temáticas | Usar subtítulos jerárquicos para dividir secciones de contenido. |
| 2.5.1 Gestos del puntero | No aplica | Uso exclusivo de gestos complejos como arrastrar y soltar | Proporcionar métodos alternativos accesibles como botones o enlaces simples; requiere soporte *JS* o *ARIA*. |
| 2.5.2 Cancelación del puntero | No aplica | Depende del comportamiento, multimedia o estilos. | No puede resolverse exclusivamente con *HTML*. Requiere *CSS*, *JS*, multimedia o ajustes de diseño externos. |

| | | | |
|---|---|---|---|
| 2.5.3 Etiqueta en el nombre | *<button>, <input>, <label>* | Etiqueta visual y atributo accesible no coinciden | Asegurar que el texto visible coincida con el atributo *aria-label*, *label* o *name* del control. |
| 2.5.4 Activación por movimiento | No aplica | Uso de sensores de movimiento sin alternativas | Ofrecer control alternativo mediante teclado o *mouse*; no es solucionable solo con *HTML*. |
| 2.5.5 Tamaño del objetivo | No aplica | Depende del comportamiento, multimedia o estilos. | No puede resolverse exclusivamente con *HTML*. Requiere *CSS*, *JS*, multimedia o ajustes de diseño externos. |
| 2.5.6 Mecanismos de entrada concurrentes | No aplica | Elementos que solo responden a un tipo de entrada (ejemplo: solo mouse) | Asegurar que todos los controles puedan usarse con teclado, pantalla táctil y voz, generalmente fuera del alcance de *HTML* puro. |
| 3.1.1 Idioma de la página | *<html>* | Ausencia del atributo *lang* en la etiqueta *<html>* | Incluir *lang="es"* (o el idioma correspondiente) en la etiqueta *<html>*. |
| 3.1.2 Idioma de las partes | *, <p>, * | Fragmentos de texto en otro idioma sin declarar idioma específico | Agregar el atributo *lang* al elemento que contiene el texto en otro idioma. |
| 3.1.3 Palabras inusuales | No aplica | Términos técnicos o poco comunes sin explicación | Se recomienda usar *<abbr>*, *<dfn>*, *<title>* u ofrecer glosario; no es solucionable solo desde *HTML*. |
| 3.1.4 Abreviaturas | *<abbr>* | Abreviaturas no explicadas | Utilizar *<abbr title="Significado">* para indicar el significado de la abreviatura. |
| 3.1.5 Nivel de lectura | No aplica | Contenido demasiado técnico o complejo | Requiere redacción clara y simplificación textual; no se soluciona directamente con *HTML*. |

| | | | |
|---|---|---|---|
| 3.1.6 Pronunciación | No aplica | Ambigüedad en la pronunciación de palabras | No existe una solución estandarizada con *HTML* actual; se requiere explicación contextual o audio complementario. |
| 3.2.1 Al recibir el foco | *<input>, <button>, <a>* | El *foco* provoca cambios inesperados (redirecciones automáticas, ventanas emergentes) | Evitar el uso de *onfocus* que cambie el contexto. No asociar eventos de cambio de página con el *foco*. |
| 3.2.2 Al recibir entradas | *<select>, <input>* | Un cambio en el valor activa una acción inesperada sin aviso | Evitar usar *onchange* para redirecciones o actualizaciones automáticas sin notificar al usuario. |
| 3.2.3 Navegación coherente | *<nav>, <ul>, <a>* | Elementos de navegación cambian de orden entre páginas similares | Mantener el mismo orden y contenido del menú de navegación en todas las páginas del sitio. |
| 3.2.4 Identificación coherente | *<button>, <link>, <input>* | Mismos elementos con diferentes nombres o funciones en páginas distintas | Usar etiquetas, nombres y funciones consistentes en todo el sitio. |
| 3.2.5 Cambios a petición | No aplica | Cambios de contexto se realizan sin el consentimiento del usuario | Requiere interacción activa del usuario para cambios de contexto, controlado vía *scripts* y lógica del sitio, no solo con *HTML*. |

| | | | |
|---|---|---|---|
| 3.3.1 Identificación de errores | *<form>, <input>, <textarea>, <select>* | No se indican errores cuando el usuario envía el formulario con campos inválidos o incompletos | Usar elementos que permitan mostrar mensajes de error visibles junto a los campos incorrectos; usar atributos como *required*, *pattern*, y mensajes con *aria-live* o etiquetas visibles. |
| 3.3.2 Etiquetas o instrucciones | *<label>, <input>, <textarea>, <select>* | Campos sin etiquetas o sin instrucciones claras de uso | Asegurar que cada campo tenga una *<label>* asociada o texto de ayuda (*aria-describedby*, *placeholder*, texto anterior al campo). |
| 3.3.3 Sugerencias ante errores | *<form>, <input>, <textarea>* | No se ofrecen sugerencias sobre cómo corregir un error en los formularios | Incluir mensajes que indiquen cómo corregir el error junto al campo afectado o usando atributos *aria-describedby* para enlazar con el texto explicativo. |
| 3.3.4 Prevención de errores (legales, financieros, datos) | No aplica | El sistema permite errores graves sin verificación previa | Requiere implementación lógica con validaciones del lado cliente o servidor, *HTML* no es suficiente por sí solo. |
| 3.3.5 Ayuda | No aplica | No se proporciona ayuda accesible para completar formularios complejos | Solución requiere mecanismos de ayuda contextual o soporte externo, no implementable solo con *HTML*. |
| 3.3.6 Prevención de errores (todos) | No aplica | Cambios irreversibles sin confirmación | Se requiere lógica adicional (*JavaScript* o *backend*) para confirmaciones antes de acciones críticas. |

| | | | |
|---|---|---|---|
| 4.1.1 Procesamiento | General (todo el *HTML*) | Uso incorrecto de elementos *HTML*, etiquetas mal anidadas, atributos mal escritos o estructuras no válidas | Validar el código *HTML* con herramientas como *W3C* Validator para asegurar que el documento sea procesado correctamente por tecnologías de asistencia. |
| 4.1.2 Nombre, función, valor | *<button>, <input>, <select>, <textarea>, <a>*, componentes personalizados | Controles sin nombre o rol accesible, atributos *aria* mal aplicados | Usar etiquetas *<label>*, atributos *aria-label*, *aria-labelledby*, *role*, *value*, *name*, según sea necesario para definir la función y valor de cada *control*. |
| 4.1.3 Mensajes de estado | * *, **, o cualquier contenedor dinámico | Cambios de estado (ejemplo: éxito al enviar un formulario) no son anunciados por tecnologías de asistencia | Utilizar *aria-live*, *aria-atomic*, y *aria-relevant* para que los mensajes dinámicos sean anunciados automáticamente por lectores de pantalla. |

A partir de los 86 criterios de éxito presentes en las *WCAG* 2.2 [2], se identificó:

- 47 criterios de éxito pueden ser abordados directamente desde *HTML*.
- 39 criterios de éxito no son aplicables exclusivamente mediante *HTML*, ya que dependen de otras tecnologías (*CSS*, *JavaScript*, *ARIA*, contenido multimedia, etc.).

Esta clasificación se basó en un análisis detallado de los criterios de éxito, presentados en la Tabla 1, y su aplicabilidad, considerando las técnicas suficientes oficialmente documentadas por el *W3C*, y excluyendo aquellos criterios de éxito que requieren tecnologías complementarias como hojas de estilo (*CSS*), contenido audiovisual, o *scripts* de interacción. Esta distinción permite enfocar la estrategia en lo que realmente puede resolverse desde el desarrollo del lenguaje de marcado (o *HTML*) en los documentos *HTML*, promoviendo una implementación más eficaz de la accesibilidad web.

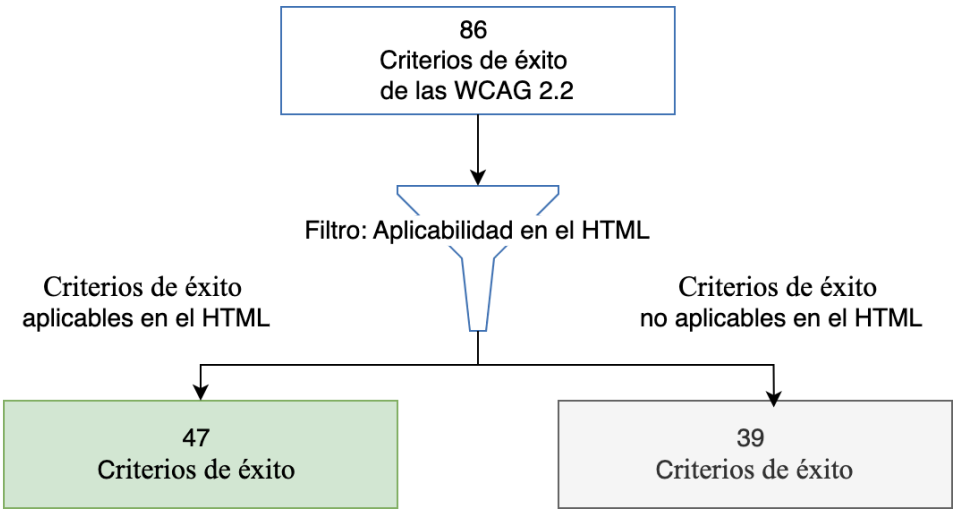


*Figura 5. Filtrado de criterios de éxito de la WCAG 2.2 aplicables a elementos HTML*

Para complementar la identificación de los criterios de éxito directamente aplicables en los documentos *HTML* (ver Tabla 3), se presenta una tabla con la relación entre dichos criterios de éxito y las técnicas suficientes establecidas oficialmente por el *W3C*. Dichas técnicas representan métodos confiables para satisfacer los requisitos de accesibilidad definidos, y ofrecen ejemplos prácticos que pueden ser implementados directamente en los documentos *HTML*. La Tabla 4 consolida esta información de forma estructurada y conforme a la *WCAG* 2.2 [2] ,[13] y [14].

Tabla 4. Relación entre criterios de éxito, etiquetas y atributos *HTML* y técnicas suficientes

| **Criterio de Éxito** | **Etiqueta o atributo del *HTML*** | **Técnica suficiente (*W3C*)** |
|---|---|---|
| 1.1.1 Contenido no textual | *<img>* | H37: Usar el atributo *alt* en el elemento *<img>*. |
| 1.1.1 Contenido no textual | *<input type="image">* | H36: Usar el atributo *alt* en el elemento *<input type="image">*. |
| 1.1.1 Contenido no textual | *<area>* | H67: Usar el atributo alt en el elemento <area> de mapas de imagen. |
| 1.1.1 Contenido no textual | *<area>* | H24: Proporcionar texto alternativo para áreas de un mapa de imagen. |
| 1.1.1 Contenido no textual | *<object>* | G94: Proporcionar texto alternativo para imágenes que transmiten información. |

| | | |
|---|---|---|
| 1.1.1 Contenido no textual | *<object>* | G196: Usar una alternativa textual para los elementos gráficos que no transmiten información (decorativos). |
| 1.1.1 Contenido no textual | *<svg>* | G94: Proporcionar texto alternativo para imágenes que transmiten información. |
| 1.1.1 Contenido no textual | *<svg>* | G196: Usar una alternativa textual para los elementos gráficos que no transmiten información (decorativos). |
| 1.1.1 Contenido no textual | *<applet>* | H53: Usar el atributo *alt* en el elemento *<applet>*. |
| 1.1.1 Contenido no textual | *<button>* | G95: Proporcionar texto alternativo para botones gráficos. |
| 1.1.1 Contenido no textual | *<a>* | G95: Proporcionar texto alternativo para botones gráficos. |
| 1.1.1 Contenido no textual | *<figure>* | H49: Uso de marcado semántico para asociar texto con contenido no textual (en este caso, *<figcaption>* dentro de *<figure>*). |
| 1.1.1 Contenido no textual | *<iframe>* | H64: Usar el atributo *title* en el elemento *<iframe>* para identificar su propósito. |
| 1.1.1 Contenido no textual | *<embed>* | H95: Proporcionar una alternativa accesible cuando se usa *<embed>*. (Usar contenido alternativo o asegurar la equivalencia mediante etiquetas externas). |
| 1.1.1 Contenido no textual | *<applet>* | H53: Usar el atributo alt en el *<applet>*. |
| 1.1.1 Contenido no textual | *<canvas>* | H96: Proveer contenido de texto alternativo dentro de un *<canvas>* usando contenido *fallback* entre las etiquetas *<canvas></canvas>*. |
| 1.1.1 Contenido no textual | *<audio>* | G158: Proporcionar alternativa textual (una transcripción textual del contenido de audio). |
| 1.1.1 Contenido no textual | *<object>* | G158: Proporcionar alternativa textual para solo audio cuando el objeto inserta contenido solo de audio. |
| 1.1.1 Contenido no textual | *<object>* | G159: Proporcionar alternativa textual para solo video cuando el objeto inserta contenido solo de video. |
| 1.1.1 Contenido no textual | *<video>* | G159: Proporcionar alternativa textual (una transcripción textual que describa el contenido del video). |

| | | |
|---|---|---|
| 1.2.2 Subtítulos (grabados) | <*video*> | G87: Usar subtítulos sincronizados para contenido de vídeo. |
| 1.2.2 Subtítulos (grabados) | <*video*> | G93: Proporcionar archivo de subtítulos sincronizado referenciado mediante <*track*>. |
| 1.2.2 Subtítulos (grabados) | <*track*> | G93: Usar <*track*> con atributo *kind*="*subtitles"* para asociar archivo de subtítulos sincronizados al <*video*>. |
| 1.2.3 Audiodescripción o medio alternativo | <*video*> | G78: Proporcionar audiodescripción sincronizada mediante una pista de audio adicional o incorporada. |
| 1.2.3 Audiodescripción o medio alternativo | <*video*> | G173: Proporcionar descripción textual alternativa en la página que explique lo que sucede visualmente en el vídeo. |
| 1.2.3 Audiodescripción o medio alternativo | <*object*> | G78: Usar objeto que incluya audiodescripción sincronizada dentro del contenido multimedia. |
| 1.2.3 Audiodescripción o medio alternativo | <*object*> | G173: Incluir una descripción textual alternativa externa vinculada al contenido del <*object*>. |
| 1.2.5 Audiodescripción (grabado) | <*video*> | G78: Incluir una pista de audiodescripción sincronizada mediante el elemento <*track*> con *kind*="*descriptions"*. |
| 1.2.5 Audiodescripción (grabado) | <*video*> | G173: Proporcionar una transcripción descriptiva del contenido visual en texto fuera del vídeo. |
| 1.2.5 Audiodescripción (grabado) | <*object*> | G78: Incorporar audiodescripción sincronizada dentro del archivo embebido. |
| 1.2.5 Audiodescripción (grabado) | <*object*> | G173: Ofrecer una descripción textual alternativa vinculada al contenido embebido. |
| 1.3.1 Información y relaciones | <*table*> | H51: Usar <*table*> para marcar datos tabulares. |
| 1.3.1 Información y relaciones | <*table*> | H39: Usar <*caption*> para identificar la tabla. |
| 1.3.1 Información y relaciones | <*table*> | H73: Usar <*thead*>, <*tbody*>, <*tfoot*> para agrupar filas. |

| | | |
|---|---|---|
| 1.3.1 Información y relaciones | *<table>* | H63: Usar *<th>* para encabezados. |
| 1.3.1 Información y relaciones | *<table>* | H77: Asociar celdas de datos y encabezados mediante *headers* y *id*. |
| 1.3.1 Información y relaciones | *<caption>* | H39: Usar *<caption>* para proporcionar título de la tabla. |
| 1.3.1 Información y relaciones | *<th>* | H63: Usar *<th>* para identificar encabezados. |
| 1.3.1 Información y relaciones | *<th>* | H77: Asociar celdas de datos con encabezados mediante *headers* y *id*. |
| 1.3.1 Información y relaciones | *<fieldset>* | H85: Usar *<fieldset>* para agrupar controles de formulario. |
| 1.3.1 Información y relaciones | *<legend>* | H85: Usar *<legend>* dentro de *<fieldset>* para describir el grupo. |
| 1.3.1 Información y relaciones | *<label>* | H71: Usar *<label>* para asociar texto con campos de formulario. |
| 1.3.1 Información y relaciones | *<h1>–<h6>* | H42: Usar encabezados para estructurar jerárquicamente el contenido. |
| 1.3.1 Información y relaciones | *<ul>*, *<ol>*, *<dl>* | H48: Usar listas para agrupar ítems relacionados. |
| 1.3.1 Información y relaciones | *<section>* | H69: Debe incluir *<h1>–<h6>* como título descriptivo. |
| 1.3.1 Información y relaciones | *<article>* | H69: Debe incluir *<h1>–<h6>* como título descriptivo. |
| 1.3.2 Secuencia significativa | *<ol>* | H49: Uso correcto de listas *HTML* semánticas (*<ol>*, *<ul>*, *<li>*) para representar contenido estructurado. |
| 1.3.2 Secuencia significativa | *<ol>* | G57: Asegurar que el orden de presentación del contenido en el código sea lógico y significativo para los usuarios. |
| 1.3.2 Secuencia significativa | *<ul>* | H49: Uso correcto de listas *HTML* semánticas (*<ol>*, *<ul>*, *<li>*) para representar contenido estructurado. |
| 1.3.2 Secuencia significativa | *<ul>* | G57: Asegurar que el orden de presentación del contenido en el código sea lógico y significativo para los usuarios. |
| 1.3.2 Secuencia significativa | *<li>* | H49: Uso correcto de listas *HTML* semánticas (*<ol>*, *<ul>*, *<li>*) para representar contenido estructurado. |

| | | |
|---|---|---|
| 1.3.2 Secuencia significativa | *<li>* | G57: Asegurar que el orden de presentación del contenido en el código sea lógico y significativo para los usuarios. |
| 1.3.2 Secuencia significativa | *<table>* | G57: Asegurar que el orden de presentación del contenido en el código sea lógico y significativo para los usuarios. |
| 1.3.2 Secuencia significativa | *<thead>* | G57: Asegurar que el orden de presentación del contenido en el código sea lógico y significativo para los usuarios. |
| 1.3.2 Secuencia significativa | *<tbody>* | G57: Asegurar que el orden de presentación del contenido en el código sea lógico y significativo para los usuarios. |
| 1.3.3 Características sensoriales | *<p>*, **, **, ** | G96: Usar instrucciones que no dependan solo del color, forma o posición. |
| 1.3.5 Identificación del propósito de entrada | *<input>*, *<select>*, *<textarea>*, *<form>* | G98: Utilizar *autocomplete* para campos de formulario con propósito conocido. |
| 1.3.6 Identificación del propósito | *<input>*, *<select>*, *<textarea>* | G211: Usar atributos *autocomplete* apropiados para proporcionar información semántica sobre el propósito del campo de entrada, como *autocomplete*="*name"*, *autocomplete*="*email"*, *autocomplete*="*bday"*, etc. |
| 1.4.1 Uso del color | *<a>*, **, *<button>* y cualquier otro con estilo aplicado. | G14: Usar más que solo color para transmitir información (por ejemplo, añadir subrayado, iconos o texto adicional). |
| 1.4.1 Uso del color | *<a>*, **, *<button>* y cualquier otro con estilo aplicado. | G205: Incluir indicadores visuales además del color para diferenciar contenido (como bordes o estilos de texto). |
| 1.4.2 Control del audio | *<audio>* | G60: Proporcionar un mecanismo para pausar, detener o controlar el volumen del audio que se reproduce automáticamente por más de 3 segundos. Esto puede hacerse incluyendo los controles nativos del elemento *<audio>* usando *controls*, o mediante controles personalizados visibles. |
| 1.4.3 Contraste (mínimo) | *<a>*, *<h1>*–*<h6>*, *<p>*, ** | G18: Asegurar una relación de contraste de al menos 4.5:1 entre el texto y el fondo. Puede validarse con herramientas como el *Contrast Checker*. |

