UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Grado en Ingeniería de la Salud
Mención en Ingeniería Biomédica

Aplicación de algoritmos de inteligencia artificial para la identificación y solución de problemas de accesibilidad en interfaces de usuario

Application of artificial intelligence algorithms for identifying and solving accessibility issues in user interfaces

Realizado por
Carla Fernández Navarro

Tutorizado por
José Francisco Chicano García

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, junio 2025

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADA EN INGENIERÍA DE LA SALUD

Aplicación de algoritmos de inteligencia artificial para la identificación y solución de problemas de accesibilidad en interfaces de usuario

Application of artificial intelligence algorithms for identifying and solving accessibility issues in user interfaces

Realizado por
Carla Fernández Navarro

Tutorizado por
José Francisco Chicano García

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2025

Fecha defensa: julio de 2025

A mis padres, por haberme permitido estudiar la mejor carrera del mundo fuera de casa sin ningún tipo de preocupación. Sin vosotros no hubiera sido posible.

A mi hermana, por ser fuente inagotable de inspiración desde el día en que la conocí. Te debo todo.

A mis amigos Nerea, Alex, Raúl, Miguel Ángel, Paloma y Claudia, por estar de principio a fin, y confiar en mí cuando nadie más lo hacía.

A Elena, por ser mi luz cuando todas las demás se apagan. Sin ti, yo habría abandonado hace tiempo. Gracias.

Y a Francis, mi tutor, por su paciencia y dedicación. Por transmitir seguridad y confianza. Gracias por acompañarme en este largo camino.

Resumen

Este Trabajo de Fin de Grado muestra el diseño y desarrollo de una herramienta automática cuyo objetivo es evaluar y optimizar la accesibilidad web, fundamentada en la normativa WCAG 2.2 nivel A. La accesibilidad web es crucial para asegurar la inclusión de las personas con discapacidad en el ámbito web, y este proyecto tiene como meta facilitar su cumplimiento a través de técnicas de automatización e inteligencia artificial.

La herramienta está desarrollada en Python, utilizando librerías como Selenium, Axe-core, BeautifulSoup y la API de OpenAI, e implementa un proceso modular que posibilita examinar una página web, identificar sus problemas de accesibilidad, aplicar soluciones automáticas y generar informes comparativos. Entre sus tareas se incluyen la descripción automática de imágenes sin texto alternativo, la identificación de fallos en la estructura y la creación del HTML corregido listo para su validación posterior.

En el diseño de la aplicación se ha empleado una estructura modular en la que cada módulo desempeña una función: navegación, extracción del HTML, análisis y corrección de errores, y generación de resultados. El sistema utiliza el método RAG para proporcionarle las imágenes al modelo de lenguaje y generar las descripciones, mientras que se utilizan otras técnicas para mejorar las respuestas del modelo de lenguaje y elevar la calidad de las correcciones.

Finalmente, la validación de páginas web indica que la herramienta es efectiva en la identificación y corrección de errores fundamentales de accesibilidad y establece las bases para las futuras expansiones.

Palabras clave: **Accesibilidad Web, Inteligencia Artificial, WCAG 2.2, Modelos de Lenguaje, Interfaces de Usuario**

Abstract

This Final Degree Project shows the design and development of an automatic tool whose objective is to evaluate and optimize web accessibility, based on the WCAG 2.2 level A standard. Web accessibility is crucial to ensure the inclusion of people with disabilities on the web, and this project aims to facilitate its compliance through automation and artificial intelligence techniques.

The tool is developed in Python, using libraries such as Selenium, Axe-core, Beautiful Soup and the OpenAI API, and implements a modular process that makes it possible to examine a web page, identify its accessibility problems, apply automatic solutions and generate comparative reports. Tasks include automatically describing images without alternative text, identifying flaws in the structure and creating corrected HTML ready for further validation.

A modular structure has been used in the design of the application, with each module playing a role: navigation, HTML extraction, parsing and error correction, and output generation. The system uses the RAG method to provide the images to the language model and to generate the descriptions, while other techniques are used to improve the responses of the language model and to improve the quality of the corrections.

Finally, the validation of web pages indicates that the tool is effective in identifying and correcting fundamental accessibility errors and lays the groundwork for future expansions.

Keywords: Web Accessibility, Artificial Intelligence, WCAG 2.2, Language Models, User Interfaces

Índice

Resumen I
Abstract II
Índice III
Índice de tablas V
Índice de ilustraciones V
1. Introducción 1
1.1 Motivación 1
1.2 Objetivos 2
1.3. Metodología 3
1.3.1. Fases de trabajo 3
1.4. Estructura del documento 5
2. Fundamentos 7
2.1. Discapacidad y accesibilidad digital 8
2.1.1. Concepto de discapacidad 8
2.1.2. Tipos de discapacidad 9
2.1.3. Discapacidad en el entorno web 10
2.2. Normativa WCAG 2.2 10
2.2.1. Introducción a WCAG 10
2.2.2. Pautas de accesibilidad 11
2.2.3. Niveles de conformidad 12
2.3. Inteligencia artificial en la ingeniería del software 13
2.3.1. Concepto de inteligencia artificial 13
2.3.2. Procesamiento de lenguaje natural (NLP) 13
2.3.3. Modelos de lenguaje de gran tamaño (LLMs) 14
2.3.4. Retrieval-Augmented Generation (RAG) 15
2.4. Herramientas y estudios existentes 16
2.4.1. Herramientas 16
2.4.2. Estudios del estado del arte 17
2.4.3. Limitaciones 19
2.4.4. Aportaciones al avance de la accesibilidad web mediante IA 20
3. Recursos aplicados en el proyecto 21
3.1. Introducción al catálogo 21
3.2. Análisis del catálogo 22
3.3. Herramientas utilizadas 23
3.4. Sitios web utilizados para la evaluación 24
4. Diseño de la herramienta 27
4.1. Arquitectura de la herramienta 27
4.1.1. Descripción de la arquitectura modular 27
4.1.2. Manejo de errores robusto 28
4.1.3. Flujo de trabajo del sistema 28

4.2. Componentes principales 30
4.2.1. `Main.py` 30
4.2.2. Carpeta `core` 31
4.2.3. Carpeta `utils` 31
4.2.4. Carpeta `config` 31
4.2.5. Carpeta `templates` 31
4.3. Integración de tecnologías externas 32
4.4. Archivos generados 33
4.5. Personalización 34
5. Implementación de la herramienta 37
5.1. Fases del desarrollo de la herramienta 37
6. Validación de la herramienta 49
6.1. Casos de prueba 50
6.1.1. Hospital Clínico de Aragón 50
6.1.2. Hospital VITHAS de Granada 52
6.1.3. Hospital Puerta del Mar de Cádiz 53
6.1.4. Hospital Del Mar de Cataluña 55
6.1.5. Clínica Dental Your Clinic de Málaga 56
6.1.6. Fundación del Corazón 57
6.1.7. Servicio de Salud de Castilla La Mancha 59
6.1.8. Área de Gestión Sanitaria de Serranía 60
6.2. Resumen final de resultados 62
6.3. Análisis de resultados 63
7. Conclusiones y líneas futuras 65
7.1. Conclusiones generales 65
7.2. Limitaciones de la herramienta 66
7.3. Aportes del proyecto 66
7.4. Líneas futuras 67
Referencias 69
Anexo A. Manual del Usuario 73
1. Información para usuarios 73
2. Requisitos del sistema 74
2.1. Hardware mínimo recomendado 74
2.2. Software mínimo requerido 74
3. Instalación paso a paso 74
3.1. Clonar el repositorio y crear un entorno virtual 74
3.2. Configuración de OpenAI 75
3.3 Ejecución 75
4. Proceso de análisis 75
5. Archivos generados 76
6. Apertura de servidor local 76
7. Finalización de la ejecución 76
8. Actualización del sistema 77
9. Depuración y mantenimiento 77
Anexo B. Mapeo de errores de accesibilidad y soluciones basadas en IA 79

Índice de tablas

Tabla 1: Herramientas para accesibilidad web 17
Tabla 2: Elección de páginas webs 25
Tabla 3: Tabla comparativa de resultados 62
Tabla 4: Depuración 77
Tabla 5: Catálogo WCAG 2.2 nivel A 80

Índice de ilustraciones

Ilustración 1: Diagrama de clases de la herramienta 32
Ilustración 2: Fragmento del HTML original de una página web 39
Ilustración 3: Mensaje en la consola sobre el comienzo del análisis con Axe-core 40
Ilustración 4: Inicio de la descarga de imágenes y generación de descripciones 42
Ilustración 5: Comparación entre error encontrado en la página original (izquierda) y la corrección en la página web accesible (derecha) 46
Ilustración 6: Detalle de los errores antes de la aplicación de la herramienta en la web del Hospital Clínico de Aragón 50
Ilustración 7: Detalle de los errores que persisten después de la aplicación de la herramienta en la web del Hospital Clínico de Aragón 50
Ilustración 8: Informe generado por la herramienta para la web del Hospital Clínico de Aragón 51
Ilustración 9: Imagen sin descripción textual de la página web del Hospital Clínico de Aragón 51
Ilustración 10: Imagen con texto descriptivo después de aplicar la herramienta en la web del Hospital Clínico de Aragón 52
Ilustración 11: Detalle de los errores antes de la aplicación de la herramienta en la web del Hospital Vithas de Granada 52
Ilustración 12: Detalle del análisis posterior a la aplicación de la herramienta en la web del Hospital Vithas de Granada 53
Ilustración 13: Informe generado por la herramienta para la web del Hospital Vithas de Granada 53
Ilustración 14: Detalle de los errores antes de la aplicación de la herramienta en la web del Hospital Puerta del Mar de Cádiz 54
Ilustración 15: Detalle de los errores persistentes después de la aplicación de la herramienta en la web del Hospital Puerta del Mar de Cádiz 54
Ilustración 16: Informe generado por la herramienta para la web del Hospital Puerta del Mar 54
Ilustración 17: Detalle de los errores antes de la aplicación de la herramienta en la web del Hospital del Mar de Cataluña 55

Ilustración 18: Detalle de los errores persistentes después de la aplicación de la herramienta en la web del Hospital del Mar de Cataluña 55
Ilustración 19: Informe generado por la herramienta para la web del Hospital del Mar de Cataluña..... 56
Ilustración 20: Detalle de los errores antes de la aplicación de la herramienta en la web de la Clínica Dental Your Clinic 56
Ilustración 21: Detalle de los errores persistentes después de la aplicación de la herramienta en la web de la Clínica Dental Your Clinic 57
Ilustración 22: Informe generado por la herramienta para la web de la Clínica Dental Your Clinic 57
Ilustración 23: Detalle de los errores antes de la aplicación de la herramienta en la web de la Fundación del Corazón 58
Ilustración 24: Detalle de los errores persistentes de la aplicación de la herramienta en la web de la Fundación del Corazón 58
Ilustración 25: Informe generado por la herramienta para la web de la Fundación del Corazón..... 58
Ilustración 26: Encabezado de la web de la Fundación del Corazón antes de aplicar la herramienta..... 59
Ilustración 27: Encabezado de la web de la Fundación del Corazón después de aplicar la herramienta..... 59
Ilustración 28: Detalle de los errores antes de la aplicación de la herramienta en la web principal sobre la sanidad de Castilla La Mancha 59
Ilustración 29: Detalle de los errores persistentes después de la aplicación de la herramienta en la web principal sobre la sanidad de Castilla La Mancha..... 60
Ilustración 30: Informe generado por la herramienta para la web principal sobre sanidad de Castilla La Mancha..... 60
Ilustración 31: Detalle de los errores antes de la aplicación de la herramienta en la web del Área de Gestión Sanitaria de Serranía 61
Ilustración 32: Detalle de los errores persistentes después de la aplicación de la herramienta en la web del Área de Gestión Sanitaria de Serranía..... 61
Ilustración 33: Informe generado por la herramienta para la web del Área de Gestión Sanitaria de Serranía 61

1

Introducción

1.1 Motivación

En una era cada vez más digital, la accesibilidad en las interfaces de usuario ha trascendido para consolidarse como un pilar fundamental de la inclusión social y la igualdad en la obtención de información y los servicios web. Un entorno digital completamente inclusivo es aquel que permite a todas las personas interactuar de manera efectiva, autónoma y satisfactoria con el mundo digital, independientemente de sus capacidad físicas o cognitivas. Sin embargo, la realidad revela una amplia brecha entre este ideal y la experiencia cotidiana de muchos usuarios, quienes enfrentan significativas barreras al navegar por la web y utilizar aplicaciones.

Mi interés en la inteligencia artificial se basa en su habilidad intrínseca para examinar grandes volúmenes de datos, reconocer patrones complejos y automatizar actividades que, de otro modo, requerirían un esfuerzo humano significativo. Esta potencia analítica y automatizadora me ha llevado a explorar su aplicación en un dominio de gran importancia social: la accesibilidad web. La posibilidad de emplear algoritmos para detectar de forma exhaustiva los diversos problemas que dificultan la accesibilidad y proponer soluciones adaptadas a cada contexto específico representa una vía prometedora para avanzar hacia un ecosistema digital más justo e inclusivo.

Este Trabajo de Fin de Grado busca aportar algo nuevo en un campo que está en pleno crecimiento. La idea es explorar cómo se pueden usar herramientas de inteligencia artificial, procesamiento de lenguaje natural y modelos de lenguaje visual para hacer que las páginas web sean más accesibles para todos. En lugar de centrarse en un solo

sitio web, el proyecto analiza un grupo representativo de páginas web reales de instituciones sanitarias que muestran diferentes niveles de accesibilidad. Esto ayuda a probar los algoritmos en distintos contextos y a entender mejor los retos a los que nos enfrentamos hoy en día.

Para lograr el anterior objetivo, crearemos una herramienta automática que combine grandes modelos de lenguaje (LLMs), evaluadores automáticos tipo Axe-core, modelos de lenguaje visual y normas de accesibilidad como WCAG 2.2, nivel A. Esta herramienta no solo detectará errores de accesibilidad con buena precisión, sino que también entenderá el contexto, ofrecerá correcciones específicas para mejorar y ayudará a generar informes claros y estructurados con los hallazgos. Para desarrollarla, usaremos Python, trabajando en PyCharm y combinando varias bibliotecas especializadas: unas para analizar el contenido de las páginas, otras para generar descripciones, también Selenium para pruebas automáticas y diferentes modelos de IA. Gracias a este enfoque modular, podremos experimentar con distintos tipos de sitios web, medir qué tan efectivas son las técnicas, y seguir mejorando el sistema hasta conseguir un prototipo que funcione adecuadamente, sea sólido y pueda usarse en situaciones reales. Al final, este proyecto quiere demostrar cómo la inteligencia artificial puede servir de gran ayuda para reducir las barreras digitales y contribuir a que la web sea un espacio más justo, inclusivo y que tenga en cuenta las necesidades de todas las personas.

1.2 Objetivos

El objetivo principal del Trabajo de Fin de Grado es crear una herramienta software basada en grandes modelos de lenguaje (LLMs) para valorar la accesibilidad en las interfaces de usuario, utilizando como base la normativa WCAG 2.2 y resolver los problemas encontrados.

Para conseguirlo, es necesario cumplir los siguientes objetivos específicos:

- **OE1:** desarrollar una herramienta software capaz de examinar el código de una interfaz de usuario e identificar el incumplimiento de los criterios propuestos en las guías del estándar WCAG.
- **OE2:** utilizar modelos con entrada multimodal para crear descripciones textuales de elementos no visuales (como imágenes, vídeos o audios).

- **OE3:** utilizar LLMs para crear descripciones de elementos HTML sin representación textual alternativa.
- **OE4:** combinar las herramientas anteriores para crear una nueva herramienta capaz de analizar y resolver problemas en las 13 guías de accesibilidad del estándar WCAG 2.2.
- **OE5:** recopilar un conjunto de interfaces web de usuario con problemas de accesibilidad para evaluar nuestra herramienta.
- **OE6:** validar la herramienta comparando el nivel de cumplimiento de accesibilidad antes y después de aplicarla sobre el conjunto de interfaces mencionado en el **OE5**.

1.3. Metodología

Para desarrollar el Trabajo de Fin de Grado se empleará una metodología SCRUM ajustada al contexto académico, dividiendo el proyecto en sprints de dos semanas que facilitan el avance progresivo de la herramienta y la implementación de mejoras constantes basadas en los resultados alcanzados después de cada sprint.

La elección de SCRUM se justifica por su método iterativo, lo que lo hace particularmente apropiado para proyectos de desarrollo de software en los que los requerimientos pueden variar durante el proceso.

Cada sprint de dos semanas concluye con una reunión con el tutor, en la que se revisa el trabajo realizado y se recibe una retroalimentación anticipada para modificar el desarrollo según las necesidades identificadas.

1.3.1. Fases de trabajo

El proyecto consta de seis fases, con una duración total estimada de 296 horas de trabajo:

1. **Examinación bibliográfica y reglamentaria de la normativa WCAG:** en esta fase inicial se establecerán las bases teóricas del proyecto mediante un análisis de la normativa de accesibilidad web, enfocándose en WCAG 2.2 nivel A. Se revisarán las herramientas existentes y sus limitaciones, así como se estudiarán las

técnicas de implementación y sus errores más comunes, con el fin de apoyar el futuro desarrollo de los algoritmos de detección y corrección.

2. **Recopilación de interfaces web de usuario para realizar las pruebas de validación de la herramienta:** se seleccionarán páginas web del ámbito sanitario institucional y obtener un conjunto de prueba representativo. Se aplicarán criterios que asegurarán la diversidad tecnológica y estructural, incluyendo hospitales, clínicas privadas y sitios web informativos.
3. **Análisis de la accesibilidad de sitios web utilizando herramientas especializadas:** se realizará el análisis sistemático de la accesibilidad de las páginas web recopiladas mediante herramientas automáticas existentes, evaluando sus capacidades, limitaciones y metodologías. Esto permitirá identificar errores comunes, estableciendo una base sólida para comparar posteriormente la efectivad de la herramienta desarrollada.
4. **Diseño y desarrollo de la herramienta:** esta fase es la central del proyecto. Se basa en la utilización del análisis estático, una técnica para examinar el código fuente de la página web sin interactuar dinámicamente con ella, para la detección y corrección de errores de accesibilidad. Comenzará con la elección de OpenAI como modelo de lenguaje debido a su gran capacidad contextual y su precisión. Se diseñarán prompts específicos para los criterios del nivel A de WCAG 2.2. Además, se organizará en módulos: automatización web, descarga de imágenes de la página web, detección de errores mediante análisis estático, integración modular con la API de OpenAI para generar correcciones y analizar las imágenes para generar sus descripciones y un sistema de validación automática para comprobar la efectividad de las soluciones sin añadir errores nuevos.
5. **Validación de la herramienta:** se aplicará la herramienta a cada página web seleccionada, evaluando tanto la capacidad de detección como las correcciones realizadas. Tras ello, se analizará de nuevo la accesibilidad y se hará una comparación con los resultados obtenidos antes de aplicarla.
6. **Redacción de la memoria:** esta fase final consistirá en la documentación completa del proyecto, incluyendo la redacción de la memoria técnica y los anexos requeridos, como los manuales de desarrollador y usuario. Se detallarán

las decisiones técnicas tomadas, los resultados obtenidos en cada fase y las conclusiones del trabajo realizado. La redacción se organizará siguiendo la estructura académica establecida y garantizará que cada apartado proporcione la información necesaria para la comprensión y evaluación del proyecto.

1.4. Estructura del documento

Este Trabajo de Fin de Grado está dividido en seis capítulos:

- Capítulo 1: Introducción. Una descripción de la motivación que me ha impulsado a realizar este trabajo y los objetivos a cumplir, así como los objetivos generales y específicos que se esperan cumplir en este proyecto.
- Capítulo 2: Fundamentos. Se analiza el marco teórico, incluyendo las pautas de accesibilidad WCAG 2.2 y la revisión de las herramientas existentes.
- Capítulo 3: Recursos aplicados en el proyecto. Se describen los recursos utilizados, como el catálogo sobre la normativa WCAG 2.2 y las diferentes herramientas que cubren las necesidades, el conjunto de páginas web que utilizaremos de prueba y las tecnologías utilizadas.
- Capítulo 4: Diseño de la herramienta. Se explica la estructura modular de la herramienta, el rol del modelo de lenguaje y la lógica de interacción entre los componentes.
- Capítulo 5: Implementación de la herramienta. Se describe el proceso de aplicación de la herramienta en las diferentes páginas web, desde la automatización web hasta el desarrollo de los algoritmos de detección y corrección de errores.
- Capítulo 6: Validación de la herramienta. Se presentan los resultados de la validación, comparando los errores obtenidos antes y después de su aplicación y evaluando la efectividad en el conjunto de páginas web sanitarias seleccionadas.
- Capítulo 7: Conclusiones y líneas futuras. Se resumen los logros del proyecto, se describen las limitaciones encontradas y se proponen posibles mejoras.
- Anexo A: Manual del usuario. Incluye la documentación del funcionamiento de la herramienta, la instalación y configuración de la herramienta y su ejecución.

Dirigido a los usuarios que estén interesados en entender la estructura del sistema, cómo interactúan los componentes y las tecnologías empleadas para su implementación

- Anexo B: Mapeo de errores de accesibilidad y soluciones basadas en IA. Se presenta un catálogo que relaciona los criterios de accesibilidad establecidos en el nivel A de las pautas WCAG 2.2 con soluciones concretas obtenidas a partir de artículos científicos y aplicaciones existentes.