| | | |
|---|---|---|
| 1.4.3 Contraste (mínimo) | *<a>*, *<h1>–<h6>*, *<p>*, ** **. Texto sobre imágenes o gradientes | G145: Usar una técnica para colocar texto sobre fondo sólido o proporcionar un fondo opaco para asegurar el contraste mínimo requerido. |
| 1.4.4 Cambio de tamaño del texto | *<html>*, *<body>*, *<p>*, * *, **, *<a>*, *<h1>–<h6>*, etc. | G142: Usar unidades relativas como *em*, *rem*, % para definir el tamaño del texto, lo que permite que este pueda ser redimensionado por el usuario sin pérdida de funcionalidad o contenido. |
| 1.4.4 Cambio de tamaño del texto | *<html>*, *<body>*, *<p>*, * *, **, *<a>*, *<h1>–<h6>*, etc. | C28: Asegurar que el contenido pueda escalarse hasta un 200 % sin pérdida de contenido o funcionalidad, utilizando diseños fluidos y evitando el uso exclusivo de *px* en el tamaño de fuente. |
| 1.4.5 Imágenes de texto | *<img>* | G140: Usar texto real en lugar de imágenes de texto, excepto cuando se requiera una presentación específica, como en logotipos. |
| 1.4.5 Imágenes de texto | *<img>* | G148: Garantizar que la información visual transmitida mediante imágenes de texto también esté disponible en formato de texto accesible. |
| 1.4.9 Imágenes de texto (sin excepciones) | *<img>*, *<canvas>*, *<svg>*, *<object>* | G140: Evitar imágenes de texto y usar texto *HTML* real siempre que sea posible. |
| 1.4.9 Imágenes de texto (sin excepciones) | *<img>*, *<canvas>*, *<svg>*, *<object>* | C22: Usar *CSS* para reemplazar texto visual sin perder accesibilidad (por ejemplo, ocultar texto real y mostrar imagen decorativa solo visualmente). |
| 1.4.10 Reajuste | *<a>*, *<body>*, * *, *<html>*, *<img>*, *<main>*, *<p>*, *<section>* | C32: Usar diseño fluido y contenedores que se ajusten al ancho de la pantalla sin provocar desplazamiento horizontal. |
| 1.4.10 Reajuste | *<a>*, *<body>*, * *, *<html>*, *<img>*, *<main>*, *<p>*, *<section>* | C31: Evitar el uso de elementos fijos que dificulten el reajuste del contenido en pantallas pequeñas. |

| | | |
|---|---|---|
| 1.4.11 Contraste no textual | *<button>*, *<canvas>*, *<img>*, *<input>*, *<select>*, *<svg>* | G209: Proporcionar contraste suficiente de al menos 3:1 para los componentes de la interfaz de usuario y los gráficos significativos. |
| 1.4.11 Contraste no textual | *<button>*, *<canvas>*, *<img>*, *<input>*, *<select>*, *<svg>* | G195: Usar bordes o fondos con suficiente contraste para indicar el estado del control (ejemplo: enfoque o selección). |
| 1.4.12 Espaciado del texto | *<p>*, **, *<li>*, *<h1>–<h6>* | C36: Usar *CSS* para permitir que los usuarios ajusten el espaciado de texto (interlineado, espaciado entre palabras y letras) sin pérdida de contenido o funcionalidad. |
| 1.4.13 Contenido en *hover* o *focus* | *<button>*, *<a>*, *<input>*, *<label>* | G211: Asegurar que el contenido que aparece con *hover* o *focus* sea descartable sin mover el puntero ni perder el *foco*. Aunque gran parte del comportamiento depende de *CSS* o *JavaScript*, puede cumplirse parcialmente con atributos como *title* o estructuras claras con *aria-describedby*. |
| 2.1.1 Teclado | *<button>*, *<a>*, *<input>*, *<select>*, *<textarea>* *<fieldset>*, *<label>*, *<form>* | G202: Asegurar que toda funcionalidad esté disponible desde el teclado utilizando elementos *HTML* que sean naturalmente enfocables. |
| 2.1.1 Teclado | *<a>*, *<button>*, *<input>*, *<label>*, *<select>*, *<textarea>* | SCR20: Usar eventos de teclado además de eventos de ratón para la funcionalidad personalizada. |
| 2.1.1 Teclado | *<a>*, *<button>*, *<input>*, *<label>*, *<select>*, *<textarea>* | H91: Usar elementos de formulario *HTML* estándar para asegurar la funcionalidad con teclado sin intervención adicional. |
| 2.1.2 Sin trampas para el foco | *<a>*, *<button>*, *<input>* | G21: Asegurar que si se utiliza un componente que atrapa el *foco*, debe existir una forma (como tecla *Escape* o *Tab*) de salir de él usando el teclado. |
| 2.1.4 Atajos de teclado | *<button accesskey=“”>*, *<a accesskey=“”>* | G217: Proveer mecanismos para desactivar, remapear o activar solo cuando el componente tiene *foco*. El atributo *accesskey* en *HTML* permite definir atajos directamente. |

| | | |
|---|---|---|
| 2.2.2 Pausar, detener, ocultar | *<video>*, *<audio>*, *<marquee>* | F4: No iniciar movimiento automáticamente. |
| | *<video>*, *<audio>*, *<marquee>* | G186: Proporcionar controles de reproducción, pausa y parada para contenido multimedia. |
| 2.4.1 Evitar bloques | *<a href=“#main”>*, *<nav>*, *<main>* | G1: Proporcionar un mecanismo para saltar bloques repetitivos de contenido. Usar enlaces tipo “Saltar al contenido principal” o landmarks *HTML*. |
| 2.4.2 Titulado de páginas | *<title>* | G88: Proporcionar títulos de página descriptivos y relevantes mediante la etiqueta *<title>* del *<head>*. |
| 2.4.3 Orden del foco | Cualquier elemento con *tabindex* | G59: Asegurar que el orden del foco siga una secuencia lógica mediante el uso apropiado de *tabindex* y estructura *HTML* natural. |
| 2.4.4 Propósito de los enlaces | *<a>* | H30: Proporcionar texto de enlace significativo que tenga sentido por sí solo o por contexto. Evitar frases como “haz clic aquí”. |
| 2.4.5 Múltiples vías | *<nav>*, *<a>*, *<ul>*, *<ol>* | G125: Proporcionar múltiples formas de acceder a las páginas de un sitio web (barra de navegación, mapa del sitio, buscador, etc.). |
| 2.4.5 Múltiples vías | *<nav>*, *<a>*, *<ul>*, *<ol>* | G64: Proporcionar un mapa del sitio que permita acceder a las secciones principales. |
| 2.4.6 Encabezados y etiquetas | *<h1>* a *<h6>*, *<label>*, *aria-label* | H65: Usar etiquetas *<label>* para describir controles de formulario. |
| 2.4.6 Encabezados y etiquetas | *<h1>* a *<h6>*, *<label>*, *aria-label* | H42: Usar elementos de encabezado *HTML* para comunicar la estructura de la página. |
| 2.4.7 Foco visible | Cualquier elemento interactivo (*<button>*, *<a>*, *<input>*, etc.) | G149: Asegurar que el *foco* del teclado sea visible a medida que se navega por la página. Esto puede lograrse con estilos *CSS* como *:focus { outline: 2px solid blue; }*. |
| 2.4.9 Propósito de los enlaces (solo enlaces) | *<a>* | H33: Usar texto visible en el enlace que describa claramente su propósito. |
| 2.4.9 Propósito de los enlaces (solo enlaces) | *<a>* | H30: Proporcionar texto descriptivo del propósito del enlace usando el contenido del elemento *<a>*. |

| | | |
|---|---|---|
| 2.4.9 Propósito de los enlaces (solo enlaces) | *<a>* | G91: Proporcionar texto de enlace que sea significativo por sí mismo, sin depender del contexto circundante. |
| 2.4.10 Encabezados de sección | *<h1>–<h6>* | H69: Usar elementos de encabezado para transmitir la estructura lógica del contenido. |
| 2.5.3 Etiqueta en el nombre | *<label>*, *aria-label*, *aria-labelledby* | G208: Asegurarse de que las etiquetas visibles coincidan con el nombre accesible del elemento (por ejemplo, usando *aria-label*). |
| 3.1.1 Idioma de la página | *<html lang="es">*, *<html lang="en">* | H57: Usar el atributo lang en el elemento *<html>* para indicar el idioma predeterminado del contenido de la página. |
| 3.1.2 Idioma de las partes | **, *<p lang="de">* | H58: Usar el atributo *lang* en elementos específicos para identificar el idioma de partes del contenido. |
| 3.1.4 Abreviaturas | *<abbr>* | H28: Usar el elemento *<abbr>* y el atributo *title* para identificar abreviaturas. |
| 3.2.1 Al recibir el foco | Cualquier elemento interactivo (*<input>*, *<a>*, *<button>*) | G107: Asegurar que ningún cambio de contexto ocurra únicamente al recibir el *foco*. Es decir, no se deben activar *scripts* ni redirecciones cuando un elemento solo recibe *foco* mediante teclado. |
| 3.2.2 Al recibir entradas | *<select>*, *<input>*, *<textarea>* | G80: No cambia automáticamente el contexto cuando el usuario selecciona una opción o introduce un dato. Por ejemplo, evitar que un *<select>* redirecciona la página al cambiar de opción. |
| 3.2.3 Navegación coherente | *<nav>*, *<header>*, *<footer>*, *<a>* | G61: Usar una estructura de navegación consistente en todas las páginas (ubicación, orden, elementos). El *HTML* semántico ayuda a mantener la coherencia al reutilizar menús. |
| 3.2.4 Identificación coherente | *<label>*, *<button>*, *<input>*, *<a>* | H48: Usar etiquetas y nombres consistentes para elementos que cumplen funciones similares en diferentes páginas. |
| 3.3.1 Identificación de errores | *<input>*, *<label>*, *<select>*, *<textarea>* | G83: Proporcionar mensajes de error claros cuando la entrada del usuario no es válida. |
| 3.3.1 Identificación de errores | *<input>*, *<label>*, *<select>*, *<textarea>* | G84: Mostrar mensajes de error que describan el problema. |

| | | |
|---|---|---|
| 3.3.1 Identificación de errores | *<input>*, *<label>*, *<select>*, *<textarea>* | H90: Usar marcado semántico para identificar campos obligatorios (por ejemplo, *required*). |
| 3.3.1 Identificación de errores | *<input>*, *<label>*, *<select>*, *<textarea>* | H91: Usar elementos de formulario estándar para asegurar compatibilidad con tecnologías de asistencia. |
| 3.3.2 Etiquetas o instrucciones | *<label>* | H44: Usar el atributo *'for'* en *<label>* para asociarlo a un control de formulario. |
| 3.3.2 Etiquetas o instrucciones | *<label>* | H65: Usar *<label>* para identificar los controles de entrada. |
| 3.3.2 Etiquetas o instrucciones | *<form>*, *<fieldset>*, *<input>*, *<textarea>*, *<select>* | G131: Proporcionar instrucciones antes de que el usuario interactúe con un formulario. |
| 3.3.2 Etiquetas o instrucciones | *<input>*, *<select>*, *<textarea>* | G89: Proporcionar información útil en la etiqueta de cada campo. |
| 3.3.2 Etiquetas o instrucciones | *<fieldset>*, *<legend>* | H71: Agrupar controles relacionados con *<fieldset>* y describirlos con *<legend>*. |
| 3.3.3 Sugerencias ante errores | *<form>*, *<input>*, *<select>*, *<textarea>* | G83: Proporcionar sugerencias específicas que ayuden al usuario a corregir los errores. |
| 3.3.3 Sugerencias ante errores | *<form>*, *<input>*, *<select>*, *<textarea>* | G177: Proporcionar instrucciones o sugerencias para cada error detectado, como texto junto al campo. |
| 3.3.3 Sugerencias ante errores | *<form>*, *<input>*, *<select>*, *<textarea>* | G85: Proporcionar mensajes de error personalizados que describan cómo solucionar el problema. |
| 4.1.1 Procesamiento | *<body>*, *<button>*, *<form>*, *<html>*, *<input>*, *<script>*, *<select>*, *<textarea>* | G134: Validar que el código esté bien formado para evitar problemas de procesamiento por parte de tecnologías de asistencia. |
| 4.1.1 Procesamiento | *<body>*, *<button>*, *<form>*, *<html>*, *<input>*, *<script>*, *<select>*, *<textarea>* | H74: Asegurar que todos los elementos *HTML* tienen etiquetas de apertura y cierre correctamente anidadas. |

| | | |
|---|---|---|
| 4.1.1 Procesamiento | *<body>*, *<button>*, *<form>*, *<html>*, *<input>*, *<script>*, *<select>*, *<textarea>* | H75: Asegurar que los atributos estén correctamente escritos y no duplicados. |
| 4.1.2 Nombre, función, valor | *<a>*, *<button>*, *<form>*, *<img>*, *<input>*, *<select>*, *<textarea>* | H91: Usar elementos de formulario estándar con los atributos adecuados para asegurar que su función y valor sean interpretables por tecnologías de asistencia. |
| 4.1.2 Nombre, función, valor | *<a>*, *<button>*, *<form>*, *<img>*, *<input>*, *<select>*, *<textarea>* | ARIA1: Usar atributos *ARIA* para definir nombre, función y estado de los elementos interactivos cuando no se logre con *HTML* nativo. |
| 4.1.2 Nombre, función, valor | *<a>*, *<button>*, *<form>*, *<img>*, *<input>*, *<select>*, *<textarea>* | ARIA4: Asegurar que los roles y estados *ARIA* estén correctamente aplicados y actualizados dinámicamente. |
| 4.1.2 Nombre, función, valor | *<a>*, *<button>*, *<form>*, *<img>*, *<input>*, *<select>*, *<textarea>* | H44: Usar el atributo *'for'* en *<label>* para asociarlo correctamente a un control de formulario, proporcionando así el nombre accesible. |
| 4.1.3 Mensajes de estado | * *, *<output>*, *<p>*, **, atributos *ARIA* como *role=status*, *aria-live* | ARIA22: Usar *aria-live* para anunciar actualizaciones dinámicas del contenido sin mover el *foco*. |
| 4.1.3 Mensajes de estado | * *, *<output>*, *<p>*, **, atributos *ARIA* como *role=status*, *aria-live* | ARIA19: Usar *role=status* para proporcionar mensajes de estado que se anuncien automáticamente a los usuarios de tecnologías de asistencia. |
| 4.1.3 Mensajes de estado | * *, *<output>*, *<p>*, **, atributos *ARIA* como *role=status*, *aria-live* | G108: Utilizar cambios de texto visibles que se detecten automáticamente por tecnologías de asistencia. |

La Tabla 4 puede considerarse una guía práctica para desarrolladores. Si bien en las herramientas automáticas y la documentación de las *WCAG* 2.2 permiten conocer pautas y niveles, no siempre muestran de forma clara cómo corregir un error específico detectado en una etiqueta *HTML*. Al consultar la documentación oficial de las *WCAG 2.2* o los resultados de herramientas validadoras (como los mencionados en la Sección 4 "Marco Teórico"), los desarrolladores pueden encontrar los principios, pautas, criterios de éxito y los niveles de conformidad, pero la vinculación específica de un error detectado en una etiqueta *HTML* con la técnica suficiente precisa para corregirlo no siempre es directa o fácilmente localizable. La Tabla 4 aborda esta brecha al clarificar estas conexiones entre estos elementos. Aunque esta estrategia no reemplaza una evaluación completa, sí ofrece una base clara para detectar y evitar errores comunes, ayudando a que los sitios web sean más accesibles para todas las personas.

## 6.3 DISEÑO DE LA ESTRATEGIA DE VERIFICACIÓN *HTML-WCAG*

Tras analizar los errores de accesibilidad, la identificación de criterios de éxito que se aplican directamente a documentos *HTML* y la relación explícita entre los elementos del *HTML* y criterios de éxito, se procedió a la especificación operativa de la estrategia, materializada en un conjunto de reglas de verificación concretas. La finalidad de estas reglas es transformar los criterios de éxito las *WCAG* 2.2 en reglas claras, y aplicables que permitan detectar errores directamente en el código del documento *HTML*. Cada regla ha sido construida a partir de cinco componentes fundamentales: el criterio de éxito establecido por las *WCAG* 2.2, la etiqueta *HTML* relacionada, la técnica suficiente recomendada por el *W3C* para cumplir ese criterio, el error más común que suele encontrarse en la práctica, y una recomendación que sugiere cómo resolver dicho error directamente en el código. Esta estructura tiene como finalidad facilitar tanto la comprensión como la aplicación de los criterios de éxito.

La Tabla 5 [2] ,[13], [14] presenta la propuesta de reglas, diseñada para ser comprensible para personas sin mayor experiencia en desarrollo web, ya que cada recomendación incluye ejemplos prácticos y explicaciones claras sobre cómo aplicar las buenas prácticas. Con ello, se espera cerrar la brecha entre el conocimiento normativo y la implementación técnica. El detalle de las técnicas suficientes se puede revisar en la Tabla 4, puesto que en la Tabla 5 se menciona solo cuál es la técnica suficiente, pero no el detalle de la misma. Estas reglas interpretan los criterios de éxito de las *WCAG* 2.2, transformándolos en acciones específicas para su aplicación en el lenguaje de marcado o *HTML*. Gracias a esta

especificación, es posible automatizar la validación de accesibilidad desde el propio entorno de desarrollo, como *Visual Studio Code*, facilitando su adopción continua.

Tabla 5. Reglas de verificación *HTML-WCAG*

| **Regla #** | **Criterio de Éxito** | **Etiqueta o atributo del *HTML*** | **Técnica suficiente (*W3C*)** | **Error común** | **Recomendación** |
|---|---|---|---|---|---|
| R1 | 1.1.1 Contenido no textual | *<img>* | H37 | Las imágenes no tienen texto alternativo (*alt*) que describa su contenido. | Incluir siempre el atributo *alt* en las imágenes. Si la imagen tiene un propósito informativo, el *alt* debe describir brevemente su contenido, por ejemplo:<br><br>*<img src="grafico.png" alt="Gráfico de evolución de ventas 2023">*.<br><br>Si la imagen es decorativa, se debe usar *alt*="" para que los lectores de pantalla la ignoren. |
| R2 | 1.1.1 Contenido no textual | *<input type="image" >* | H36 | El botón de imagen no tiene una descripción textual. | Usar *alt* para describir la acción del botón, como *<input type="image" src="enviar.png" alt="Enviar formulario">*. |
| R3 | 1.1.1 Contenido no textual | *<area>* | H67 | El área del mapa no tiene texto alternativo. | Incluir el atributo alt en cada *<area>*, como *<area shape="rect" coords="..." href="info.html" alt="Más información">*. |
| | | | H24 | No se proporcionó texto alternativo para cada región del mapa. | Incluir texto alternativo descriptivo en el atributo alt para cada *<area>*. |

| | | | | | |
|---|---|---|---|---|---|
| R4 | 1.1.1 Contenido no textual | <*object*> | G94 | El objeto gráfico no tiene alternativa textual. | Proporcionar una descripción como contenido alternativo dentro de <*object*>. |
| | | | G196 | Objeto decorativo sin descripción. | Agregar *alt* vacío o contenido textual oculto si no transmite información relevante. |
| R5 | 1.1.1 Contenido no textual | <*svg*> | G94 | Gráfico *SVG* que contiene información sin descripción. | Usar *aria-label* o <*title*> dentro del *SVG* para describir su contenido. |
| | | | G196 | *SVG* decorativo sin descripción. | Agregar *aria-hidden*="*true*" para que el lector de pantalla lo ignore. |
| R6 | 1.1.1 Contenido no textual | <*applet*> | H53 | El contenido *applet* no tiene texto alternativo. | Agregar *alt* en <*applet*> con descripción del contenido o propósito. |
| R7 | 1.1.1 Contenido no textual | <*button*> | G95 | El botón contiene solo una imagen y no tiene *aria-label* ni texto visible. | Agregar *aria-label* o incluir texto visible dentro del botón. |
| R8 | 1.1.1 Contenido no textual | <*a*> | G95 | El enlace contiene solo un ícono sin *aria-label*. | Usar *aria-label* para describir el propósito del enlace. |

| | | | | | |
|---|---|---|---|---|---|
| R9 | 1.1.1 Contenido no textual | <*figure*> | H49 | La figura no incluye <*figcaption*> para describir su contenido. | Agregar un <*figcaption*> dentro de <*figure*> con texto descriptivo. |
| R10 | 1.1.1 Contenido no textual | <*iframe*> | H64 | El *iframe* no tiene atributo title. | Incluir *title* que describa el contenido, cómo <*iframe src*="*mapa.html*" *title*="*Mapa de ubicación*">. |
| R11 | 1.1.1 Contenido no textual | <*embed*> | H95 | Contenido incrustado sin descripción o alternativa textual. | Usar etiquetas externas o *fallback* como <*p*> para describir el contenido. |
| R12 | 1.1.1 Contenido no textual | <*canvas*> | H96 | El contenido del *canvas* no se describe mediante texto alternativo. | Colocar contenido *fallback* dentro de <*canvas*> que describa su función. |
| R13 | 1.2.1 Solo audio y solo vídeo (grabado) | <*audio*> | G158 | El contenido de solo audio no tiene una transcripción textual accesible. | Incluir una transcripción textual que describa todo el contenido del audio, incluyendo diálogos y sonidos relevantes. |

| | | | | | |
|---|---|---|---|---|---|
| R14 | 1.2.1 Solo audio y solo vídeo (grabado) | <*object*> | G158 | El objeto que contiene solo audio no tiene una transcripción textual. | Incluir una transcripción textual accesible fuera del objeto que detalle el contenido del audio. |
| | | | G159 | El objeto que contiene solo video no tiene una descripción textual alternativa. | Agregar una descripción textual que explique lo que sucede en el video y esté disponible cerca del objeto. |
| R15 | 1.2.1 Solo audio y solo vídeo (grabado) | <*video*> | G159 | El contenido de video no tiene una descripción textual alternativa. | Proporcionar una transcripción accesible que describa las escenas y acciones que ocurren en el video. |
| R16 | 1.2.2 Subtítulos (grabados) | <*video*> | G87 | El video no incluye subtítulos sincronizados para usuarios sordos o con dificultades auditivas. | Incluir subtítulos directamente integrados en el video o utilizando la etiqueta <*track*> para mostrar texto sincronizado. |
| | | | G93 | El video no referencia ningún archivo de subtítulos sincronizados. | Añadir un elemento <*track*> dentro de <*video*> que apunte al archivo de subtítulos, por ejemplo:<br><br><*track src*=“*subtitulos.vtt*” *kind*=“*subtitles*” *srclang*=“*es*” *label*=“*Español*”>. |

| | | | | | |
|---|---|---|---|---|---|
| R17 | 1.2.2 Subtítulos (grabados) | <*track*> | G93 | El <*track*> no tiene definido el atributo *kind*="*subtitles"* o no está asociado correctamente al <*video*>. | Usar correctamente la etiqueta <*track*> con los atributos necesarios: *kind*="*subtitles"*, *src*, *srclang* y *label*. |
| R18 | 1.2.3 Audiodescripción o medio alternativo | <*video*> | G78 | El video no tiene una pista de audiodescripción para usuarios con discapacidad visual. | Incluir una pista adicional de audio que describa lo que sucede visualmente o integrar en la pista principal del video. |
| | | | G173 | El contenido visual del video no está descrito de forma textual en ninguna parte de la página. | Agregar una descripción textual detallada acerca del video que explique las acciones o escenas importantes. |
| R19 | 1.2.3 Audiodescripción o medio alternativo | <*object*> | G78 | El objeto multimedia no tiene una pista de audiodescripción incorporada. | Asegurar que el archivo multimedia embebido en el <*object*> contenga audiodescripción sincronizada. |
| | | | G173 | No se proporciona una descripción textual del contenido visual que está embebido. | Agregar una sección de texto cercana al <*object*> que describa las escenas o eventos visuales. |

| | | | | | |
|---|---|---|---|---|---|
| R20 | 1.2.5 Audiodescripción (grabado) | <*video*> | G78 | El video grabado no contiene una pista de audiodescripción sincronizada. | Añadir un <*track*> con *kind*=“*descriptions”* que proporcione una pista de audiodescripción sincronizada para describir acciones visuales importantes. |
| | | | G173 | El contenido visual del video no está descrito de forma accesible para personas ciegas o con baja visión. | Proporcionar una transcripción narrativa detallada como texto alternativo fuera del video. |
| R21 | 1.2.5 Audiodescripción (grabado) | <*object*> | G78 | El objeto embebido no ofrece audiodescripción sincronizada de los contenidos visuales. | Asegurar que el archivo multimedia embebido en el <*object*> incluya audiodescripción dentro de su contenido. |
| | | | G173 | No se incluye una descripción alternativa que describa visualmente el contenido embebido. | Añadir una descripción textual cerca del objeto que explique detalladamente las acciones o contexto visual. |

| | | | | | |
|---|---|---|---|---|---|
| R22 | 1.3.1 Información y relaciones | <*table*> | H51 | Se usa <*div*> u otras etiquetas para representar datos tabulares. | Usar <*table*> cuando se presentan datos estructurados en filas y columnas. |
| | | | H39 | La tabla no tiene un título que explique su propósito. | Agregar un <*caption*> dentro de <*table*> que describa su contenido. |
| | | | H73 | La tabla no utiliza secciones que estructuran visualmente los datos. | Organizar la tabla en <*thead*> para encabezados, <*tbody*> para datos y <*tfoot*> para resúmenes. |
| | | | H63 | Los encabezados de columnas o filas están usando <*td*> en lugar de <*th*>. | Utilizar <*th*> para identificar encabezados de columna o fila. |
| | | | H77 | Las celdas de datos no están asociadas a sus encabezados, dificultando su interpretación por lectores de pantalla. | Usar atributos id en <*th*> y headers en <*td*> para establecer asociaciones explícitas. |

| | | | | | |
|---|---|---|---|---|---|
| R23 | 1.3.1 Información y relaciones | <*caption*> | H39 | El título de la tabla está fuera del elemento o no existe. | Insertar un <*caption*> como primer hijo de <*table*> con una descripción clara. |
| R24 | 1.3.1 Información y relaciones | <*th*> | H63 | Se utilizan <*td*> para encabezados. | Reemplazar <*td*> por <*th*> en las celdas que actúan como encabezado. |
| | | | H77 | Las celdas <*td*> no indican su relación con encabezados <*th*>. | Asignar *id* a <*th*> y referenciar desde <*td*> con el atributo *headers*. |
| R25 | 1.3.1 Información y relaciones | <*fieldset*> | H85 | No se agrupan controles relacionados lógicamente. | Usar <*fieldset*> para crear grupos de campos relacionados. |
| R26 | 1.3.1 Información y relaciones | <*legend*> | H85 | No se describe el propósito del grupo de campos. | Agregar un <*legend*> dentro del <*fieldset*> que explique el grupo. |

| | | | | | |
|---|---|---|---|---|---|
| R27 | 1.3.1 Información y relaciones | <*label*> | H71 | Campos de formulario no están etiquetados correctamente. | Agregar <*label*> y asociarlo con el campo mediante *for* o envolviendo el *input*. |
| R28 | 1.3.1 Información y relaciones | <*h1*>–<*h6*> | H42 | El contenido no tiene una estructura de encabezados clara. | Usar <*h1*> a <*h6*> para organizar el contenido en niveles jerárquicos. |
| R29 | 1.3.1 Información y relaciones | <*ul*>, <*ol*>, <*dl*> | H48 | Se listan elementos sin usar listas semánticas. | Usar <*ul*>, <*ol*> o <*dl*> según corresponda para representar agrupaciones. |
| R30 | 1.3.1 Información y relaciones | <*section*> | H69 | La sección no tiene encabezado que describa su contenido. | Incluir un encabezado dentro de <*section*> para mejorar la navegación y comprensión. |
| R31 | 1.3.1 Información y relaciones | <*article*> | H69 | El artículo no tiene un título claro que indique su tema. | Incluir un encabezado dentro del <*article*> que describa su contenido. |

| | | | | | |
|---|---|---|---|---|---|
| R32 | 1.3.2 Secuencia significativa | <*ol*> | H49 | Se usa <*div*> o <*p*> en lugar de <*ol*> para secuencias ordenadas. | Utilizar <*ol*> para listas donde el orden de los elementos sea importante. |
| | | | G57 | Los elementos aparecen fuera de orden o rompen la secuencia lógica para el lector. | Estructurar el contenido en el *HTML* siguiendo un orden semántico y visual que respete la lógica del contenido. |
| R33 | 1.3.2 Secuencia significativa | <*ul*> | H49 | Se usa texto plano o separadores visuales en lugar de listas semánticas. | Utilizar <*ul*> para listas no ordenadas de elementos relacionados. |
| | | | G57 | El contenido de la lista se presenta en un orden ilógico o visualmente incoherente. | Mantener un orden natural y lógico dentro de la estructura <*ul*>. |
| R34 | 1.3.2 Secuencia significativa | <*li*> | H49 | Los ítems de lista no están contenidos en elementos <*li*>. | Usar <*li*> para cada elemento de lista dentro de <*ul*> o <*ol*>. |
| | | | G57 | Los elementos <*li*> están desordenados o intercalados con otros elementos sin lógica. | Organizar los <*li*> siguiendo una secuencia coherente con el contenido que representan. |

| | | | | | |
|---|---|---|---|---|---|
| R35 | 1.3.2 Secuencia significativa | <*table*> | G57 | El orden de las filas o columnas en la tabla no sigue una secuencia lógica. | Estructurar los datos tabulares de modo que reflejen un orden claro y comprensible. |
| R36 | 1.3.2 Secuencia significativa | <*thead*> | G57 | El encabezado no refleja adecuadamente el contenido de las columnas o está mal posicionado. | Colocar correctamente <*thead*> al inicio de la tabla y asegurar que cada columna esté representada. |
| R37 | 1.3.2 Secuencia significativa | <*tbody*> | G57 | Las filas dentro de <*tbody*> no siguen una estructura lógica o jerárquica clara. | Organizar las filas de datos en <*tbody*> respetando el flujo esperado de la información. |
| R38 | 1.3.3 Características sensoriales | <*p*>, <*span*>, <*strong*> <*strong*>, <*em*> | G96 | El contenido da instrucciones como “haz clic en el botón rojo” o “ve al ícono redondo” sin contexto adicional. | Incluir texto explicativo adicional como etiquetas visibles o descripciones que no dependan únicamente del color o la forma. |
| R39 | 1.3.5 Identificación del propósito de entrada | <*input*>, <*select*>, <*textarea*>, <*form*> | G98 | Los campos de formulario no tienen atributos *autocomplete* que definan su propósito. | Agregar atributos *autocomplete* como *autocomplete*=“*name*”, “*email*”, “*tel*”, etc. según el contexto del campo. |

| | | | | | |
|---|---|---|---|---|---|
| R40 | 1.3.6 Identificación del propósito | <*input*>, <*select*>, <*textarea*> | G211 | No se utilizan atributos *autocomplete* para facilitar el llenado automático en campos personales. | Incluir *autocomplete*="*bday*", "*name*", "*address-line1*", etc., para enriquecer la semántica del formulario. |
| R41 | 1.4.1 Uso del color | <*a*>, <*span*>, <*button*> y cualquier otro con estilo aplicado. | G14 | Se utiliza solo color (ejemplo: rojo o verde) para indicar estados o diferencias. | Complementar el color con texto adicional, subrayado o símbolos que transmiten el mismo mensaje. |
| | | | G205 | Los usuarios con daltonismo no pueden identificar diferencias que solo usan color. | Usar estilos adicionales como bordes, íconos, o texto visible para reforzar la diferenciación. |
| R42 | 1.4.2 Control del audio | <*audio*> | G60 | El audio se reproduce automáticamente y no ofrece controles de pausa, volumen o detención. | Incluir el atributo controls en la etiqueta <*audio*>, o proporcionar controles personalizados accesibles para que el usuario gestione la reproducción. |
| R43 | 1.4.3 Contraste (mínimo) | <*a*>, <*h1*>–<*h6*>, <*p*>, <*span*> | G18 | El texto tiene bajo contraste respecto al fondo, dificultando la lectura para personas con baja visión. | Verificar el contraste con herramientas automáticas y ajustar colores de texto y fondo para lograr al menos una relación de 4.5:1. |

| | | | | | |
|---|---|---|---|---|---|
| R44 | 1.4.3 Contraste (mínimo) | <*a*>, <*h1*>–<*h6*>, <*p*>, <*span*><br><br>Texto sobre imágenes o gradientes | G145 | El texto se muestra directamente sobre una imagen o gradiente, reduciendo su legibilidad. | Añadir una superposición opaca o fondo sólido detrás del texto, o aplicar sombra al texto para mejorar su contraste. |
| R45 | 1.4.4 Cambio de tamaño del texto | <*html*>, <*body*>, <*p*>, <*div*>, <*span*>, <*a*>, <*h1*>–<*h6*>, etc. | G142 | El texto está definido con unidades fijas (*px*), lo que impide el redimensionamiento efectivo por el usuario. | Utilizar unidades relativas como em, rem o % para que el texto pueda escalar según la configuración del navegador o del sistema. |
| | | | C28 | El contenido se desborda, se corta o se sobrepone al aumentar el tamaño del texto. | Diseñar usando layouts fluidos y evitar anchuras fijas o restricciones que impidan la adaptabilidad del contenido. |
| R46 | 1.4.5 Imágenes de texto | <*img*> | G140 | Se utiliza una imagen que contiene texto informativo en lugar de usar texto *HTML*. | Reemplazar la imagen con texto real estilizado con *CSS*, excepto en casos donde se requiere una presentación visual específica como logotipos. |
| | | | G148 | El texto dentro de la imagen no está disponible en un formato accesible para lectores de pantalla. | Agregar texto alternativo que describa el contenido textual de la imagen, o proveer el mismo contenido en texto junto a la imagen. |

| | | | | | |
|---|---|---|---|---|---|
| R47 | 1.4.9 Imágenes de texto (sin excepciones) | <*img*>, <*canvas*>, <*svg*>, <*object*> | G140 | Se presenta texto mediante imágenes en lugar de usar *HTML* real. | Reemplazar imágenes de texto con elementos *HTML* reales estilizados con *CSS* siempre que sea posible. |
| | | | C22 | Se oculta el texto real y se muestra solo la imagen, sin garantizar accesibilidad. | Mantener el texto real en el *HTML* y usar *CSS* para ocultar visualmente si es necesario, asegurando su disponibilidad para tecnologías de asistencia. |
| R48 | 1.4.10 Reajuste | <*a*>, <*body*>, <*div*>, <*html*>, <*img*>, <*main*>, <*p*>, <*section*> | C32 | El contenido requiere desplazamiento horizontal al aumentar el tamaño de la ventana. | Usar contenedores con anchos flexibles y evitar valores fijos para permitir un diseño fluido y adaptable. |
| | | | C31 | Elementos como <*div*> o <*img*> usan tamaños fijos que no se adaptan bien en dispositivos móviles. | Aplicar unidades relativas (%, *em*) y evitar fijar tamaños absolutos con *px* para mejorar el reajuste. |

| | | | | | |
|---|---|---|---|---|---|
| R49 | 1.4.11 Contraste no textual | *<button>*, *<canvas>*, *<img>*, *<input>*, *<select>*, *<svg>* | G209 | Los elementos interactivos tienen bajo contraste respecto al fondo, dificultando su visibilidad. | Verificar que botones, inputs y gráficos tengan al menos una relación de contraste de 3:1 con el fondo. |
| | | | G195 | El estado de foco o selección no es visible para personas con baja visión. | Aplicar estilos como bordes destacados o fondos contrastantes en los estados *:focus*, *:hover* o activos. |
| R50 | 1.4.12 Espaciado del texto | *<p>*, **, *<li>*, *<h1>–<h6>* | C36 | El texto se superpone o se corta cuando el usuario ajusta el interlineado o espaciado. | Diseñar con suficiente espacio entre elementos y permitir ajustes personalizados de espaciado sin romper el diseño. |
| R51 | 1.4.13 Contenido en *hover* o *focus* | *<button>*, *<a>*, *<input>*, *<label>* | G211 | El contenido emergente no puede ser descartado fácilmente o impide la navegación con teclado. | Asegurar que el contenido que aparece al pasar el *mouse* o enfocar pueda ser cerrado sin mover el cursor o perder el *foco*, por ejemplo, con botones visibles o eventos bien gestionados. |
| R52 | 2.1.1 Teclado | *<button>*, *<a>*, *<input>*, *<select>*, *<textarea>* *<fieldset>*, *<label>*, *<form>* | G202 | Algunas funciones solo están disponibles mediante *clic* del ratón y no pueden ser activadas con el teclado. | Utilizar elementos interactivos estándar como *<button>*, *<a>*, *<input>*, etc., que ya son accesibles desde el teclado sin configuración adicional. |

| | | | | | |
|---|---|---|---|---|---|
| R53 | 2.1.1 Teclado | *<a>*, *<button>*, *<input>*, *<label>*, *<select>*, *<textarea>* | SCR20 | Los eventos como *onClick* no tienen equivalentes para teclado como *onKeyDown* o *onKeyPress*. | Agregar controladores de eventos de teclado junto con los de ratón para permitir la interacción sin *mouse*. |
| | | | H91 | Se crean botones o enlaces personalizados con * * o **, que no pueden recibir *foco* ni activarse con teclado. | Preferir elementos semánticos *HTML* nativos que ya poseen soporte de accesibilidad como *<button>* en lugar de elementos genéricos. |
| R54 | 2.1.2 Sin trampas para el foco | *<a>*, *<button>*, *<input>* | G21 | El usuario no puede salir de un componente con el teclado porque atrapa el foco. | Implementar control con teclas como *Esc* o *Tab* para permitir salir del componente interactivo sin necesidad del *mouse*. |
| R55 | 2.1.4 Atajos de teclado | *<button accesskey="" >*<br><br>*<a accesskey="" >* | G217 | Los atajos de teclado definidos con *accesskey* interfieren con combinaciones del navegador y no se pueden personalizar. | Permitir personalizar o limitar el uso de *accesskey* y documentar claramente qué teclas se usan. |