2

Fundamentos

En este capítulo se presentan los conceptos fundamentales y el estado del arte que sostienen el trabajo desarrollado en este proyecto. Para ello, debemos introducir el concepto de discapacidad así como los diferentes tipos que existen, la discapacidad en el entorno web y la importancia de garantizar la accesibilidad web para todos los usuarios, independientemente de poseer diferentes capacidades. A continuación, se expone la normativa internacional que regula esta materia, las Pautas de Accesibilidad para el Contenido Web (WCAG, *Web Content Accessibility Guidelines*), describiendo su estructura, niveles y principios orientadores.

Posteriormente, se explica la función de la inteligencia artificial como instrumento para detectar y resolver los problemas de accesibilidad que aparecen en las interfaces de usuario. Se revisan las principales ramas de la inteligencia artificial involucradas en este contexto, como el procesamiento de lenguaje natural (NLP) y los modelos de lenguaje de gran tamaño (LLMs), destacando así su potencial en la automatización de tareas complejas en la ingeniería del software.

Para finalizar, se ofrece un análisis crítico del estado actual de la investigación en este área, revisando herramientas y otros trabajos que ya aplican o han aplicado inteligencia artificial a la accesibilidad web. Esto permite ver qué avances ha habido hasta ahora, qué limitaciones existen todavía y por qué este proyecto puede aportar innovaciones.

2.1. Discapacidad y accesibilidad digital

2.1.1. Concepto de discapacidad

El término "discapacidad" ha evolucionado paulatinamente en la historia. Al principio, se asociaba a un castigo divino, generando un aislamiento a las personas que la poseían. Más tarde, en torno al siglo XV, empezaron a aparecer los denominados manicomios, ocasionando un punto de vista sesgado y estigmatizante.

En el siglo XX se comienzan a crear los centros de educación especial, aunque desde una perspectiva paternalista, atenuando así la discriminación tanto social como laboral, lo que da lugar a la creación de asociaciones formadas por personas con discapacidad y sus familiares que se unen para defender sus derechos como humanos. De este modo, las personas con discapacidad empiezan a empoderarse, pero no es hasta 1982 cuando se produce el punto de inflexión. En España se aprueba la LISMI (Ley de Integración Social del Minusválido), actualmente conocida como Ley General de la Discapacidad. En este documento, se reconocen los derechos de las personas con discapacidad y se establece la obligación de incluir un porcentaje superior al 2% de dichas personas en empresas de más de 50 trabajadores.

Sin embargo, tras todo esto, no es hasta los años 2000 que se empieza a dejar de lado la perspectiva paternalista previamente mencionada y se avanza hacia el enfoque de que la persona que posee capacidades diferentes cuenta con habilidades, recursos y competencias si se les brinda el apoyo necesario [14].

Es por esto por lo que la Organización Mundial de la Salud (OMS) establece una definición de "discapacidad" que representa un cambio significativo: cualquier limitación o dificultad para llevar a cabo una actividad de la manera que se considera normal para el ser humano [25].

Entonces, la discapacidad es un fenómeno complejo que no se limita a considerar al individuo de manera aislada, sino que se entiende en relación con la interacción que mantiene con la sociedad en la que se encuentra. Así, el contexto social se convierte en un elemento clave para comprender la discapacidad de una persona.

2.1.2. Tipos de discapacidad

Existen cuatro categorías principales de discapacidad: física, sensorial, intelectual y psicosocial. Esta división está estructurada, según la OMS, en función del área corporal afectada y reconoce que la discapacidad no se define solamente por la salud de los individuos, sino también por factores sociales y ambientales [15]. A continuación, se explican las cuatro categorías principales [11]:

- **Discapacidad física:** también conocida como discapacidad motora, abarca las restricciones en la movilidad. Puede ser consecuencia de una afección de salud o trastorno que perjudique al sistema músculo esquelético y neuromuscular del individuo. Dentro de esta categoría encontramos dos subtipos:
    - Funcional: implica dificultades en el funcionamiento motor, como caminar, moverse o manipular objetos.
    - Orgánica: puede no ser visible al afectar a los órganos internos. Por ejemplo, las enfermedades respiratorias que limitan la capacidad física.
- **Discapacidad sensorial:** supone la existencia de restricciones en alguno de los sentidos, modificando la percepción de los estímulos del entorno.
    - Discapacidad visual: conlleva desde la pérdida parcial de la visión hasta la ceguera total.
    - Discapacidad auditiva: conlleva desde la disminución en la capacidad para percibir sonidos hasta la pérdida total de la audición, conocida como anacusia o sordera profunda. También puede afectar a la comunicación verbal.
    - Discapacidad olfativa: pérdida de la capacidad olfativa, lo que afecta la percepción del entorno y puede poner en riesgo la seguridad de quienes lo padecen.
    - Discapacidad gustativa: se refiere a la alteración o pérdida del sentido del gusto, lo que puede derivar en complicaciones de tipo nutricional.
- **Discapacidad intelectual:** también se conoce como discapacidad cognitiva. Implica una alteración en las capacidades intelectuales, como la memoria o la atención. Generalmente se suele manifestar antes de los 18 años y tiene como consecuencia dificultades en el aprendizaje, retraso en el desarrollo del lenguaje

o motricidad así como problemas relacionados con las habilidades sociales y comunicativas.

- **Discapacidad psicosocial:** aborda problemas de salud mental que afectan al equilibrio emocional y bienestar social de las personas. Implica la dificultad en el funcionamiento mental y emocional, afectando a su habilidad para gestionar las emociones o relaciones sociales. Un ejemplo de esto es la esquizofrenia.

2.1.3. Discapacidad en el entorno web

La discapacidad en el entorno web se refiere a las barreras que encuentran las personas con limitaciones visuales, auditivas, motoras o cognitivas a la hora de interactuar con páginas webs, aplicaciones o servicios digitales. El mundo digital está cada día más avanzado, pero sigue habiendo muchos problemas relacionados con el diseño, la estructura del contenido, la navegación web o la falta de adaptaciones tecnológicas.

Muchos de estos problemas surgen cuando no se sigue la normativa reguladora WCAG, que se explicará a continuación. Esto puede implicar que, por ejemplo, una persona ciega no pueda utilizar correctamente un lector de pantalla por falta de descripciones textuales o que una persona con discapacidad motora no pueda navegar por internet si no puede manejar el teclado.

Por tanto, garantizar la accesibilidad web permite que todos los usuarios puedan ejercer su derecho a la información, comunicación y ocio, además de mejorar la experiencia para todos, promoviendo un internet más inclusivo y equitativo.

2.2. Normativa WCAG 2.2

2.2.1. Introducción a WCAG

Las *Web Content Accessibility Guidelines* (WCAG) son las directrices de diseño y desarrollo de una página o aplicación web, con el objetivo de mejorar la accesibilidad de los sitios web y las aplicaciones, incluyendo aquellas con algún tipo de discapacidad. Establecen una serie de normas que los desarrolladores deben encargarse de cumplir para asegurarse de que el contenido de los sitios web que crean o de las aplicaciones sea accesible para todo el mundo, facilitando la inclusión digital [28].

Es importante recalcar que no solo favorecen a los usuarios con limitaciones, sino que también optimizan la experiencia para todos, incluyendo a quienes utilizan dispositivos móviles, su conexión a internet es lenta o utilizan tecnología asistencial.

Se desarrollan a través del W3C en colaboración con individuos y organizaciones de todo el mundo, proporcionado un estándar singular y común, cumpliendo así con las necesidades de todos a escala global [30]. Aclaran cómo hacer que el contenido web sea más accesible, entendiendo por contenido a la información que podemos encontrar en una página web o aplicación, como textos, imágenes y sonidos o código que define la estructura o presentación.

2.2.2. Pautas de accesibilidad

WCAG 2.2 se compone de 4 principios fundamentales, que ya existían en WCAG 2.0 y WCAG 2.1, 13 pautas y 86 criterios de conformidad, clasificaciones en diferentes niveles [32].

Los principios son:

- **Perceptibilidad:** la información y componentes de la interfaz de usuario deben mostrarse de forma que los usuarios puedan percibirlos claramente y comprender su propósito. Esto se refiere a las pautas que rigen el uso de texto alternativo, contenido multimedia que proporcione alternativas dependientes del tiempo, que se pueda adaptar a diferentes formas y no se pierda ni información ni escritura y que permita a los usuarios ver y escuchar el contenido, diferenciando entre lo más y lo menos importante.
- **Operabilidad:** los componentes de la interfaz y navegación deben ser manejables. Por ejemplo, el teclado debe ser accesible. Es decir, que todo se pueda controlar desde él. También debe proporcionar el tiempo adecuado para poder leer, entender y utilizar el contenido, evitando así el diseño de contenidos que puedan causar ataques epilépticos. Permite a los usuarios utilizar la funcionalidad mediante diversos métodos de entrada, no solo desde el teclado, y proponer formas para asistir a los usuarios en la navegación, búsqueda de contenido y determinar su ubicación actual.

- **Comprensibilidad:** toda la información y las operaciones deben ser comprensibles, utilizando texto legible, diseño uniforme y una forma de navegar en las páginas webs previsibles, además de ofrecer asistencia en la entrada de datos.
- **Robustez:** el contenido debe ser sólido para que pueda ser comprendido por una amplia gama de agentes de usuario, incluyendo las tecnologías de apoyo.

2.2.3. Niveles de conformidad

Una vez que se establecen los estándares internacionales, los criterios de accesibilidad se clasifican en tres niveles de conformidad: A, AA y AAA [31]. Entender la diferencia entre estos niveles es crucial a la hora de desarrollar webs o aplicaciones.

- **Nivel A:** es el nivel más básico de accesibilidad. Cumplir con este nivel significa que se satisfacen los criterios mínimos que deben ser implementados para asegurar que un sitio web sea utilizable para personas con discapacidad. Este nivel incluye aspectos como proporcionar texto alternativo para imágenes no decorativas, asegurar que todos los elementos sean operables mediante teclado y que los vídeos tengan subtítulos.
- **Nivel AA:** incluye todos los criterios del nivel A y requisitos extra para tratar problemas de accesibilidad frecuentes y significativos. Este nivel establece un estándar más exigente e incorpora requisitos como mayor contraste entre el texto y el fondo, navegación más intuitiva y predecible y mayor compatibilidad con tecnologías de asistencia como los lectores de pantalla. Si se cumple este nivel, se puede garantizar que el sitio web es accesible para un espectro amplio de usuarios con capacidades diferentes, como limitaciones visuales, auditivas o cognitivas.
- **Nivel AAA:** es el grado más alto y completo de accesibilidad web. Entre sus criterios se encuentra la incorporación de interpretación en lengua de señas para vídeos, instrucciones precisas sobre la pronunciación de texto y un contraste aún más elevado entre texto y elementos visuales. Aun así, alcanzar este nivel suele ser poco práctico, por lo que no se considera un requisito obligatorio para la mayoría de los sitios web.

2.3. Inteligencia artificial en la ingeniería del software

2.3.1. Concepto de inteligencia artificial

La inteligencia artificial se describe como la capacidad de una máquina para mostrar habilidades iguales o similares a la de los humanos, tales como el razonamiento, aprendizaje o habilidad de planear [27]. Para conseguir esto, se utiliza una combinación de algoritmos, modelos matemáticos y grandes volúmenes de datos para permitir a las máquinas aprender de la experiencia, adaptarse a la información nueva y actuar con un cierto grado de autonomía.

Hay diferentes tipos de inteligencia artificial, según los expertos Stuart Russell y Peter Norvig [17]:

- **Tecnologías que razonan como humanos:** automatizan tareas como la toma de decisiones, solución de problemas y aprendizaje.
- **Aplicaciones que operan como humanos:** llevan a cabo tareas de forma similar a las personas.
- **Modelos que piensan de manera racional:** buscan imitar el pensamiento lógico y racional de las personas.
- **Agentes que actúan de manera racional:** reproducen el comportamiento humano de forma lógica.

2.3.2. Procesamiento de lenguaje natural (NLP)

El procesamiento de lenguaje natural es una especialidad de la inteligencia artificial centrada en la relación entre humanos y computadores a través del lenguaje natural [5]. Para ello, utilizan aprendizaje automático, brindando a los ordenadores la capacidad de interpretar, manipular y comprender el lenguaje humano.

Esta rama surgió en la década de los 50 debido a la necesidad de traducir textos de forma automatizada en distintos idiomas, ya que el sistema estaba fundamentado en reglas implantadas y en una recopilación manual, por lo que había limitaciones para adaptarse al lenguaje humano natural [18].

El procesamiento del lenguaje natural juega un papel crucial en el progreso y evolución de la inteligencia artificial ya que, al facilitar que las máquinas entiendan y produzcan el lenguaje humano, las vuelve más accesibles y funcionales. También consideran las complejidades del idioma, como las metáforas o sarcasmo, volviendo su utilización esencial para las aplicaciones de traducción inmediata y chatbots [18].

Algunas de las tareas del día a día y del ámbito empresarial donde puede aplicarse son:

- **Análisis empresarial:** analizando reseñas, publicaciones en redes sociales y otros datos.
- **Chatbots:** ofrecen un servicio de atención al cliente automatizado. Atiende preguntas frecuentes y dirige las más complejas a humanos.
- **Filtros de detección de spam:** las organizaciones de correo electrónico lo utilizan para detectar y clasificar los mensajes según si son deseados o no.
- **Traducción automática:** convierten texto o voz de un lenguaje a otro con exactitud.

2.3.3. Modelos de lenguaje de gran tamaño (LLMs)

Estos modelos de inteligencia artificial son entrenados para poder procesar grandes cantidades de texto, permitiéndole así realizar tareas complejas de procesamiento de lenguaje natural. Se fundamentan en arquitecturas avanzadas de redes neuronales profundas, especialmente en aquellas que incorporan mecanismos de atención.

Un aspecto de su funcionamiento es el aprendizaje autorregresivo, donde el modelo predice la siguiente palabra de una secuencia textual tomando el contexto de las palabras anteriores. Para entrenarlos, se utilizan una gran cantidad de datos contextuales, permitiéndoles aprender patrones complejos, relaciones semánticas y estructuras sintácticas del lenguaje.

A medida que aumentan los parámetros del modelo, también aumentan la precisión y la capacidad para generar y comprender los textos de manera más efectiva. Sin embargo, el aumento de precisión implica una mayor complejidad y, por ende, una mayor inversión en recursos para el entrenamiento y despliegue del modelo [16].

Entre los beneficios que tiene el uso de estos modelos encontramos:

- **Mayor precisión y coherencia:** generan textos precisos y coherentes, por lo que ayuda a clasificar textos correctamente y la traducción automática de un idioma a otro.
- **Capacidad de generalización y transferencia:** los modelos son capaces de aprender un dominio nuevo y aplicarlo a otros, permitiendo la adaptación a nuevas tareas y la reducción del tiempo y recursos necesarios.
- **Reducción de la necesidad de datos etiquetados:** son entrenados con grandes volúmenes de datos no etiquetados, por lo que disminuye la dependencia de los datos etiquetados y aumenta la eficiencia en el aprendizaje.

Los más utilizados hoy en día son Gemini, Copilot, LLaMA, DeepSeek, Claude y GPT-4o. Cada uno fue desarrollado por diferentes empresas. Gemini pertenece a Google, mientras que Copilot pertenece a Microsoft. GPT-4o fue creado por OpenAI y LLaMA pertenece a Meta.

No obstante, los LLMs avanzan rápidamente y transforman cada día la manera de interactuar con la tecnología y el procesamiento del lenguaje. En un futuro serán capaces de analizar modelos aún más grandes y sofisticados, enfocados en la mitigación de sesgos, la optimización del ahorro energético y la creación de aplicaciones seguras y responsables.

2.3.4. Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) es un método que integra la generación de texto con la recuperación de datos, utilizado para aumentar la exactitud y el contexto de las respuestas de los modelos LLM. [34]

Los modelos de lenguaje de gran tamaño (LLMs) han mostrado un buen rendimiento en tareas como la generación de texto, resumen automático o traducción. No obstante, también presentan limitaciones significativas: suelen generar respuestas erróneas (denominadas alucinaciones), no cuentan con métodos internos para acceder a datos actualizados y, una vez que han sido entrenados, no pueden añadir nueva información sin un proceso de reajuste.

Esta metodología se presenta como una alternativa efectiva para aumentar la exactitud y adaptabilidad de estos modelos, facilitando el acceso a datos externos mientras produce texto.

Según Wu et al. [34], RAG combina un sistema de recuperación de información semántica con un modelo generador, fusionando así lo mejor de las dos áreas: la contextualización de los LLMs y la exactitud de los datos de fuentes externas. En vez de depender únicamente de la información almacenada en el modelo RAG, accede a una base de conocimiento indexada, como manuales técnicos o artículos, y emplea los resultados como texto adicional para crear respuestas más precisas.

El funcionamiento del sistema RAG se fundamenta en tres elementos clave. Primero, el *retriever* transforma la solicitud del usuario en una representación numérica, denominada *embedding*, y encuentra los fragmentos más relevantes dentro de una base de datos preprocesada. Después, el módulo *retrieval fusion* integra esa información obtenida en la entrada original, creando un contexto ampliado. Finalmente, el *generator* (normalmente es un LLM) crea la respuesta, utilizando como fundamento tanto la pregunta como los textos obtenidos.

Esta estructura modular facilita la actualización del contenido del sistema sin requerir el reentrenamiento total del modelo, lo que representa un gran beneficio en contextos dinámicos o especializados.

2.4. Herramientas y estudios existentes

2.4.1. Herramientas

Existen numerosas herramientas que nos ayudan a evaluar cómo de accesible es una página web o una aplicación para personas con discapacidad. Normalmente, son programas o aplicaciones que suelen funcionar en cualquier momento, desde que se empieza a desarrollar la web o la aplicación hasta que está en funcionamiento, y miden si se cumple la normativa WCAG [20].

Para saber cuál nos podría ayudar mejor, solo es necesario mirar las características de cada una y decidir cuál se ajusta mejor al contexto en que la requerimos.

Las más comunes se encuentran listadas en la Tabla 1:

Tabla 1: Herramientas para accesibilidad web (Fuente: elaboración propia a partir de la información de la web MailChimp)

HERRAMIENTA	DESCRIPCIÓN
WAVE	Suite de aplicaciones para ayudar a diseñadores y desarrolladores a garantizar que su contenido sea accesible.
DYNO Mapper	Asiste en la validación del contenido, estructura del sitio y accesibilidad. Incorpora la revisión del sitio, catálogo de contenidos y monitoreo diario de *keywords*.
AXE	Realiza pruebas de accesibilidad durante la fase de desarrollo. Es útil para los desarrolladores que están directamente involucrados en la codificación del sitio.
JAWS	Lector de pantalla destinado a individuos con discapacidad visual. Puede ser utilizado por programadores para probar la accesibilidad de la web o app durante su desarrollo.
Tota11y	Revisa el contraste de color, las descripciones de texto, los puntos de regencia y la conformidad con la normativa WCAG 2.2.
Comprobador de accesibilidad de Intent Based	Escanea los sitios web en busca de incumplimiento de la normativa WCAG actual.
Google Lighthouse	Mejora la calidad de las páginas web. Ofrece auditorías en rendimiento, accesibilidad y aplicaciones web progresivas.

2.4.2. Estudios del estado del arte

En este subapartado, se analizan cinco artículos relevantes que tratan sobre el uso de inteligencia artificial en la accesibilidad web. La selección de estos artículos responde a los criterios específicos que ofrecen una visión representativa y diversa del uso de la inteligencia artificial. Se ha priorizado la variedad de enfoques metodológicos, el contexto de la aplicación y la actualidad de los trabajos, donde se han incluido tanto investigaciones académicas como desarrollos prácticos. Así, se asegura una cobertura equilibrada desde perspectivas teóricas y sociales, teniendo en cuenta soluciones tecnológicas concretas, permitiendo un análisis integral del estado del arte en esta área.

En el primer artículo, el Dr. Décima [9] destaca la responsabilidad de los organismos públicos, concretamente de Buenos Aires, de mejorar el acceso a los servicios electrónicos y garantizar el uso de todos los usuarios, implementando proyectos basados en IA, Big Data y análisis de datos para mejorar servicios como centros de

atención telefónica o sistemas de mensajería con agentes virtuales. El objetivo es asegurar que la interacción con estos sistemas sea natural, promoviendo una accesibilidad más inclusiva.

El Trabajo de Fin de Grado de Nicolás Daniel Vegas Rodríguez [3] se enfoca en el desarrollo de una herramienta web que permite ejecutar algoritmos de optimización para la gestión del transporte de manera accesible, mejorando así la experiencia del usuario. La interfaz visual está diseñada para recoger un fichero de entrada y ejecutar el algoritmo de optimización que ha sido configurado por el propio usuario. En este estudio se demuestra cómo combinar IA y diseño centrado en el usuario puede dar como resultado aplicaciones más accesibles y eficientes.

El artículo escrito por Edson Rufino de Souza [10] habla sobre la importancia de la accesibilidad web para la total participación de los humanos en la sociedad y destaca que la mayoría de los sitios no cumplen con la normativa mínima de accesibilidad. Edson Rufino de Souza sugiere que la integración de técnicas de IA puede ofrecer soluciones rápidas, precisas y adaptativas, permitiendo una evaluación más efectiva.

En el trabajo de Mowar et al. [24] se presenta una extensión denominada CodeA11y, diseñada para ser integrada en asistentes de programación como GitHub Copilot y así fomentar el desarrollo de páginas web accesibles. Es capaz de evaluar en tiempo real la conformidad del código con la normativa WCAG y, como resultado, los desarrolladores que la utilizan suelen generar código más accesible, incluso sin tener experiencia previa en este ámbito.