| | | | | | |
|---|---|---|---|---|---|
| R56 | 2.2.2 Pausar, detener, ocultar | <*video*>, <*audio*>, <*marquee*> | F4 | El contenido multimedia se reproduce automáticamente sin opción de detenerlo. | Evitar *autoplay* y permitir al usuario iniciar el contenido manualmente. |
| | | | G186 | No se ofrecen controles visibles para detener, pausar o ocultar contenido en movimiento. | Agregar controles mediante atributos como *"controls"* o elementos interactivos personalizados. |
| R57 | 2.4.1 Evitar bloques | <*a href="#mai"*><br><br><*nav*>, <*main*> | G1 | El usuario debe navegar por menús o encabezados en cada página sin una vía directa al contenido. | Incluir un enlace "Saltar al contenido" que redireccione a una sección principal mediante anclas o *landmarks*. |
| R58 | 2.4.2 Titulado de páginas | <*title*> | G88 | El documento *HTML* no tiene un título o el título no es representativo del contenido. | Agregar un título significativo dentro del <*head*> con la etiqueta <*title*>. |
| R59 | 2.4.3 Orden del foco | Cualquier elemento con *tabindex* | G59 | El *foco* del teclado salta de forma inesperada o no sigue el flujo visual del contenido. | Organizar el contenido *HTML* en un orden lógico y usar *tabindex* solo cuando sea necesario para ajustar el flujo del *foco*. |

| | | | | | |
|---|---|---|---|---|---|
| R60 | 2.4.4 Propósito de los enlaces | *<a>* | H30 | El texto del enlace no tiene sentido por sí solo (ejemplo: "haz clic aquí"). | Usar textos descriptivos que indiquen el destino o la acción del enlace, como "Ver detalles del producto". |
| R61 | 2.4.5 Múltiples vías | *<nav>*, *<a>*, *<ul>*, *<ol>* | G125 | La navegación solo es posible mediante una única vía. | Agregar elementos como menú de navegación, buscador o mapa del sitio para facilitar múltiples formas de acceso. |
| | | | G64 | El sitio no ofrece una visión estructurada de su contenido. | Crear y vincular un *sitemap* que permita explorar fácilmente las secciones importantes del sitio. |
| R62 | 2.4.6 Encabezados y etiquetas | *<h1>*-*<h6>*, *<label>*, *aria-label* | H65 | Los campos del formulario no están correctamente etiquetados. | Vincular cada *<input>* con una etiqueta *<label>* que describa su propósito, usando for o conteniéndolo directamente. |
| | | | H42 | La jerarquía visual del contenido no coincide con el uso de encabezados *HTML*. | Usar *<h1>*-*<h6>* para marcar títulos y subtítulos según su jerarquía en la página. |

| | | | | | |
|---|---|---|---|---|---|
| R63 | 2.4.7 Foco visible | Cualquier elemento interactivo (<*button*>, <*a*>, <*input*>, etc.) | G149 | No se muestra visualmente qué elemento tiene el *foco* del teclado. | Estilizar el estado *:focus* con bordes visibles o cambios de color que permitan ubicar el *foco* fácilmente. |
| R64 | 2.4.9 Propósito de los enlaces (solo enlaces) | <*a*> | H33 | El enlace tiene texto ambiguo o vacío (por ejemplo, solo un ícono). | Incluir texto visible o atributos accesibles como *aria-label* para indicar el propósito del enlace. |
| | | | H30 | El texto del enlace depende del contexto visual o es genérico. | Redactar texto de enlace que sea comprensible sin depender del contexto circundante. |
| | | | G91 | El enlace necesita contexto adicional para entender su propósito. | Revisar todos los enlaces y asegurar que su texto tenga sentido sin necesidad de contexto externo. |
| R65 | 2.4.10 Encabezados de sección | <*h1*>–<*h6*> | H69 | Se usan estilos visuales en lugar de encabezados *HTML* para estructurar contenido. | Emplear encabezados semánticos en orden jerárquico para representar la estructura del contenido. |

| | | | | | |
|---|---|---|---|---|---|
| R66 | 2.5.3 Etiqueta en el nombre | *<label>*, *aria-label*, *aria-labelledby* | G208 | El texto visible no coincide con el nombre accesible anunciado por tecnologías de asistencia. | Sincronizar el contenido visible con los atributos accesibles como *aria-label* para evitar confusión al usuario. |
| R67 | 3.1.1 Idioma de la página | *<html lang="es">*, *<html lang="en">* | H57 | El atributo *lang* está ausente o mal configurado en el elemento *<html>*. | Agregar el atributo *lang* en la etiqueta *<html>* con el código del idioma predominante del documento. |
| R68 | 3.1.2 Idioma de las partes | **, *<p lang="de">* | H58 | Se incluyen frases o palabras en otro idioma sin especificar el atributo *lang*. | Usar el atributo lang en las etiquetas que contengan contenido en un idioma diferente al principal del documento. |
| R69 | 3.1.4 Abreviaturas | *<abbr>* | H28 | El significado de las abreviaturas no está disponible para lectores de pantalla. | Utilizar la etiqueta *<abbr>* con el atributo title para describir la abreviatura. |
| R70 | 3.2.1 Al recibir el foco | Cualquier elemento interactivo (*<input>*, *<a>*, *<button>*) | G107 | Se activa una redirección o cambio de contenido automáticamente al enfocar un elemento. | Evitar ejecutar scripts que alteren el contexto (como redirecciones) al usar eventos como *onfocus*. |
| R71 | 3.2.2 Al recibir entradas | *<select>*, *<input>*, *<textarea>* | G80 | Seleccionar una opción en un *<select>* redirige inmediatamente a otra página sin confirmación. | Requerir acción explícita del usuario (ejemplo: botón de enviar) en lugar de cambiar el contexto automáticamente. |

| | | | | | |
|---|---|---|---|---|---|
| R72 | 3.2.3 Navegación coherente | <*nav*>, <*header*>, <*footer*>, <*a*> | G61 | Los menús o encabezados cambian de orden o estilo entre páginas. | Reutilizar la misma estructura de navegación y mantener consistencia en posición, orden y etiquetas. |
| R73 | 3.2.4 Identificación coherente | <*label*>, <*button*>, <*input*>, <*a*> | H48 | Botones con la misma función usan etiquetas diferentes en distintas páginas. | Nombrar de forma uniforme todos los elementos que realicen la misma función para evitar confusión. |
| R74 | 3.3.1 Identificación de errores | <*input*>, <*label*>, <*select*>, <*textarea*> | G83 | Los mensajes de error son genéricos o no indican qué campo contiene el problema. | Mostrar mensajes que expliquen claramente el error y cómo corregirlo. |
| | | | G84 | El usuario comete un error, pero no se le informa qué ocurrió ni cómo solucionarlo. | Añadir mensajes específicos en el mismo contexto del error. |
| | | | H90 | Los campos obligatorios no están claramente indicados o no usan el atributo *required.* | Aplicar el atributo *required* en los campos del formulario que lo necesiten. |
| | | | H91 | Se crean formularios personalizados con <*div*> en lugar de elementos nativos. | Utilizar etiquetas *HTML* de formularios estándar para mantener la accesibilidad y funcionalidad. |

| | | | | | |
|---|---|---|---|---|---|
| R75 | 3.3.2 Etiquetas o instrucciones | <*label*> | H44 | El <*label*> no está asociado al campo correspondiente. | Asegurarse de que el atributo *'for'* en <*label*> coincida con el id del campo correspondiente. |
| | | | H65 | Campos de entrada sin etiquetas visibles o accesibles. | Incluir una etiqueta clara y visible para cada campo, ya sea envolviendo el input o mediante el atributo *'for'*. |
| R76 | 3.3.2 Etiquetas o instrucciones | <*form*>, <*fieldset*>, <*input*>, <*textarea*>, <*select*> | G131 | El usuario no sabe cómo llenar el formulario correctamente por falta de instrucciones. | Añadir instrucciones breves al inicio del formulario o junto a los campos que lo requieran. |
| R77 | 3.3.2 Etiquetas o instrucciones | <*input*>, <*select*>, <*textarea*> | G89 | La etiqueta no comunica lo suficiente sobre qué se espera del usuario. | Redactar etiquetas que indiquen claramente el tipo de dato esperado (ejemplo: "Correo electrónico", "Teléfono"). |
| R78 | 3.3.2 Etiquetas o instrucciones | <*fieldset*>, <*legend*> | H71 | Controles relacionados están agrupados sin estructura semántica o sin descripción. | Usar <*fieldset*> para agrupar y <*legend*> para nombrar el grupo de campos. |

| | | | | | |
|---|---|---|---|---|---|
| R79 | 3.3.3 Sugerencias ante errores | <*form*>, <*input*>, <*select*>, <*textarea*> | G83 | El usuario no sabe cómo corregir el error cometido. | Añadir mensajes que expliquen lo que se espera y cómo corregir los errores detectados. |
| | | | G177 | El campo muestra un mensaje genérico sin orientación clara. | Mostrar mensajes de ayuda contextual justo al lado del campo con error. |
| | | | G85 | Los mensajes de error no dan pistas sobre cómo resolver el problema. | Personalizar los mensajes de error para que sean comprensibles y accionables. |
| R80 | 4.1.1 Procesamiento | <*body*>, <*button*>, <*form*>, <*html*>, <*input*>, <*script*>, <*select*>, <*textarea*> | G134 | El código *HTML* contiene errores de sintaxis que afectan su interpretación. | Verificar y corregir errores de marcado como etiquetas mal cerradas, atributos inválidos o estructuras mal anidadas. |
| | | | H74 | Existen etiquetas anidadas incorrectamente o no cerradas adecuadamente. | Revisar el *DOM* y asegurar que todas las etiquetas *HTML* estén correctamente abiertas y cerradas. |
| | | | H75 | Se repiten atributos o se usan atributos mal escritos, lo que puede causar fallos de interpretación. | Revisar que cada atributo esté correctamente escrito y aparezca una sola vez por etiqueta. |

| | | | | | |
|---|---|---|---|---|---|
| R81 | 4.1.2 Nombre, función, valor | <*a*>, <*button*>, <*form*>, <*img*>, <*input*>, <*select*>, <*textarea*> | H91 | El comportamiento de un campo de formulario no es comprendido por lectores de pantalla. | Utilizar elementos *HTML* semánticos y atributos como type, name y *value* correctamente asignados. |
| | | | ARIA1 | Los elementos personalizados no son reconocibles por tecnologías de asistencia. | Aplicar roles y propiedades *ARIA* adecuadas como *role*, *aria-label*, *aria-checked*, etc. |
| | | | ARIA4 | El estado de los componentes no se actualiza correctamente para los lectores de pantalla. | Actualizar dinámicamente los valores de los atributos *ARIA* según cambie el estado del componente. |
| | | | H44 | El nombre accesible del campo no está definido o no está vinculado. | Asociar correctamente las etiquetas <*label*> con los *inputs* mediante el atributo *'for'*. |

| | | | | | |
|---|---|---|---|---|---|
| R82 | 4.1.3 Mensajes de estado | * *, *<output>*, *<p>*, **, atributos *ARIA* como *role=status*, *aria-live* | ARIA22 | Cambios importantes en la interfaz no son anunciados a usuarios con lector de pantalla. | Agregar el atributo *aria-live=“polite”* o *“assertive”* en los contenedores de mensajes dinámicos. |
| | | | ARIA19 | El usuario no es informado automáticamente sobre el resultado de una acción. | Aplicar *role=“status”* a los elementos que contienen mensajes de confirmación, éxito o error. |
| | | | G108 | Actualizaciones importantes no son percibidas por el lector de pantalla. | Asegurarse de que los mensajes visibles estén en contenedores accesibles o notificados mediante *ARIA*. |

Al revisar la Tabla 5, se puede observar que algunos criterios de éxito de las *WCAG* 2.2 pueden dar lugar a múltiples reglas que aplican a diferentes etiquetas *HTML* o a distintos contextos de una misma etiqueta (por ejemplo, el Criterio 1.1.1 Contenido no textual se refleja en las reglas R1 para *<img>*, R2 para *<input type=“image”>*, R3 para *<area>*, etc.). Esta granularidad es intencional y responde a varias razones:

- **Especificidad de las técnicas suficientes:** Aunque un criterio de éxito sea el mismo, el *W3C* a menudo proporciona técnicas suficientes distintas y específicas para cada tipo de elemento *HTML* (ej. H37 para *<img>*, H36 para *<input type=“image”>*). Las reglas de la tabla buscan reflejar la aplicación de estas técnicas específicas.
- **Contextos de error particulares:** El error común y la recomendación pueden variar sutilmente según la etiqueta. Por ejemplo, la forma de proveer una alternativa textual para un ***<svg>*** (que podría usar *aria-label* o *<title>*) difiere de la de un *<img>* (que usa *alt*).

- **Claridad para el desarrollador:** Presentar reglas separadas por etiqueta o agrupar técnicas relevantes para un mismo criterio, tiene como objetivo proporcionar una guía más directa y menos ambigua al desarrollador, facilitando la identificación y corrección del error en el contexto específico de la etiqueta o el problema con el que está trabajando.

Este enfoque detallado busca maximizar la utilidad de la Tabla 5 como una herramienta práctica para la implementación de la accesibilidad directamente en los documentos *HTML*.

**Ejemplo 1**

A continuación, se muestra un ejemplo de aplicación para la etiqueta <*img*>:

1. Una persona está revisando un documento *HTML* y encuentra una etiqueta como la que aparece en la Figura 6 <*img src="foto1.jpg"*>, pero no está segura de si está cumpliendo con las pautas de accesibilidad, puede usar la Tabla 5 como punto de partida.

```
<img src="foto1.jpg">
```

*Figura 6. Ejemplo de etiqueta img del HTML sin atributo alt*

2. En lugar de buscar por el criterio de éxito de las *WCAG* 2.2 (que quizá no conoce), puede buscar directamente la etiqueta <*img*> en la columna "Etiqueta o atributo del *HTML*" . En la columna hay varias reglas alusivas a la etiqueta <*img*>, como **R1** (relacionada con la falta de texto alternativo), **R46** (sobre el uso de mapas de imagen) y **R47** (referente a imágenes que provocan destellos). Analizando el contexto del ejemplo de la Figura 6, no es un mapa de imagen, y descartando la opción que sea una imagen que provoca destellos, la regla que aplica para el error es la **R1**. Esta regla indica que para la etiqueta <*img*>, el criterio de éxito asociado es el 1.1.1 (Contenido no textual), y que un error común es no incluir el atributo *alt* en las imágenes.

3. En la columna de "Recomendación" para la Regla **R1**, la tabla sugiere: Incluir siempre el atributo *alt* en las imágenes. Si la imagen tiene un propósito informativo, el *alt* debe describir brevemente su contenido... En caso de que la imagen sea decorativa, el atributo *alt* debe utilizarse vacío (*alt=""*). Siguiendo esta regla, la solución sería actualizar el código como se puede observar en la Figura 7.

```
<img src="foto1.jpg" alt="Retrato de una
persona usando un computador portátil">
```

*Figura 7. Ejemplo de etiqueta img del HTML aplicando el atributo alt*

**Ejemplo 2**

Ejemplo de aplicación para una etiqueta de formularios <*input*>:

1. Una persona revisando el documento *HTML*, se encuentra con la etiqueta de formulario como se muestra en la Figura 8, <*input type="text" id="apellido" name="apellido"*>, pero no está segura si cumple con las pautas de accesibilidad.

```
<input type="text" id="apellido" name="apellido">
```

*Figura 8. Ejemplo de etiqueta input del HTML*

2. Al revisar la Tabla 5 buscando etiquetas de formularios como la etiqueta <*input*>, se encuentra varias reglas como la **R27** (señala la ausencia de la etiqueta <*label*>), y la **R75** (señala una asociación por atributos entre la etiqueta <*input*> y la etiqueta <*label*>). En la Figura 8, existe una ausencia de la etiqueta <*label*>, por lo que la regla a aplicar es la **R27**. Una vez incluida la etiqueta <*label*> se puede aplicar la regla **R75**, asegurando la correcta asociación mediante atributos entre las dos etiquetas.

3. Según la Tabla 5, la recomendación es "Agregar <*label*> y asociarlo con el campo mediante *for* o envolviendo el *input*" o "Asegurarse de que el atributo *for* en <*label*> coincida con el *id* del campo correspondiente". El desarrollador podría corregir el código como se ve en la Figura 9. De esta manera, incluso sin conocer en detalle las *WCAG*, la persona puede seguir una recomendación clara y útil, haciendo el contenido más accesible para quienes usan tecnologías de asistencia.

```html
<label for="apellido">Apellido:</label>
<input type="text" id="apellido" name="apellido">
```

*Figura 9. Ejemplo de etiqueta input de HTML con etiqueta <label>*

En la Sección 7 “Aplicación técnica de la propuesta”, se muestra una implementación en forma de extensión para *Visual Studio Code*, como ejemplo de implementación técnica de la estrategia. Aunque, la estrategia ha sido diseñada con la intención de ser independiente de cualquier entorno o herramienta específica, podría aplicarse en otros contextos, como diferentes editores de código, librerías de validación de software, herramientas de auditoría automática o soluciones personalizadas.

# 7. APLICACIÓN TÉCNICA DE LA PROPUESTA

Esta sección detalla la implementación de la estrategia concebida en la Sección 6 “Diseño y desarrollo de la propuesta”, la cual apunta a facilitar la implementación de la accesibilidad web en los documentos *HTML*, de acuerdo con los criterios de éxito definidos por las *WCAG* 2.2. Este proceso para llevar a cabo la implementación técnica y la evaluación de la propuesta, partiendo de los conceptos definidos en la fase de diseño, se estructuró en dos etapas (ver Figura 10).

- **Implementación técnica en *Visual Studio Code***: Se describe el desarrollo de una extensión funcional para el editor *Visual Studio Code*, la cual materializa la estrategia aplicando las reglas definidas para la validación de código.
- **Evaluación comparativa de la estrategia:** Se presentan los resultados de probar la estrategia en documentos *HTML* con errores comunes, incluyendo una comparación con otras herramientas de validación.

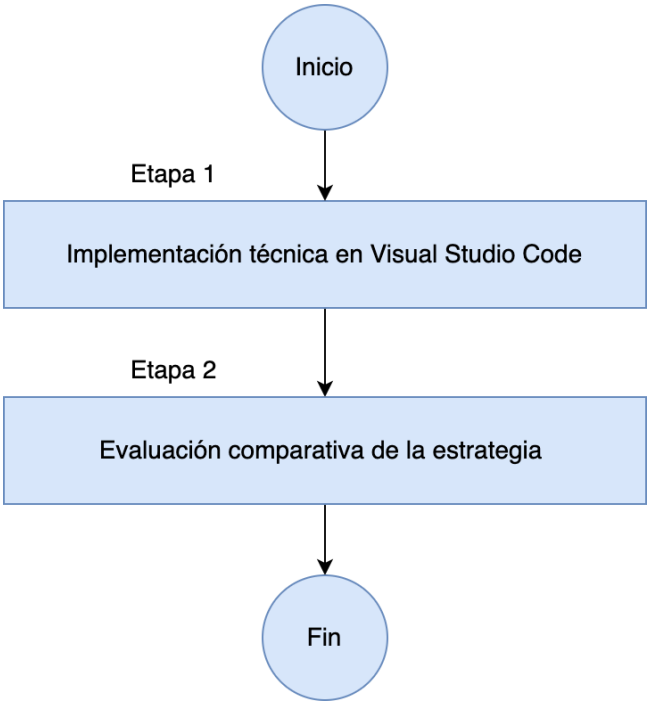


*Figura 10. Etapas de la aplicación técnica de la propuesta*

Cada una de estas etapas busca demostrar que es posible verificar la accesibilidad desde el momento en que se escriben las etiquetas y atributos en los documentos *HTML*.

## 7.1 IMPLEMENTACIÓN TÉCNICA EN *VISUAL STUDIO CODE*

A partir de las reglas de accesibilidad definidas en la Tabla 5, se desarrolló una extensión llamada "*Semantic Checker*" para *Visual Studio Code*, usando el lenguaje de programación *Javascript*, un editor ampliamente utilizado por desarrolladores web. La implementación muestra que las reglas definidas pueden utilizarse directamente durante el desarrollo de un documento *HTML*, sin necesidad de conocimientos avanzados en accesibilidad o desarrollo web. La función principal de la extensión es verificar el cumplimiento de las reglas definidas, aplicando estas validaciones directamente sobre etiquetas y atributos del *HTML*. Para realizar esta tarea, se utiliza una técnica basada en expresiones regulares (*regex*). Estas permiten identificar patrones específicos en el contenido del documento *HTML* y detectar errores frecuentes, como por ejemplo imágenes sin el atributo *alt*, formularios sin etiquetas asociadas, o tablas sin elementos <*caption*>.

La extensión se activa desde el menú de comandos de *Visual Studio Code*. Una vez activada, analiza el contenido del archivo *HTML* en pantalla y genera advertencias (conocidas como *warnings*) cuando detecta posibles errores de accesibilidad. Cada advertencia incluye un mensaje explicativo del problema y, cuando es posible, una sugerencia concreta para corregirlo. Esta retroalimentación permite realizar mejoras inmediatas mientras se escribe o edita el contenido del documento *HTML*.

La forma en la que opera la extensión, desde su activación y la detección del documento *HTML*, ejecución y análisis del *HTML*, se detalla a continuación:

- **Activación de la extensión:** La extensión está configurada para activarse manualmente cuando se trabaja con documentos *HTML*. Esto asegura que las reglas de accesibilidad se apliquen solo en documentos donde son pertinentes.
- **Ejecución del comando principal:** Al activar manualmente la extensión, se inicia la rutina de verificación. Esta función analiza el contenido del documento *HTML* abierto y lo prepara para ser examinado mediante las reglas definidas.
- **Análisis del código *HTML*:** Una vez que el contenido del documento está listo para ser analizado, la extensión inicia el núcleo de su lógica de verificación. Este proceso implica:

    - **Preprocesamiento del texto:** Para optimizar la detección y evitar falsos positivos, la extensión primero "neutraliza" comentarios *HTML* y el contenido de las etiquetas <*script*> y <*style*>, reemplazandolos temporalmente con espacios sin alterar la

numeración de líneas.

- **Aplicación secuencial de reglas:** A continuación, la extensión itera sobre cada una de las 82 reglas de verificación definidas en la Tabla 5 (que están implementadas en el archivo *rules.js*). Cada regla encapsula una expresión regular (*regex*) específica y una función de validación (*validate*).
- **Búsqueda de patrones y validación:** Para cada regla, se ejecuta su expresión regular sobre el texto *HTML* preprocesado para encontrar todas las posibles coincidencias. Cada coincidencia (*matchedTagString*) es luego sometida a la función *validate* de esa regla. Esta función realiza una comprobación contextual más precisa para confirmar si el patrón detectado constituye efectivamente una violación de accesibilidad según los criterios de la Tabla 5.
- **Generación de diagnósticos:** Si la validación es positiva (se confirma un error), la extensión crea un "diagnóstico" o advertencia. Este diagnóstico no solo identifica el problema, sino que también almacena la ubicación exacta del error en el código (línea y carácter de inicio y fin), el mensaje de error, el nivel de conformidad de las *WCAG* 2.2, la recomendación de la Tabla 5 para su corrección, y el identificador de la regla infringida (ej. R1), facilitando así la posterior visualización al desarrollador.

La retroalimentación se visualiza en el código *HTML* del editor, permitiendo al desarrollador aplicar las mejoras siguiendo las recomendaciones que proporciona la extensión, sin necesidad de configuraciones adicionales. Las reglas implementadas en la extensión están codificadas de manera simple y pueden modificarse o ampliarse fácilmente, lo que permite adaptar la herramienta a nuevos criterios o contextos de desarrollo.

La extensión proporciona un mecanismo útil para detectar errores en los documentos *HTML* y su integración directa en el entorno de trabajo facilita su uso continuo, ayudando a adoptar buenas prácticas de accesibilidad de forma natural y constante.

El código de la extensión se encuentra disponible en el siguiente enlace de *GitHub*: https://github.com/vanessamarely/vscode-checksemantic, que al ingresar en la url se verá como en la Figura 11, este repositorio facilita la revisión, reutilización y mejora continua del proyecto por parte de otros desarrolladores interesados en la accesibilidad web que deseen realizar una contribución.

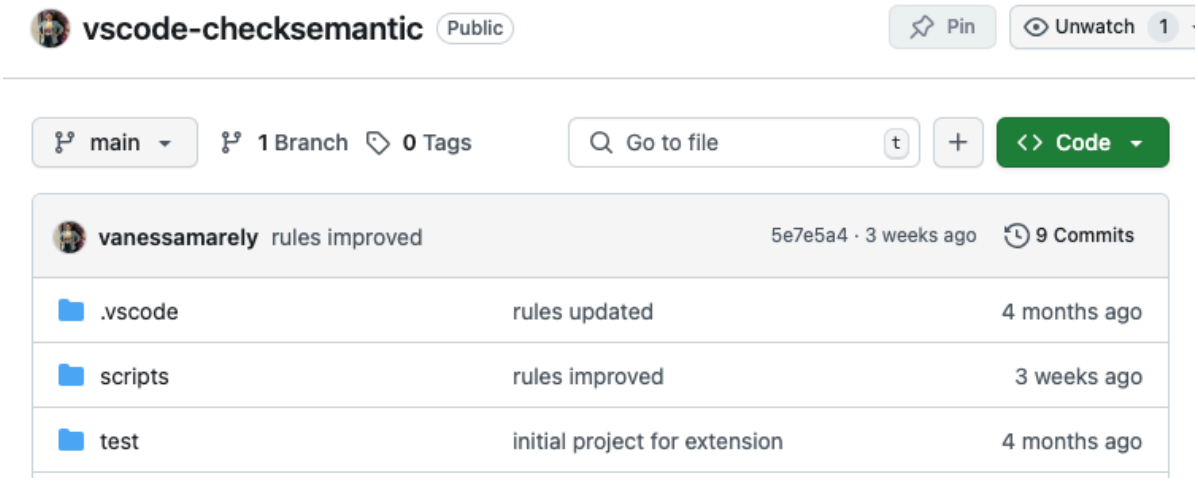


*Figura 11. Captura del repositorio de la extensión de Visual Studio Code*

La extensión está disponible para ser descargada para el editor de *Visual Studio Code*, desde el *Visual Studio Marketplace* a través del siguiente enlace, como se ve en la Figura 12: https://marketplace.visualstudio.com/items?itemName=VanessaAristizabal.semantic-checker.

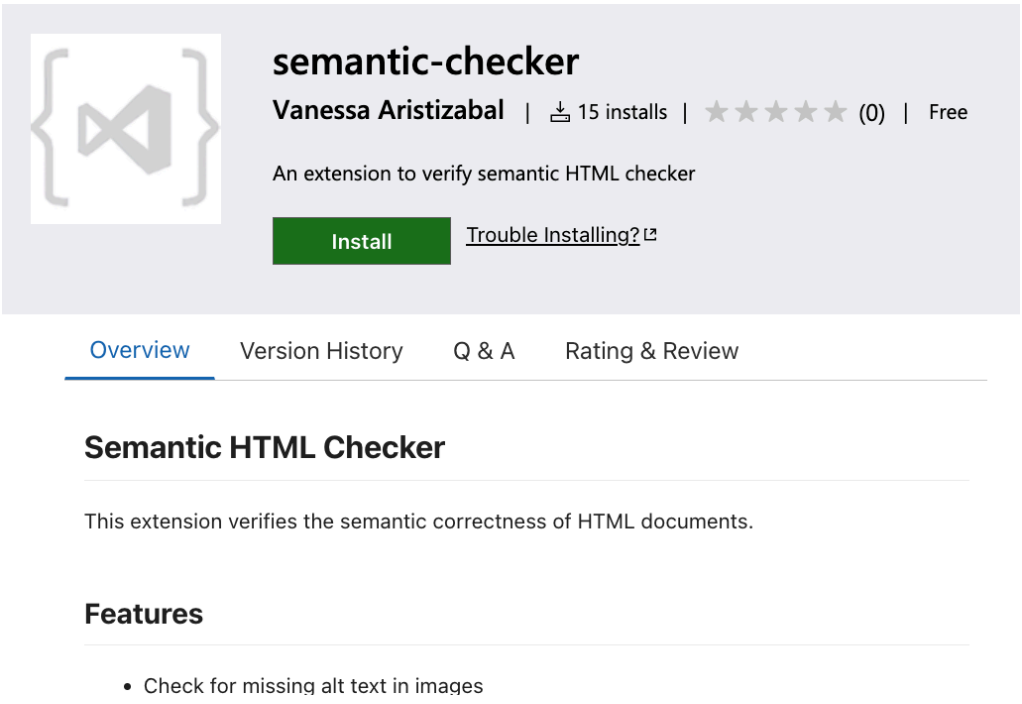


*Figura 12. Captura de la extensión en el Visual Studio Marketplace*

## 7.2 EVALUACIÓN COMPARATIVA DE LA ESTRATEGIA

Una vez implementada técnicamente la estrategia a través de "*Semantic Checker*" para *Visual Studio Code*, se evalúa la efectividad y utilidad práctica. La evaluación determina la capacidad de la estrategia para identificar correctamente errores de accesibilidad comunes en documentos *HTML*, y verificar su desempeño en relación con otras herramientas.

Para llevar a cabo esta evaluación, se diseñó un proceso comparativo utilizando un conjunto de documentos *HTML* de prueba. Estos documentos fueron creados específicamente con fines de prueba y diseñados intencionalmente para incluir errores comunes de accesibilidad, tomando como referencia los problemas frecuentemente identificados en informes de la industria como el *WebAIM Million Report 2025*. Este enfoque permitió validar la efectividad de las reglas definidas en la Tabla 5 al ser aplicadas en escenarios controlados.

Se seleccionaron tres herramientas para realizar un análisis comparativo con la extensión creada en la aplicación técnica de la propuesta, cada uno con una justificación específica para su inclusión:

- **La extensión "*Semantic Checker*":** Esta extensión es generada en la implementación técnica de la propuesta en este trabajo, su análisis es esencial para valorar la aplicación práctica de las reglas de la Tabla 5.

- ***axe DevTools* (Extensión de Navegador):** Se eligió esta herramienta por ser un referente consolidado en la industria para auditorías de accesibilidad. Su análisis sobre el *DOM* renderizado proporciona un punto de comparación con una herramienta estándar ampliamente utilizada por los desarrolladores [40]. Es importante destacar que *axe DevTools* opera con su propio motor de reglas exhaustivo (*axe-core*), el cual, si bien se basa en las *WCAG*, implementa un conjunto de pruebas y heurísticas diferentes en alcance y especificidad a las 82 reglas propuestas en la estrategia centradas en el *HTML*, definidas en la Tabla 5. Esta diferencia en los conjuntos de reglas es fundamental para interpretar los resultados de la comparación.

- **Modelos de Inteligencia Artificial:** Considerando la creciente aplicación de la IA como asistentes de código y la generación de documentación técnica, se incluyeron los modelos de IA *ChatGPT* y *Gemini* [44], para explorar su capacidad de aplicar de manera controlada las reglas

específicas de la Tabla 5.

El diseño de la evaluación comparativa se centró en medir el rendimiento de cada herramienta, para ello se realizaron las siguientes acciones:

- **Preparación del material evaluado:** La elaboración de un conjunto de documentos *HTML*, diseñados específicamente para contener errores representativos de accesibilidad.
- **Proceso detallado de análisis:** La aplicación de las herramientas seleccionadas a cada documento para realizar la evaluación comparativa.
- **Definición de métricas:** El establecimiento de los criterios cuantitativos para medir y valorar la efectividad de la estrategia propuesta.

El proceso detallado de cómo se aplicó cada una de estas herramientas de análisis y cómo se recopilaron los datos se describe en las subsecciones siguientes.

### 7.2.1 PREPARACIÓN DEL MATERIAL DE EVALUADO

Se elaboraron 10 documentos *HTML* de prueba. Cada documento contiene una selección deliberada de errores comunes de accesibilidad (alineados con las reglas de la Tabla 5).

La Tabla 6 detalla los errores que fueron introducidos intencionalmente en cada uno de los 10 documentos *HTML,* llamados: "*Example1*", "*Example2*", "*Example3*", "*Example4*", "*Example5*", "*Example6*", "*Example7*", "*Example8*", "*Example9*" y "*Example10*".

Tabla 6. Errores incluidos por documento *HTML*

| **Nombre del documento *HTML*** | **Tipo de error** |
|---|---|
| Example1 | Imagen sin *alt*, enlaces vacíos o ambiguos, botón vacío, falta etiqueta <*label*> en formulario, uso incorrecto de *ARIA*, falta atributo *lang*. |
| Example2 | Imagen con *alt* vacío, enlaces ambiguos, botón sin nombre accesible, saltos en jerarquía de encabezados, falta etiqueta <*label*> en formulario, contraste de color insuficiente, uso incorrecto de *ARIA*, falta *skip link*. |

| | |
|---|---|
| Example3 | Imagen decorativa sin *alt*, enlace sin *href*, texto inaccesible en botón, encabezado poco descriptivo, formulario sin etiquetas, uso redundante de *ARIA*, problemas de navegación por teclado, contraste insuficiente,falta *meta viewport*. |
| Example4 | Título de página no descriptivo, *alt* de imagen engañosa, enlace sin indicador de *foco*, formulario con *placeholders* en lugar de etiquetas, contenido multimedia sin subtítulos ni transcripción, contraste de color deficiente, uso redundante de *ARIA*, problemas de navegación por teclado, falta *meta viewport*. |
| Example5 | Imagen sin *alt*, enlace con texto vago, tabla sin encabezados, formulario sin botón de envío, contraste deficiente, *ARIA* mal usado, falta manejo de *foco*, audio *autoplay* sin controles, falta atributo *lang*. |
| Example6 | Título vacío, *SVG* sin *aria-label* o *aria-hidden*, *canvas* sin contenido alternativo, *object* sin texto alternativo. |
| Example7 | Formulario sin etiquetas, botón con solo ícono, sin *aria-label*. |
| Example8 | Texto con bajo contraste, texto con unidad fija (*px*), texto sobre fondo de imagen sin sombra. |
| Example9 | *Canvas* sin *fallback*, formulario sin etiquetas. |
| Example10 | Imagen con *alt* inadecuado, enlace sin nombre accesible, formulario sin etiquetas, tabla sin encabezados ni *caption*, video *autoplay* sin subtítulos, contraste bajo, problemas de navegación por teclado, *ARIA* mal usado, falta atributo *lang*, falta *meta viewport*. |

Los 10 documentos *HTML* de prueba usados en la evaluación, están disponibles en la siguiente *URL*, como se puede ver en la Figura 13: https://vanessamarely.github.io/docs-examples/.


*Figura 13. Captura de los ejemplos de documentos HTML en el navegador*

Los códigos correspondientes a los 10 documentos *HTML* de prueba, se encuentran en el siguiente repositorio *GitHub*, como se ve en la Figura 14, que es una captura de pantalla del repositorio: https://github.com/vanessamarely/docs-examples.

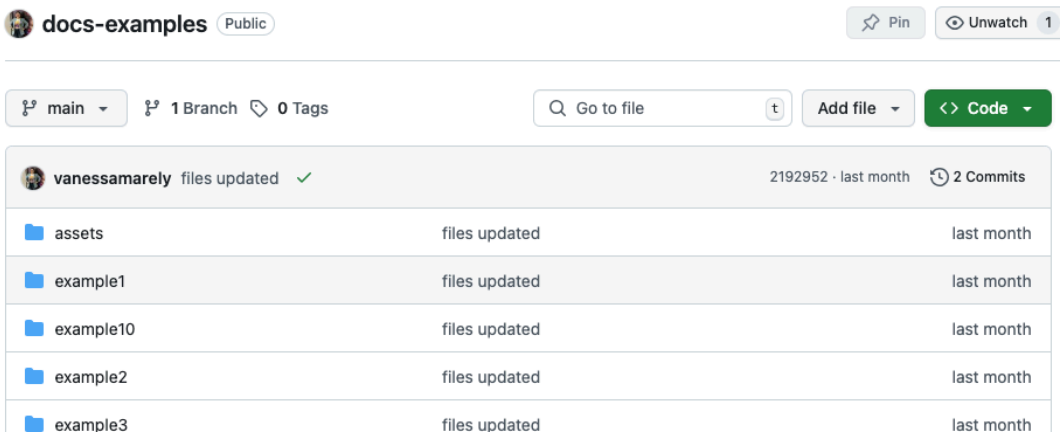


*Figura 14. Captura del repositorio del código de los documentos HTML*

### 7.2.2 PROCESO DETALLADO DE ANÁLISIS

Para ilustrar el proceso de evaluación, se toma como ejemplo el análisis del documento llamado "*Example1*". Este mismo procedimiento se replicó para los 10 documentos *HTML*. Cada documento fue analizado con las tres herramientas seleccionadas: la extensión "*Semantic Checker", axe DevTools* y los modelos de Inteligencia Artificial, para comparar sus respectivas capacidades de detección de errores, resumido en la Tabla 7.