Complementando estos enfoques, el artículo de María Gabriela Miranda, Adriana Elba Martín y Gabriela Gaetan [22] ofrece un estudio detallado de herramientas basadas en inteligencia artificial destinadas a la accesibilidad web. Subraya la importancia de incluir soluciones automatizadas para identificar y corregir obstáculos de accesibilidad, y examina ejemplos específicos de implementación que demuestran la capacidad de la inteligencia artificial para optimizar la experiencia de los usuarios con discapacidad. Proporciona una mejora de la visión global del estado actual de la investigación en este campo.

2.4.3. Limitaciones

A pesar del creciente desarrollo de herramientas y de modelos de inteligencia artificial que están orientados a la mejora de la accesibilidad web, hoy en día todavía hay numerosas limitaciones que dificultan su aplicación efectiva.

En primer lugar, muchas de las herramientas de evaluación de accesibilidad, como WAVE o Axe-core, que son capaces de detectar errores básicos como la ausencia de etiquetas alternativas, siguen siendo incompletas porque no captan las deficiencias más complejas relacionadas con la experiencia de usuario. Tal como explica Edson Rufino de Souza [10], estas herramientas no comprenden el contexto en el que se presenta la información, llevando a la omisión de problemas realmente críticos para los usuarios.

Además, aunque la IA agiliza los procesos de evaluación, no sustituye la necesidad de revisión manual por parte de los especialistas en accesibilidad. Esto es debido a que muchas de las soluciones generadas automáticamente no son lo suficientemente precisas, lo que obliga a los desarrolladores a intervenir para adaptar o validar los cambios. Por tanto, la aparición de la IA en este ámbito no ha eliminado la carga de trabajo humano, sino que la ha desplazado hacia otras tareas, como de validación y ajuste.

Otra limitación importante es la relacionada con los propios modelos de inteligencia artificial, según José Mauricio Décima [9]. Las arquitecturas de dichos modelos requieren una cantidad masiva de datos y recursos computacionales para su entrenamiento y despliegue. Esto supone una enorme barrera para muchas instituciones, donde no siempre se cuenta con los medios o recursos necesarios para implementarlos y mantenerlos.

En el caso de la extensión CodeA11 presentada por Mowar et al. [24], la dependencia de herramientas como GitHub Copilot significa que su efectividad está sujeta a la calidad y habilidad del modelo baso, por lo que no puede captar todos los detalles de accesibilidad o generar falsos positivos que requieran una revisión manual regular. Además, la conformidad del código puede llevar a pasar por alto elementos dinámicos que no se evidencian en el código estático.

En contraste, el estudio de Miranda, Martín y Gaetán [22] se enfoca en soluciones automatizadas para identificar y solucionar las barreras, aunque se enfrenta a otros desafíos como la complejidad para interpretar diversas situaciones y la exigencia de conjuntos amplios de datos para entrenar modelos exactos. De igual forma, la aplicación práctica puede estar restringida por la diversidad de plataformas web y la ausencia de uniformidad en la implementación de las directrices de accesibilidad, complicando la generalización de resultados y su incorporación en contextos reales.

Finalmente, otro obstáculo importante es la ausencia de estándares claros. Las herramientas aplican distintos criterios y metodologías para valorar y mejorar la accesibilidad, lo que impide un desarrollo común y dificulta su integración en procesos existentes.

2.4.4. Aportaciones al avance de la accesibilidad web mediante IA

Según el enfoque de este Trabajo de Fin de Grado, se pueden aportar innovaciones significativas al integrar modelos de lenguaje de gran tamaño con técnicas de procesamiento multimodal para analizar y mejorar la accesibilidad de interfaces web. Una de las contribuciones principales es la automatización del análisis del cumplimiento del estándar WCAG 2.2, no solo desde la perspectiva textual, sino también mediante la interpretación visual y semántica de los componentes de la interfaz de usuario. Esto nos permite identificar barreras de accesibilidad que otras herramientas no son capaces de detectar, como imágenes con una descripción que no es comprensible, botones sin etiquetas claras o estructuras de navegación confusas.

Además, esta propuesta plantea soluciones generadas automáticamente, adaptadas a los problemas que se detectan, facilitando a los desarrolladores y diseñadores la implementación de mejoras concretas sin un conocimiento especializado previo sobre accesibilidad.

Esta combinación representa un avance práctico y útil en un ámbito donde todavía existen muchas carencias tecnológicas.

3

Recursos aplicados en el proyecto

En este capítulo se detallan los recursos esenciales que respaldan el desarrollo y validación de la herramienta. Se ofrece un catálogo que conecta los criterios de conformidad nivel A de WCAG 2.2 con las soluciones propuestas en diferentes artículos académicos y aplicaciones existentes. Este catálogo actúa como guía para crear y dirigir las estrategias de detección y corrección aplicadas en el proyecto.

Asimismo, se describen las herramientas tecnológicas utilizadas en el desarrollo, incluyendo programas y marcos que permiten la automatización, análisis y verificación de la accesibilidad web.

Finalmente, se presenta el conjunto de páginas web seleccionadas para la aplicación de la herramienta y su posterior evaluación, garantizando una variedad suficiente en cuanto a tecnología, contenido y estructura, facilitando la validación de la efectividad de la herramienta en situaciones reales.

3.1. Introducción al catálogo

En este apartado se presenta una descripción del catálogo elaborado, el cual se encuentra disponible en el anexo B, que relaciona cada uno de los criterios de conformidad del nivel A de las pautas WCAG 2.2 con las soluciones que proponen algunos artículos académicos y aplicaciones disponibles actualmente. Su objetivo es

identificar qué aspectos de la accesibilidad web han sido ya abordados por la inteligencia artificial y tecnologías automatizadas, detectar carencias o criterios que aún no cuentan con solución o cuya solución no está lo suficientemente desarrollada.

Para elaborar este catálogo, se han consultado fuentes científicas y plataformas especializadas en accesibilidad digital. Cada entrada del catálogo especifica el criterio de accesibilidad correspondiente, el tipo de solución propuesta y la fuente de la que procede.

A través del análisis de dicho catálogo se demuestra que, si bien existen múltiples herramientas que solucionan un problema de accesibilidad, hay otros criterios que todavía presentan limitaciones importantes o no se ha desarrollado aún una solución.
Este trabajo no solo permite observar cómo es el estado actual de la accesibilidad web, sino que sirve también para justificar las decisiones técnicas que se adoptan a lo largo del desarrollo de este trabajo.

3.2. Análisis del catálogo

Tras realizar el análisis del catálogo, podemos identificar una serie de patrones. En primer lugar, algunos criterios, como el criterio 1.4.1 y el criterio 2.4.4 están cubiertos por herramientas como WAVE o Axe-core, permitiendo detectar rápidamente los problemas de contraste o la ambigüedad en los enlaces. Otros criterios, como el 1.1.1, han sido abordados no solo por numerosas herramientas, sino también por propuestas recientes que utilizan modelos de lenguaje e inteligencia artificial para generar descripciones automáticas de imágenes e incluso vídeos.

Sin embargo, es importante resaltar que muchos criterios no cuentan aún con soluciones automatizas, motivo por el cual no se encuentran reflejados en el catálogo. Por ejemplo, los criterios como el 3.2.2 y el 2.1.2 requieren un análisis más contextual y dinámico del comportamiento de las interfaces, por lo que las herramientas aún no han sido capaces de afrontar con precisión, o necesitan de intervención humana para su validación final.

Este análisis nos deja ver la necesidad de hallar soluciones más personalizables. Algunas herramientas simplemente aplican las recomendaciones generales, sin tener en cuenta los distintos tipos de discapacidades o las necesidades del usuario final.

Como hemos podido ver, hay tres enfoques principales que, aunque todavía presenten muchas limitaciones, permiten detectar numerosas barreras:

- **Análisis automatizado de código:** esto está presente en las herramientas como WAVE o Axe-core, que detectan errores rápidamente pero no comprenden el contexto.
- **Técnicas de procesamiento de lenguaje natural:** son muy útiles para evaluar la claridad del lenguaje o generar textos alternativos.
- **Técnicas multimodales:** estas técnicas combinan análisis textual y visual y evalúan interfaces de forma más completa.

Finalmente, también se evidencia que, aunque ha habido importantes avances, existen criterios fundamentales de nivel A cuya automatización están en una fase inicial o no estandarizada. Por tanto, es aquí donde el proyecto pretende intervenir, aportando soluciones basadas en modelos multimodales y lenguaje natural para facilitar la valoración y optimización de la accesibilidad.

3.3. Herramientas utilizadas

Para la realización de este trabajo se han empleado un conjunto de herramientas y tecnologías que permiten el análisis de la accesibilidad web y su mejora a través del uso de inteligencia artificial. La elección de estas herramientas es debido a la necesidad de automatizar los procesos, evaluar interfaces web y aplicar modelos inteligentes que detecten y corrijan los problemas existentes de accesibilidad. Las herramientas que se han utilizado son:

- **Python:** ha sido utilizado como lenguaje principal de programación por su versatilidad y amplia disposición de bibliotecas especializadas en procesamiento de lenguaje natural y visión por computador.
- **PyCharm:** es el IDE escogido para desarrollar el código, ya que facilita la organización y depuración del proyecto mediante las herramientas de *testing*.

- **Axe-core:** se utiliza para realizar el análisis de accesibilidad en páginas web reales. Es una de las más utilizadas actualmente.
- **Selenium:** permite la automatización de navegadores web para simular la navegación de un usuario real. Gracias a ella, la herramienta Axe-core ha podido ser integrada en flujos de trabajo.
- **Modelos de lenguaje de gran tamaño (LLMs):** se han empleado a la hora de interpretar elementos no textuales, generar descripciones alternativas y analizar la estructura semántica del contenido, así como proponer cambios en el código HTML.
- **Modelos de lenguaje visual:** se han investigado para identificar componentes visuales, como botones, y ver si cumplen con los requisitos mínimos de accesibilidad visual.
- **Script automatizado:** se ha desarrollado una herramienta propia, en este caso un *script*, que combina estas tecnologías para ejecutar el análisis, recopilar los resultados y proponer soluciones. Esto es uno de los principales aportes del proyecto y representa la aproximación práctica a la aplicación de inteligencia artificial en la accesibilidad web.

3.4. Sitios web utilizados para la evaluación

Con el fin de verificar la eficacia práctica de las herramientas desarrolladas, se han escogido varias páginas web reales con diferentes grados de accesibilidad y estructuras. Esto facilita la evaluación del sistema en diversos contextos y la identificación de problemas habituales. Estas páginas web facilitarán la implementación de los *scripts* del proyecto, el uso de herramientas como Axe-core y Selenium y la generación de mejoras con modelos LLMs y de visión por computador. Por lo tanto, se podrá analizar si se cumplen las WCAG 2.2 de nivel A e identificar las limitaciones actuales, subrayando la importancia de tener soluciones basadas en inteligencia artificial más precisas y personalizadas.

Para evaluar el funcionamiento de la herramienta, se han seleccionado ocho páginas web relacionadas con la salud. Esto ha sido motivado por el interés en aplicarla en un

contexto importante, donde la accesibilidad es un factor clave para garantizar la igualdad entre los individuos.

Los principales criterios que se han seguido han sido:

1. **Conexión directa con el sector salud:** se han seleccionado páginas web de hospitales, clínicas, servicios médicos y plataformas de datos sobre enfermedades y recursos enfocados en pacientes y profesionales sanitarios.
2. **Diversos niveles de complejidad técnica y visual:** se han incorporado webs con estructuras sencillas y otras con mayor carga interactiva, dando la posibilidad de evaluar la herramienta ante distintos escenarios.
3. **Diversidad institucional:** se abarcan tanto sitios oficiales del sector público como del privado, con el propósito de analizar variaciones en el cumplimiento de las pautas de accesibilidad y entre diferentes clases de entidades.

Estos criterios permiten verificar el funcionamiento técnico adecuado e investigar su aplicación en un área específica, como la sanitaria, donde la accesibilidad tiene una elevada importancia debido al impacto directo en la vida de los usuarios.

Las webs elegidas son las que aparecen en la Tabla 2.

Tabla 2: Elección de páginas webs (Fuente: elaboración propia)

N.º	Comunidad/Entidad	Sitio Web	Tipo de sitio
1	Junta de Andalucía	https://hospitalpuertadelmar.com/	Hospital Público Andaluz
2	Hospital Vithas	https://vithas.es/centro/vithas-hospital-granada/	Hospital Privado
3	Junta de Comunidades de Castilla La Mancha	https://sanidad.castillalamancha.es/	Portal de Salud Autonómico
4	Generalitat de Cataluña	https://www.hospitaldelmar.cat/ca/	Hospital Público Catalán
5	Gobierno de Aragón	https://hospitalclinico.salud.aragon.es/	Hospital público

6	Fundación del Corazón	https://fundaciondelcorazon.com/	Fundación del Sistema Cardiovascular
7	Clínica Dental Your Clinic	https://yourclinic.es/	Clínica Dental Your Clinic
8	Junta de Andalucía	https://areasanitariaserrania.es/	Área Sanitaria de Serranía

4

Diseño de la herramienta

En este capítulo se explica el proceso de diseño de la herramienta propuesta para la evaluación y optimización de la accesibilidad web, cumpliendo con la normativa WCAG 2.2, en el nivel A. La herramienta ha sido pensada como una solución automatizada e inteligente, siendo capaz de identificar carencias o deficiencias de accesibilidad, generar descripciones textuales de imágenes mediante técnicas de procesamiento multimodal y proponer correcciones guiadas por los modelos de lenguaje de última generación.

El diseño se ha estructurado en varias fases, las cuales están explicadas a lo largo del capítulo.

4.1. Arquitectura de la herramienta

4.1.1. Descripción de la arquitectura modular

Para lograr el desarrollo de una herramienta de análisis y evaluación de la accesibilidad web, se ha diseñado una arquitectura modular, flexible y escalable. Debe ser capaz de integrar herramientas de análisis automático, técnicas de inteligencia artificial y mecanismos de evaluación multimodal. Así, esta estructura permite realizar un análisis profundo de las páginas web seleccionadas, detectar los errores de accesibilidad que presentan y proponer las soluciones adecuadas.

Esta estructura se clasifica en tres niveles fundamentales:

- **Nivel de coordinación:** representado por la clase `Main`, la cual funcionará como punto de acceso y coordinador del sistema.
- **Nivel de procesamiento:** conformado por los elementos centrales que llevan a cabo las funciones principales.
- **Nivel de soporte:** compuesto por herramientas que ofrecen funciones complementarias.

4.1.2. Manejo de errores robusto

El diseño de la herramienta incorpora un manejo de errores sólidos para garantizar que el proceso de análisis y mejora de accesibilidad opere de manera estable, incluso ante fallos o situaciones inesperadas. Esto significa identificar y gestionar fallos como dificultades de conexión con la API de OpenAI, problemas al cargar páginas web con Selenium o errores en el procesamiento del contenido HTML.

Para lograr esto se llevan a cabo reintentos automáticos, se generan registros minuciosos de errores y proporcionan mensajes claros al usuario, previniendo que la aplicación se cierre de forma inesperada.

Este enfoque permite que la herramienta siga operando adecuadamente, proporcionando una análisis confiable y simplificando su mantenimiento.

4.1.3. Flujo de trabajo del sistema

El sistema mantiene un proceso de trabajo lineal y claramente establecido. A continuación, se detalla cada uno de los módulos de la herramienta:

a) Módulo de adquisición de contenido web

El primer paso consiste en acceder al contenido de las páginas web que se han escogido para su evaluación. Esto se hace con la ayuda de Selenium, una herramienta que permite automatizar la interacción de las páginas web de manera parecida a como lo haría un usuario. Gracias a esto, se puede capturar tanto el HTML estático como los elementos generados dinámicamente, lo que garantiza una visión completa de la interfaz de la web. También es responsable

de extraer el árbol DOM de cada página, algo que será fundamental para los análisis posteriores.

b) Módulo de detección automatizada de errores

Constituye la base inicial del análisis, proporcionando una primera capa de evaluación sobre la que el resto de los módulos puede operar. Una vez que hemos obtenido la estructura de la web, se ejecuta un análisis automático mediante la integración de Axe-core, una de las soluciones más utilizadas en la evaluación de la accesibilidad web. Identifica los errores más comunes, como errores de contraste de color, problemas de navegación mediante teclado o falta de etiquetas `alt`. Cada error se registra con su tipo, ubicación y explicación, facilitando su posterior tratamiento.

c) Módulo de análisis visual multimodal

Este módulo se encarga de asegurar que los elementos visuales de las páginas web tengan descripciones alternativas correctas. El procedimiento inicia reconociendo de manera automática los elementos visuales de la página web. Luego, cada imagen es codificada y evaluada con inteligencia artificial para crear descripciones textuales precisas, breves y adecuadas a las necesidades de los individuos con discapacidad visual. Estas descripciones se almacenan para facilitar su uso posterior y prevenir repeticiones en análisis futuros.

d) Módulo de interpretación y corrección con LLMs

El proyecto incorpora un módulo que se apoya en modelos de lenguaje de gran tamaño (LLMs). Estos están entrenados para comprender y generar lenguaje natural.

Esta parte tiene como función interpretar los errores detectados por Axe-core, entender el contexto de los elementos afectados y generar soluciones. Por ejemplo, si se detecta una imagen sin descripción, el modelo puede generar esta descripción en función del contexto o lo que interprete en la imagen. La inteligencia artificial no solo hace una comprobación técnica, sino que actúa

como un asistente que convierte los descubrimientos en acciones tangibles y alcanzable para los desarrolladores.

Además, el sistema permite generar una versión corregida del código HTML original, donde se aplican las soluciones propuestas por el modelo de inteligencia artificial. Esta copia se guarda como archivo adicional, facilitando la revisión técnica y la validación visual de los cambios que se han realizado. Esto optimiza el trabajo de los desarrolladores y acelera la implementación de mejoras de accesibilidad en entornos reales.

e) Módulo de generación de informes

Para finalizar, los resultados obtenidos se integran en un informe estructurado. Cada evaluación produce un documento que recoge el número de errores de accesibilidad detectados, el número de errores que corrige y el porcentaje de mejora. Este informe permite valorar el estado actual de una web y comprobar la evolución tras haber aplicado las correcciones. Además, estos datos son fundamentales para la validación de la herramienta y el análisis empírico del impacto, pudiendo realizar una comparativa entre webs y visualizar patrones de inaccesibilidad.

4.2. Componentes principales

Esta sección explica la estructura interna del sistema creado para la valoración y mejora de la accesibilidad web. La solución se estructura en diversos módulos y scripts que se comunicar para automatizar el análisis de accesibilidad con Axe-core, implementar correcciones, crear descripciones para imágenes sin etiquetar y elaborar un informe comparativo.

4.2.1. `Main.py`

Controla el flujo del sistema: se lanza el navegador con la URL introducida y comienza el análisis con Axe-core.

Una vez analizado, se obtienen los errores, se comienza a descargar las imágenes y generar las descripciones correspondientes. Se realiza la corrección de los errores.

Por último, se genera el informe comparativo de antes y después de aplicar la herramienta.

4.2.2. Carpeta `core`

- **`analyzer.py`:** ejecuta Axe-core, analiza los errores de accesibilidad y los recoge en un archivo JSON.
- **`html_generator`:** aplica las correcciones propuestas al HTML original.
- **`image_processing.py`:** detecta las imágenes sin descripción, las descarga, las guarda en la caché y solicita descripciones a GPT-4o.
- **`report.py`:** genera el informe con los datos de los errores antes y después de aplicar la herramienta.
- **`webdriver_setup.py`:** inicializa Selenium y realiza la configuración del navegador.

4.2.3. Carpeta `utils`

- **`violation_utils.py`:** clasifica los errores según la normativa WCAG 2.2 nivel A.
- **`io_utils.py`:** proporciona funcionalidades para la gestión de archivos y directorios, manejo del sistema de caché y operaciones de entrada/salida.
- **`html_utils.py`:** funciones para manipular el DOM.

4.2.4. Carpeta `config`

- **`constants.py`:** almacena las variables reutilizables como el tiempo de espera, claves o directorios.

4.2.5. Carpeta `templates`

Contiene una plantilla Jinja2 para generar el informe final en formato HTML. Esta plantilla facilita la exposición clara y ordenada de los resultados obtenidos después del análisis y la corrección de la página web. El sistema completa la plantilla con el número de errores detectados, corregidos y no corregidos, y un porcentaje de mejora de la accesibilidad de la página web.

Jinja2 facilita la distinción entre la lógica del programa y el contenido visual, lo que ayuda en la personalización y reutilización del informe.

La Ilustración 1 muestra un diagrama de la arquitectura del sistema, en el que la clase principal coordina el flujo de ejecución y los componentes centrales que se ocupan del análisis de accesibilidad, la corrección del HTML, el procesamiento de imágenes y la generación de informes. Estos módulos se basan en herramientas concretas para manejar el DOM, clasificar errores y administrar archivos, promoviendo un diseño modular, limpio y sostenible.