Tabla 7. Análisis de accesibilidad aplicado a las herramientas

| **Paso** | **Descripción General del Paso** | **Extensión "Semantic Checker"** | **axe DevTools (Extensión de Navegador)** | **Inteligencia Artificial (ChatGPT y Gemini)** |
|---|---|---|---|---|
| 1 | Preparar el entorno | Se abre el documento *HTML* en el editor de código *Visual Studio Code*. | Se carga la página *HTML* renderizada en un navegador web (Google Chrome). | Se prepara un prompt que incluye las reglas de la Tabla 5 y el contenido del código *HTML* del documento a analizar. |
| 2 | Activar la herramienta | Se ejecuta la extensión manualmente desde la paleta de comandos del editor. | Se activan las herramientas de desarrollador en el navegador y se inicia el escaneo de accesibilidad con la extensión *axe DevTools*. | Se ejecuta la interacción con el modelo de IA enviando el prompt completo a la interfaz correspondiente. |
| 3 | Ejecutar el análisis | La extensión analiza el código *HTML* aplicando las 82 reglas. | La herramienta analiza el *DOM* de la página renderizada aplicando su propio conjunto de reglas (*axe-core*). | El modelo de IA procesa el prompt basándose en las 82 reglas proporcionadas en el prompt. |
| 4 | Revisar los resultados | Se visualizan los errores detectados como advertencias en el editor, en el panel de "Problemas" y como notificaciones en el editor. | Se revisa el listado de problemas de accesibilidad generados en la interfaz de la herramienta, incluyendo su severidad. | Se recopila y examina la respuesta del modelo, que incluye el listado de errores y el conteo total. |

A. **Análisis con la extensión "*Semantic Checker*" para *Visual Studio Code*:**

1. **Preparar el entorno:** Se abrió el archivo *HTML* de la página "*Example1*" en el editor *Visual Studio Code*. El contenido del documento es el siguiente:

```
<!DOCTYPE html>
<html>
 <head>
   <title>Test Accessibility</title>
   <link rel="stylesheet" href="styles.css" />
   <link rel="preconnect" href="https://fonts.googleapis.com" />
   <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
   <link

href="https://fonts.googleapis.com/css2?family=Noto+Sans:ital,wght@0,100..900;1,100..
900&display=swap"
     rel="stylesheet"
   />
 </head>
 <body class="test-accessibility">
   <!-- Missing lang attribute in <html> -->
   <header>
     <h1>Welcome to Our Website</h1>
   </header>
   <main>
     <section class="test-accessibility__content">

       <!-- Image without alt text -->
       <img src="./../assets/image.jpg" />
       <!-- Empty link -->
       <a href="#">Click here</a>
       <!-- Empty button -->
       <button></button>
       <!-- Ambiguous link (short text) -->
       <a href="#">Go</a>
       <!-- Missing skip link -->
       <section id="maincontent">
         <h2>Main Content</h2>
         <p>This is the main content section.</p>
       </section>

       <!-- Form field without label -->
```

```
      <form>
        <input type="text" />
      </form>

      <!-- Incorrect ARIA usage (missing region role) -->
      <div class="content">
        <p>This is a content section with no ARIA roles.</p>
      </div>
    </section>
  </main>
  <footer>
    <p>Contact us for more information.</p>
  </footer>
 </body>
</html>
```

2. **Activar la herramienta:** Se ejecutó el comando para activar la extensión desde la paleta de comandos en *Visual Studio Code*. En *MacOS*, la combinación de teclas es ***Cmd*** + ***Shift*** + ***P***, y en *Windows*, ***Ctrl*** + ***Shift*** + ***P***. A continuación, se elige la opción "*Verify Semantic HTML*", como se detalla en la Figura 15. Este comando activa la extensión "*Semantic Checker*", que verifica que es un documento *HTML*.

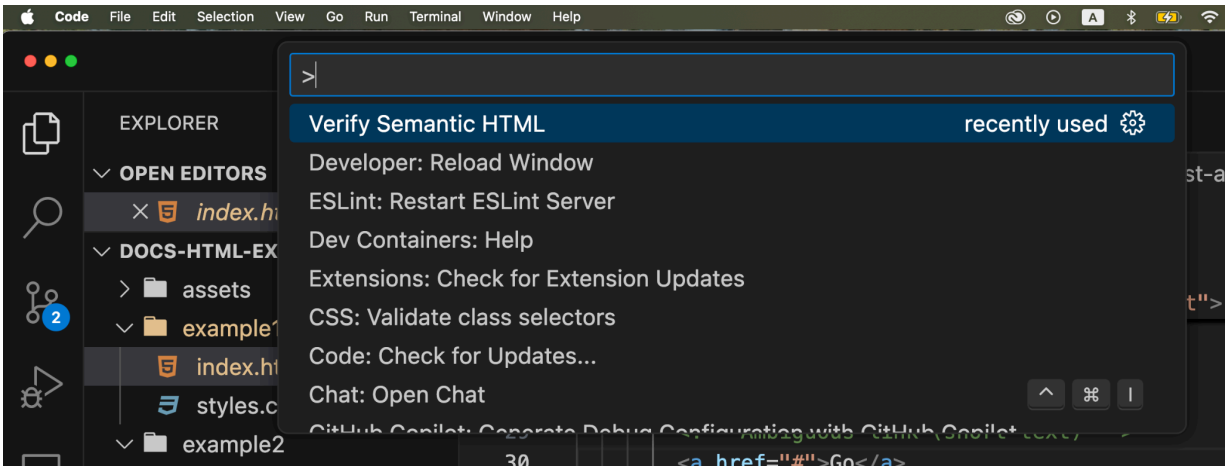


*Figura 15. Paleta de comandos de Visual Studio Code*

3. **Ejecutar el análisis:** La extensión analiza el contenido del documento *HTML*, utilizando una lista de comprobaciones basadas en expresiones regulares, identifica patrones que coinciden con errores definidos en la Tabla 5. Por cada coincidencia, se genera un mensaje de advertencia

(*warning*) que incluye: el número de la regla (por ejemplo, R1), la descripción del error, una recomendación, y una referencia para ampliar información.

4. **Revisar los resultados:** Los errores detectados por la extensión se muestran directamente en el editor, subrayando las líneas de código problemáticas y presentando una advertencia en el panel de "Problemas", como se puede ver en la Figura 16.

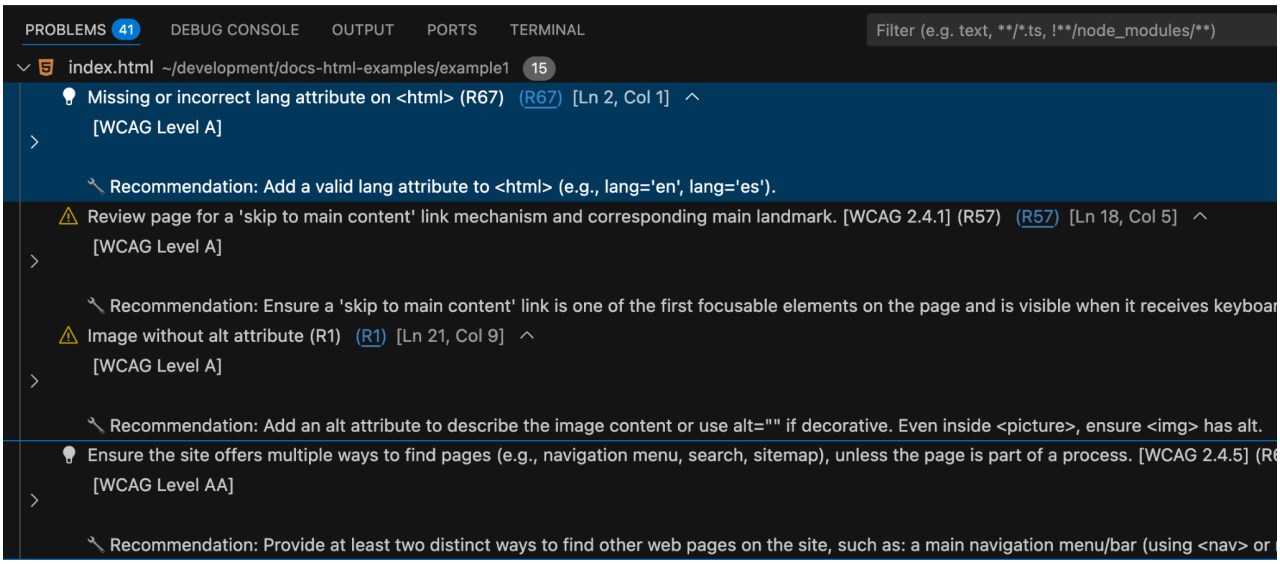


*Figura 16. Panel de problemas en Visual Studio Code*

Cada advertencia incluye el número de reglas de la Tabla 5, una descripción del error y la recomendación para su corrección como lo muestra la Figura 17.

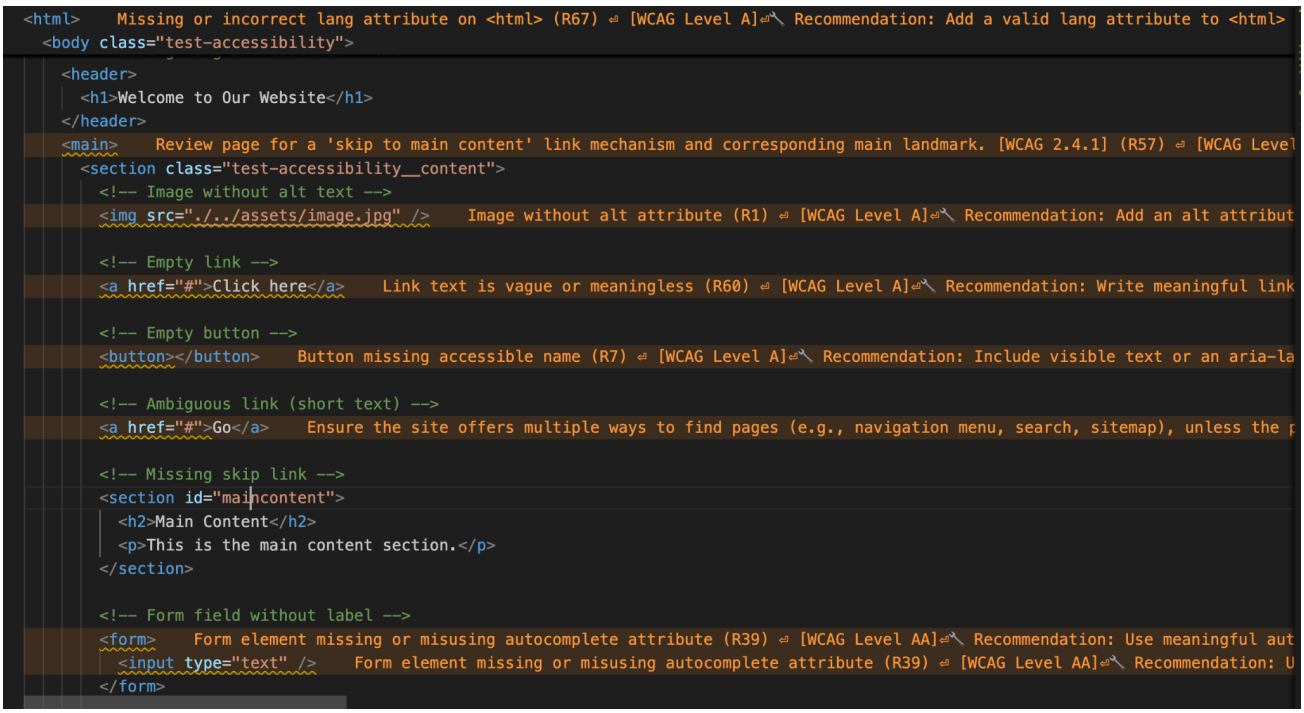


*Figura 17. Captura visualización de resultados en Visual Studio Code*

La cantidad de errores reportados por la extensión para el documento *HTML* "*Example1*", fue de un total de 15 errores. Esta cantidad y el detalle se muestran como una notificación en el editor de *Visual Studio Code*, como se puede ver en la Figura 18, donde también se ofrece un desglose por nivel de conformidad *WCAG* 2.2, mostrando 6 errores asociados al nivel de conformidad A, 8 errores al nivel AA y un único error al nivel AAA.

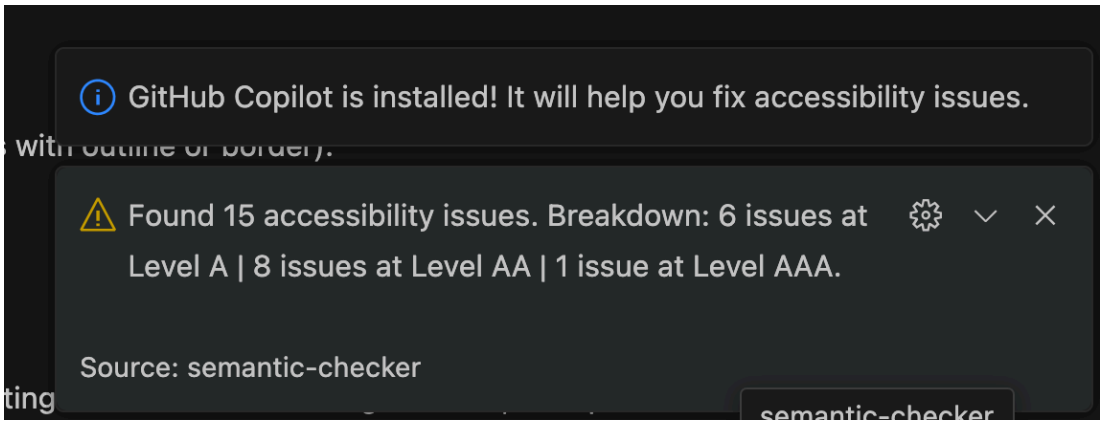


*Figura 18. Visualización de errores en el editor de Visual Studio Code*

B. **Análisis con *axe DevTools* (Extensión de Navegador):**

1. **Preparar el entorno:** La página de prueba "*Example1*" se abrió en el navegador *Google Chrome*, donde la extensión *axe DevTools* estaba previamente instalada, usando la siguiente *URL*: https://vanessamarely.github.io/docs-examples/example1/index.html

2. **Activar la herramienta:** Se activó *axe DevTools* desde las herramientas de desarrollador del navegador como se puede ver en la Figura 19.

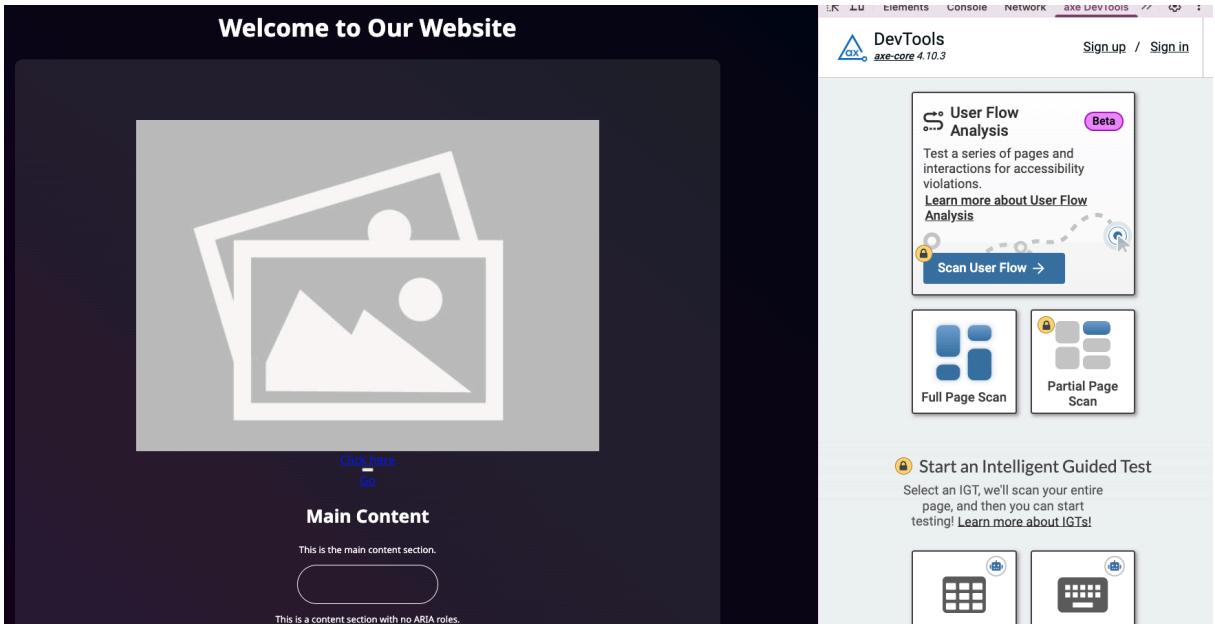


*Figura 19. Interfaz de axe DevTools en el navegador*

3. **Ejecutar el análisis:** Se inició un escaneo de accesibilidad en la página renderizada.

4. **Revisar los resultados:** Se examinaron los problemas de accesibilidad reportados por *axe DevTools*, que presenta un listado de los problemas identificados, detallando para cada uno su descripción, el impacto o severidad (crítico, serio, moderado, menor), y las recomendaciones específicas para su corrección. Para la página "*Example1*", *axe DevTools* reportó un total de 3 errores, como se puede ver en la Figura 20. Se destaca que estos hallazgos se basan en el conjunto de reglas propio de *axe DevTools* (*axe-core*), el cual, si bien está alineado con las *WCAG* 2.1, difiere en su alcance y especificidad de las 82 reglas la Tabla 5, utilizadas por la extensión "*Semantic Checker*".

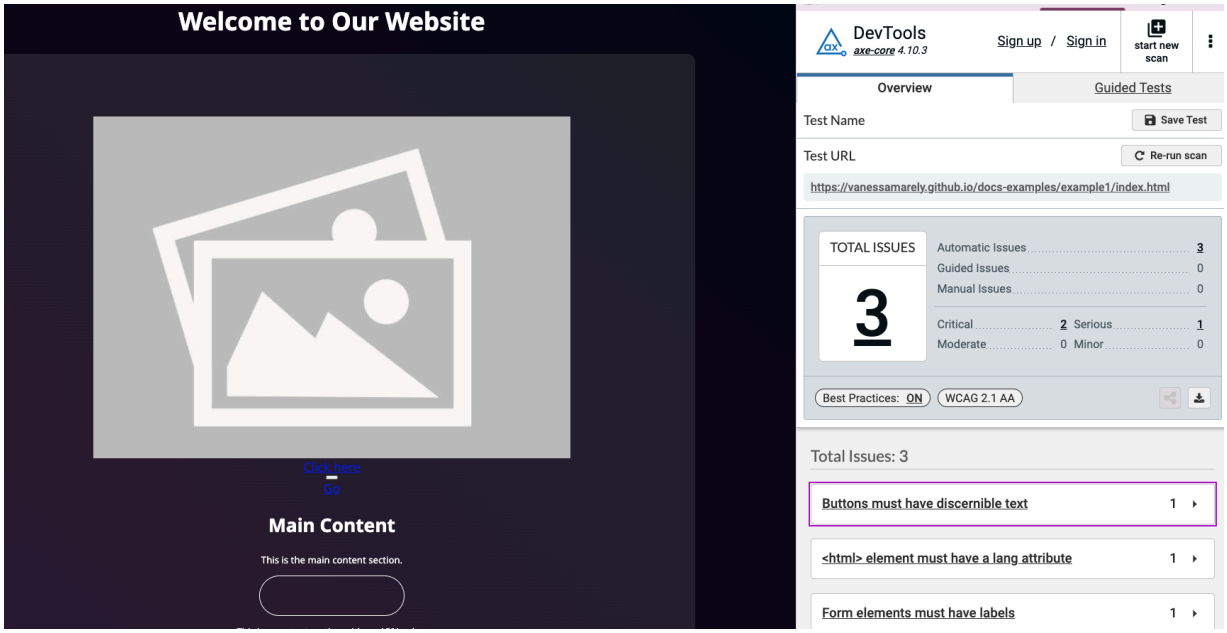


*Figura 20. Lista de errores detectados por axe DevTools*

C. **Análisis con Modelos de Inteligencia Artificial (*ChatGPT* y *Gemini*):**

1. **Preparar el entorno:** Para el documento *HTML* (*Example1*) de prueba, se preparó el prompt para los modelos de IA (*ChatGPT* y *Gemini*), que se puede ver en la Tabla 8. Esto incluyó:

- El contenido textual completo de la Tabla 5.
- El código *HTML* completo del documento de prueba a analizar.
- El prompt de la Tabla 8, diseñado para instruir a los modelos que basan su análisis estricta y exclusivamente en las reglas de la Tabla 5.