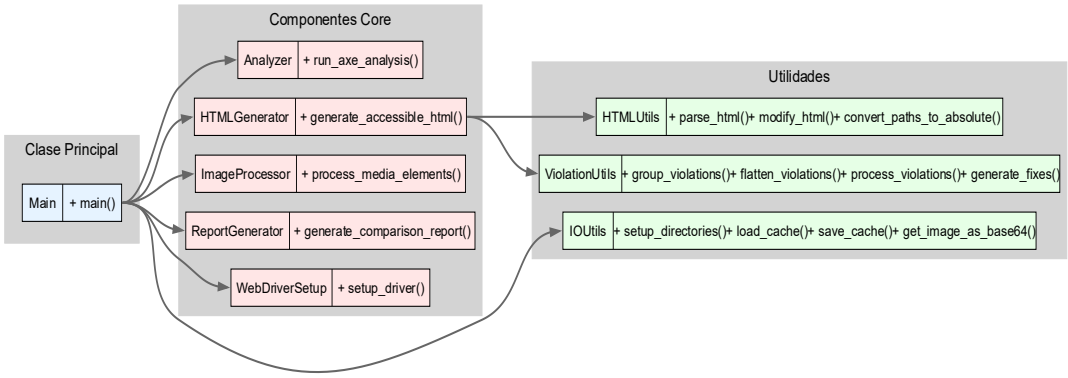

Ilustración 1: Diagrama de clases de la herramienta

4.3. Integración de tecnologías externas

El sistema creado se basa en la combinación de distintas tecnologías externas que enriquecen y amplifican sus habilidades en relación con el análisis, la creación y la automatización, asegurando de este modo un proceso completo de optimización de la accesibilidad web.

- **Integración con OpenAI:** es una entidad enfocada en la investigación y desarrollo de inteligencia artificial avanzada, que proporciona modelos de lenguaje como GPT-4o, capaces de entender y generar texto de manera coherente y contextual. La conexión con la API de OpenAI permite al sistema hacer uso de estas funcionalidades a través de una clave de la API que valida y permite las solicitudes a la plataforma. Esta relación es esencial para llevar a cabo actividades importantes en el ámbito de la accesibilidad web, como la creación automática de las descripciones de las imágenes, generando textos alternativos precisos y completos que enriquecen la experiencia de los usuarios con

discapacidad visual. Asimismo, la API se emplea para perfeccionar y optimizar textos ya existentes, modificándolos para que se adhieran a las normas de accesibilidad y sean más claros y comprensibles.

- **Integración con Axe-core:** es una biblioteca de código abierto creada para llevar a cabo auditorías automáticas de accesibilidad en sitios web. Se encarga de examinar el código HTML y la organización de la página web para detectar problemas frecuentes de accesibilidad, como elementos sin etiquetas correctas. Gracias a esta integración, el sistema es capaz de realizar un análisis detallado de la accesibilidad, identificando los errores de manera precisa y ofreciendo informes claros sobre los elementos que requieren mejora. Así, Axe-core verifica que la página cumpla con las normas internacionales de accesibilidad, especialmente las pautas WCAG 2.2.
- **Integración con Selenium:** es un conjunto de herramientas creado para la automatización de navegadores web, posibilitando la simulación de comportamientos de usuarios como clics, desplazamientos y navegación entre páginas. En el marco de esta herramienta, Selenium se emplea para automatizar la navegación en la web, garantizando que el sistema pueda comunicarse con páginas dinámicas y cargar todos los elementos requeridos para su evaluación. Esto abarca la capacidad de abrir páginas, explorar menús y recoger el contenido más reciente junto con el código fuente de la página. Con esta herramienta, el análisis y la corrección se llevan a cabo sobre la versión precisa que el usuario final vería, abarcando elementos dinámicos creados a través de JavaScript o interacciones particulares. Igualmente permite la captura exacta del contenido y el DOM, lo cual es esencial para el funcionamiento adecuado de los módulos de análisis y creación de descripciones.

4.4. Archivos generados

Los archivos que se generan en el flujo completo son:

- **`Original_page.html`:** contiene el HTML original de la página web a analizar.

- **`Initial_report.json`:** informe con los errores detectados antes de la aplicación de la herramienta.
- **`Accesible_page.html`:** página con el HTML corregido.
- **`Final_report.json`:** informe posterior a la corrección automática.
- **`Comparison_report.html`:** informe comparativo de cambios y mejoras antes y después de aplicar la herramienta.

4.5. Personalización

La accesibilidad web no puede plantearse como una solución estática y uniforme. Cada persona tiene unas necesidades diferentes dependiendo del tipo de discapacidad, nivel de alfabetización digital, idioma o cultura. Por tanto, uno de los aspectos fundamentales de la herramienta desarrollada es su capacidad de personalización. Esto es fundamental para garantizar que las soluciones cumplan con la normativa, aporten valor y mejoren la experiencia de navegación del usuario.

En primer lugar, el sistema permite configurar perfiles personalizados de evaluación, orientados a los distintos tipos de discapacidad existentes. Estos perfiles actúan como filtros que priorizan el análisis sobre los errores más comunes de cada colectivo. Así, se ha definido un conjunto enfocado a:

- **Usuarios con discapacidad visual:** la herramienta se centra en ausencia de descripciones alternativas para imágenes, estructuras semánticas incorrectas, niveles de contraste no adecuados o falta de etiquetas en formularios y elementos interactivos.
- **Usuarios con discapacidad motora:** se detectan problemas que limitan la navegación mediante teclado, botones o enlaces inaccesibles.

Esta clasificación por perfiles permite generar evaluaciones precisas y relevantes, adaptándose a las necesidades reales de cada usuario y no solo a la verificación de la normativa WCAG 2.2.

En segundo lugar, la herramienta ofrece una disposición flexible de la normativa. Aunque esté diseñada en torno a los 31 criterios del nivel A, puede progresar a los niveles AA y AAA. También admite guías específicas de accesibilidad para entornos

concretos, como portales educativos o sistemas de información sanitaria. Esto permite la aplicación de la herramienta en diferentes contextos internacionales, respetando las diferencias entre legislaciones y normativas locales.

Además, la personalización se refleja en cómo se presentan los resultados. Tanto los informes como las páginas web corregidas pueden exportarse en distintos formatos y niveles de detalle. Para un desarrollador, puede mostrarse el código HTML con los cambios propuestos resaltados. Para un diseñador, se puede priorizar la vista previa visual con marcadores de errores. Y para quienes gestionan la accesibilidad, se puede ofrecer una puntuación global y una lista de mejoras implementadas o pendientes de implementarse. Esa flexibilidad en la presentación hace que la herramienta sirva a diferentes perfiles profesionales y fomente la colaboración entre equipos.

En conclusión, la personalización es un pilar clave de la herramienta. Gracias a estos mecanismos, el sistema va más allá de evaluar solo lo técnico, convirtiéndose en un sistema inteligente que ayuda a los desarrolladores a crear soluciones accesibles que respondan a necesidades concretas, normativas vigentes y realidades diversas.

5

Implementación de la herramienta

5.1. Fases del desarrollo de la herramienta

La herramienta ha sido desarrollada en el entorno PyCharm, utilizando Python como lenguaje principal por si versatilidad y amplio contenido de librerías para automatización y análisis de páginas web. Se ejecuta aplicando varias fases que permiten el análisis exhaustivo de la accesibilidad de una página web, identificar los errores conforme a los criterios de la WCAG 2.2 en su nivel A y generar una versión nueva de la web con las soluciones incorporadas.

Este proceso nos permite automatizar gran parte del trabajo al unir la identificación automática de errores con la interpretación textual a través de modelos de lenguaje y análisis visual. De esta manera, se reconocen de manera ágil las limitaciones de accesibilidad, se entiende su efecto y se crean sugerencias concretas para su solución. Asimismo, la herramienta tiene la capacidad de implementar de manera automática las soluciones en el código, lo que permite una resolución eficiente y reduce la carga manual para los desarrolladores.

A continuación, se detalla cada una de las fases del proceso:

1) **Fases del módulo de adquisición de contenido web**

- **Fase 1: Ejecución de la herramienta**

  Para iniciar el programa, se ejecuta en una terminal el comando:

  ```
  python main.py --url http://ejemplo.com/
  ```

  y se presiona Enter.

- **Fase 2: Configuración del entorno con Selenium**

  Cuando se ejecuta el programa se lleva a cabo la configuración del entorno usando Selenium, una biblioteca que permite automatizar la apertura del navegador Google Chrome en modo *headless*, es decir, sin interfaz gráfica, y simular la interacción entre el usuario y la interfaz. Esto posibilita una ejecución más eficiente en segundo plano. En esta fase se lleva a cabo la inicialización del WebDriver pertinente. En concreto, se utiliza Chromedriver, un controlador específico que permite a Selenium controlar Google Chrome con el uso de las bibliotecas Selenium y Webdriver_manager, encargado de manejar la navegación automática en cada página web.

- **Fase 3: Simulación de navegación**

  Selenium accede a cada página imitando la experiencia de usuario activo. Esta simulación realiza una espera activa de tres segundos para que los elementos dinámicos que pueden no estar presentes en el código fuente se carguen adecuadamente. En ciertos casos, puede ser fundamental realizar acciones simples como la interacción con elementos para garantizar que toda la información pertinente esté visible y accesible para un análisis posterior.

- **Fase 4: Obtención del contenido web**

  Al finalizar la carga total de la página, se obtiene el contenido HTML que ha sido renderizado. A diferencia del código fuente original, este HTML incorpora los elementos creados de forma dinámica, lo que ofrece una perspectiva más auténtica del contenido que ve el usuario. Asimismo, se obtiene el árbol DOM en su totalidad, incluyendo atributos importantes como clases, identificadores (IDs), roles y estructuras semánticas, fundamentales en las etapas posteriores de evaluación. Aquí,

BeautifulSoup se utiliza para estructurar, parsear y limpiar el HTML para realizar una manipulación más fina del contenido.

- **Fase 5: Almacenamiento del contenido web obtenido**

  Finalmente, el HTML de la URL proporcionada se guarda en una carpeta local como archivo HTML de forma ordenada y limpia, preparada para ser manejada por los siguientes módulos del sistema.

En la Ilustración 2 podemos observar un fragmento del código fuente de una página web antes de aplicar la herramienta desarrollada.

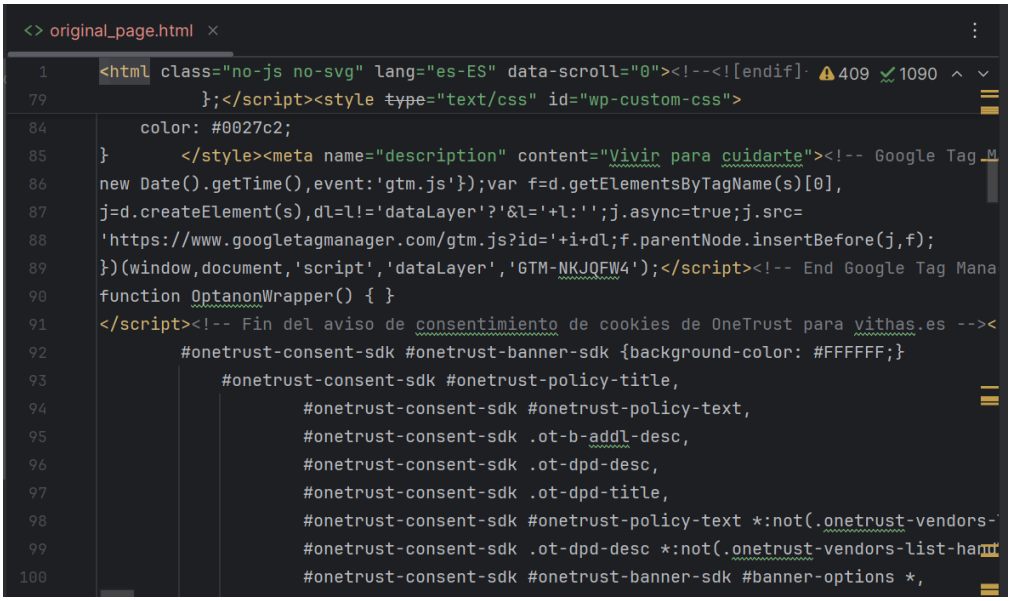

Ilustración 2: Fragmento del HTML original de una página web

2) Fases del módulo de detección automatizada de errores

- **Fase 1: Configuración del entorno de análisis**

  Al obtener el contenido estructurado de la página web, se configura el entorno necesario para poner en marcha el análisis automático.

  Se incorporan herramientas como Axe-core, que actúa como una biblioteca diseñada para analizar la conformidad de los elementos del DOM con las pautas WCAG. Se asegura la carga completa del DOM y la sincronización entre las bibliotecas Selenium y Axe-core para que el análisis se efectúe sobre la versión renderizada de la página.

- **Fase 2: Realización del escaneo con Axe-core**

  Una vez que el entorno configurado, se inicia el proceso de escaneo automático. Axe-core examina la estructura del DOM del sitio web para identificar incumplimientos de la normativa de accesibilidad WCAG 2.2 de nivel A. Entre los errores que se pueden identificar están las imágenes con falta de texto alternativo, el uso incorrecto de encabezados y la presencia de botones sin etiquetas accesibles. Este procedimiento es ágil y se lleva a cabo directamente sobre el DOM que ha sido cargado por Selenium.

  En la Ilustración 3 podemos observar el mensaje en consola que indica que el análisis con Axe-core de la página original se está realizando

```
--- PASO 1: Análisis de la página original ---
Analizando con Axe: https://vithas.es/centro/vithas-hospital-granada/
```

Ilustración 3: Mensaje en la consola sobre el comienzo del análisis con Axe-core

- **Fase 3: Documentación detallada de errores identificados**

  Cada violación identificada por Axe-core se documenta de forma estructurada. Esto abarca el tipo de error, el nodo particular del DOM donde sucede, la gravedad aproximada (crítico, moderado, leve) y una descripción técnica de la cuestión. Esta información es fundamental para las etapas posteriores del proyecto, ya que conecta cada error con soluciones automáticas proporcionadas por IA. Para almacenarla, se utiliza la biblioteca JSON.

- **Fase 4: Categorización y agrupación de resultados**

  Después de la recopilación, se clasifican los errores en grupos, lo que simplifica el análisis. Por ejemplo, todos los errores de botones sin etiquetas accesibles se relacionan con el criterio 4.1.2 de WCAG 2.2. Esta categorización posibilita relacionar directamente los errores con los criterios normativos y simplifica su gestión en los módulos de recomendación y corrección.

- **Fase 5: Depósito para análisis posterior**

  Por último, los resultados se guardan en un archivo JSON. Esto asegura la trazabilidad de los errores identificados y permite que otros módulos del sistema utilicen estos datos sin requerir un nuevo escaneo del contenido

web. Asimismo, esta etapa contempla un registro temporal para analizar la evolución de la accesibilidad después de implementar correcciones.

3) Fases del módulo de análisis visual multimodal

- **Fase 1: Obtención de imágenes a través de Selenium**

  La herramienta navega por la página web a través de Selenium para replicar la experiencia de un usuario. En este proceso, se localizan todos los elementos `‹img›` del DOM, que son los encargados de cargar imágenes. Después, se obtiene su atributo `src`, el cual contiene la URL de la imagen que se utilizará para generar la descripción.

  Además, se utiliza la biblioteca Time para introducir esperas eventos del navegador y asegurar que las imágenes estén completamente cargadas.

- **Fase 2: Selección por dominios**

  Antes de proceder a la descarga de las imágenes, se implementa un filtro de exclusión utilizando una lista de dominios restringidos configurados manualmente, como servicios de CDN privados, sitios con licencias restrictivas o fuentes externas conocidas por no permitir descargas. Para ello, se utiliza la biblioteca Urllib.parse, comparando su dominio con la lista de exclusión. Esto posibilita eliminar las imágenes que provienen de fuentes externas no relevantes o cuya descarga está prohibida por el autor.

- **Fase 3: Descarga y codificación**

  Las imágenes correctas se obtienen desde sus URLs correspondientes con la biblioteca Requests, a menos que ya estén guardadas en la memoria caché local. Mediante la biblioteca hashlib, se crea un identificador único con el objetivo de evitar duplicados. Esta caché es un mecanismo de almacenamiento transitorio que conserva documentos y datos ya procesados para evitar ejecuciones repetidas. En esta situación, las imágenes se guardan de manera local por lo que, si se observa que una imagen se encuentra en la caché, no se descarga ni procesa nuevamente. A continuación, se convierten al formato base64 para cumplir con los requisitos de ingreso de la API de OpenAI.

- **Fase 4: Creación de descripciones con GPT-4o**

  Cada imagen codificada se envía al modelo multimodal GPT-4o de OpenAI a través de la biblioteca OpenAI para obtener una descripción breve y clara, creada especialmente para personas con discapacidad visual. Con el envío de la imagen se utiliza el prompt

  `"Describe esta imagen para el texto alternativo ('alt') de una página web. Sé conciso y útil para una persona con discapacidad visual."`

  El modelo analiza visualmente la imagen y devuelve la descripción, pensada para ser utilizada como valor en los atributos alt y title en la etiqueta ‹img›.

  En la Ilustración 4, observamos el mensaje en consola sobre el número de imágenes encontradas, el inicio de la descarga y su procesamiento.

```
--- PASO 2: Generando página accesible ---
Procesando elementos de medios (imágenes)...
Se encontraron 51 imágenes.
Descargando imagen: https://vithas.es/wp-content/uploads/flags/spain.png
Generando descripción para la imagen: 06a5e3581a7677c1188e59d0842be2c58ce5f851180cc763d7c6ba7c8acfeb69.png
Created TensorFlow Lite XNNPACK delegate for CPU.
  > Imagen procesada y cacheada.
```

Ilustración 4: Inicio de la descarga de imágenes y generación de descripciones

- **Fase 5: Enriquecimiento contextual mediante RAG (*Retrieval-Augmented Generation*)**

  En esta etapa, el sistema lleva a cabo un proceso de optimización de la accesibilidad web que combina la biblioteca BeautifulSoup con la inteligencia artificial.

  BeautifulSoup se encarga del análisis y manipulación del HTML, extrayendo las partes importantes que necesitan ajuste. Estos fragmentos se procesan utilizando un sistema de indicaciones contextuales creado para un modelo de lenguaje, que detecta y soluciona infracciones de accesibilidad, como la falta de descripciones de imágenes o atributos ARIA. Esta metodología facilita no solo la identificación de problemas, sino también la creación de soluciones personalizadas según el contenido particular, lo que mejora notablemente la experiencia del usuario para individuos con discapacidad.

La combinación de BeautifulSoup con el modelo de lenguaje permite una retroalimentación constante donde cada ajuste se incorpora directamente en el documento HTML, preservando su estructura original y garantizando que las mejoras en accesibilidad se implementen de manera coherente y eficaz.

- **Fase 6: Cacheo y devolución estructurada**

  Las descripciones creadas se guardan en una caché local, utilizando como clave el atributo `src` y como valor, la descripción generada. Esto previene la descarga o procesamiento de las mismas imágenes en ejecuciones posteriores, mejorando así la eficiencia.

  Asimismo, se modifica el DOM original: a cada elemento sin descripción se le añaden dos atributos, `alt` y `title`, con el texto generado.

  Este proceso se lleva a cabo a través de la biblioteca BeautifulSoup, que posibilita modificar directamente los nodos del DOM. Por último, el HTML corregido se guarda en archivos mediante los módulos `os` y `pathlib`, garantizando una escritura organizada y compatible con múltiples plataformas.

4) Fases del módulo de interpretación y corrección con LLMs

- **Fase 1: Filtrado de errores**

  Después de llevar a cabo el análisis automático utilizando Axe-core y obtener una lista de los problemas de accesibilidad encontrados, debemos filtrar los errores en relevantes e irrelevantes. No todos los errores requieren el mismo enfoque y es necesario enfocarse en aquellos que realmente importan en la experiencia del usuario.

  En esta etapa, la herramienta evalúa cada uno de los componentes de la página web relacionados con los errores y determina si es necesario solucionarlo. Técnicamente, estos componentes son nodos del DOM, la estructura interna que refleja todo lo visible de una página web. La característica fundamental es si ese nodo es perceptible o no para el usuario. Si no lo es, no tiene sentido solucionarlo. Por ejemplo, si un

botón no es visible o está oculto, solucionarlo podría dar lugar a confusión.

Para llevar a cabo este filtrado, se utiliza la librería BeautifulSoup. Es una herramienta de análisis estructurado de HTML que se encarga de recorrer el DOM y eliminar directamente los nodos que no son visibles o no tienen impacto real en la navegación.

- **Fase 2: Agrupación de errores por tipo**

  Una vez tenemos clasificados los errores, se realiza una agrupación según el identificador estandarizado que Axe-core asigna a un tipo particular de violación de accesibilidad (ID). De este modo, todas las instancias que compartan el mismo tipo de violación se procesarán de manera conjunta y permitirá al sistema generar soluciones más coherentes y generalizables para ese tipo de problema. Por ejemplo, si hay cinco errores cuyo ID es "botones sin etiqueta accesible", se agruparán y serán procesados de manera conjunta.

- **Fase 3: División en bloques manejables**

  Para prevenir exceder los límites de tokens en la interacción con la API de OpenAI, los errores agrupados se separan en segmentos de cinco nodos, siendo este número el más adecuado en cuanto a limitación de tokens. Esta división facilita la conservación del contexto requerido para cada categoría de errores sin afectar la eficacia de las respuestas producidas por la inteligencia artificial.

- **Fase 4: Generación asistida por LLM**

  Cada bloque se envía al modelo de lenguaje GPT-4o junto con un prompt cuidadosamente diseñado. Esto orienta al modelo para que funcione como un experto en el desarrollo de accesibilidad web, ofreciendo una versión mejorada del código HTML original que soluciona todas las instancias del error especificado.

  El prompt utilizado es el siguiente:

```
"Eres un desarrollador web experto en accesibilidad
   (WCAG 2.2, nivel A). Tu tarea es reescribir el
   siguiente código HTML para corregir múltiples
instancias del mismo tipo de error de accesibilidad.
```

```
Tipo de error: {description}
ID del error: {violation_id}

Elementos afectados:
{node_list_str}

Mapeo de descripciones de imágenes:
```json
{descriptions_json}
```

HTML original:
```html
{current_html}
```
"
```

- **Fase 5: Generación de la web corregida**

  Después de corregir todos los errores de accesibilidad identificados, el sistema pasa a la creación del código HTML accesible completo.

  Esta fase no se trata únicamente de proporcionar secciones aisladas que han sido corregidas, sino de construir un texto cohesionado y operativo. Para lograr esto, se utiliza la biblioteca BeautifulSoup, manipulando el DOM de forma que encuentra los nodos originales que contienen errores y los reemplaza directamente por sus equivalentes corregidos. Gracias a su capacidad para editar y limpiar HTML, se garantiza una reconstrucción fiel y funcional del código fuente.

  A lo largo de este proceso, se llevan a cabo varias transformaciones adiciones que son cruciales para asegurar la calidad, como son:

  - **Reemplazo de fragmentos corregidos:** cada nodo HTML que mostraba un error es sustituido por su versión accesible. Esta versión fue generada anteriormente por el modelo GPT-4o.
  - **Incorporación de atributos accesibles (alt, aria-label, role):** por ejemplo, si se identifica que un botón carece de una etiqueta accesible, el modelo sugiere una solución apropiada que posteriormente añade al nodo correspondiente. Igual sucede con las imágenes o enlaces.

- **Limpieza del material no visible:** se eliminan del código final los elementos del HTML considerados invisibles o irrelevantes para mitigar el ruido innecesario y disminuir la carga cognitiva.
- **Transformación de rutas relativas en absolutas:** gracias a `urllib.parse` y una función particular como `urljoin`, todos los enlaces o direcciones que anteriormente eran relativos (`/img/logo.png`) se convierten en absolutas (`https://example.com/img/logo.png`). Esto garantiza que los recursos se carguen adecuadamente al visualizar el HTML desde contextos diferentes al original.

Al finalizar esta etapa, se genera un documento HTML ordenado, accesible y que cumple con la normativa WCAG 2.2 de nivel A, preparado para ser analizado nuevamente con Axe-core. Su objetivo es verificar qué errores han sido corregidos y cuáles no han podido ser solucionados. Este HTML generado se guarda en una carpeta local, con el nombre de la página web y una marca de tiempo.

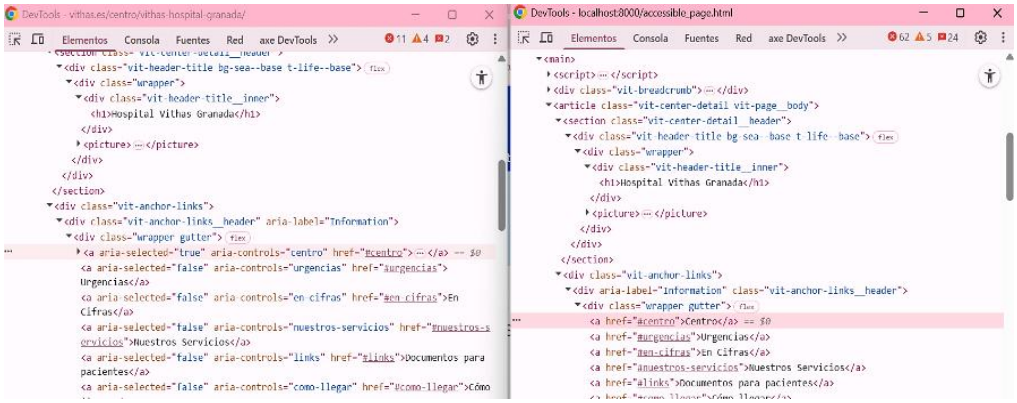

Ilustración 5: Comparación entre error encontrado en la página original (izquierda) y la corrección en la página web accesible (derecha)

En la Ilustración 5, observamos la corrección del error encontrado en la página web original, mostrada en la izquierda, sobre el atributo ARIA-LABEL. A la derecha, se observa la página web corregida.

Para poder visualizar la página web accesible, se ofrece al usuario la previsualización de la página corregida. Si se acepta, se inicia un servidor

local en el puerto 8000 o en el siguiente disponible hasta 8050, y aparece un mensaje en consola si quieres que se abra el navegador predeterminado con la URL correspondiente.

Para esto, se emplea `http.server.SimpleHTTPRequestHandler` para manejar peticiones HTTP, `socketserver.TCPserver` para crear y gestionar el servidor loca y `webbrowser` para abrir el navegador. El servidor permanece activo hasta que el usuario lo detiene manualmente o cierre la terminal.

5) Fases del módulo de generación de informes

- **Fase 1: Cálculo de métricas y resultados de mejora**

  Al corregir la página web, el sistema recoge y mide los datos más significativos:

  - **Errores identificados:** número total de incumplimientos de accesibilidad encontrados en el primer análisis.
  - **Errores rectificados:** cantidad de problemas resueltos después de la aplicación de la herramienta.
  - **Errores no corregidos:** número de errores que no han podido ser corregidos tras aplicar la herramienta.
  - **Porcentaje de mejora:** proporción entre fallos corregidos y fallos totales, mostrando el impacto real de la herramienta.

  Esta información permite evaluar de manera objetiva el progreso de la accesibilidad en cada situación específica.

- **Fase 2: Creación del informe técnico**

  A partir de la información proporcionada, se elabora de forma automática un informe claro y detallado en formato HTML. Este contiene el resumen numérico de los parámetros calculados anteriormente, permitiendo una comparación entre el HTML original y el HTML corregido. Para ello, se utiliza la biblioteca Jinja2, que permite construir dinámicamente el contenido del informe y garantizar que tenga una estructura coherente y fácil de interpretar.

6

Validación de la herramienta

Para comprobar la eficacia de la herramienta desarrollada para la mejora de la accesibilidad web, se ha llevado a cabo un proceso sistemático utilizando la extensión Axe DevTools, basada en la biblioteca Axe-core, incorporada en las herramientas del desarrollador del navegador.

Una vez que accedemos a la página web que queremos analizar, realizamos un escaneo de la web completa a través de Axe DevTools. Esto facilita la identificación y el listado de los errores de accesibilidad.

Después, aplicamos la herramienta desarrollada en este proyecto para corregir las violaciones de nivel A de las pautas WCAG 2.2 que han sido encontradas y, una vez que obtenemos la página web corregida, volvemos a realizar el escaneo completo utilizando dicha extensión.

Con esto, verificamos la eliminación completa o parcial de las incidencias detectadas en el primer análisis, facilitando una evaluación objetiva de la mejora de la accesibilidad que proporciona la herramienta creada.

6.1. Casos de prueba

6.1.1. Hospital Clínico de Aragón

URL analizada: https://hospitalclinico.salud.aragon.es

a. Estado inicial

En primer lugar, se ha realizado un análisis de la accesibilidad de esta página web utilizando Axe-core. Antes de aplicar la herramienta, se han detectado los siguientes errores de nivel A de las pautas WCAG 2.2, mostrados en la Ilustración 6:

- Falta de atributos alt en las imágenes informativas y decorativas.
- Enlaces con textos ambiguos sin contexto.
- Uso incorrecto de etiquetas semánticas, como `‹main›`, `‹header›` o `‹nav›`.
- Botones sin descripción *aria-label*.

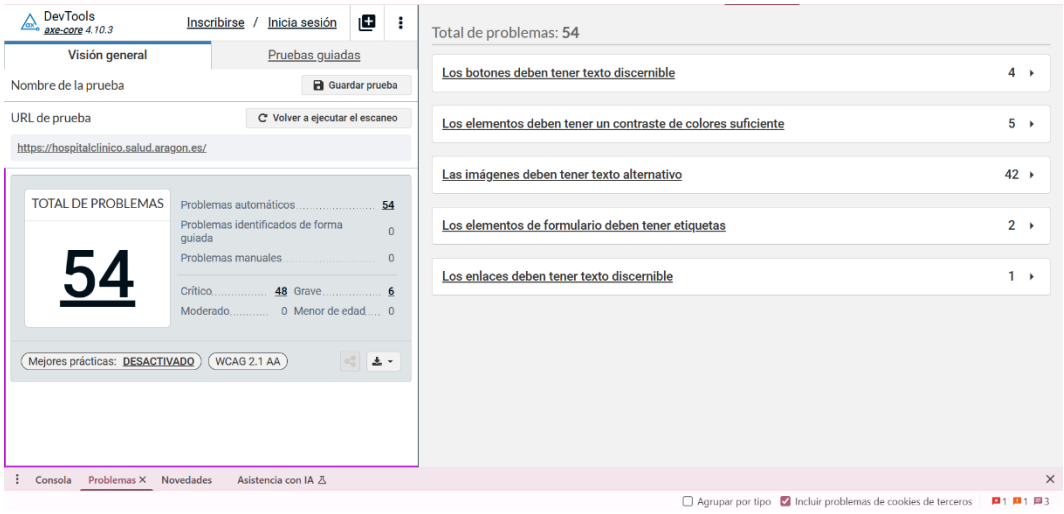

Ilustración 6: Detalle de los errores antes de la aplicación de la herramienta en la web del Hospital Clínico de Aragón

b. Fase de corrección y resultado

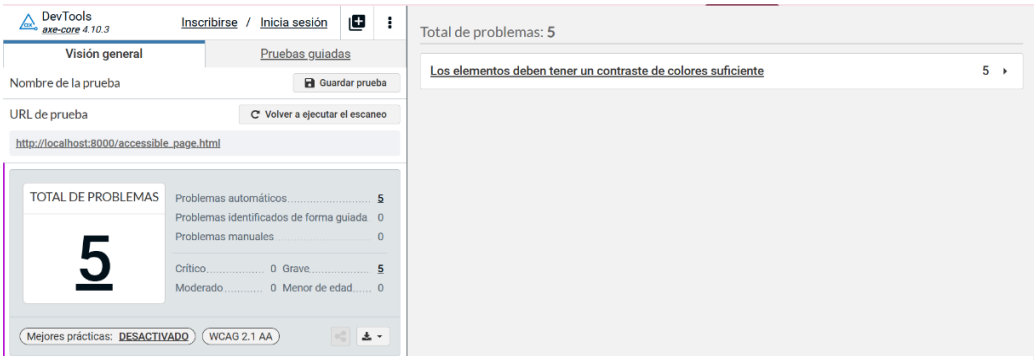

Ilustración 7: Detalle de los errores que persisten después de la aplicación de la herramienta en la web del Hospital Clínico de Aragón

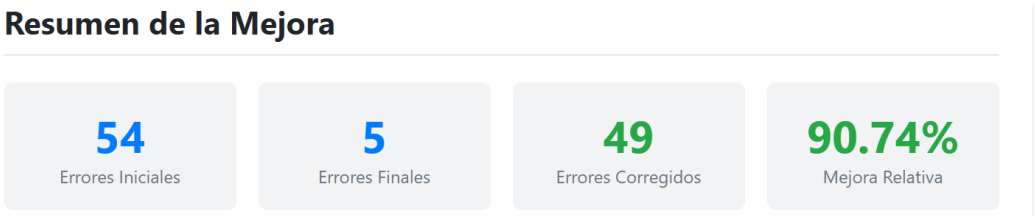

Ilustración 8: Informe generado por la herramienta para la web del Hospital Clínico de Aragón

Como podemos observar en el informe mostrado en la Ilustración 8, se han corregido 49 errores de 54 detectados, lo que representa un 90,74%. Los errores que no han sido corregidos, mostrados en la figura 7, son debido a que no son del nivel A, sino que pertenecen al nivel AA o AAA.

A continuación, en la Ilustración 9, se muestra una imagen de la página web del Hospital Clínico de Aragón antes de aplicar la herramienta, por lo que las imágenes aún no tienen texto descriptivo.

Ilustración 9: Imagen sin descripción textual de la página web del Hospital Clínico de Aragón

Una vez que aplicamos la herramienta, podemos observar en la Ilustración 10 cómo se ha insertado la descripción en la imagen correspondiente.

Ilustración 10: Imagen con texto descriptivo después de aplicar la herramienta en la web del Hospital Clínico de Aragón

6.1.2. Hospital VITHAS de Granada

URL analizada: https://vithas.es/centro/vithas-hospital-granada/

a. Estado inicial

En primer lugar, se ha realizado un análisis de la accesibilidad de esta página web utilizando Axe-core. Antes de aplicar la herramienta, se han detectado los siguientes errores de nivel A de las pautas WCAG 2.2, mostrados en la Ilustración 11:

- Enlaces descriptivos sin `alt`.
- Elementos interactivos sin etiquetas *aria-label*.
- Uso incorrecto de encabezados jerárquicos.

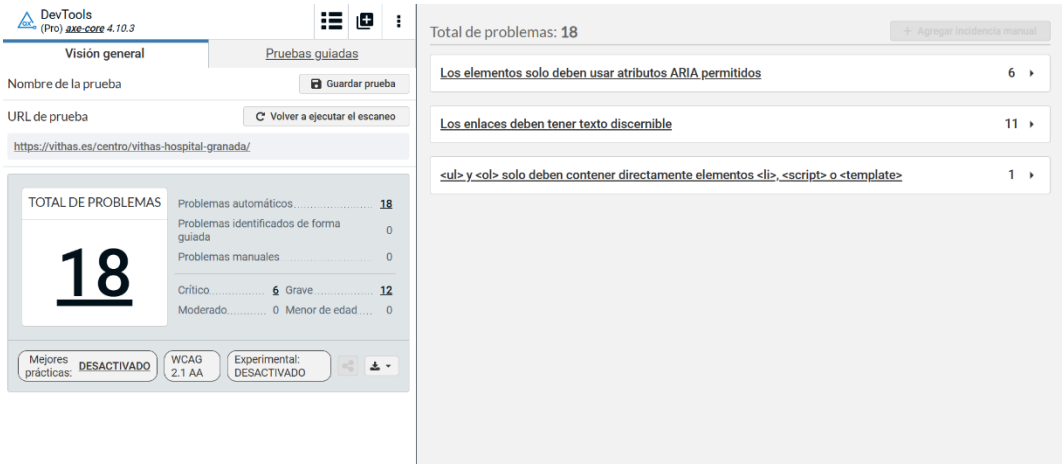

Ilustración 11: Detalle de los errores antes de la aplicación de la herramienta en la web del Hospital Vithas de Granada

b. Fase de corrección y resultado

Ilustración 12: Detalle del análisis posterior a la aplicación de la herramienta en la web del Hospital Vithas de Granada

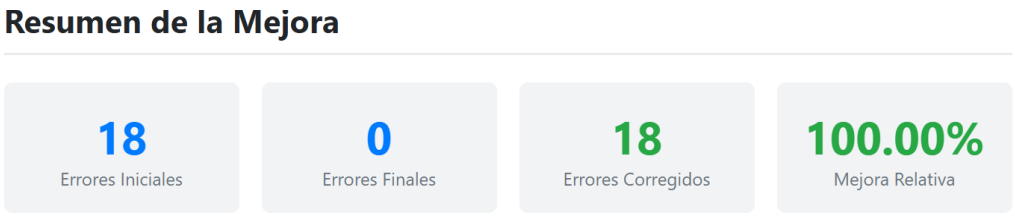

Ilustración 13: Informe generado por la herramienta para la web del Hospital Vithas de Granada

Como vemos en la Ilustración 12, en esta página web se han corregido todos los errores de accesibilidad que se habían encontrado. Esto es debido a que todos pertenecían al nivel A de WCAG 2.2.

El informe mostrado en la Ilustración 13 refleja un 100% de mejora.

6.1.3. Hospital Puerta del Mar de Cádiz

URL analizada: https://hospitalpuertadelmar.com/

a. Estado inicial

En primer lugar, se ha realizado un análisis de la accesibilidad de esta página web utilizando Axe-core. Antes de aplicar la herramienta, se han detectado los siguientes errores de nivel A de las pautas WCAG 2.2, mostrado en la Ilustración 14:

- Imágenes descriptivas sin `alt`.
- Elementos interactivos, como enlaces y botones, sin etiquetas *aria-label*.
- Uso incorrecto de encabezados jerárquicos.
- Ausencia de roles semánticos para las tecnologías asistidas.

- Marcos sin atributo `title`.

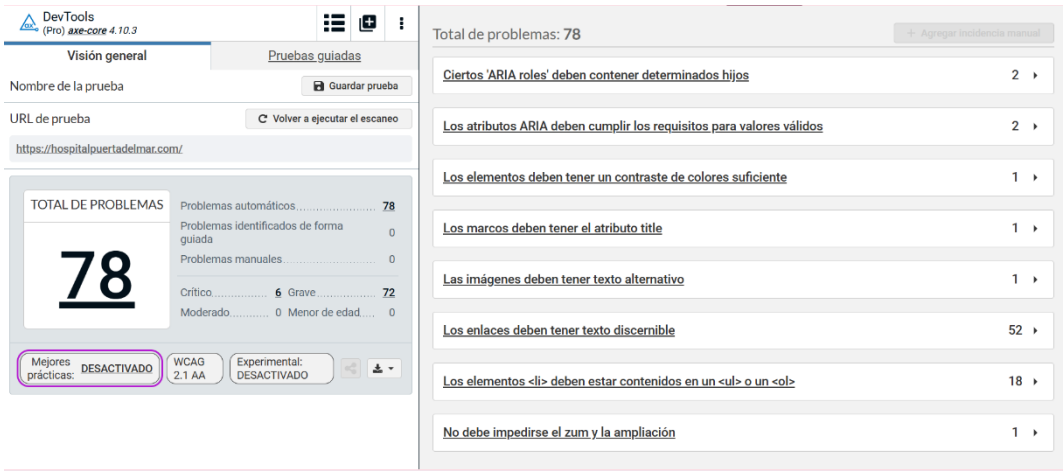

Ilustración 14: Detalle de los errores antes de la aplicación de la herramienta en la web del Hospital Puerta del Mar de Cádiz

b. Fase de corrección y resultado

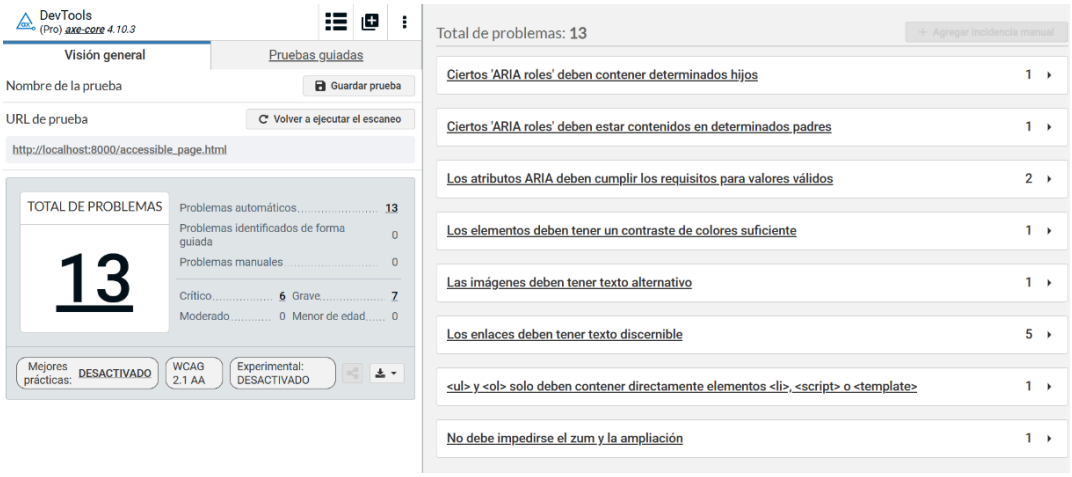

Ilustración 15: Detalle de los errores persistentes después de la aplicación de la herramienta en la web del Hospital Puerta del Mar de Cádiz

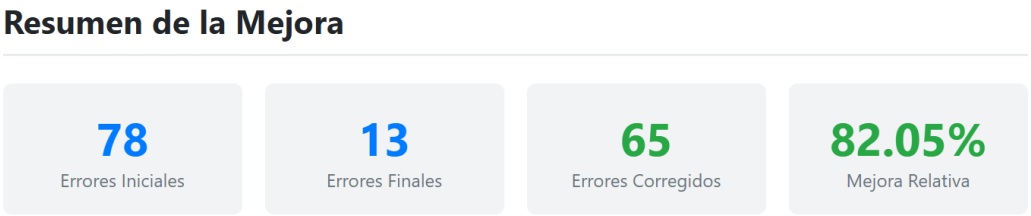

Ilustración 16: Informe generado por la herramienta para la web del Hospital Puerta del Mar

En esta página web se han corregido un total de 65 errores de nivel A, representando un 82% de mejora, como se observa en la Ilustración 16. Los errores no corregidos, mostrados en la Ilustración 15, aunque hay algunos que son de nivel A, son de nivel AA

o AAA. Los que son de nivel A que no se han corregido es debido a que son vídeos y audios, y la herramienta solo está preparada para corregir imágenes.

6.1.4. Hospital Del Mar de Cataluña

URL analizada: https://www.hospitaldelmar.cat/ca/

a. Estado inicial

En primer lugar, se ha realizado un análisis de la accesibilidad de esta página web utilizando Axe-core. Antes de aplicar la herramienta, se han detectado los siguientes errores de nivel A de las pautas WCAG 2.2, mostrados en la Ilustración 17:

- Imágenes descriptivas sin `alt`.
- Enlaces con texto genérico no descriptivo.