Tabla 8. Prompt para los modelos de IA

| **Prompt para los modelos de IA** |
|---|
| You are an AI assistant specializing in web accessibility auditing. Your task is to analyze the *HTML* document that will be provided below.<br><br>**Core Instructions:**<br><br>1. You will use a reference document titled 'Tabla 5.pdf' (Reglas de verificación *HTML-WCAG*), which I will ensure you have access to or provide its content.<br>2. Your analysis to identify accessibility errors in the *HTML* must be based **strictly and |

exclusively** on the rules ('Regla #') defined within 'Tabla 5.pdf'.
3. Do not use any other accessibility knowledge, heuristics, or external *WCAG* guidelines. If a potential issue is observed in the *HTML* but is not explicitly covered by a rule in 'Tabla 5.pdf', it should **not** be reported as an error for this task. Assume 'Tabla 5.pdf' is the complete and sole source of truth for identifying errors.

**Input:**

* **Reference Rules:** 'Tabla 5.pdf' (Reglas de verificación *HTML-WCAG*)
* ***HTML* Document to Analyze:**
  ```html
  [Contenido del documento *HTML*]
  ```

**Output Requirements:**

For the provided *HTML* document, please:

1. Identify every instance where a rule from 'Tabla 5.pdf' is violated.
2. For each identified violation, provide:
   * The 'Regla #' (Rule Number) from 'Tabla 5.pdf'.
   * The 'Error común' (Common Error) or a brief description of the issue as stated in 'Tabla 5.pdf' for that specific rule and *HTML* tag.
   * The specific *HTML* code snippet from the analyzed document that demonstrates the violation.
3. Conclude your analysis with a **total numerical count** of all distinct violations found based *only* on the rules in 'Tabla 5.pdf'.

Please proceed with the analysis once the *HTML* document and 'Tabla 5.pdf' (or its content) are available.

2. **Activar la herramienta:** Se interactuó con *ChatGPT* y *Gemini* por separado, abriendo el entorno de cada uno.

3. **Ejecutar el análisis:** Se proporciona la información preparada a los modelos de IA, solicitando el análisis según el prompt de la Tabla 8, como se puede ver en la Figura 21 y Figura 22 respectivamente.

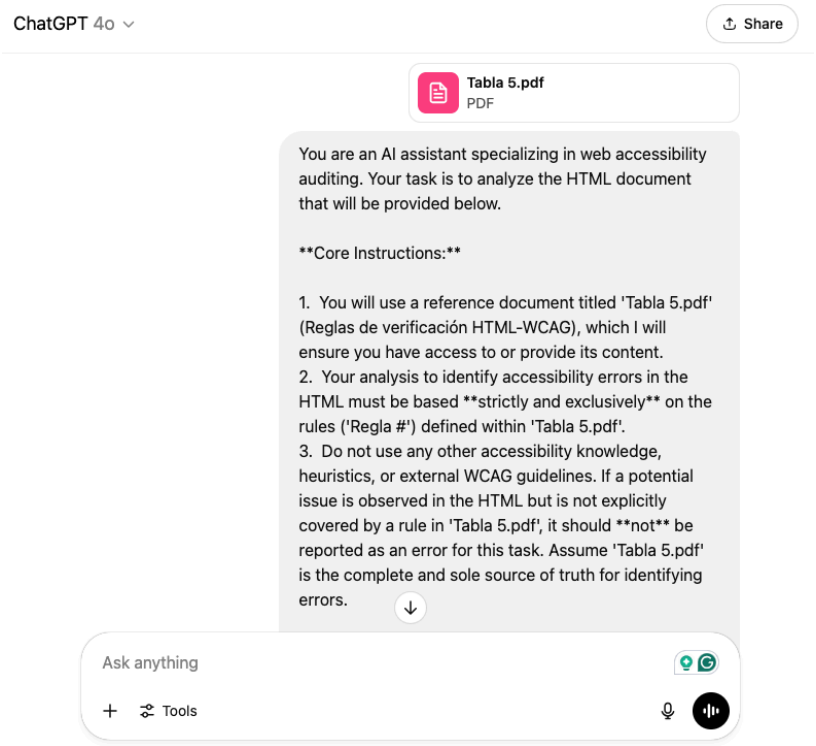


*Figura 21. Extracto del prompt en ChatGPT*

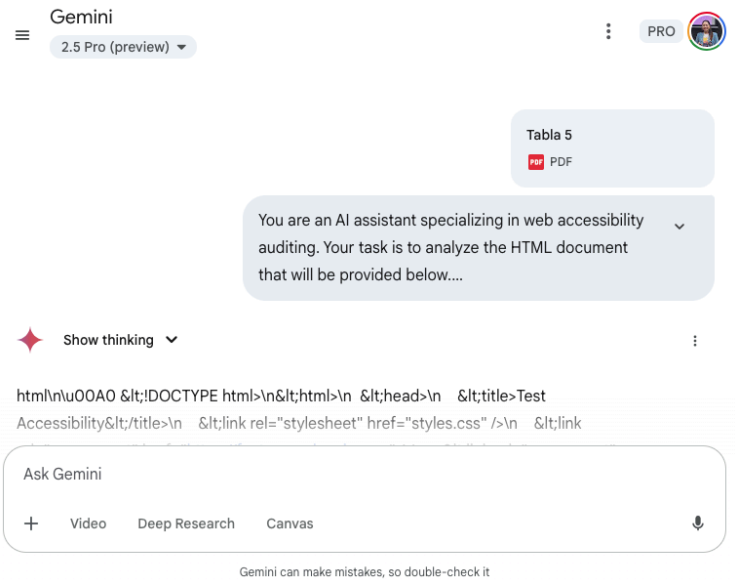


*Figura 22. Extracto del prompt en Gemini*

4. **Revisar los resultados:** De las respuestas obtenidas de cada modelo de Inteligencia Artificial para la página "*Example1*", se extrajo el conteo total de errores identificados según las reglas de la Tabla 5.

- **Para *ChatGPT*:** Específicamente el modelo *GPT-4o*, como se observa en la Figura 23, se reportó un total de 7 violaciones y un resumen de las reglas específicas que el modelo identificó como infringidas (R1, R7, R27, R57, R60, R64 y R67).

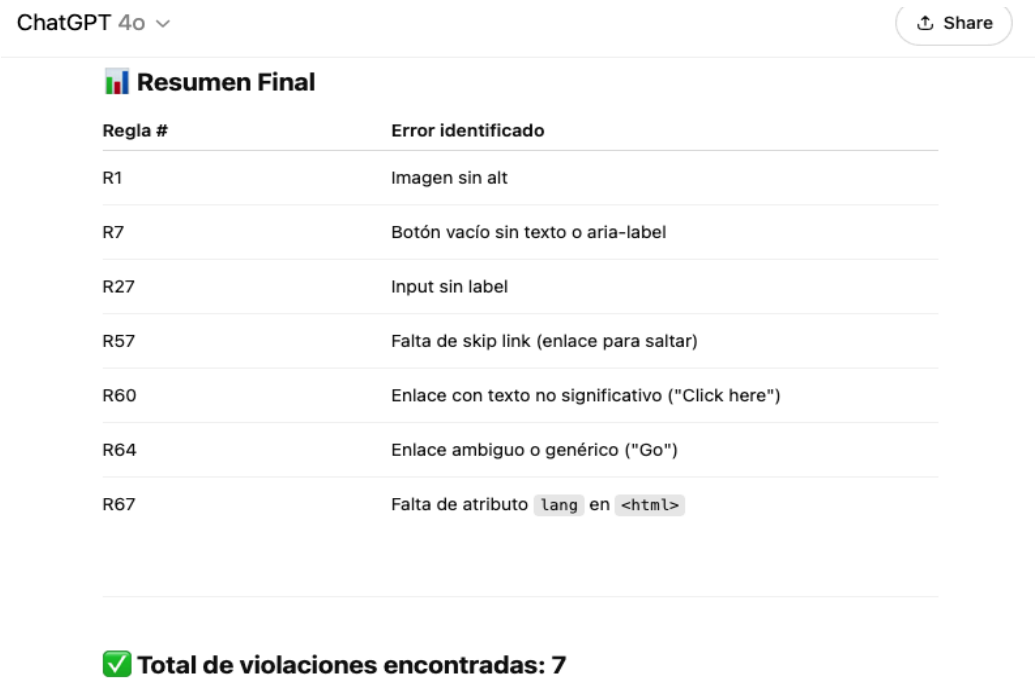


*Figura 23. Extracto de la respuesta ChatGPT*

- **Para *Gemini*:** utilizando el modelo *2.5 Pro* (*preview*), según la Figura 24, indicó un total de 8 violaciones distintas encontradas, basadas exclusivamente en las reglas de la Tabla 5.

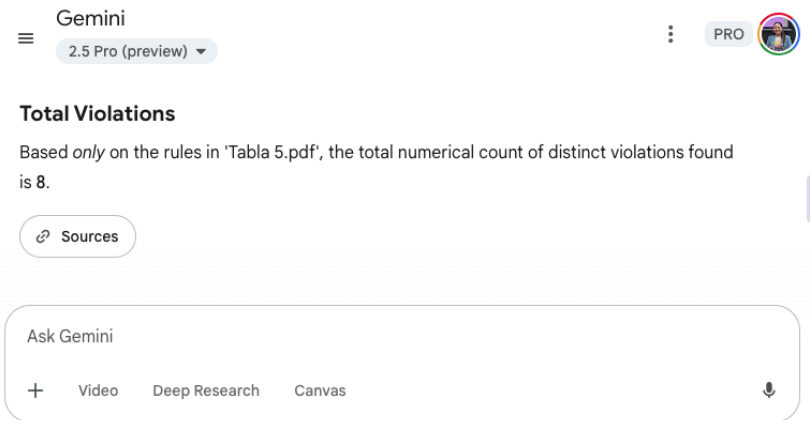


*Figura 24. Extracto de la respuesta Gemini*

### 7.2.3 DEFINICIÓN DE MÉTRICAS

La valoración de la estrategia y su implementación en la extensión "*Semantic Checker*", se basó en un análisis comparativo de la detección de errores. Por lo tanto, se definieron las siguientes métricas principales:

- **Número de errores detectados por herramienta**: El recuento total de los errores de accesibilidad hallados se realizó mediante:
    - La extensión de *Visual Studio Code:* "*Semantic Checker*", basado en las 82 reglas de la Tabla 5.
    - La extensión de navegador *axe DevTools*, que aplica su propio conjunto de reglas.
    - Los modelos de inteligencia artificial, *ChatGPT* y *Gemini*, instruidos con la Tabla 5 y el prompt especificado en la Tabla 8.
- **Análisis comparativo de la capacidad de detección**: Se realizó una comparación de los hallazgos entre las diferentes herramientas.

Estas métricas permiten evaluar no solo la capacidad de la extensión desarrollada para identificar los errores definidos en la estrategia, sino también revisar su desempeño frente a herramientas establecidas y enfoques basados en IA.

En resumen, esta sección detalla la implementación práctica de la extensión "*Semantic Checker*" y la evaluación de la misma comparándola con otras herramientas (*axe DevTools, ChatGPT* y *Gemini)* en diferentes documentos *HTML*, definiendo unas métricas de recolección de datos. Este diseño asegura que la validación de la estrategia es comparable y objetiva.

Los hallazgos y datos recopilados a través de la aplicación de este proceso se presentan y analizan en detalle en la siguiente Sección 8 "Resultados".

# 8. RESULTADOS

En esta sección se exponen los resultados derivados de la implementación de la estrategia de accesibilidad web en documentos *HTML* mediante el uso de la extensión desarrollada para *Visual Studio Code:* "*Semantic Checker*" basada en las reglas de la Tabla 5, para detectar errores en documentos *HTML* de prueba. El desempeño de "*Semantic Checker*" se comparó con *axe DevTools* (ejecutada como extensión de navegador) y con el análisis realizado en modelos de Inteligencia Artificial (*ChatGPT* y *Gemini*), los cuales fueron instruidos para utilizar exclusivamente las reglas de la Tabla 5.

Los resultados se presentan de la siguiente manera:

- **Resultados cuantitativos:** Se detalla la cantidad de errores identificados por cada herramienta de validación aplicada, incluyendo la extensión desarrollada, *axe DevTools* y los modelos de IA (*ChatGPT* y *Gemini*).
- **Resultados cualitativos:** Se realiza una comparación de las herramientas, destacando sus fortalezas, limitaciones y utilidad práctica para los desarrolladores.

La estructura propuesta permite organizar el análisis de resultados, comenzando por el contexto de evaluación (documentos y métricas) y culminando con un análisis comparativo que sustenta las conclusiones del estudio.

## 8.1 RESULTADOS CUANTITATIVOS

Tras aplicar las tres herramientas de análisis (Extensión "*Semantic Checker*", *axe DevTools*, y los modelos de Inteligencia Artificial *ChatGPT* y *Gemini*) a los 10 documentos *HTML* de prueba, se contabilizó el número total de errores detectados por cada uno, como se puede ver en la Tabla 9 de la comparación entre herramientas.

Tabla 9. Comparación de resultados de errores detectados en los documentos *HTML*

| **Nombre del documento *HTML*** | **Errores detectados (*Semantic Checker*)** | **Errores detectados (*axe DevTools*)** | **Errores detectados por *ChatGPT*** | **Errores detectados por *Gemini*** |
|---|---|---|---|---|
| Example1 | **15** | 3 | 7 | 8 |
| Example2 | **17** | 3 | 8 | 11 |
| Example3 | **21** | 6 | 8 | 10 |
| Example4 | **24** | 1 | 9 | 14 |
| Example5 | **21** | 5 | 9 | 9 |
| Example6 | **18** | **7** | 4 | 6 |
| Example7 | **15** | 4 | 3 | 5 |
| Example8 | **5** | 3 | 3 | **5** |
| Example9 | **14** | 5 | 2 | 6 |
| Example10 | **26** | 4 | 8 | 14 |

Estos datos cuantitativos se consolidan en la Figura 25, las cuales detallan el número de errores identificados por cada herramienta, según las condiciones de evaluación descritas en la Sección 7 “Aplicación técnica de la propuesta”.

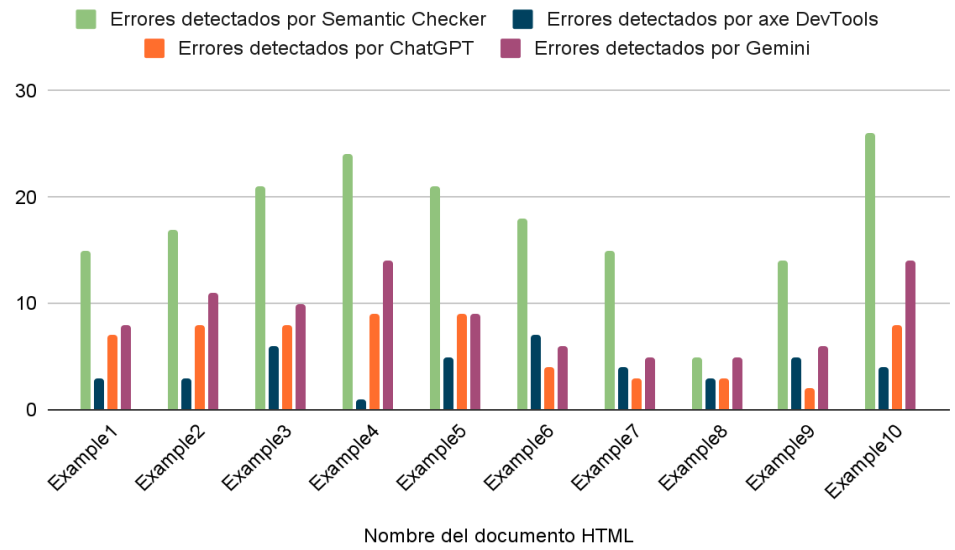


*Figura 25. Errores detectados* en los documentos *HTML*

## 8.2 RESULTADOS CUALITATIVOS

El análisis de los datos presentados en la Tabla 9 se complementa con una interpretación cualitativa de las diferencias observadas en la detección de errores entre las herramientas evaluadas. Esta interpretación considera el alcance, el método de detección y el contexto de análisis de cada una:

- **Alcance:**
  - La extensión de *"Semantic Checker"* de *Visual Studio Code* (basada en expresiones regulares), se basa estrictamente en las 82 reglas definidas en la Tabla 5 y utiliza expresiones regulares para la detección. El alcance está limitado a lo que estas reglas cubren y a la capacidad de las expresiones regulares para identificar patrones de error en el código *HTML*.
  - *axe DevTools* opera sobre el renderizado en el navegador y aplica un conjunto de reglas mucho más amplio, que van más allá de la verificación de patrones en el *HTML* estático.
  - Los modelos de IA (*ChatGPT* y *Gemini*), pueden tener dificultades para adherirse estrictamente a un conjunto limitado de reglas y evitar el uso de su conocimiento general sobre accesibilidad. Su interpretación de las reglas y del código *HTML* también puede variar, llevando a diferencias en los errores reportados.

- **Método de Detección:**
  - La extensión de *"Semantic Checker"* de *Visual Studio Code* (basada en expresiones regulares) es eficaz para errores sintácticos o de atributos faltantes claramente definidos en la Tabla 5, pero puede ser menos precisa para errores contextuales o que requieren una comprensión más profunda de la semántica o el comportamiento de la página. La precisión es un factor limitante; algunas reglas pueden ser difíciles de traducir perfectamente a patrones de las expresiones regulares, llevando a posibles falsos negativos o positivos.
  - *axe DevTools* combina análisis estático del *DOM* lo que le permite realizar un análisis de los problemas amplio en cuanto a que no solo aborda el *HTML*, pero podría no detectar errores que él no considere importantes.
  - Los modelos de IA (*ChatGPT* y *Gemini*), procesan el lenguaje natural de las reglas y el código, lo que puede ser una ventaja para ciertos errores contextuales, pero también una desventaja si se interpretan o se desvían de las reglas proporcionadas.

- **Contexto de Análisis:**
    - La extensión de *"Semantic Checker"* de *Visual Studio Code* (basada en expresiones regulares), analiza el código de *HTML* directamente en el editor.
    - *axe DevTools* analiza la página tal como se renderiza en el navegador, incluyendo el impacto de *CSS* y *JavaScript* en la estructura y accesibilidad final.
    - Los modelos de IA (*ChatGPT* y *Gemini*), analizan el código de *HTML* incluido en el prompt que se le provee a los chats.

En conclusión, los resultados demuestran que una herramienta desarrollada como *"Semantic Checker"* con reglas claras y aplicables desde el entorno de desarrollo puede ser una opción efectiva y más al alcance para los desarrolladores al estar en el editor. Además, evidencia que el enfoque basado en código *HTML* y expresiones regulares resulta altamente funcional para detectar errores reales y recurrentes de accesibilidad. Esta validación respalda la viabilidad de la estrategia propuesta y sienta las bases para su aplicación y evolución futura.

# 9. CONCLUSIONES

En respuesta a la pregunta de investigación sobre cómo facilitar la implementación de la accesibilidad web en documentos *HTML,* conforme a las *WCAG* 2.2, este trabajo definió una estrategia que aborda esta necesidad. Para alcanzar el objetivo general de definir dicha estrategia, se adoptó la metodología de *Design Science Research* (*DSR*), que guió el proceso desde la identificación del problema, de la brecha existente entre los estándares en cuanto a la accesibilidad y su aplicación, hasta el diseño, desarrollo y evaluación de los componentes de la solución.

A continuación, se resaltan las contribuciones de este trabajo:

1. La identificación de las barreras técnicas a las que se enfrentan los desarrolladores a la hora de aplicar la accesibilidad web y los errores comunes que dificultan la aplicación de la accesibilidad en los documentos de *HTML*.

2. A partir del análisis de las *WCAG* 2.2 (abarcando sus principios, pautas y criterios de éxito), se seleccionaron 47 criterios de éxito aplicables a las etiquetas y atributos del *HTML*, excluyendo 39 criterios de éxito relacionados con *CSS*, *Javascript* y multimedia.

3. El diseño de una estrategia que traduce 47 criterios de éxito de las *WCAG* 2.2 previamente elegidos, convertidos en 82 reglas, detalladas en la Tabla 5, creando así un puente directo entre la teoría y la aplicación. Esta tabla provee un vínculo entre los criterios de éxito, sus elementos de aplicación en el *HTML*: las etiquetas y atributos, los errores comunes y las técnicas suficientes del *W3C*.

4. La implementación de la estrategia de forma práctica en una extensión para *Visual Studio Code* denominada "*Semantic Checker*", que implementa las 82 reglas definidas mediante expresiones regulares para ofrecer a los desarrolladores retroalimentación sobre errores de accesibilidad directamente en el editor de *Visual Studio Code*. La extensión "*Semantic Checker*" fue probada en los 10 documentos *HTML*, que se crearon con los errores comunes que se identificaron en accesibilidad.

5. La evaluación de "*Semantic Checker*", comparándolo con herramientas como *axe DevTools* y modelos de IA (*ChatGPT y Gemini*), arrojó los siguientes hallazgos:

- La extensión "*Semantic Checker*" demostró ser eficaz en la detección de errores relacionados con las 82 reglas específicas de la Tabla 5 para las cuales fue diseñada, identificando un número considerable de los errores predefinidos en las páginas de prueba.
- Se observaron diferencias en el número y tipo de errores detectados al comparar la extensión con *axe DevTools* y los modelos de IA (*ChatGPT* y *Gemini*). Estas variaciones se atribuyen al alcance de los conjuntos de reglas, la granularidad del conteo de errores, y la capacidad de las IA para adherirse estrictamente a un conjunto limitado de reglas.
- Se determinó que el enfoque basado en expresiones regulares resultó funcional para identificar errores sintácticos y de atributos faltantes comunes directamente en los documentos *HTML*, aunque presenta limitaciones para errores contextuales o que dependan de una necesaria evaluación humana para verificar otros aspectos adicionales a algunos criterios de éxito.

Los resultados encontrados permiten concluir que la estrategia propuesta y su implementación como una extensión para el editor de código contribuyen a facilitar la incorporación temprana de prácticas de accesibilidad web. Al proporcionar a los desarrolladores una herramienta que señala errores específicos de *HTML* alineados con *WCAG* 2.2 directamente durante el proceso de creación de etiquetas y sus atributos en los documentos *HTML*, se promueve la corrección proactiva. La Tabla 5, en particular, se constituye como un recurso práctico y educativo de la accesibilidad web.

**Limitaciones**

Es importante reconocer las limitaciones y alcance de este estudio y de la herramienta desarrollada, "*Semantic Checker*":

1. La evaluación se basó en un conjunto limitado de 10 documentos *HTML* de prueba creados en este trabajo. Los resultados encontrados en los documentos varían de acuerdo a la herramienta aplicada, y no abarcan la complejidad de errores de documentos publicados en sitios web del mundo real, como los de ámbitos de nivel corporativo, que podrían ser considerablemente mayores.
2. Por diseño, la estrategia focaliza su atención en los criterios de éxito *WCAG* 2.2, aplicables etiquetas y atributos *HTML*. Aunque no todos los criterios de éxito son exclusivos del *HTML*, hay otros que abarcan el *CSS* y *JavaScript*, que no fueron incluidos en este trabajo y quedaron fuera del alcance.

3. La herramienta desarrollada "*Semantic Checker*" fue creada como un apoyo para los desarrolladores cuando están en el proceso de edición del código *HTML*, pero no reemplaza las auditorías de accesibilidad exhaustivas realizadas con herramientas especializadas o por expertos en accesibilidad.
4. Las expresiones regulares identifican patrones de forma precisa, pero no tienen la capacidad para evaluar el contexto o la calidad del contenido. Por ello, las reglas implementadas podrían requerir un refinamiento y pruebas adicionales para mejorar su efectividad.

**Trabajo futuro**

Considerando los hallazgos y las limitaciones de este estudio, se identifican varias líneas para trabajo futuro que permitirían expandir y potenciar la estrategia y la extensión "*Semantic Checker*" desarrolladas:

- **Ampliación del alcance de las reglas:**
  - **Cobertura de *CSS* y *JavaScript*:** Creación de reglas para los criterios de éxito de las *WCAG* 2.2 para el *CSS* y *JavaScript*. Esto implicaría analizar las técnicas suficientes correspondientes para estas tecnologías y traducir los criterios de éxito en validaciones aplicables a la presentación y el comportamiento de las páginas web, complementando el enfoque actual del *HTML*.
  - **Refinamiento de reglas *HTML*:** Continuar con la mejora de la precisión de las 82 reglas de expresiones regulares existentes para *HTML*, con el objetivo de minimizar aún más la incidencia de falsos positivos y falsos negativos.
- **Mejoras funcionales en la extensión "*Semantic Checker*":**
  - **Asistencia Inteligente para correcciones:** Explorar la integración de modelos de Inteligencia Artificial dentro de la extensión, no solo para la detección, sino también para que la herramienta pueda sugerir fragmentos de código *HTML* corregido o incluso permitir la aplicación de soluciones automáticas para errores comunes y no ambiguos.
  - **Integración con documentación detallada y ejemplos:** Mejorar la retroalimentación al desarrollador vinculando las advertencias directamente con explicaciones más profundas de los criterios *WCAG* 2.2, las técnicas suficientes y ejemplos de buenas prácticas.

En definitiva, este trabajo ha demostrado la utilidad de una estrategia enfocada sobre los documentos *HTML* para mejorar la accesibilidad web. Se espera que tanto la metodología como la extensión

desarrollada sirvan de base para futuras investigaciones y desarrollos que busquen hacer la web un lugar más inclusivo para todas las personas.

# REFERENCIAS


[1] W3C Web Accessibility Initiative, “Introduction to Web Accessibility”, W3C. [Online]. Available: https://www.w3.org/WAI/fundamentals/accessibility-intro/. [Accessed: Sep. 08, 2024].

[2] W3C, “Web Content Accessibility Guidelines (WCAG) 2.2”, W3C Recommendation, Dec. 12, 2024. [Online]. Available: https://www.w3.org/TR/WCAG22/. [Accessed: Sep. 08, 2024].

[3] WebAIM, “The WebAIM Million: The 2025 Report on the Accessibility of the Top 1,000,000 Home Pages”, Feb, 2025. [Online]. Available: https://webaim.org/projects/million/. [Accessed: Mar. 02, 2025].

[4] H. Y. Abuaddous, M. Z. Jali, and N. Basir, “Web Accessibility Challenges,” *Int. J. of Adv. Comp. Sci.e and Appl.*, vol. 7, no. 10, pp. 172–181, 2016. [Online]. Available: https://thesai.org/Downloads/Volume7No10/Paper_23-Web_Accessibility_Challenges.pdf. [Accessed: Mar. 08, 2025].

[5] M. Halpin, “Understanding the Barriers to Accessibility,” Recite Me, 2024. [Online]. Available: https://reciteme.com/news/barriers-to-accessibility/. [Accessed: Sep. 08, 2024].

[6] Accessibility Guidelines Working Group (AG WG), “Introduction to Understanding WCAG 2.2,” W3C Web Accessibility Initiative (WAI), Apr. 01, 2025. [Online]. Available: https://www.w3.org/WAI/WCAG22/Understanding/intro. [Accessed: Apr. 08, 2025].

[7] W3C Web Accessibility Initiative, “Accessibility Principles,” W3C, Jul. 15, 2024. [Online]. Available: https://www.w3.org/WAI/fundamentals/accessibility-principles/. [Accessed: Apr. 08, 2025].

[8] W3C Web Accessibility Initiative, “How to Meet WCAG (Quick Reference)”, W3C, Oct. 03, 2024. [Online]. Available: https://www.w3.org/WAI/WCAG22/quickref/. [Accessed: Apr. 08, 2025].

[9] W3C Web Accessibility Initiative, “WCAG 2 Overview,” W3C, May. 6, 2025. [Online]. Available: https://www.w3.org/WAI/standards-guidelines/wcag/. [Accessed: May 08, 2025].

[10] W3C Web Accessibility Initiative, “Accessibility Conformance Testing (ACT) Overview,” W3C, Oct. 30, 2020. [Online]. Available: https://www.w3.org/WAI/standards-guidelines/act/. [Accessed: Mar. 08, 2025].

[11] W3C Web Accessibility Initiative, “Web Accessibility Evaluation Tools List,” W3C, [Online]. Available: https://www.w3.org/WAI/test-evaluate/tools/list/. [Accessed: Mar. 08, 2025].

[12] World Wide Web Consortium (W3C), “About us,” 2025. [Online]. Available: https://www.w3.org/about/. [Accessed: Mar. 08, 2025].

[13] WCAG 2.2 Understanding, “Understanding Techniques for WCAG 2.2 Success Criteria,” W3C, Apr. 01, 2025. [Online]. Available: https://www.w3.org/WAI/WCAG22/Understanding/understanding-techniques. [Accessed: Apr. 08, 2025].

[14] WCAG 2.2 Techniques, "Techniques for WCAG 2.2," W3C, Apr. 01, 2025. [Online]. Available: https://www.w3.org/WAI/WCAG22/Techniques/. [Accessed: Apr. 09, 2025].

[15] WHATWG, "DOM Living Standard," Mar. 09, 2025. [Online]. Available: https://www.w3.org/TR/dom/. [Accessed: Mar. 09, 2025].

[16] WHATWG, "HTML Living Standard". Mar. 09, 2025. [Online]. Available: https://html.spec.whatwg.org/multipage/. [Accessed: Mar. 09, 2025].

[17] M. York, "Survey Finds 40 Percent of European Product Developers Do Not Build Accessibility into Their Design Plans," AT Today, May 16, 2023. [Online]. Available: https://attoday.co.uk/survey-finds-40-percent-of-european-product-developers-do-not-build-accessibility-into-their-design-plans/. [Accessed: Mar. 09, 2025].

[18] A11y Collective, "Web Accessibility Training for Developers: Why It Matters," a11y-collective.com, [Online]. Available: https://www.a11y-collective.com/information/web-accessibility-training-for-developers-why-it-matters. [Accessed: Apr. 09, 2025].

[19] G. A. Giannoumis, "Implementing Web Accessibility Policy", Case Studies of the United Kingdom, Norway, United States, Ph.D. dissertation, Dept. of Information Science and Media Studies, Univ. of Bergen, Bergen, Norway, 2019. [Online]. Available: https://bora.uib.no/bora-xmlui/handle/1956/21054. [Accessed: Apr. 09, 2025].

[20] S. Werren, H. Grieder, and C. Scherb, "Towards an Inclusive Digital Society: Digital Accessibility Framework for Visually Impaired Citizens in Swiss Public Administration," arXiv preprint arXiv:2503.09824, Mar. 2025. doi: 10.48550/arXiv.2503.09824. [Online]. Available: https://arxiv.org/html/2503.09824v1. [Accessed: Apr. 09, 2025].

[21] M. Halpin, "Making the Business Case for Accessibility," Recite Me, 2024, [Online]. Available: https://reciteme.com/news/business-case-for-accessibility/. [Accessed: Apr. 09, 2025].

[22] E. Serrano Mascaraque, A. Moratilla Ocaña, and I. Olmeda Martos, "Directrices técnicas referidas a la accesibilidad web," *Rev. Esp. Doc. Cient.*, vol. 37, no. 2, pp. 167–183, 2014. [Online]. Available: https://www.redalyc.org/articulo.oa?id=63511932014. [Accessed: Apr. 10, 2025].

[23] D. R. González, M. Á. Macías, and A. Holgado, "Análisis de accesibilidad web y diseño web accesible para instituciones socias del proyecto ESVI-AL," in *Actas del Congreso Internacional ATICA*, pp. 55–62, 2012. [Online]. Available: http://www.esvial.org/wp-content/files/Atica2012_pp55-62.pdf. [Accessed: Apr. 11, 2025].

[24] Y. Yesilada, M. Exton, D. Sloan, and S. Harper, "Understanding Web Accessibility and Its Drivers," in *Proc. Int. Cross-Disciplinary Conf. Web Accessibility (W4A '12)*, Lyon, France, Apr. 2012, Art. no. 5, pp. 1-4. doi: 10.1145/2207016.2207021. [Online]. Available: https://www.ambuehler.ethz.ch/CDstore/www2012/W4A/01%20-%20Technical%20Papers/yesilada.pdf. [Accessed: Apr. 11, 2025].

[25] M. Campoverde Molina, "La accesibilidad web: un reto en el entorno educativo ecuatoriano," *Rev. Cient. Tecnol. UPSE*, vol. 3, no. 3, pp. 86–92, 2016. [Online]. Available: https://repositorio.upse.edu.ec/bitstream/46000/8323/1/UPSE-RCT-2016-Vol.3-No.3-010.pdf. [Accessed: Apr. 11, 2025].

[26] E. Fernández-Díaz, M. C. Jambrino Maldonado, and P. P. Iglesias Sánchez, "Web Accessibility: The New Era of WCAG 2.1, the Transition to Future WCAG 3.0," *Rev. Estud. sobre Cienc. Tecnol. Inf. (Ge-ConTIC)*, vol. 10, no. 2, pp. 38–50, 2022. [Online]. Available: https://upo.es/revistas/index.php/gecontec/article/view/4069/3385. [Accessed: Apr. 11, 2025].

[27] E. Álvarez and C. M. Salinas, "Bibliotecas nacionales y accesibilidad web: situación en América Latina," *Rev. Esp. Doc. Cient.*, vol. 43, no. 2, e282, 2020. [Online]. Available: https://redc.revistas.csic.es/index.php/redc/article/view/822/1025. [Accessed: Apr. 11, 2025].

[28] C. E. Cernadas Ramos y G. I. Moyano López, "Accesibilidad web centrada en revisiones manuales: Estudio de un EVA de formación docente continua," Trabajo Final de Especialización, Facultad de Informática, Univ. Nac. de La Plata, La Plata, Argentina, 2023. doi: 10.24215/18509959.36.e5. [Online]. Available: https://sedici.unlp.edu.ar/handle/10915/162345. [Accessed: Apr. 11, 2025].

[29] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2014. doi:10.2753/MIS0742-1222240302. [Online]. Available: https://doi.org/10.2753/MIS0742-1222240302. [Accessed: Apr. 11, 2025].

[30] WCAG 2.2 Understanding, "Understanding Conformance," W3C, Apr. 01, 2025. [Online]. Available: https://www.w3.org/WAI/WCAG22/Understanding/conformance. [Accessed: Apr. 11, 2025].

[31] J. Waltner, "Digital Accessibility in 2024: Some Progress, Some Plateaus," applause.com, May 08, 2024. [Online]. Available: https://www.applause.com/blog/digital-accessibility-2024-survey-results/. [Accessed: May 09, 2025].

[32] L. Pinedo, M. A. Valles Coral, and R. Injante, "Estado actual de la accesibilidad web en Latinoamérica: Una revisión exploratoria de las evaluaciones y herramientas utilizadas," ResearchGate, *Rev. Esp. Doc. Cient.*, vol. 47, no. 1, e378, 2024. doi: 10.3989/redc.2024.1.1464 [Online]. Available: https://www.researchgate.net/publication/378655104. [Accessed: May 09, 2025].

[33] A. Ahmed, M. Fresco, F. Forsberg, and H. Grotli, "From Code to Compliance: Assessing ChatGPT's Utility in Designing an Accessible Webpage—A Case Study," arXiv preprint arXiv:2501.03572, 2025. doi: 10.48550/arXiv.2501.03572. [Online]. Available: https://www.researchgate.net/publication/387797297_From_Code_to_Compliance_Assessing_ChatGPT's_Utility_in_Designing_an_Accessible_Webpage_--_A_Case_Study. [Accessed: May 09, 2025].

[34] E. Calì, T. Fulcini, R. Coppola, L. Laudadio, and M. Torchiano, "A Prototype VS Code Extension to Improve Web Accessible Development," arXiv preprint arXiv:2503.09673, Mar. 2025. doi: 10.48550/arXiv.2503.09673. [Online]. Available:https://www.researchgate.net/publication/389821742_A_Prototype_VS_Code_Extension_to_Improve_Web_Accessible_Development. [Accessed: May 09, 2025].

[35] Deque Systems, “axe Accessibility Linter,” Visual Studio Marketplace, 2025. [Online]. Available: https://marketplace.visualstudio.com/items?itemName=deque-systems.vscode-axe-linter. [Accessed: May 14, 2025].

[36] Microsoft Corporation, “Visual Studio Code,” Visual Studio Code Official Website, *2025*. [Online]. Available: https://code.visualstudio.com/. [Accessed: May 14, 2025].

[37] OpenAI, “ChatGPT,” 2025. Available: https://chat.openai.com/. [Accessed: May 14, 2025].

[38] WebAIM, “WAVE Web Accessibility Evaluation Tool,” WebAIM.org. [Online]. Available: https://wave.webaim.org/. [Accessed: May 14, 2025].

[39] Google, “Lighthouse overview,” Chrome for Developers. [Online]. Available: https://developer.chrome.com/docs/lighthouse/. [Accessed: May 14, 2025].

[40] Deque Systems, “axe DevTools: Web Accessibility Testing,” Deque Systems. [Online]. Available: https://www.deque.com/axe/devtools/. [Accessed: May 14, 2025].

[41] International Organization for Standardization, *ISO/IEC 40500:2012 – Information technology — W3C Web Content Accessibility Guidelines (WCAG) 2.0*. Geneva, CH: ISO, Oct. 2012. [Online]. Available: https://www.iso.org/standard/58625.html. [Accessed: May 14, 2025].

[42] European Telecommunications Standards Institute (ETSI), *EN 301 549 V3.2.1 (2021-03) – Accessibility requirements for ICT products and services*. Sophia Antipolis, FR: ETSI, Mar. 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf. [Accessed: May 14, 2025].

[43] U.S. General Services Administration, “Section508.gov,” Section508.gov. [Online]. Available: https://www.section508.gov/. [Accessed: May 14, 2025].

[44] Google, “Gemini,” 2025. Available: https://gemini.google.com/. [Accessed: May 14, 2025].

[45] Accessibility Insights, “Accessibility Insights for Web Overview,” Accessibility Insights. Available: https://accessibilityinsights.io/docs/web/overview/. [Accessed: May 14, 2025].