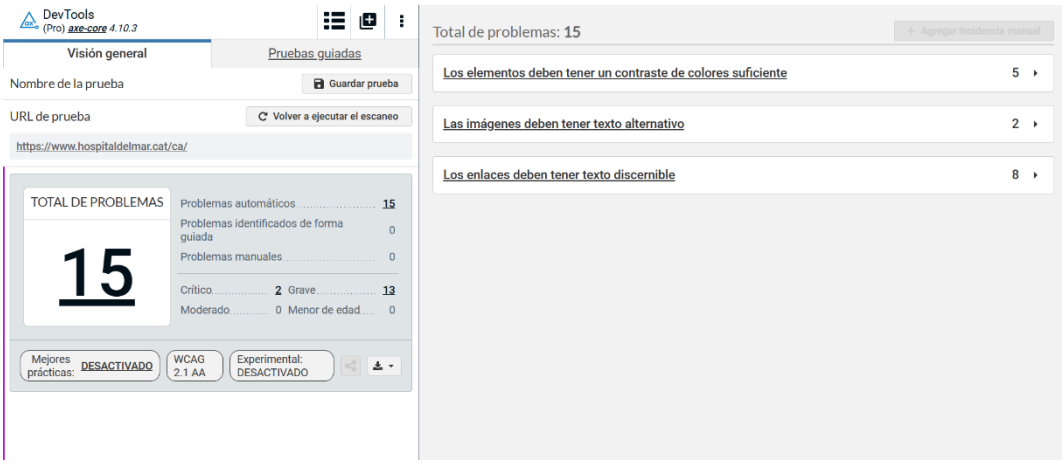

Ilustración 17: Detalle de los errores antes de la aplicación de la herramienta en la web del Hospital del Mar de Cataluña

b. Fase de corrección y resultado

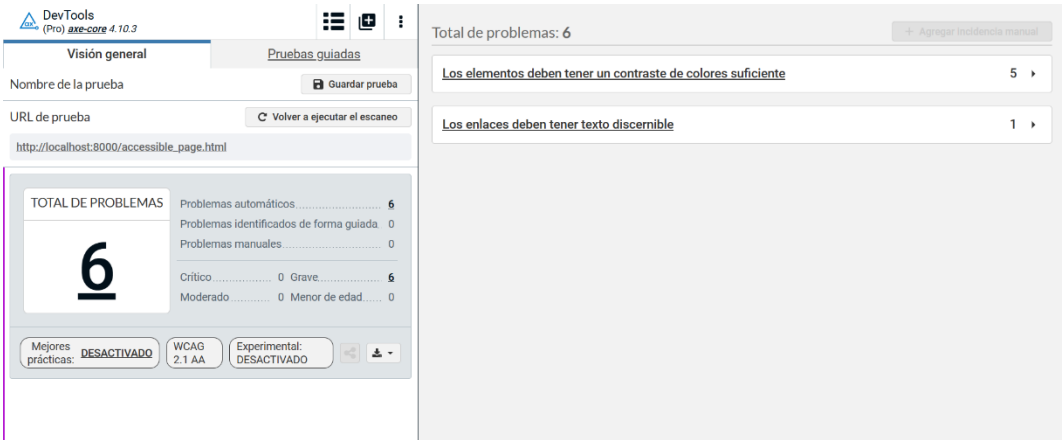

Ilustración 18: Detalle de los errores persistentes después de la aplicación de la herramienta en la web del Hospital del Mar de Cataluña

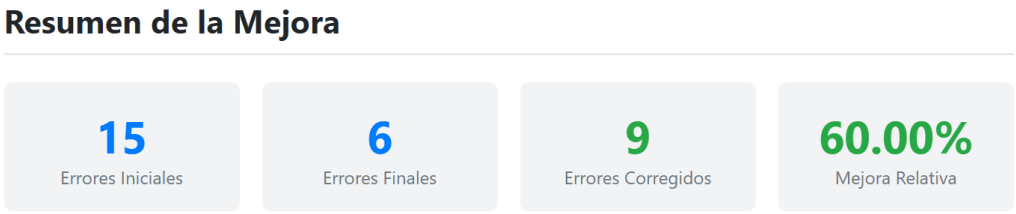

Ilustración 19: Informe generado por la herramienta para la web del Hospital del Mar de Cataluña

Los errores que no se pueden corregir, mostrados en la Ilustración 18, son los pertenecientes al nivel AA. Además, hay un error del nivel A que no se corrige debido a que el enlace pertenece a un vídeo y la herramienta solo está preparada para imágenes.

El informe de la Ilustración 19 muestra un porcentaje de mejora del 60%.

6.1.5. Clínica Dental Your Clinic de Málaga

URL analizada: https://yourclinic.es/

a. Estado inicial

En primer lugar, se ha realizado un análisis de la accesibilidad de esta página web utilizando Axe-core. Antes de aplicar la herramienta, se han detectado los siguientes errores de nivel A de las pautas WCAG 2.2, mostrados en la Ilustración 20:

- Elementos interactivos sin atributos *aria-**.
- Enlaces sin *aria-label*.

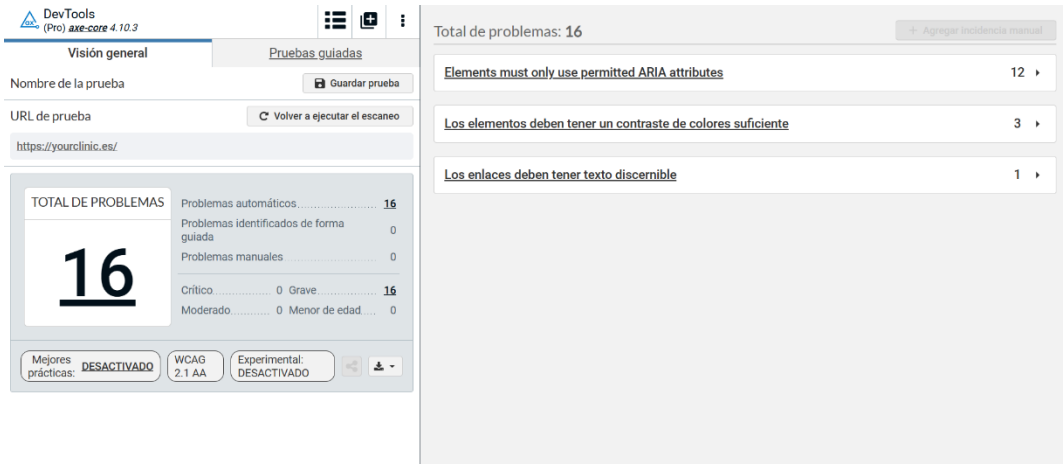

Ilustración 20: Detalle de los errores antes de la aplicación de la herramienta en la web de la Clínica Dental Your Clinic

b. Fase de corrección y resultado

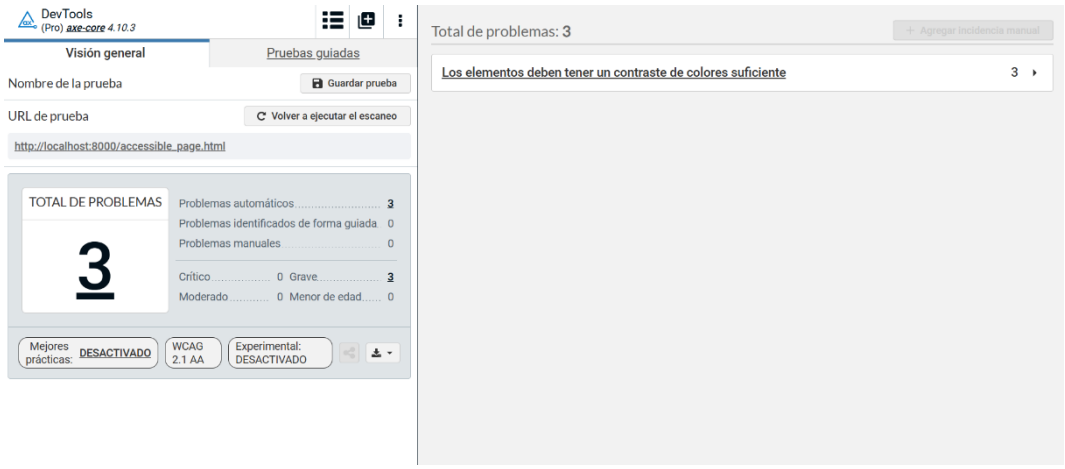

Ilustración 21: Detalle de los errores persistentes después de la aplicación de la herramienta en la web de la Clínica Dental Your Clinic

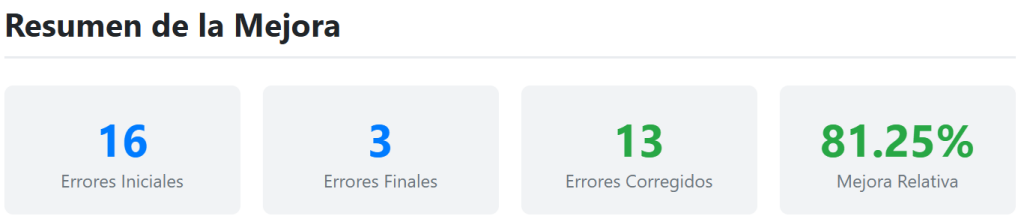

Ilustración 22: Informe generado por la herramienta para la web de la Clínica Dental Your Clinic

En esta página web, la herramienta funciona de manera eficaz, con un 81% de mejora relativa, según el informe de la Ilustración 22. El grupo de errores que no se corrigen, mostrados en la Ilustración 21 pertenecen al nivel AA, por lo que la herramienta no está configurada para corregirlo.

6.1.6. Fundación del Corazón

URL analizada: https://fundaciondelcorazon.com/

a. Estado inicial

En primer lugar, se ha realizado un análisis de la accesibilidad de esta página web utilizando Axe-core. Antes de aplicar la herramienta, se han detectado los siguientes errores de nivel A de las pautas WCAG 2.2, mostrados en la Ilustración 23:

- Roles sin determinados hijos.
- Botones y enlaces con ausencia de `alt`.

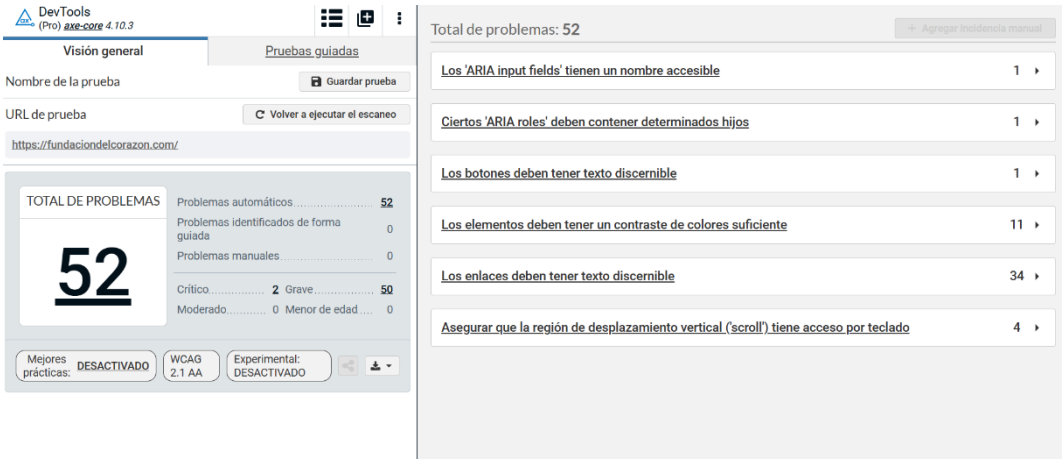

Ilustración 23: Detalle de los errores antes de la aplicación de la herramienta en la web de la Fundación del Corazón

b. Fase de corrección y resultado

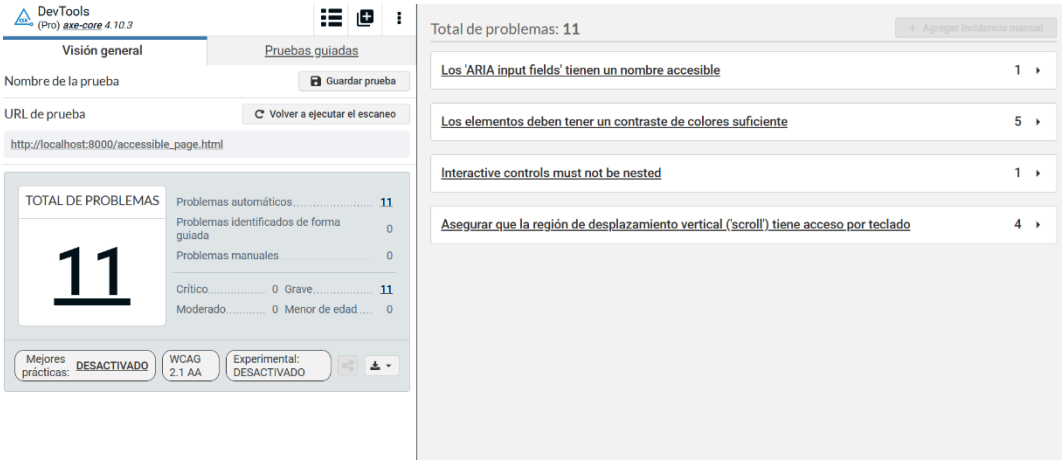

Ilustración 24: Detalle de los errores persistentes de la aplicación de la herramienta en la web de la Fundación del Corazón

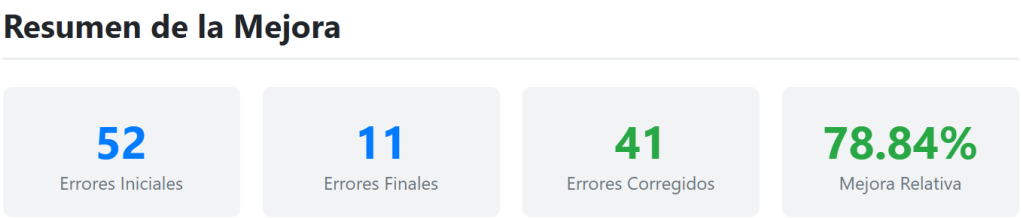

Ilustración 25: Informe generado por la herramienta para la web de la Fundación del Corazón

Tras aplicar la herramienta, vemos en el informe de la Ilustración 25 que se han corregido los 41 errores de nivel A que había. Sin embargo, siguen quedando 11 de nivel AA que la herramienta no es capaz de corregir, reflejados en la Ilustración 24.

Antes de aplicar la herramienta, podemos observar que hay un tipo de error llamado "Los enlaces deben tener texto discernible". En la Ilustración 26, se observa el encabezado de la página web.

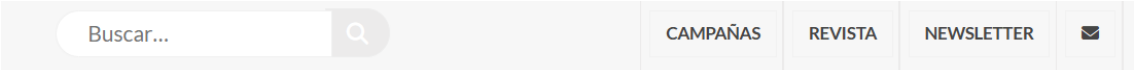

Ilustración 26: Encabezado de la web de la Fundación del Corazón antes de aplicar la herramienta

Después, cuando aplicamos la herramienta, vemos que ese error se ha corregido. En la Ilustración 27, observamos cómo ha añadido el texto "Contacto" a la derecha del icono del sobre.

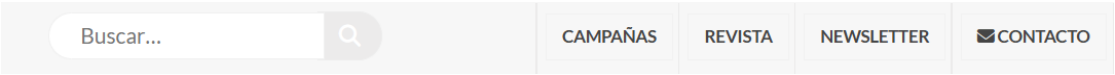

Ilustración 27: Encabezado de la web de la Fundación del Corazón después de aplicar la herramienta

6.1.7. Servicio de Salud de Castilla La Mancha

URL analizada: https://sanidad.castillalamancha.es/

a. Estado inicial

En primer lugar, se ha realizado un análisis de la accesibilidad de esta página web utilizando Axe-core. Antes de aplicar la herramienta, se han detectado los siguientes errores de nivel A de las pautas WCAG 2.2, mostrados en la Ilustración 28:

- Enlaces sin texto descriptivo.

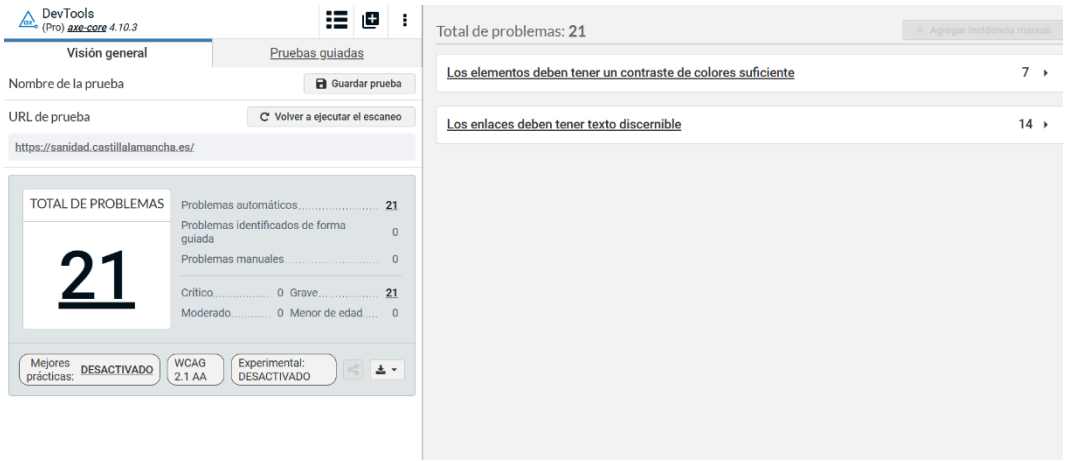

Ilustración 28: Detalle de los errores antes de la aplicación de la herramienta en la web principal sobre la sanidad de Castilla La Mancha

b. Fase de corrección y resultado

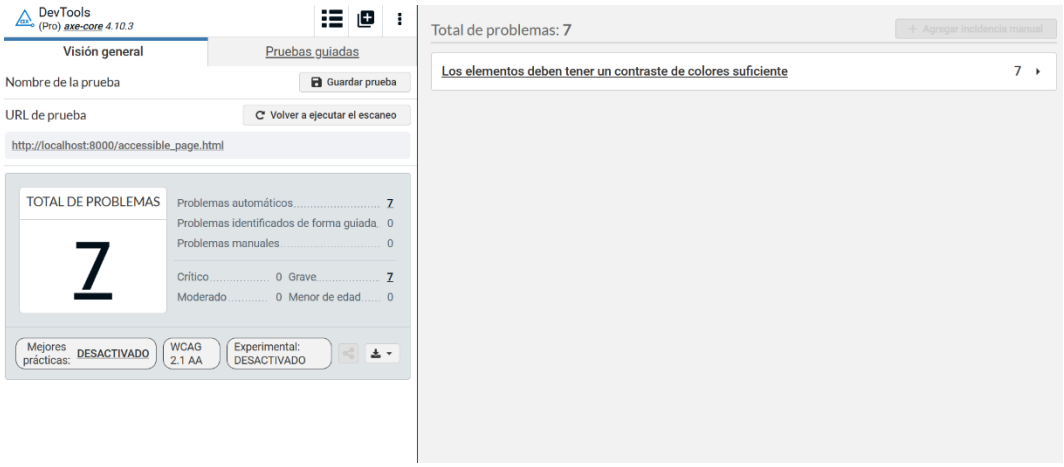

Ilustración 29: Detalle de los errores persistentes después de la aplicación de la herramienta en la web principal sobre la sanidad de Castilla La Mancha

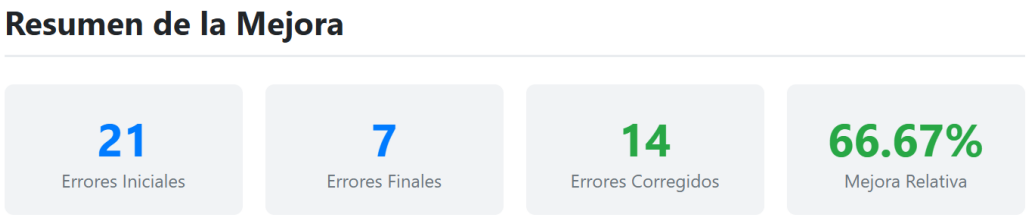

Ilustración 30: Informe generado por la herramienta para la web principal sobre sanidad de Castilla La Mancha

De los 21 errores iniciales, se han corregido los 14 que pertenecen al nivel A, como se observa en el informe de mejora de la Ilustración 30. Los errores pertenecientes a otros niveles, reflejados en la Ilustración 29, no se pueden corregir con la herramienta.

6.1.8. Área de Gestión Sanitaria de Serranía

URL analizada: https://areasanitariaserrania.es/

a. Estado inicial

En primer lugar, se ha realizado un análisis de la accesibilidad de esta página web utilizando Axe-core. Antes de aplicar la herramienta, se han detectado los siguientes errores de nivel A de las pautas WCAG 2.2, mostrados en la figura 31:

- Imágenes descriptivas sin `alt`.
- Enlaces sin distinción con el texto adyacente.
- Enlaces sin texto alternativo.

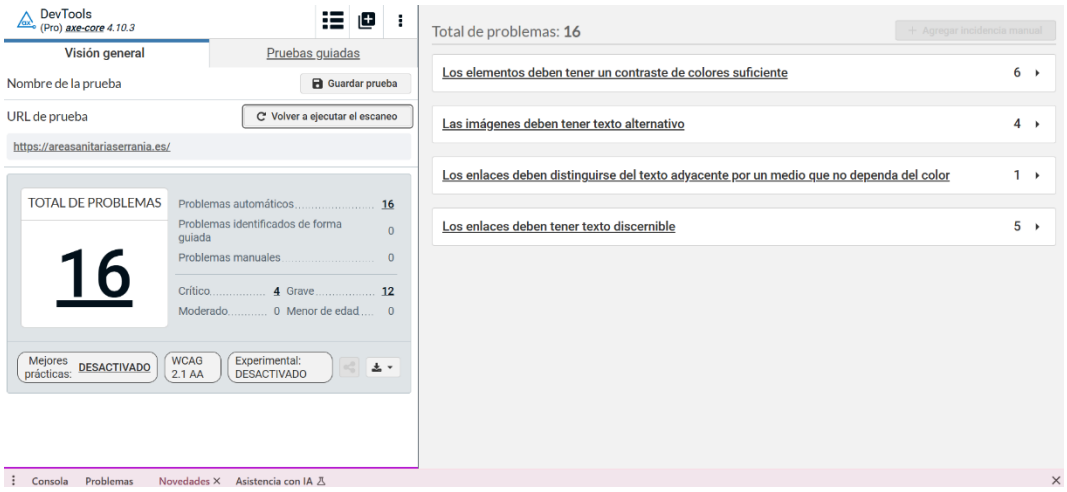

Ilustración 31: Detalle de los errores antes de la aplicación de la herramienta en la web del Área de Gestión Sanitaria de Serranía

b. Fase de corrección y resultado

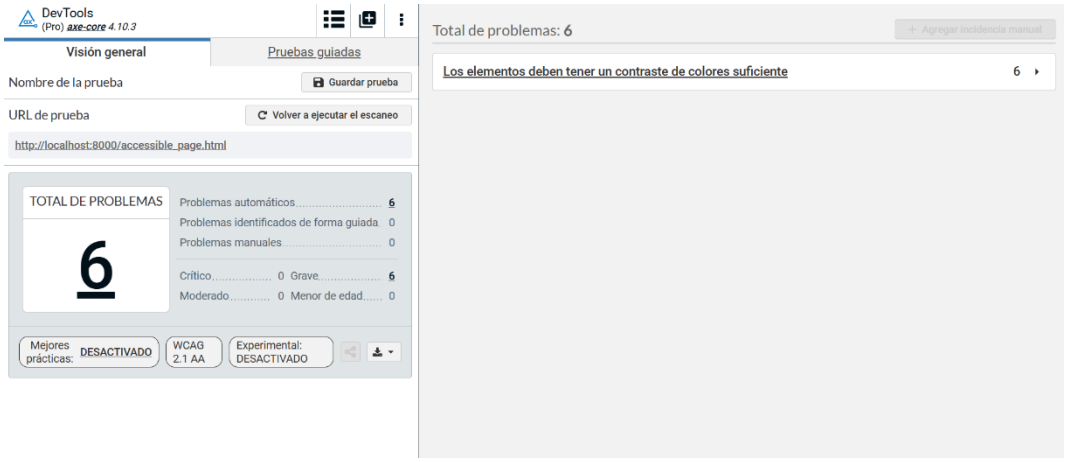

Ilustración 32: Detalle de los errores persistentes después de la aplicación de la herramienta en la web del Área de Gestión Sanitaria de Serranía

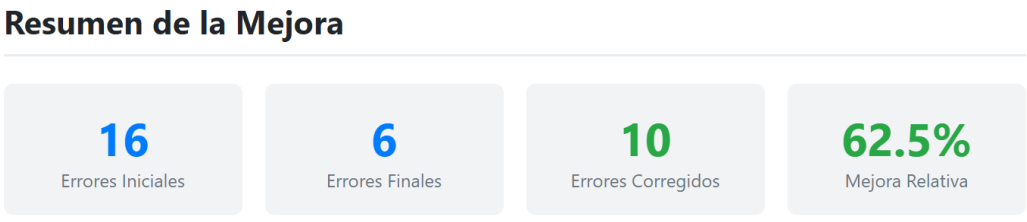

Ilustración 33: Informe generado por la herramienta para la web del Área de Gestión Sanitaria de Serranía

Como podemos observar en la Ilustración 32, persisten 6 errores, ya que son de nivel AA y la herramienta no está preparada para corregirlos. Aun así, en la Ilustración 33 se muestra un porcentaje de mejora del 62.5%.

6.2. Resumen final de resultados

En esta sección se muestra una tabla comparativa con los resultados obtenidos antes y después de aplicar la herramienta, además de un gráfico para comprender mejor los resultados.

Tabla 3: Tabla comparativa de resultados (Fuente: elaboración propia a partir de análisis de páginas web)

Sitio web	Errores iniciales	Errores finales	Errores corregidos	Mejora (%)
Hospital Clínico de Aragón	54	5	49	91
Hospital VITHAS (Granada)	18	0	18	100
Hospital Puerta del Mar (Cádiz)	78	13	65	82
Hospital del Mar (Cataluña)	15	6	9	60
Clínica Dental Your Clinic	16	3	13	81
Fundación Española del Corazón	52	11	41	79
Servicio de Salud de Castilla La Mancha	21	7	14	67
Área Sanitaria Serranía	16	6	10	63

6.3. Análisis de resultados

Tras observar la tabla comparativa y el gráfico, podemos concluir que existe una mejora significativa de la página web tras aplicarle la herramienta. En promedio, la herramienta ha logrado corregir un 78,88% de los errores detectados.

El caso más favorable ha sido el del Hospital VITHAS de Granada, con un 100% de mejora relativa, mientras que el más bajo ha sido el Hospital del Mar de Cataluña, con tan solo un 60%.

Este porcentaje de mejora nos indica el gran potencial de la solución propuesta para automatizar y mejorar, al menos de manera parcial, la accesibilidad web, especialmente en sitios complejos donde se puede requerir muchos recursos humanos y tiempo.

Además, en los sitios donde no se han corregido el 100% de los errores, vemos cómo los errores restantes están ligados con otros niveles de accesibilidad según las pautas WCAG 2.2. La herramienta solo está preparada inicialmente para corregir los de nivel A, por lo que, cuando encuentre errores de otro tipo de nivel, no será capaz de corregirlos y los omitirá. También, estos errores son aspectos más complejos y se necesitaría la intervención humana.

Esto muestra las limitaciones actuales de este tipo de soluciones automáticas, justificando la necesidad de la revisión humana complementaria para alcanzar el 100% de corrección de errores según la normativa a seguir.

En conclusión, los datos muestran una mejora significativa de la accesibilidad tras la aplicación de la herramienta. En un gran número de páginas web se han corregido más del 80%, destacando un caso con el 100% de corrección.

Esta variación de porcentajes puede deberse a la complejidad del HTML de cada web o elementos que requieren intervención humana.

7

Conclusiones y líneas futuras

7.1. Conclusiones generales

El presente Trabajo de Fin de Grado ha tenido como objetivo el desarrollo de una herramienta capaz de analizar y mejorar la accesibilidad de las páginas web de manera automática y eficaz, aplicando técnicas de inteligencia artificial y análisis estático del código. Se ha utilizado como base la normativa WCAG 2.2 de nivel A.

A lo largo de este proyecto, se ha diseñado e implementado una solución basada en Selenium, Axe-core, OpenAI y BeautifulSoup, entre otras, permitiendo el escaneo de la página web, la detección de errores, la generación de descripciones de las imágenes, la corrección de errores de accesibilidad y la generación de una nueva versión de la web.

Los resultados obtenidos tras aplicar la herramienta en ocho páginas web diferentes han sido muy positivos. En todos los casos se ha notado una mejora, con porcentajes que superan el 80% e incluso llegando a alcanzar el 100%.

Además, la herramienta también incorpora un sistema de caché eficiente, manejo de errores robusto y una estructura modular que permite su extensión en un futuro. Esto la convierte en una herramienta sólida para posibles aplicaciones reales e investigación continuada.

7.2. Limitaciones de la herramienta

A pesar de que se han obtenido resultados muy buenos, también hemos encontrado limitaciones.

No todos los errores que se encuentran son posibles de corregir automáticamente. Algunos necesitan intervención humana o un rediseño del contenido.

También, el análisis está enfocado en la estructura HTML, por lo que el contenido generado dinámicamente no puede repararse.

Además, la herramienta depende de la calidad del HTML, y de una buena conexión a internet y el uso de la API de OpenAI tiene un coste asociado, por lo que podría ser una barrera para los proyectos que no tienen presupuesto o es bajo.

7.3. Aportes del proyecto

Este proyecto ha contribuido en la intersección entre ingeniería del software, accesibilidad web y el uso de la inteligencia artificial de manera positiva. Uno de los principales aportes ha sido el diseño y desarrollo de una herramienta capaz de corregir los errores que presentan diferentes páginas web conforme al criterio WCAG 2.2 nivel A. Esto no solo permite la detección de problemas, sino una generación automática de propuestas de corrección, facilitando el cumplimiento de la normativa por parte de desarrolladores que no son expertos en accesibilidad.

Desde el punto de vista técnico, se combinan tecnologías de automatización web como Selenium, bibliotecas especializadas en análisis de accesibilidad, y modelos de lenguaje grandes como los de OpenAI para la generación de contenido textual accesible.

Otra aportación relevante es la incorporación de un sistema que genera un informe visual y claro, permitiendo entender el antes y el después de la corrección de los errores. Además, el uso de plantillas HTML facilita su reutilización sin requerir un diseño manual.

Este Trabajo de Fin de Grado también presenta una validación sólida y empírica, ya que se ha probado en ocho páginas web, mostrando una mejora significativa de cada una, corrigiendo en la mayoría de los casos más del 80% de problemas encontrados. El

enfoque práctico contribuye al valor aplicado del trabajo y se aleja de propuestas que son teóricas exclusivamente.

Para finalizar, desde la perspectiva metodológica, se propone una arquitectura modular y escalable, facilitando su evolución futura. Gracias a su estructura clara y separación por módulos, puede ser extendida para incorporar nuevos tipos de análisis y soportar otros niveles de accesibilidad.

En conclusión, este proyecto ofrece una solución funcional, con una base teórica sólida, una implementación técnica que proporciona buenos resultados y una validación práctica para comprobar su eficacia.

7.4. Líneas futuras

Entre las posibles ampliaciones de la herramienta se encuentra la incorporación de los niveles AA y AAA de accesibilidad de WCAG 2.2, permitiendo abordar problemas más complejos y cubrir las necesidades de los usuarios al completo.

Otra línea de mejora sería la integración de validadores gráficos, de modo que el análisis y corrección se puedan visualizar fácilmente por desarrollador no técnicos.

En cuanto a inteligencia artificial, se podría reemplazar la API de OpenAI por modelos de lenguaje *Open Source* ejecutados localmente, lo que permitiría el uso de la herramienta de forma *offline*, sin depender de una buena conexión a internet.

Otra línea interesante sería incorporar modelos de inteligencia artificial multimodal, capaces de analizar y generar descripciones en audios y vídeos, y no limitarse solo a la descripción de imágenes.

Además, integrar la herramienta en sistemas de integración continua para facilitar el uso automático durante su desarrollo web, asegurando que cada actualización mantenga la accesibilidad como requisito fundamental.

Otra mejora importante sería la incorporación del soporte multilingüe i18n, permitiendo aplicar las descripciones, correcciones e informes en diferentes idiomas.

Finalmente, la validación del sistema con usuarios con discapacidad reales permitiría evaluar la utilidad de las correcciones y garantizar que el impacto sea significativo en la experiencia del usuario.

Referencias

[1] Abou-Zahra, S., Brewer, J., & Cooper, M. (2018). *Artificial intelligence (AI) for web accessibility: Is conformance evaluation a way forward?* Proceedings of the 15th International Web for All Conference. https://doi.org/10.1145/3192714.3192823

[2] Acosta-Vargas, P., Acosta-Vargas, G., Salvador-Acosta, B., & Jadán-Guerrero, J. (2024). *Addressing web accessibility challenges with generative artificial intelligence tools for inclusive education*. 2024 Tenth International Conference on eDemocracy & eGovernment (ICEDEG) (pp. 1-7). IEEE. https://doi.ieeecomputersociety.org/10.1109/ICEDEG61611.2024.10702085

[3] Andrés, M. P. J., Ginés, L. R., & Informática, G. E. I. (2023). *Interfaz de usuario web para la accesibilidad de los diferentes desarrollos basados en optimización e inteligencia artificial* [Trabajo de fin de grado, Universidad de la Laguna]. RIULL. https://riull.ull.es/xmlui/handle/915/33851

[4] Bhagat, S., Joshi, P., Agarwal, A., & Gupta, S. (2024). *Accessibility evaluation of major assistive mobile applications available for the visually impaired*. arXiv. https://arxiv.org/abs/2407.17496

[5] Botpress. (2025). *¿Qué es el procesamiento del lenguaje natural (PLN) en la IA?* https://botpress.com/es/blog/natural-language-processing-nlp

[6] Chemnad, K., & Othman, A. (2024). *Digital accessibility in the era of artificial intelligence: Bibliometric analysis and systematic review*. Frontiers in Artificial Intelligence, 7, 1349668. https://doi.org/10.3389/frai.2024.1349668

[9] Décima, J. M. (2018). *La inteligencia artificial como habilitador de la inclusión digital* [Trabajo de especialización, Universidad Nacional de La Plata]. Repositorio Digital UNLP. https://sedici.unlp.edu.ar/handle/10915/72014

[10] De Souza, E. R. (2023). *Evaluación de la accesibilidad web: oportunidades con inteligencia artificial y aprendizaje automático*. Dialnet. https://dialnet.unirioja.es/servlet/articulo?codigo=9183592

[11] Femcet. (2023). *Clasificación de los tipos de discapacidad según la OMS*. https://femcet.com/es/que-tipos-de-discapacidad-existen/

[12] Figueira, I., Cisneros, J. M., Leachman, M., Peña, E. D., & Branham, S. M. (2024). *Informing accessible design of AI literacy apps through practices of blind parents reading with sighted children*. The 26th International ACM SIGACCESS Conference on Computers and Accessibility (pp. 1-5). ACM. https://doi.org/10.1145/3663548.3675650

[13] Frug, S., & Bruce, T. (s.f.). *Artificial intelligence and accessibility for administrative applications. CEUR Workshop Proceedings*. https://ceur-ws.org/Vol-2471/paper3.pdf

[14] Fundación Adecco. (2022). *¿Qué es la discapacidad? Evolución histórica y cultural*. https://fundacionadecco.org/blog/que-es-la-discapacidad-evolucion-historica/

[15] Grupo Social ONCE. (2024). *Tipos de discapacidad: Conócelos todos*. https://gruposocialonce.com/b/tipos-discapacidad

[16] Iberaval. (2024). *¿Qué son los LLMs?* https://www.iberaval.es/blog/que-son-los-llms/

[17] Iberdrola. (2025). *¿Qué es la Inteligencia Artificial?* https://www.iberdrola.com/innovacion/que-es-inteligencia-artificial

[18] KSchool. (2023). *La importancia del Natural Language Processing en la IA*. https://kschool.com/blog/big-data/importancia-del-natural-language-processing-en-la-ia/

[19] Kubullek, A.-K., & Dogangün, A. (2023). *Creating accessibility 2.0 with artificial intelligence.* Mensch und Computer 2023 (pp. 437-441). ACM. https://doi.org/10.1145/3603555.360854

[20] Mailchimp. (2023). *8 herramientas para comprobar la accesibilidad de una web*. https://mailchimp.com/es/resources/test-website-accessibility/

[21] Miesenberger, K., Edler, C., Heumader, P., & Petz, A. (2019). *Tools and applications for cognitive accessibility*. Human–Computer Interaction Series (pp. 523-546). Springer. https://doi.org/10.1007/978-1-4471-7440-0_28

[22] Miranda, M. G., Martin, A. E., & Gaetan, G. (2014). *Mejora de la accesibilidad web mediante el uso de agentes inteligentes*. Informes Científicos - Técnicos UNPA, 5(2), 133-160. https://doi.org/10.22305/ict-unpa.v5i2.75

[23] Morrison, C., Cutrell, E., Dhareshwar, A., Doherty, K., Thieme, A., & Taylor, A. (2017). *Imagining artificial intelligence applications with people with visual disabilities using tactile ideation*. Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility (pp. 81-90). ACM. https://doi.org/10.1145/3132525.3132530

[24] Mowar, P., Peng, Y., Wu, J., Steinfeld, A., & Bigham, J. P. (2025, 15 de febrero). *CodeA11y: Making AI coding assistants useful for accessible web development*. arXiv. https://arxiv.org/abs/2502.10884

[25] Organización Mundial de la Salud. (2023). *Discapacidad*. https://www.who.int/es/news-room/fact-sheets/detail/disability-and-health

[26] Ozarkar, S., Chetwani, R., Devare, S., Haryani, S., & Giri, N. (2020). *AI for accessibility: Virtual assistant for hearing impaired*. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-7). IEEE. https://doi.org/10.1109/ICCCNT49239.2020.9225392

[27] Parlamento Europeo. (2021). *¿Qué es la inteligencia artificial y cómo se usa?* https://www.europarl.europa.eu/topics/es/article/20200827STO85804/que-es-la-inteligencia-artificial-y-como-se-usa

[28] Smowl Tech. (2024). *WCAG: qué son y para qué sirven*. https://smowl.net/es/blog/wcag/

[29] Tseng, C.-E., Jung, S. H., Elglaly, Y. N., Liu, Y., & Ludi, S. (2022*). Exploration on integrating accessibility into an AI course*. Proceedings of the 53rd ACM Technical

Symposium on Computer Science Education (pp. 683-689). ACM. https://doi.org/10.1145/3478431.3499399

[30] Web Accessibility Initiative. (2023). *Sumario de WCAG 2*. https://www.w3.org/WAI/standards-guidelines/wcag/es

[31] World Wide Web Consortium. (2023). *Understanding conformance. Web Accessibility Initiative.* https://www.w3.org/WAI/WCAG21/Understanding/conformance#levels

[32] World Wide Web Consortium. (2025). *Introduction to Understanding WCAG 2.2.* Web Accessibility Initiative. https://www.w3.org/WAI/WCAG22/Understanding/intro#understanding-the-four-principles-of-accessibility

[33] Wu, S., Wieland, J., Farivar, O., & Schiller, J. (2017). *Automatic alt-text: Computer-generated image descriptions for blind users on a social network service*. Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (pp. 1180-1192). ACM. https://doi.org/10.1145/2998181.2998364

[34] Wu, S., Xiong, Y., Cui, Y., Wu, H., Chen, C., Yuan, Y., Huang, L., Liu, X., Kuo, T., Guan, N., & Xue, C. J. (2024, 18 de julio). *Retrieval-augmented generation for natural language processing: A survey*. arXiv. https://arxiv.org/abs/2407.13193

Anexo A

Manual del Usuario

Este manual de usuario ha sido elaborado para orientar de manera clara y accesible a cualquier persona interesada en emplear el sistema de mejora automática de accesibilidad web, una herramienta creada para detectar y solucionar fallos de accesibilidad en sitios web, permitiendo su utilización por individuos con discapacidad. Se describen todos los procedimientos necesarios para la instalación y se incluyen las directrices sobre cómo establecer adecuadamente el entorno, emplear los parámetros disponibles, entender el proceso de análisis de un sitio web desde la identificación temprana de fallos hasta la creación del informe comparativo y mostrar la página corregida a través de un servidor local.

Además, se tratan posibles inconvenientes habituales durante la ejecución, proporcionando soluciones efectivas para optimizar la eficiencia del análisis.

Este manual no solo ofrece las instrucciones técnicas requeridas, sino que también actúa como una guía integral de buenas prácticas, mantenimiento y uso eficiente del sistema.

1. Información para usuarios

Todos los *scripts* están elaborados en lenguaje Python y separados por su funcionalidad. Por eso, la clave API y las constantes deben almacenarse en los archivos `.env` y `constants.py`, respectivamente.

Además, la caché de las imágenes está centralizada en una carpeta, mejorando su eficiencia y permitiendo la reutilización de aquellas que se encuentran descargadas.

También, los informes se generan en HTML, pero están configurados para adaptarse a otros formatos, como PDF.

Por último, las llamadas a OpenAI se hacen con un control de los errores y se pueden sustituir por otros modelos de lenguaje si se cree conveniente.

2. Requisitos del sistema

2.1. Hardware mínimo recomendado

- Procesador Intel Core i5 o equivalente.
- Memoria RAM: 4 GB.
- Almacenamiento libre: 2 GB.
- Conexión a Internet activa.

2.2. Software mínimo requerido

- Python 3.10 o superior.
- Sistema operativo: Linux, Windows o macOS.
- Google Chrome.
- Chromedriver compatible con la versión de Google Chrome.

3. Instalación paso a paso

3.1. Clonar el repositorio y crear un entorno virtual

En la terminal de PyCharm, ejecuta:

```
git clone https://github.com/CarlaFernandez12/accessibility.git
```

```
cd <NOMBRE-DEL-DIRECTORIO>
```

En el directorio del proyecto, se crea y activa un entorno virtual. Esto es opcional pero recomendado:

```
# Si usas Windows, escribe esto en la terminal:

python -m venv venv

venv\Scripts\activate

# Si usas Mac o Linux, escribe esto en su lugar:

python -m venv venv

source venv/bin/activate
```

Y luego , instalar las dependencias necesarias:

```
pip install -r requirements.txt
```

3.2. Configuración de OpenAI

La herramienta requiere una clave API de OpenAI. Se recomienda por seguridad guardar en un archivo `.env` en el directorio:

```
OPENAI_API_KEY="tu-clave-api-aquí"
```

3.3 Ejecución

Para ejecutar el flujo completo de la herramienta, se ejecuta el script principal, poniendo en la terminal del entorno:

```
python main.py --url https://ejemplo.com/
```

4. Proceso de análisis

Primero, se realiza un análisis preliminar con la biblioteca Axe-core , que identifica los fallos de accesibilidad conforme a las directrices WCAG 2.2 de nivel A. Posteriormente, se lleva a cabo la recolección de imágenes, donde se descargan todas las encontradas en el sitio web para que luego puedan ser descritas por la API de OpenAI. Después, se realiza un análisis estático del HTML, lo que permite revisar el código fuente en busca de estructuras incorrectas sin tener que interactuar con la página web.

Durante la etapa de corrección con IA, el sistema genera soluciones para cada error identificado utilizando las indicaciones elaboradas específicamente para cada uno de ellos, aprovechando la habilidad contextual del modelo GPT-4o.

Una vez se han obtenido las correcciones, se realiza la reconstrucción del HTML, modificando el DOM directamente con BeautifulSoup, sustituyendo las partes defectuosas por sus correcciones. Finalmente, se lleva a cabo una valoración definitiva a través de un segundo análisis con Axe-core para verificar qué errores se han solucionado y medir la mejora lograda.

La duración prevista del procedimiento varía según el tamaño del sitio web: de 2 a 4 minutos para páginas web pequeñas, de 5 a 10 minutos para páginas web medianas y hasta 30 minutos para grandes sitios web con contenido abundante.

5. Archivos generados

Tras el flujo completo del sistema, se obtienen las siguientes carpetas y archivos:

```
results/
└── nombre_de_la_web/
      └── fecha_y_hora/
            ├── initial_report.json
            ├── accessible_page.html
            ├── final_report.json
            └── comparison_report.html
```

6. Apertura de servidor local

Una vez finaliza la ejecución del programa, aparece un mensaje en consola:

```
¿Deseas previsualizar la página corregida en el navegador? (y/n)
```

En caso de que la respuesta sea "y", se abre un servidor local con la página web accesible. En caso contrario, no se abre la página web corregida.

7. Finalización de la ejecución

Para finalizar el flujo del programa, es necesario pulsar en el teclado Ctrl + C.

8. Actualización del sistema

Para actualizar el sistema, se ejecuta el siguiente comando:

```
pip install -r requirements.txt
```

9. Depuración y mantenimiento

Esta sección se enfoca en reconocer y registrar los problemas más comunes que pueden surgir durante el uso de la herramienta, junto con sus posibles orígenes y las soluciones sugeridas.

Mediante la Tabla 4, se permite la solución ágil de fallos vinculados a diferentes elementos del sistema, como la compatibilidad y actualización del navegador automático, los permisos de lectura y escritura de archivos para la adecuada creación de informes o la conexión y configuración de la API de OpenAI. Esto es clave para garantizar un mantenimiento efectivo y una continuada apropiada del proyecto, permitiendo a desarrolladores y usuarios identificar y soluciones problemas sin requerir una depuración completa del código en cada caso. De esta manera, se asegura la consistencia y confianza de la herramienta a lo largo del tiempo.

Tabla 4: Depuración (Fuente: elaboración propia)

Problema	**Posible causa**	**Solución**
El navegador no abre	ChromeDriver desactualizado	Actualizar el navegador
No se generan informes correctamente	Error al escribir archivos	Comprobar los permisos de lectura
Descripciones vacías	Problemas con OpenAI	Revisar archivo `.env` y conexión a Internet
Tiempo de espera insuficiente	Páginas con mucho contenido dinámico	Aumentar *sleep* en webdriver_setup.py

Anexo B

Mapeo de errores de accesibilidad y soluciones basadas en IA

Este anexo presenta un catálogo que relaciona los criterios de accesibilidad establecidos en el nivel A de las pautas WCAG 2.2 con soluciones concretas obtenidas a partir de artículos científicos y aplicaciones existentes. El objetivo principal de crear este catálogo es comprender cómo cada uno de los 31 criterios puede ser abordado técnicamente, identificando los errores más comunes y utilizar estrategias actuales para su detección y corrección.

Sin embargo, no todos los criterios han podido ser relacionados con un artículo o herramienta que solucione el problema, por lo que no están incluidos en el catálogo.

Tabla 5: Catálogo WCAG 2.2 nivel A

Criterio	Artículo	Problema	Solución	Aplicaciones
1.1.1	**[12] Informing Accessible Design of AI Literacy Apps through Practices of Blind Parents Reading with Sighted Children** **[2] Addressing Web Accessibility Challenges with Generative Artificial Intelligence Tools for Inclusive Education** **[6] Digital accessibility in the era of artificial intelligence—Bibliometric analysis and systematic review** **[4] Accessibility evaluation of major assistive mobile applications available for the visually impaired** **[29] Exploration on Integrating Accessibility into an AI Course** **[26] AI for Accessibility: Virtual Assistant for Hearing Impaired** **[21] Tools and Applications for Cognitive Accessibility** **[13] Artificial Intelligence and Accessibility for Administrative Applications** **[1] Artificial Intelligence (AI) for Web Accessibility: Is Conformance Evaluation a Way Forward?** **[23] Imagining Artificial Intelligence Applications with People with Visual Disabilities using Tactile Ideation** **[33] Automatic Alt-text: Computer-generated Image Descriptions for Blind Users on a Social Network Service**	1. Las aplicaciones de alfabetización infantil suelen incluir imágenes y elementos gráficos sin descripciones textuales, dificultando que los padres ciegos participen plenamente en la lectura con sus hijos 2. Las aplicaciones de IA suelen generar contenido visual sin proporcionar descripciones textuales, lo que dificulta el acceso para personas con discapacidades visuales 3. Imágenes sin texto alternativo (alt text) no pueden ser interpretadas por usuarios con discapacidad visual 4. Imágenes e iconos sin descripciones accesibles 5. Desigualdad en el acceso a contenido visual para estudiantes con discapacidades visuales 6. Falta de accesibilidad para personas	**Solución 1:** incorporar descripciones textuales para todos los elementos visuales en las aplicaciones de lectura, permitiendo que los lectores de pantalla transmitan la información visual a los padres ciegos **Solución 2:** implementar herramientas que generen automáticamente descripciones textuales para las imágenes creadas por IA, permitiendo que los lectores de pantalla transmitan esta información a los usuarios con discapacidad visual **Solución 3**: uso de IA para generar automáticamente descripciones alternativas para imágenes mediante reconocimiento de objetos o análisis de contexto visual **Solución 4:** añadir etiquetas accesibles (contentDescription, accessibilityLabel) para todos los elementos **Solución 5:** proponer el uso de texto alternativo para imágenes, diagramas y otros elementos	1. Sonar Glass-Artificial Vision 2. AI Based Automated Image Caption Tool 3. Image/Video Summarization (Android)

		con discapacidad auditiva en contenidos visuales o auditivos 7. Personas con dificultades cognitivas pueden no interpretar correctamente elementos visuales 8. Las imágenes sin texto alternativo impiden a los usuarios con discapacidades visuales comprender el contenido visual 9. Muchos sitios web no cumplen con las pautas de accesibilidad, y la IA puede detectarlos 10. Dificultad de las personas con discapacidades visuales para percibir imágenes sin texto alternativo 11. Las imágenes sin descripciones alternas son inaccesibles para usuarios ciegos y muchas redes sociales no tienen implementadas descripciones automáticas	visuales en los materiales del curso **Solución 6:** el asistente virtual puede describir o traducir contenido visual y auditivo a texto, haciéndolo accesible para personas con pérdida auditiva **Solución 7:** herramientas que proporcionan alternativas textuales para imágenes y otros elementos visuales, mejorando la comprensión de contenido **Solución 8:** la IA puede generar descripciones automáticas para imágenes, proporcionando texto alternativo adecuado **Solución 9:** la IA genera descripciones automáticas para imágenes sin texto alternativo **Solución 10:** uso de ideación táctil para representar imágenes, permitiendo que los usuarios con discapacidad visual las perciban mediante el tacto **Solución 11:** generación automática de descripciones (alt-text) para las imágenes mediante inteligencia artificial	

1.2.1	**[6] Digital accessibility in the era of artificial intelligence—Bibliometric analysis and systematic review** **[26] AI for Accessibility: Virtual Assistant for Hearing Impaired** **[1] Artificial Intelligence (AI) for Web Accessibility: Is Conformance Evaluation a Way Forward?**	1. El contenido sólo auditivo o solo visual no es accesible si no hay alternativas textuales 2. Inaccesibilidad de contenido multimedia que depende exclusivamente de sonido 3. Incompatibilidad con tecnologías de asistencia como los lectores de pantalla	**Solución 1:** la IA puede generar transcripciones para audios y descripciones para vídeos sin audio mediante análisis automático del contenido **Solución 2:** ofrece subtítulos o transcripciones de contenido auditivo, permitiendo que las personas con discapacidad auditiva comprendan el contenido multimedia **Solución 3:** la IA evalúa la compatibilidad con tecnologías de asistencia (ej. lectores de pantalla)	
1.2.2	**[6] Digital accessibility in the era of artificial intelligence—Bibliometric analysis and systematic review**	1. Los vídeos sin subtítulos excluyen a personas con discapacidad auditiva	**Solución 1**: algoritmos de reconocimiento de voz permiten crear subtítulos automáticos sincronizados para vídeos grabados	1. Real-Time Fire Warning System 2. Sign language recognition using artificial intelligent
1.3.1	**[12] Informing Accessible Design of AI Literacy Apps through Practices of Blind Parents Reading with Sighted Children** **[2] Addressing Web Accessibility Challenges with Generative Artificial Intelligence Tools for Inclusive Education** **[6] Digital accessibility in the era of artificial intelligence—Bibliometric analysis and systematic review** **[4] Accessibility evaluation of major assistive mobile applications available for the visually impaired** **[19] Creating Accessibility 2.0 with Artificial Intelligence** **[23] Imagining Artificial Intelligence Applications with People with Visual Disabilities using Tactile Ideation**	1. La estructura y organización de la información en las aplicaciones de lectura pueden no ser evidentes para los usuarios que dependen de tecnologías asistidas 2. La estructura y organización del contenido generado por IA puede no tener una jerarquía clara, dificultando la	**Solución 1:** diseñar aplicaciones con una estructura clara y semántica, utilizando encabezados y etiquetas adecuadas, para que las relaciones entre los elementos sean comprensibles a través de lectores de pantalla. **Solución 2:** desarrollar modelos de IA que generen contenido con una estructura semántica	1. Context-Aware Artificial Intelligence-based System 2. Banknote Object Detection

		comprensión para usuarios que dependen de tecnologías asistidas 3. El contenido visual complejo (gráficos, diagramas, formularios) no transmite relaciones semánticas a usuarios con lectores de pantalla 4. Estructura de interfaz poco clara para lectores de pantalla 5. Las relaciones visuales (como agrupaciones de botones o etiquetas) no siempre están disponibles para lectores de pantalla 6. Falta de adaptación de contenido para usuarios con discapacidades visuales	adecuada, utilizando encabezados y etiquetas que faciliten la navegación y comprensión mediante lectores de pantalla **Solución 3:** la visión por computadora basada en IA puede analizar elementos visuales y generar descripciones que expliciten relaciones **Solución 4:** uso de jerarquías visuales bien definidas y componentes nativos accesibles **Solución 5:** análisis semántico de interfaces con IA para identificar estructuras lógicas y relaciones jerárquicas en la interfaz **Solución 6:** creación de interfaces accesibles que se adapten a las necesidades de los usuarios con discapacidades visuales	
1.3.3	**[6] Digital accessibility in the era of artificial intelligence—Bibliometric analysis and systematic review**	1. Algunos usuarios no pueden interpretar diferencias de color o forma usadas como única vía para transmitir información	**Solución 1:** la IA puede detectar estos casos y ofrecer descripciones alternativas textuales o alertas cuando se depende solo de una característica sensorial	

2.1.1	**[12] Informing Accessible Design of AI Literacy Apps through Practices of Blind Parents Reading with Sighted Children** **[2] Addressing Web Accessibility Challenges with Generative Artificial Intelligence Tools for Inclusive Education** **[6] Digital accessibility in the era of artificial intelligence—Bibliometric analysis and systematic review** **[4] Accessibility evaluation of major assistive mobile applications available for the visually impaired** **[19] Creating Accessibility 2.0 with Artificial Intelligence** **[29] Exploration on Integrating Accessibility into an AI Course** **[13] Artificial Intelligence and Accessibility for Administrative Applications** **[1] Artificial Intelligence (AI) for Web Accessibility: Is Conformance Evaluation a Way Forward?** **[23] Imagining Artificial Intelligence Applications with People with Visual Disabilities using Tactile Ideation**	1. Algunas aplicaciones requieren interacciones táctiles o gestuales que no son accesibles para usuarios que dependen del teclado 2. Algunas aplicaciones de IA requieren interacciones que no son accesibles mediante teclado, limitando su uso para personas con discapacidades motoras 3. Algunos elementos no pueden ser activados sin un ratón, impidiendo su uso a personas que dependen del teclado 4. Algunas funciones no accesibles sin interacción táctil 5. Elementos inaccesibles por teclado 6. Inaccesibilidad para estudiantes con movilidad reducida que no pueden usar un ratón 7. Las personas con movilidad reducida pueden no ser capaces de usar un ratón, haciendo que ciertas funcionalidades sean inaccesibles 8. Navegación	**Solución 1:** garantizar que todas las funciones de la aplicación sean operables mediante teclado, permitiendo que los padres ciegos naveguen y utilicen la aplicación sin necesidad de interacciones táctiles **Solución 2**: asegurar que todas las funciones de las aplicaciones de IA sean operables mediante teclado, permitiendo que los usuarios naveguen y utilicen las herramientas sin necesidad de un ratón u otros dispositivos de apuntado **Solución 3:** los modelos de IA pueden simular navegación y detectar estos puntos de falla, sugiriendo accesos por teclado o ajustes de foco **Solución 4:** asegurar que toda funcionalidad sea usable con gestos o navegación tipo teclado **Solución 5:** evaluación automática del árbol de accesibilidad para detectar elementos sin navegación por teclado, proponiendo mejoras automáticas	1. VisionX-Virtual Assistant 2. Navigation and Augmented Reality Sistem

		inaccesible sin el uso de un mouse 9. Interfaz no accesible para usuarios con discapacidades visuales, dificultando su interacción con la tecnología	**Solución 6:** diseñar el curso para que se pueda navegar y realizar todas las tareas mediante el teclado **Solución 7:** la IA puede analizar la interfaz y garantizar que todos los controles y funciones sean accesibles mediante teclado **Solución 8:** la IA evalúa la accesibilidad del contenido solo con el teclado **Solución 9:** implementación de controles táctiles accesibles para facilitar la interacción de usuarios con discapacidades visuales	
2.1.2	**[6] Digital accessibility in the era of artificial intelligence—Bibliometric analysis and systematic review** **[29] Exploration on Integrating Accessibility into an AI Course**	1. Un usuario queda atrapado en un componente (por ejemplo, un modal) sin poder salir mediante el teclado 2. Estudiantes con discapacidades cognitivas o motoras pueden necesitar más tiempo para completar tareas	**Solución 1:** la IA puede automatizar pruebas de navegación por teclado y detectar si hay trampas de foco o bucles **Solución 2:** ofrecer tiempo adicional o la opción de extender el tiempo de finalización de actividades	
2.2.1	**[1] Artificial Intelligence (AI) for Web Accessibility: Is Conformance Evaluation a Way Forward?**	1. Límites de tiempo en contenido que afectan a usuarios con discapacidades cognitivas	**Solución 1:** la IA identifica y sugiere ajustes para eliminar o extender límites de tiempo en el contenido	1. Real-Time Fire Warning System (solo si permite al usuario personalizar el tiempo de visualización de las alertas)
2.2.2				1. Real-Time Fire Warning System

2.3.1	**[13] Artificial Intelligence and Accessibility for Administrative Applications**	1. El contenido que parpadea o destella a alta frecuencia puede causar molestias o convulsiones a personas con epilepsia fotosensible	**Solución 1:** la IA puede identificar y eliminar o modificar el contenido que parpadea o destella, asegurando que cumpla con los requisitos de accesibilidad	1. AI-Vision
2.4.1	**[19] Creating Accessibility 2.0 with Artificial Intelligence**	1. Usuarios con lectores de pantalla tienen que escuchar todo el contenido repetitivo antes de llegar al principal	**Solución 1:** identificación automática de bloques repetitivos usando IA para sugerir saltos estructurales o enlaces de navegación rápida	
2.4.2	**[12] Informing Accessible Design of AI Literacy Apps through Practices of Blind Parents Reading with Sighted Children** **[2] Addressing Web Accessibility Challenges with Generative Artificial Intelligence Tools for Inclusive Education** **[4] Accessibility evaluation of major assistive mobile applications available for the visually impaired**	1. La falta de títulos descriptivos en las páginas o secciones de la aplicación puede dificultar la navegación y orientación de los usuarios ciegos 2. La falta de títulos descriptivos en las interfaces de las aplicaciones de IA puede dificultar la orientación y navegación de los usuarios 3. Páginas o secciones sin títulos claros, dificultando la orientación	**Solución 1:** proporcionar títulos claros y descriptivos para cada página o sección, facilitando la comprensión del contenido y la navegación dentro de la aplicación **Solución 2:** incluir títulos claros y descriptivos en cada sección de las aplicaciones, facilitando la identificación del contenido y mejorando la experiencia del usuario **Solución 3:** usar encabezados descriptivos y etiquetas claras para cada pantalla	
2.4.3	**[6] Digital accessibility in the era of artificial intelligence—Bibliometric analysis and systematic review**	1. El foco del teclado no sigue un orden lógico, dificultando la navegación para personas con lector de pantalla	**Solución 1:** simulación de navegación con IA que detecta secuencias ilógicas y recomienda reordenamientos en la estructura del DOM	

2.4.4	**[21] Tools and Applications for Cognitive Accessibility**	1. Enlaces confusos o mal etiquetados pueden generar incertidumbre o dificultad para navegar	**Solución 1:** herramientas que mejoran la claridad de los enlaces y proporcionan etiquetas comprensibles, facilitando la navegación	1. AI Based Automated Imagen Caption Tool
3.1.1	**[21] Tools and Applications for Cognitive Accessibility**	La información en un idioma no especificado puede ser confusa para los usuarios con discapacidades cognitivas	aplicaciones que simplifican el lenguaje y ayudan a los usuarios a comprender mejor el contenido en un idioma claro y consistente	1. Development of Multi-Language Interactive Device using AI
3.3.1	**[6] Digital accessibility in the era of artificial intelligence—Bibliometric analysis and systematic review** **[4] Accessibility evaluation of major assistive mobile applications available for the visually impaired** **[19] Creating Accessibility 2.0 with Artificial Intelligence** **[23] Imagining Artificial Intelligence Applications with People with Visual Disabilities using Tactile Ideation**	1. Formularios no informan claramente cuándo ocurre un error (por ejemplo, campo vacío o mal formato) 2. Formularios que no informan de errores al usuario de forma clara 3. Los errores en formularios no se explican claramente 4. Dificultad para los usuarios con discapacidades visuales para entender la interfaz sin retroalimentación adecuada	**Solución 1**: los algoritmos de IA pueden detectar formularios mal estructurados y sugerir mensajes de error accesibles automáticamente **Solución 2:** proveer mensajes de error accesibles mediante texto visible y para lectores de pantalla **Solución 3:** IA que identifica patrones de errores y genera mensajes comprensibles y contextualizados para el usuario Solución 4: uso de ideación táctil para proporcionar retroalimentación y sugerencias en tiempo real, mejorando la comprensión de la interfaz	1. Intelligent Platform for Visually Impaired Children (solo si incluye ejercicios interactivos que indican errores al responder)

3.3.2	**[6] Digital accessibility in the era of artificial intelligence—Bibliometric analysis and systematic review** **[21] Tools and Applications for Cognitive Accessibility** **[1] Artificial Intelligence (AI) for Web Accessibility: Is Conformance Evaluation a Way Forward?**	1. Campos de formulario sin etiquetas o instrucciones claras dificultan su uso 2. Las instrucciones o etiquetas ambiguas pueden causar confusión a los usuarios con discapacidades cognitivas 3. Formularios sin etiquetas adecuadas o instrucciones claras	**Solución 1**: la IA puede identificar campos sin etiquetas y generar sugerencias de etiquetas accesibles basadas en contexto y contenido **Solución 2:** herramientas que proporcionan etiquetas claras y comprensibles, así como instrucciones detalladas para completar formularios o interacciones **Solución 3:** la IA identifica campos mal etiquetados y sugiere correcciones en las etiquetas o instrucciones	
4.1.2	**[4] Accessibility evaluation of major assistive mobile applications available for the visually impaired** **[19] Creating Accessibility 2.0 with Artificial Intelligence**	1. Controles que no exponen su función a tecnologías asistidas (botones sin rol definido, etc.) 2. Los lectores de pantalla no reconocen correctamente los controles interactivos	**Solución 1:** implementar controles con roles, nombres y estados claros para tecnologías como TalkBack o VoiceOver **Solución 2:** análisis de atributos ARIA y detección de roles mal definidos mediante modelos de aprendizaje automático	1. Sonar Glass - Artificial Vision 2. Avatar to Person (ATP)

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA