



UNIVERSIDAD NACIONAL DE HURLINGHAM

Licenciatura en Informática

Accesify: diseño e implementación de un widget de accesibilidad web

Téc. Facundo Matias Ferrara

Director del Instituto: Ing. Fernando Puricelli

Tutor: Mag. Ing. Pablo Pandolfo

Tesina para obtener el Título de

Licenciado en Informática

Buenos Aires, Argentina

mayo 2026

Agradecimientos

Deseo expresar mi más profundo agradecimiento a todas las personas e instituciones que me acompañaron y brindaron su apoyo durante el desarrollo de esta tesis. En primer lugar, extendo mi gratitud al Mag. Ing. Pablo Pandolfo, director de este trabajo, por su guía constante, su paciencia y su valioso acompañamiento a lo largo de todo el proceso. Su experiencia y compromiso fueron clave para alcanzar los objetivos propuestos.

Asimismo, agradezco sinceramente al director del Instituto, Ing. Fernando Puricelli, quien supo estar presente en los momentos importantes y fue una figura de referencia y motivación a lo largo de toda la formación.

También quiero reconocer a los docentes y compañeros de la Universidad Nacional de Hurlingham, cuyas enseñanzas, intercambios y apoyo contribuyeron significativamente tanto a mi desarrollo académico como personal.

Índice General

1 Introducción.....	7
1.1 Descripción del problema.....	7
1.2 Marco metodológico.....	8
1.2.1 Objetivo.....	8
1.2.2 Alcance.....	8
2 Marco conceptual y el estado del arte.....	11
2.1 Accesibilidad Web.....	11
2.2 Discapacidad.....	13
2.2.1 Concepto.....	13
2.2.2 Tipos de discapacidad.....	13
2.2.3 Datos y cifras.....	14
2.3 Discapacidad visual.....	15
3 Widget de accesibilidad web.....	19
3.1 Concepto.....	19
3.2 Funciones claves.....	19
3.3 Ventajas.....	21
4 Desarrollo del trabajo.....	24
4.1 Especificación del problema.....	24
4.2 Análisis y diseño.....	25
4.2.1 Requerimientos funcionales.....	26
4.2.2 Requerimientos no funcionales.....	36
4.2.3 Diagrama de caso de uso.....	38
4.2.4 Diagrama de contexto.....	39
4.2.5 Diagrama de funcionalidad del sistema.....	40
4.3 Tecnologías.....	40
4.3.1 Lenguajes de desarrollo.....	41
4.3.2 Framework.....	42
4.4 Fase de desarrollo.....	43
4.4.1 Arquitectura MVC del sistema.....	43
4.4.2 Organización del proyecto.....	44
4.4.3 Empaquetado y bundling con Webpack.....	45
4.4.4 Gestión de dependencias con npm.....	46
4.4.5 Versionado y publicación en npm.....	47
4.4.6 Automatización de instalación.....	48
4.4.8 Diseño de interfaz y experiencia de usuario.....	49
4.4.9 Subir el widget a producción.....	53
4.4.10 Diseño modular y desacoplamiento del componente.....	54
4.5 Proceso de pruebas.....	55
4.5.1 Pruebas funcionales.....	56
4.5.2 Casos de prueba.....	57
4.5.3 Pruebas de integración.....	58

4.5.4 Pruebas de usabilidad.....	59
5 Conclusiones.....	61
5.1 Resumen final.....	61
5.2 Futuras investigaciones.....	62
6 Anexo.....	63
6.1 Guía de acrónimos y términos.....	63
7 Apéndice.....	68
7.1 Configuración de Webpack.....	68
7.2 Inicialización y configuración del paquete npm.....	69
7.3 Comandos de publicación y versionado.....	70
7.4 Script de instalación automatizada.....	70
8 Bibliografía.....	72

Índice de figuras

Figura 1. Claves para mejorar la Accesibilidad Web.....	12
Figura 2. Estimación de personas con deterioro de la visión.....	15
Figura 3. Página actual de ajuste “modo contraste”.....	26
Figura 4. Página con “Contraste inteligente” aplicado.....	27
Figura 5. Página actual “Ajustes de texto”.....	28
Figura 6. Página con “Fuente negrita” aplicado.....	28
Figura 7. Página actual “Modos de lectura”.....	29
Figura 8. Página con “Modo de lectura” aplicado.....	30
Figura 9. Página actual “Resaltar enlaces”.....	31
Figura 10. Página con “Resaltar enlaces” aplicado.....	32
Figura 11. Página con “Resaltar títulos” aplicado.....	32
Figura 12. Página actual “Pruebas de animaciones”.....	33
Figura 13. Página con “Pausar animaciones” aplicado.....	33
Figura 14. Página de Inicio Actual (sin estilos aplicados).....	34
Figura 15. Página con modo “Perfil Ceguera” activado.....	34
Figura 16. Página con modo “Perfil Daltonismo” activado.....	35
Figura 17. Página con modo “Perfil Visión baja” activado.....	35
Figura 18. Página con modo “Perfil Dislexia” activado.....	36
Figura 19. Diagrama de caso de uso.....	38
Figura 20. Diagrama de caso de contexto.....	39
Figura 21. Diagrama de caso de funcionalidad.....	40
Figura 22. Fundamentos de Laravel.....	44
Figura 23. Funcionalidades de Accesify.....	51

Índice de tablas

Tabla 1. Tipos de discapacidades.....	9
Tabla 2. Tipos de lectores de pantalla:.....	17

1 Introducción

1.1 Descripción del problema

Cuando los sitios web y las herramientas en línea están diseñadas y programadas de manera adecuada, todos los usuarios son capaces de usarlas. Sin embargo, en la actualidad, muchas se desarrollan con barreras de accesibilidad que los hacen difíciles o imposibles de usar para algunas personas.

A pesar de los avances en accesibilidad digital, persisten obstáculos para las personas con discapacidad visual, auditiva, motriz o cognitiva, afectando su acceso a información y servicios esenciales. La escasez de opciones de personalización en la mayoría de los sitios web impide que los usuarios ajusten la visualización, navegación y funcionalidades según sus necesidades específicas. La falta de adaptaciones inclusivas limita la autonomía de las personas con discapacidad en el acceso a plataformas de educación, comercio, salud y administración pública, restringiendo así su participación plena en la sociedad digital.

La accesibilidad web constituye una práctica que promueve la inclusión digital, ya que permite que personas con discapacidad y adultos mayores, entre otros grupos, puedan acceder y utilizar la Web, tal como se describe en materiales del Instituto Nacional de la Administración Pública [2].

Asimismo, se estima que sólo tres de cada cien páginas web son aptas para personas con discapacidad visual, según datos publicados por la Organización de las Naciones Unidas [3].

En Argentina, el Estado Nacional, entiéndanse los tres poderes que lo constituyen, sus organismos descentralizados o autárquicos, los entes públicos no estatales, las empresas del Estado y las empresas privadas concesionarias de servicios públicos, empresas prestadoras o contratistas de bienes y servicios, deberán respetar en los diseños de sus páginas web las normas y requisitos sobre accesibilidad de la información que faciliten el acceso a sus contenidos, a todas las personas con

discapacidad con el objeto de garantizarles la igualdad real de oportunidades y trato, evitando así todo tipo de discriminación, de acuerdo con lo establecido en la Ley 26.653 [4].

1.2 Marco metodológico

1.2.1 Objetivo

Desarrollar un widget de accesibilidad visual que permita a los sitios web ofrecer opciones de personalización adaptadas a diversos perfiles de usuarios con dificultades visuales, tanto permanentes como transitorias.

Accesify busca ajustar las páginas web a los estándares internacionales establecidos por las Pautas de Accesibilidad para el Contenido Web (WCAG 2.2), incorporando funcionalidades como modificación del contraste de color, aumento del tamaño del texto, mejoras en la visibilidad de enlaces y ajustes de espaciado y tipografía.

El proyecto incorpora el análisis de buenas prácticas, marcos normativos y la implementación de una interfaz gráfica intuitiva, flexible e inclusiva.

1.2.2 Alcance

El presente proyecto se enfoca en el desarrollo de **Accesify**, un widget de accesibilidad visual diseñado para ser incorporado en sitios web sin requerir modificaciones estructurales significativas. Su objetivo es mejorar la experiencia de navegación de personas que presentan distintos tipos de dificultades visuales, tanto permanentes como transitorias.

El widget está orientado a brindar soluciones específicas y configurables, ajustadas a distintos perfiles de usuarios. Entre las condiciones visuales contempladas se incluye la siguiente tabla que corresponde a las diferentes discapacidades.

Tabla 1. *Tipos de discapacidades*

Tipo		Necesidad
Ceguera total	Ausencia completa de percepción visual, que impide procesar información presentada en pantalla.	Compatibilidad con lectores de pantalla, navegación mediante teclado, textos alternativos en imágenes y correcta semántica HTML.
Daltonismo	Alteración en la percepción de los colores, con dificultad para diferenciar combinaciones específicas (rojo/verde, azul/amarillo, etc.).	Uso de contrastes adecuados, paletas accesibles, diferenciación de información mediante formas, iconos o patrones, no solo colores.
Dislexia	Dificultad específica del aprendizaje que afecta la decodificación, comprensión y fluidez lectora.	Tipografías legibles, espaciado adecuado, modos de lectura simplificada, minimización de distracciones visuales.
Visión baja	Limitación significativa de la agudeza visual, que dificulta la lectura de textos y la identificación de elementos.	Ampliación de texto, alto contraste, fuentes claras, resaltado de elementos interactivos, posibilidad de aumentar el tamaño general de la interfaz.

En la **tabla 1** se presentan las siguientes funcionalidades de forma modular, permitiendo que el usuario final seleccione y combine las opciones más adecuadas según su necesidad. El enfoque está centrado en la autonomía del usuario, respetando la estructura original del sitio web y evitando interferencias con su diseño base.

El proyecto no aborda otras dimensiones de la accesibilidad web (como las relacionadas con discapacidades auditivas, motrices o cognitivas), sino que se restringe al campo de la accesibilidad visual. A través de esta solución, se busca contribuir a la inclusión digital, facilitando el acceso equitativo a la información en entornos digitales para una población diversa que presenta desafíos visuales específicos, tal como se describe en el artículo de Mique [5].

2 Marco conceptual y el estado del arte

2.1 Accesibilidad Web

La accesibilidad web, inclusión digital, implica que personas con discapacidad y adultos mayores, entre otros, puedan hacer uso de la Web. Desde la perspectiva de la accesibilidad web, se entiende que un sitio o contenido web es accesible cuando cualquier tipo de usuario puede percibir, entender, navegar e interactuar con él, tal como se plantea en materiales del Instituto Nacional de la Administración Pública [2].

La Web está diseñada fundamentalmente para funcionar para todas las personas, independientemente de su hardware, software, idioma, ubicación o capacidad. Cuando la Web cumple este objetivo, es accesible para personas con diversas capacidades auditivas, motoras, visuales y cognitivas.

Por lo tanto, el impacto de la discapacidad en el entorno digital puede diferir del que se presenta en el mundo físico, ya que muchas de las barreras de comunicación e interacción existentes en este último pueden eliminarse mediante el uso de tecnologías adecuadas. Sin embargo, cuando los sitios web, las aplicaciones o las herramientas digitales no se diseñan teniendo en cuenta criterios de accesibilidad, pueden generar nuevas barreras que dificultan o incluso impiden el acceso y uso de la Web por parte de las personas con discapacidad.

Entre estos casos se incluyen:

La accesibilidad web no solo beneficia a personas con discapacidad, sino también a usuarios en diferentes contextos de uso, como aquellos que utilizan dispositivos móviles, presentan limitaciones temporales o cuentan con conexiones a Internet restringidas, tal como señalan guías del World Wide Web Consortium [1].

- Personas utilizando teléfonos móviles, relojes inteligentes, televisores

inteligentes y otros dispositivos con pantallas pequeñas, diferentes modos de entrada, etc.

- Personas mayores cuyas habilidades cambian con la edad.
- Personas con “discapacidades temporales”, como puede ser un brazo roto o la pérdida de anteojos.
- Personas con “limitaciones por su ubicación”, como puede ser bajo la luz del sol o en un entorno donde no se puede escuchar audio.
- Personas con conexión lenta a Internet o que tienen ancho de banda limitado o costoso.

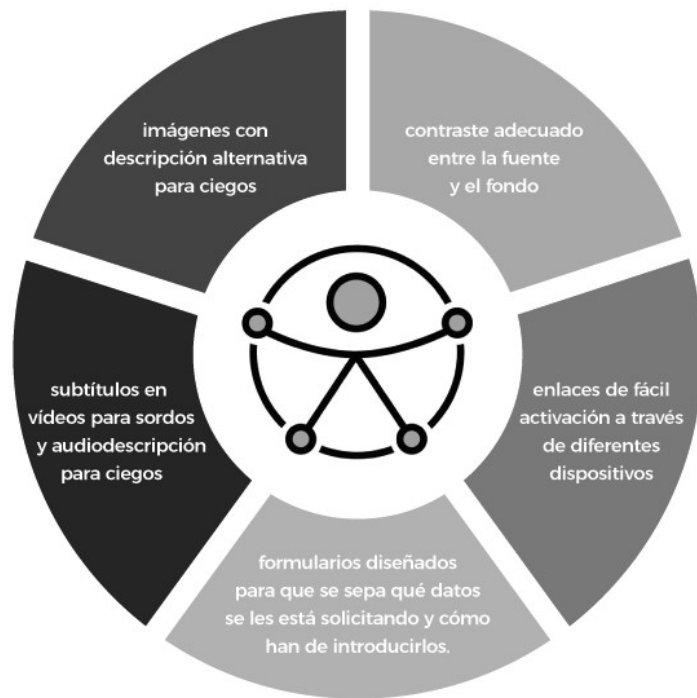


Figura 1. *Claves para mejorar la Accesibilidad Web*

La accesibilidad Web no sólo busca incluir a quienes tienen alguna discapacidad, sino que también mejora la experiencia para todos. Al construir sitios más fáciles de usar y sin barreras, se logra que cualquier persona sin importar sus habilidades o el

dispositivo que use pueda navegar cómodamente, haciendo que la Web sea mejor para toda la sociedad, tal como se señala en World Wide Web Consortium [6].

2.2 Discapacidad

2.2.1 Concepto

La palabra discapacidad se define como “situación de la persona que, por sus condiciones físicas, sensoriales, intelectuales o mentales duraderas, encuentra dificultades para su participación e inclusión social”, según el Real Academia Española [7].

La Ley N.º 22.431 define a una persona con discapacidad como aquella que padece “una alteración funcional permanente, transitoria o prolongada, motora, sensorial o mental que, en relación a su edad y medio social, implique desventajas considerables para su adecuada integración social o laboral” [8].

2.2.2 Tipos de discapacidad

La Organización Mundial de la Salud (OMS) identifica cinco grandes categorías de discapacidad: motriz, visual, auditiva, cognitiva y psicosocial. Cada una de ellas implica desafíos específicos que requieren adaptaciones particulares, tanto en entornos físicos como digitales [9].

- **La discapacidad motriz** afecta la movilidad total o parcial, e incluye desde dificultades leves para desplazarse hasta en casos de parálisis.
- **La discapacidad auditiva** se refiere a pérdidas parciales o totales de audición, lo cual puede dificultar la recepción de información si no hay alternativas visuales.
- **La discapacidad cognitiva** afecta el procesamiento de la información, la comprensión o la memoria, mientras que la psicosocial está relacionada con

condiciones de salud mental que pueden limitar la interacción social o la autonomía funcional.

- **La discapacidad visual**, que será el foco del presente trabajo, comprende desde alteraciones leves de la agudeza visual hasta la ceguera completa. Incluye también dificultades para distinguir colores, problemas con el contraste, sensibilidad a la luz o limitaciones en la lectura de textos pequeños. Estas condiciones pueden dificultar el acceso autónomo a contenidos digitales si no están diseñados con criterios de accesibilidad visual adecuados.

Es importante destacar que las barreras que enfrentan las personas con discapacidad no derivan únicamente de su condición, sino de un entorno que no contempla la diversidad funcional. En ese sentido, la Convención sobre los Derechos de las Personas con Discapacidad, adoptada por las Naciones Unidas, establece que la accesibilidad es un derecho fundamental para garantizar la igualdad de oportunidades y la plena participación en la sociedad [10].

2.2.3 Datos y cifras

En el mundo hay al menos 2200 millones de personas con deterioro de la visión cercana o lejana. En 1000 millones de esos casos, como mínimo, la discapacidad visual podría haberse evitado o todavía no se ha tratado.

Las principales causas a nivel mundial de la discapacidad visual y la ceguera son los errores de refracción y las cataratas.

Se estima que, a nivel mundial, sólo el 36% de las personas con un deterioro de la visión lejana debido a errores de refracción y el 17% de las personas con discapacidad visual debida a las cataratas han tenido acceso a una intervención adecuada.

La discapacidad visual supone una enorme carga económica mundial, ya que se estima que el costo anual en términos de productividad asciende a US\$ 411 000 millones.

Aunque la pérdida de visión puede afectar a personas de todas las edades, la mayoría de las personas con discapacidad visual y ceguera superan los 50 años [9].

Porcentaje

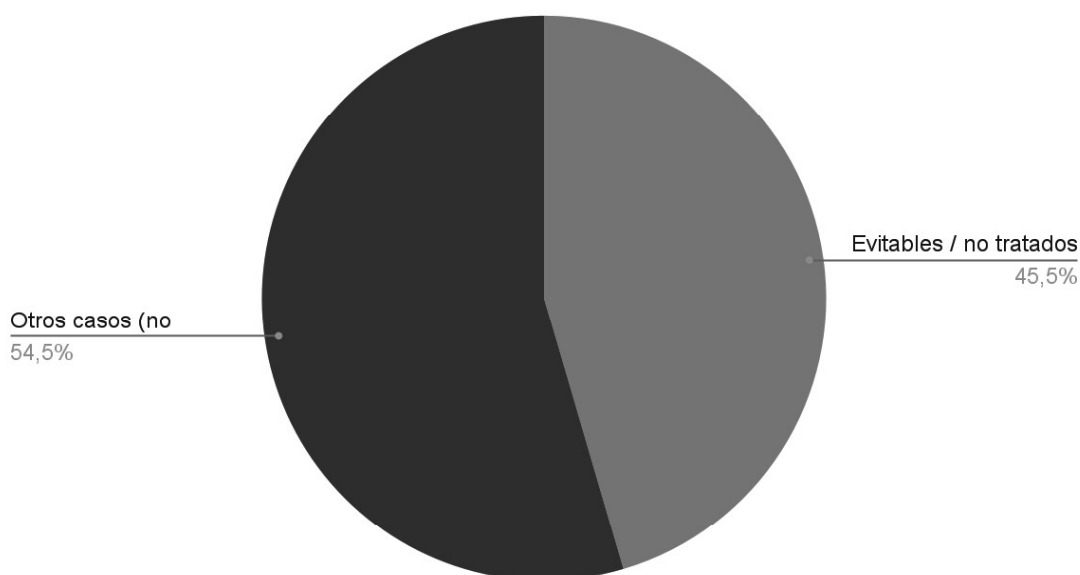


Figura 2. *Estimación de personas con deterioro de la visión*

2.3 Discapacidad visual

La discapacidad visual comprende un amplio espectro de condiciones que afectan total o parcialmente la capacidad de una persona para percibir estímulos visuales. De acuerdo con la Organización Mundial de la Salud (OMS), el deterioro de la visión constituye un problema de salud global que afecta a una proporción significativa de la población mundial [9].

En el contexto digital, estas condiciones generan obstáculos significativos para acceder de forma autónoma a la información, interactuar con plataformas web o

navegar por interfaces no adaptadas. Frente a este escenario, el desarrollo de herramientas específicas resulta fundamental para promover la equidad y la inclusión digital.

El widget **Accesify** se diseña con el objetivo de responder a los distintos perfiles de discapacidad visual, contemplando desde la baja visión hasta la ceguera total. A través de funcionalidades como el ajuste de contraste, el aumento del tamaño de fuente, el resaltado del cursor y la compatibilidad con lectores de pantalla, la herramienta permite realizar adaptaciones visuales durante la navegación, con el fin de facilitar el acceso a la información. De este modo, no sólo beneficia a las personas con discapacidad visual, sino también a otros grupos con dificultades asociadas, como adultos mayores o personas con dislexia, fortaleciendo un enfoque inclusivo y universal.

Lector de pantalla:

El lector de pantalla es una tecnología asistiva que permite a los usuarios con ceguera u otra discapacidad visual utilizar sus dispositivos digitales, como una computadora o un teléfono inteligente. Se fundamenta en realizar una lectura en voz alta de todos los elementos textuales que aparecen en la pantalla de estos dispositivos.

Gracias a los avances tecnológicos, esta herramienta va más allá de leer texto online o documentos. Hoy en día el usuario de un lector de pantalla puede, tal como se describe en UserWay [11]:

- Saltar entre diferentes secciones de un documento o web.
- Seleccionar si desea navegar por encabezados, enlaces o listas.
- Conocer qué tipo de elemento está enfocando.
- Completar un formulario de envío de datos.

Estas funcionalidades se presentan de manera accesible, sin necesidad de instalar software adicional ni realizar modificaciones complejas en el sitio web. El objetivo es

permitir que cada usuario configure su experiencia de forma autónoma, adaptando la interfaz según sus necesidades visuales.

El enfoque modular y escalable de **Accesify** responde a los principios de accesibilidad universal, contribuyendo a reducir la exclusión digital y a promover un entorno web más equitativo.

Tabla 2. *Tipos de lectores de pantalla:*

Nombre del lector	Sistema Operativo	Características principales
Talkback	Android	Está preinstalado en la mayoría de dispositivos Android. Permite control por gestos, navegación por elementos, lectura de texto en pantalla, compatibilidad con apps de Google.
Asistente de voz (Samsung)	Android (Samsung)	Similar a TalkBack pero adaptado por Samsung. Funciones extra integradas en la capa de Samsung (One UI).
VoiceOver	iOS (iPhone/iPad)	Lector de pantalla nativo de Apple. Muy optimizado, con gestos táctiles avanzados, excelente integración con apps de Apple.
Narrador	Windows (PC)	Viene integrado en Windows. Útil como lector básico o de emergencia. Ha mejorado en Windows 10 y 11, pero menos usado que NVDA o JAWS.
NVDA (NonVisual Desktop Access)	Windows (PC)	Gratuito y de código abierto. Muy usado por su accesibilidad, soporte en navegadores y aplicaciones de oficina.

Como se observa en la **Tabla 2**, existen diferentes lectores de pantalla desarrollados para distintos sistemas operativos, tanto en dispositivos móviles como en computadoras personales. Estas herramientas permiten a las personas con discapacidad visual interactuar con interfaces digitales mediante síntesis de voz que describe los elementos presentes en la pantalla, como textos, botones, enlaces o formularios. Los lectores de pantalla interpretan la estructura semántica del contenido web y permiten la navegación a través de teclado o gestos táctiles, facilitando el acceso a la información digital. Su utilización es fundamental para garantizar la accesibilidad web, ya que constituyen una de las principales tecnologías de asistencia empleadas por personas ciegas o con baja visión para navegar por sitios y aplicaciones digitales, tal como se describe en la documentación de Microsoft y en guías de accesibilidad web [11,12].

3 Widget de accesibilidad web

3.1 Concepto

Un widget de accesibilidad es un componente reutilizable que se integra en sitios web con el objetivo de mejorar la experiencia de usuarios con discapacidades. Proporciona herramientas como ajustes de contraste, configuración del tamaño y estilo del texto, y modos de lectura asistida, alineándose con las Pautas de Accesibilidad para el Contenido Web (WCAG 2.1).

En el contexto de la Web, este tipo de widgets suele presentarse como un elemento visible e interactivo dentro de la página, que permite a los usuarios realizar ajustes de forma directa mientras navegan por el sitio. A través de estas herramientas es posible modificar distintos aspectos visuales del contenido, aplicando los cambios de manera inmediata sobre la página sin necesidad de recargar ni salir de la experiencia de navegación.

3.2 Funciones claves

Entre las funcionalidades más relevantes incorporadas en el widget de accesibilidad **Accesify**, se destacan las siguientes:

- El uso del color no debe constituir el único medio visual para transmitir información, indicar acciones o distinguir elementos dentro de la interfaz. La incorporación de recursos adicionales, como iconografía, texto descriptivo o subrayados, resulta fundamental para garantizar la comprensión del contenido por parte de personas con daltonismo o discapacidad visual.
- Los widgets de accesibilidad permiten redimensionar el texto hasta un 200 % sin pérdida de contenido ni funcionalidad, exceptuando los subtítulos y las imágenes que contienen texto. Esta funcionalidad resulta especialmente relevante para usuarios con baja visión o dificultades de lectura, facilitando el

acceso a la información sin requerir tecnologías de asistencia adicionales, tal como se establece en las pautas de accesibilidad para el contenido web publicadas por el Gobierno de Argentina [14].

- La mejora del contraste visual entre el texto y el fondo constituye un aspecto esencial de la accesibilidad. En términos generales, se busca alcanzar una relación de contraste mínima de 4,5:1 para textos estándar y de 3:1 para textos de gran tamaño. No se aplican requisitos de contraste a elementos puramente decorativos ni a componentes de interfaz inactivos, dado que no aportan información relevante al usuario.
- Algunos widgets incorporan funcionalidades de narración automática que permiten transformar el contenido textual de la página en audio. Esta característica facilita el acceso a la información a personas con discapacidad visual severa, así como a usuarios con dificultades de lectura o comprensión del texto escrito, tal como se describe en recursos sobre accesibilidad del World Wide Web Consortium [1].
- La posibilidad de activar tipografías específicas, como el modo de dislexia, contribuye a mejorar la legibilidad del texto y a reducir la confusión entre caracteres visualmente similares. Esta funcionalidad está orientada principalmente a usuarios con dislexia u otras dificultades de procesamiento lector, según lo expuesto por OpenDyslexic [12].
- El ajuste del espaciado entre líneas, palabras y márgenes permite mejorar la organización visual del contenido. Esta funcionalidad beneficia a personas con baja visión, dislexia o dificultades cognitivas, favoreciendo una lectura más clara y estructurada del texto, en línea con recomendaciones de accesibilidad [12].

3.3 Ventajas

La incorporación de widgets de accesibilidad en sitios web presenta múltiples ventajas desde el punto de vista social, técnico y normativo, entre las que se destacan las siguientes:

- Los widgets de accesibilidad permiten a los usuarios personalizar la experiencia de navegación mediante el ajuste de parámetros visuales y funcionales, como el tamaño del texto, el contraste de colores u otros elementos de la interfaz. Esta capacidad resulta especialmente relevante para personas con discapacidades visuales o cognitivas, ya que facilita el acceso a la información de acuerdo con sus necesidades específicas.
- Estas herramientas suelen caracterizarse por su facilidad de uso e implementación, permitiendo que personas con distintos niveles de alfabetización digital puedan beneficiarse de sus funcionalidades sin enfrentar barreras técnicas adicionales. La simplicidad en la interacción contribuye a una navegación más accesible e inclusiva.
- La incorporación de widgets de accesibilidad contribuye a promover la igualdad de oportunidades en el acceso a los contenidos digitales, en concordancia con los principios de inclusión y diversidad. En este sentido, su implementación favorece el desarrollo de entornos web más equitativos y socialmente responsables.
- Desde una perspectiva normativa, la implementación de soluciones de accesibilidad contribuye al cumplimiento de las leyes y regulaciones vigentes en materia de accesibilidad digital. En este sentido, la adopción de herramientas como los widgets de accesibilidad puede colaborar en la reducción de riesgos legales y en el fortalecimiento de la reputación institucional.
- Un sitio web accesible no sólo beneficia a las personas con discapacidades, sino que también mejora la experiencia general de todos los usuarios. Las funcionalidades como el ajuste de texto y contraste resultan útiles para

personas mayores o para quienes navegan en condiciones ambientales desfavorables, incrementando la usabilidad del sitio.

- La accesibilidad web favorece la ampliación del público potencial al facilitar el acceso a los contenidos digitales a personas con discapacidades y a otros grupos que podrían enfrentar barreras de acceso. Esta ampliación del alcance contribuye a una mayor difusión de la información y a un uso más equitativo de los recursos digitales.
- Muchos de los principios asociados a la accesibilidad web tienen un impacto positivo en la optimización para motores de búsqueda (SEO), dado que promueven una estructura clara del contenido y el uso adecuado de descripciones alternativas. Estas prácticas favorecen tanto a los usuarios como a los sistemas de indexación.
- La accesibilidad debe entenderse como un proceso de mejora continua. En este sentido, algunos widgets incorporan mecanismos de retroalimentación que permiten identificar dificultades de uso y realizar ajustes progresivos, contribuyendo a la optimización constante de la experiencia de navegación [10].

3.3.2 Ejemplos de barreras de accesibilidad a sitios WEB

Las barreras de accesibilidad web son obstáculos que dificultan que personas con discapacidades o condiciones particulares puedan percibir, comprender, navegar, interactuar o contribuir de manera eficaz en un sitio web. Entre las barreras más comunes se encuentran las siguientes, tal como se describe en pautas y recomendaciones de accesibilidad web como WCAG 2.2 [15]:

- **Contraste de color deficiente**, situación que impide que personas con visión reducida o daltonismo puedan leer correctamente el contenido cuando no existe una diferenciación adecuada entre el texto y el fondo, como ocurre con textos claros sobre fondos claros.

- **Uso exclusivo del color para transmitir información**, práctica que dificulta el acceso a personas con daltonismo y a usuarios de lectores de pantalla, ya que el color por sí solo no es un medio accesible para comunicar información relevante, como la identificación de campos obligatorios en formularios.
- **Ausencia de alternativas textuales en imágenes**, situación impide que personas ciegas o con discapacidad visual comprendan el contenido o la finalidad de imágenes, ilustraciones y gráficos cuando estos no cuentan con descripciones textuales.
- **Videos sin subtítulos**, limitando el acceso a la información audiovisual para personas con discapacidad auditiva que dependen de textos escritos para comprender el contenido.
- **Formularios en línea inaccesibles**, que dificultan o imposibilitan su correcta utilización cuando carecen de etiquetas asociadas a los campos, instrucciones claras o indicadores de error comprensibles para tecnologías de asistencia.
- **Navegación dependiente exclusivamente del uso del mouse**, práctica excluye a personas que no pueden utilizar dispositivos apuntadores y requieren navegar mediante el teclado u otras herramientas de apoyo.

4 Desarrollo del trabajo

4.1 Especificación del problema

La accesibilidad web busca garantizar que cualquier persona, independientemente de sus capacidades físicas, sensoriales o cognitivas, pueda navegar e interactuar con los contenidos digitales en condiciones de igualdad. Esta premisa no sólo responde a principios éticos y de inclusión, sino que también se encuentra respaldada por normativas específicas en el ámbito argentino, como la Ley 26.653 y la Disposición ONTI N° 6/2019, que exige el cumplimiento de al menos los niveles de conformidad “A” y “AA” de las pautas internacionales WCAG para tecnologías como HTML y CSS [16].

A pesar de esta normativa, en la práctica muchos sitios web siguen presentando barreras de accesibilidad, ya sea por desconocimiento, falta de herramientas adecuadas o dificultades para implementar soluciones técnicas que no alteren la estructura general del sitio. Si bien existen herramientas comerciales que permiten mejorar la accesibilidad (como UserWay), muchas de ellas dependen de servicios externos, no se adaptan a las necesidades locales o no cumplen completamente con las exigencias legales nacionales, como la validación de al menos 30 criterios definidos por ONTI [17].

Ante este escenario, surge la necesidad de desarrollar una herramienta accesible, autónoma y compatible con la legislación vigente, que permita incorporar mejoras de accesibilidad directamente en el sitio, sin depender de servicios de terceros ni alterar su diseño o funcionamiento.

Accesify fue desarrollado con el objetivo de cubrir las siguientes necesidades: brindar un widget de accesibilidad eficiente, reutilizable e integrable en cualquier sitio web, que contribuya a reducir barreras digitales y facilite el cumplimiento normativo en proyectos públicos o institucionales.

4.2 Análisis y diseño

Accesify es un widget de accesibilidad web de código abierto, desarrollado para integrarse de forma sencilla en sitios web modernos que utilizan PHP 7.3, Laravel 8 y JavaScript, sin depender de servicios externos ni de plataformas de terceros para su funcionamiento.

El objetivo principal del sistema es proporcionar mecanismos de adaptación visual y funcional que permitan mejorar la experiencia de navegación de usuarios con distintas necesidades mediante la modificación dinámica de la interfaz del sitio donde se integra.

Al tratarse de una herramienta de código abierto, se pone especial énfasis en la documentación clara de su diseño, arquitectura y estructura interna, con el propósito de facilitar su extensión, reutilización e integración en otros proyectos que busquen cumplir con las normativas de accesibilidad vigentes en la República Argentina, así como con regulaciones de otros países basadas en las pautas internacionales WCAG.

El código fuente del proyecto será publicado como paquete reutilizable en npm, promoviendo la transparencia, la colaboración comunitaria y la mejora continua del sistema.

Accesify toma como referencia buenas prácticas observadas en bibliotecas distribuidas a través de npm. En particular, se consideró como inspiración conceptual la librería *accessibility*, que ofrece una barra de herramientas configurable orientada a la mejora de la experiencia de usuario. No obstante, el desarrollo de Accesify es independiente de dicha implementación.

El sistema ofrece un conjunto de mecanismos de adaptación visual orientados a mejorar la accesibilidad de la interfaz mediante perfiles configurables y ajustes dinámicos aplicados en tiempo real.

4.2.1 Requerimientos funcionales

Las funcionalidades del widget **Accesify** se estructuran en distintos módulos orientados a mejorar la accesibilidad visual de los sitios web donde se integra. Estas funcionalidades se aplican dinámicamente sobre el documento cargado en el navegador mediante la manipulación del DOM y la incorporación de clases CSS específicas, sin modificar el código fuente original del sitio anfitrión. De este modo, los cambios son reversibles y se ejecutan únicamente en tiempo de ejecución.

Ajustes de contraste:

El sistema permite modificar dinámicamente el esquema de color del sitio web a través de cuatro niveles configurables: contraste original, inversión de colores, contraste oscuro y contraste claro.

La implementación técnica se basa en la alternancia de clases CSS específicas aplicadas al elemento raíz del documento, permitiendo modificar de forma global la apariencia visual sin alterar la estructura HTML existente.

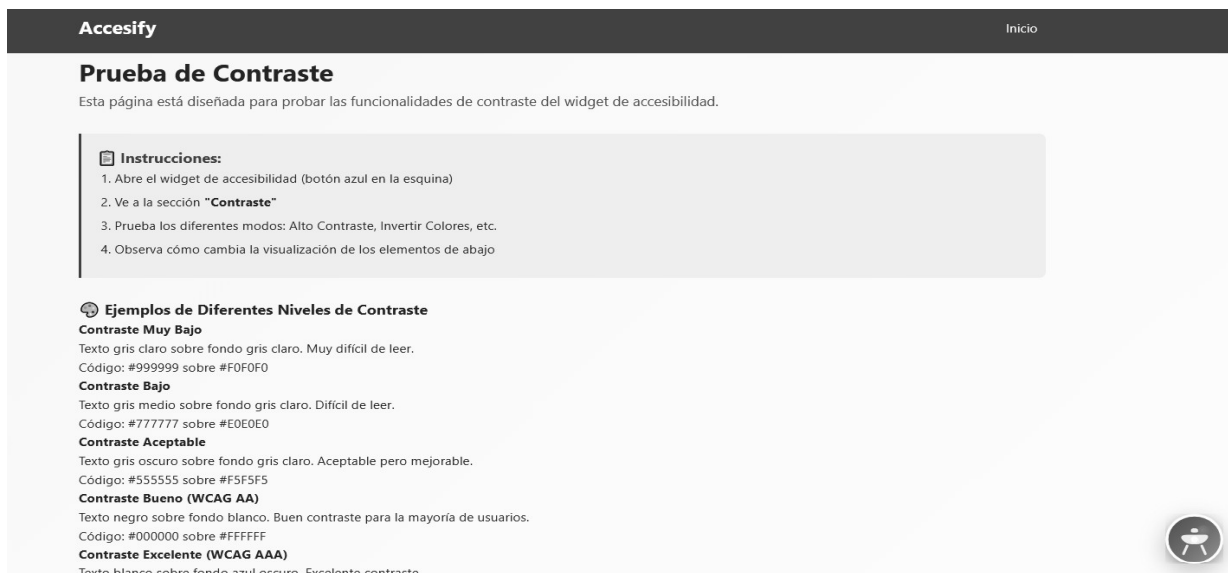


Figura 3. *Página actual de ajuste “modo contraste”*

Esta funcionalidad se encuentra alineada con las recomendaciones de contraste mínimo establecidas por las Pautas de Accesibilidad para el Contenido Web (WCAG 2.1) del W3C (2018), que definen criterios para garantizar la legibilidad del contenido.

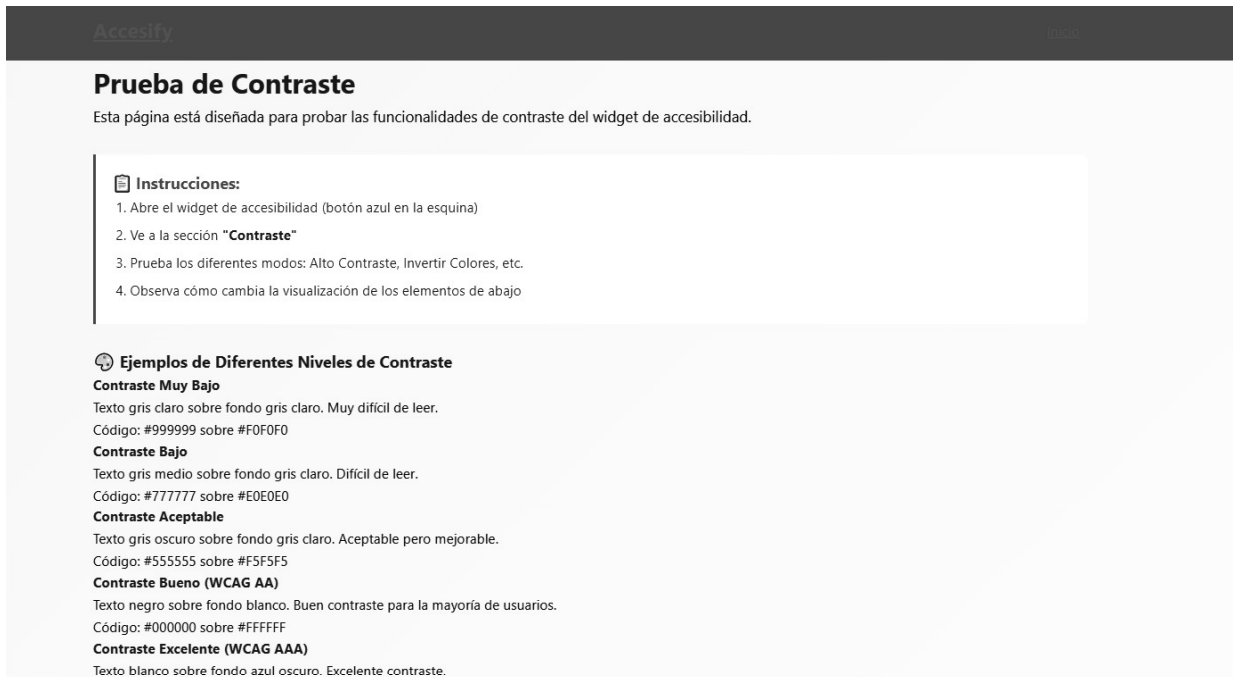


Figura 4. *Página con “Contraste inteligente” aplicado*

Ajustes tipográficos:

El widget incorpora un conjunto de adaptaciones tipográficas orientadas a mejorar la legibilidad del contenido textual. Entre ellas se incluyen:

- Escalado progresivo del tamaño de texto, con múltiples niveles de aumento.
- Cambio de tipografía, permitiendo alternar entre fuente estándar, fuente adaptada para dislexia y fuente de alta claridad.
- Incremento del espaciado entre líneas y caracteres.
- Alineación izquierda del texto para evitar el justificado completo.

Estas adaptaciones responden a principios de accesibilidad cognitiva y legibilidad recomendados por el W3C, especialmente en contextos vinculados a dislexia y dificultades de procesamiento visual.

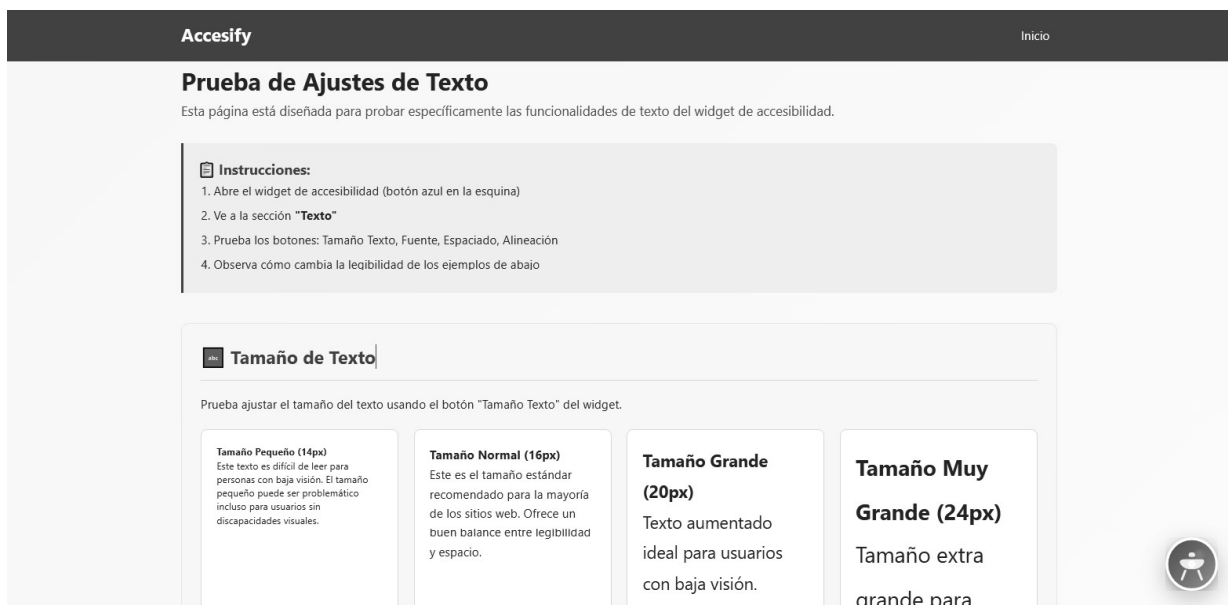


Figura 5. Página actual "Ajustes de texto"



Figura 6. Página con "Fuente negrita" aplicado

Funcionalidades de lectura:

Accesify incorpora herramientas orientadas a optimizar la experiencia de lectura del usuario. Entre ellas se destacan el modo lectura, la lectura simplificada, la guía visual de seguimiento y el resaltado de títulos.



Figura 7. *Página actual “Modos de lectura”*

Estas funciones buscan reducir la sobrecarga visual, mejorar la concentración y facilitar la comprensión del contenido textual, particularmente en entornos con alta densidad informativa.



Figura 8. *Página con “Modo de lectura” aplicado*

Funcionalidades de navegación:

El sistema incluye mecanismos que mejoran la orientación espacial y la interacción del usuario con la interfaz. Entre ellos se encuentran el resaltado de enlaces, la ampliación del cursor y el refuerzo visual del enfoque en elementos interactivos.

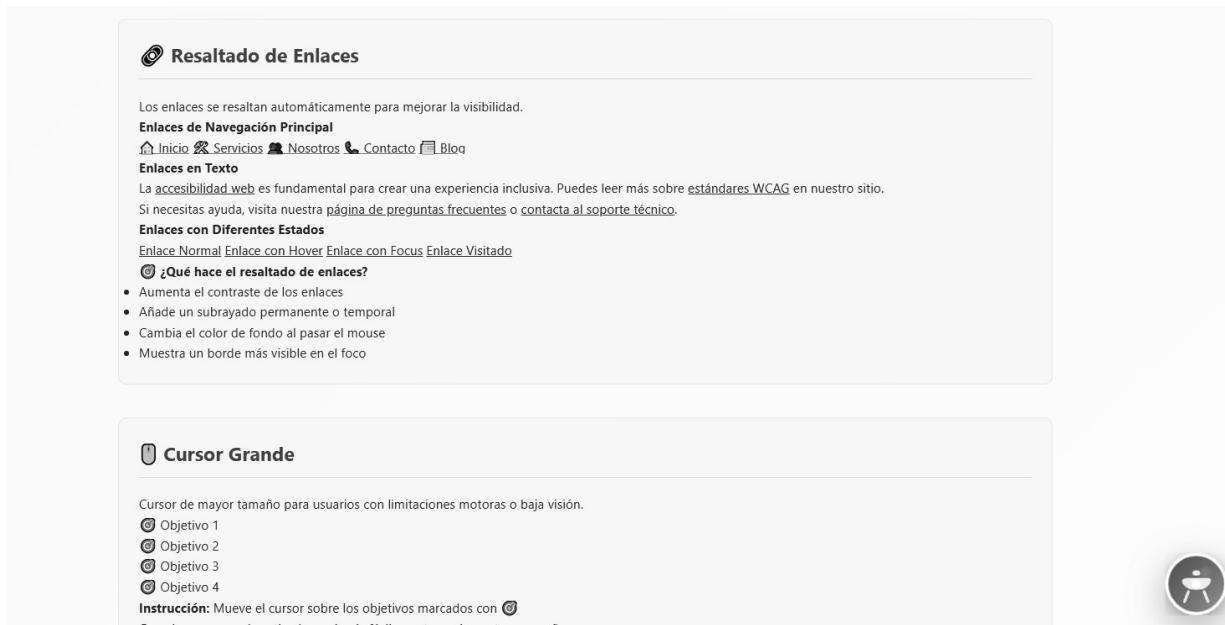


Figura 9. *Página actual “Resaltar enlaces”*

Estas herramientas resultan especialmente útiles para personas con baja visión o dificultades motoras leves, ya que facilitan la identificación de elementos clicables y mejoran la precisión en la interacción.

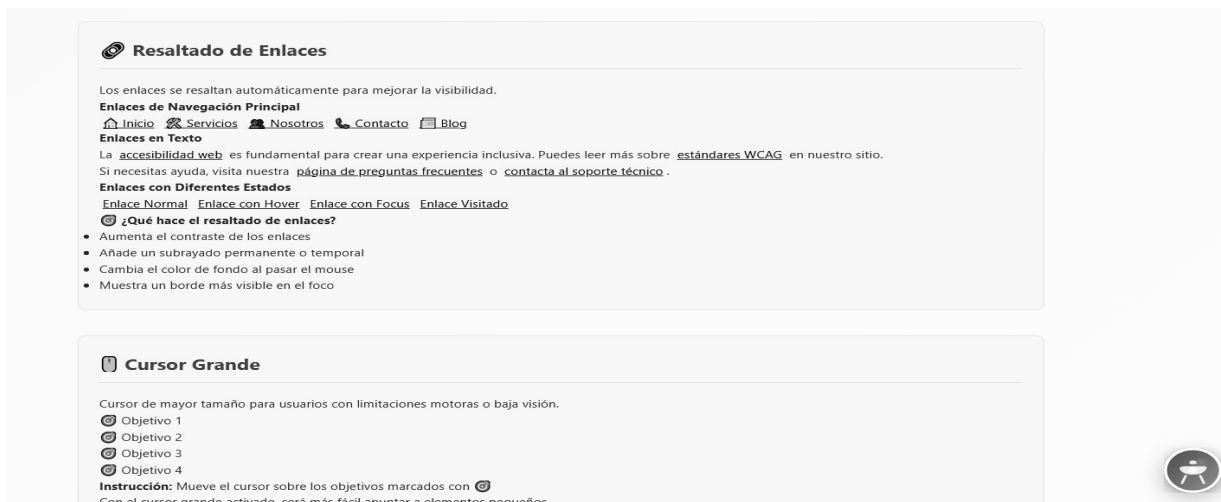


Figura 10. Página con “Resaltar enlaces” aplicado



Figura 11. Página con “Resaltar títulos” aplicado

Control de animaciones:

El widget permite pausar animaciones o reducir el movimiento presente en la interfaz. Esta funcionalidad responde a las recomendaciones internacionales destinadas a minimizar riesgos asociados a la epilepsia fotosensible y a trastornos vestibulares, conforme a los lineamientos establecidos por WCAG 2.1 (W3C, 2018).

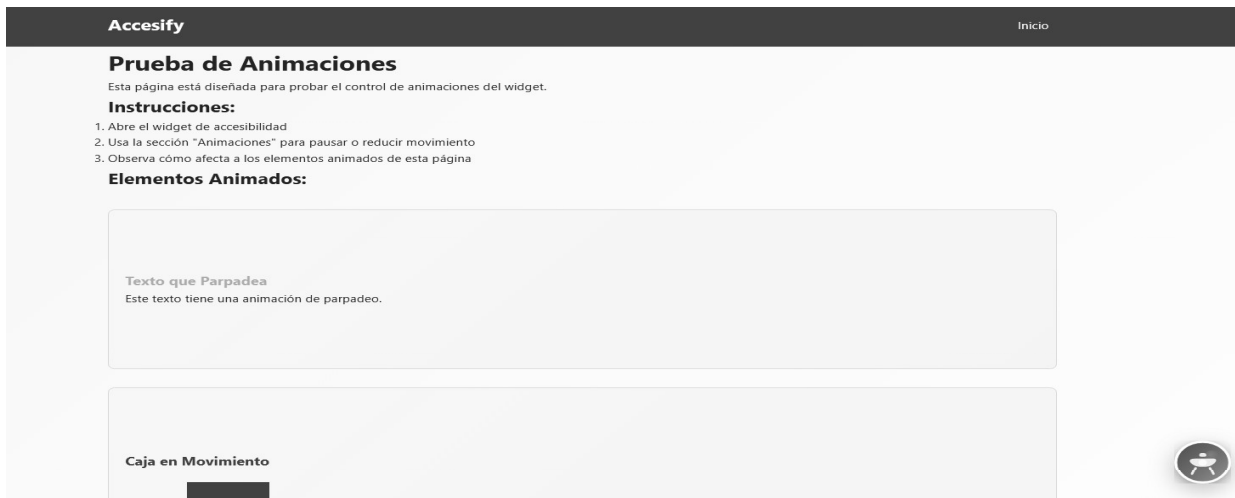


Figura 12. Página actual “Pruebas de animaciones”

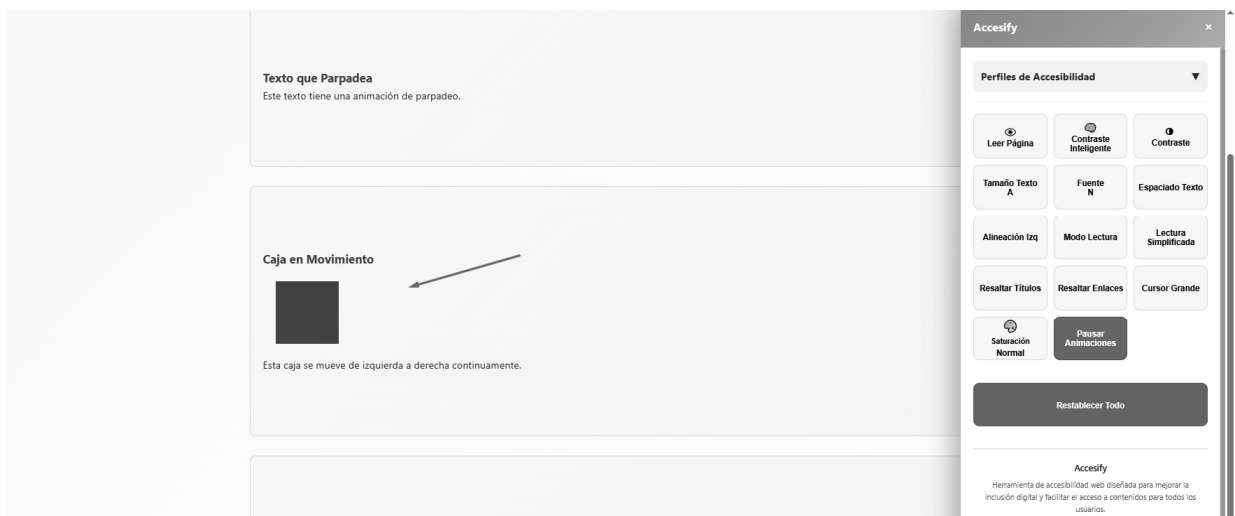


Figura 13. Página con “Pausar animaciones” aplicado

Perfiles de accesibilidad predefinidos:

Además de los ajustes individuales, el sistema implementa perfiles de accesibilidad configurables que activan simultáneamente múltiples funcionalidades según el tipo de necesidad del usuario. Entre los perfiles contemplados se incluyen baja visión, daltonismo, dislexia, epilepsia, dificultades de aprendizaje y TDAH.



Figura 14. *Página de Inicio Actual (sin estilos aplicados)*

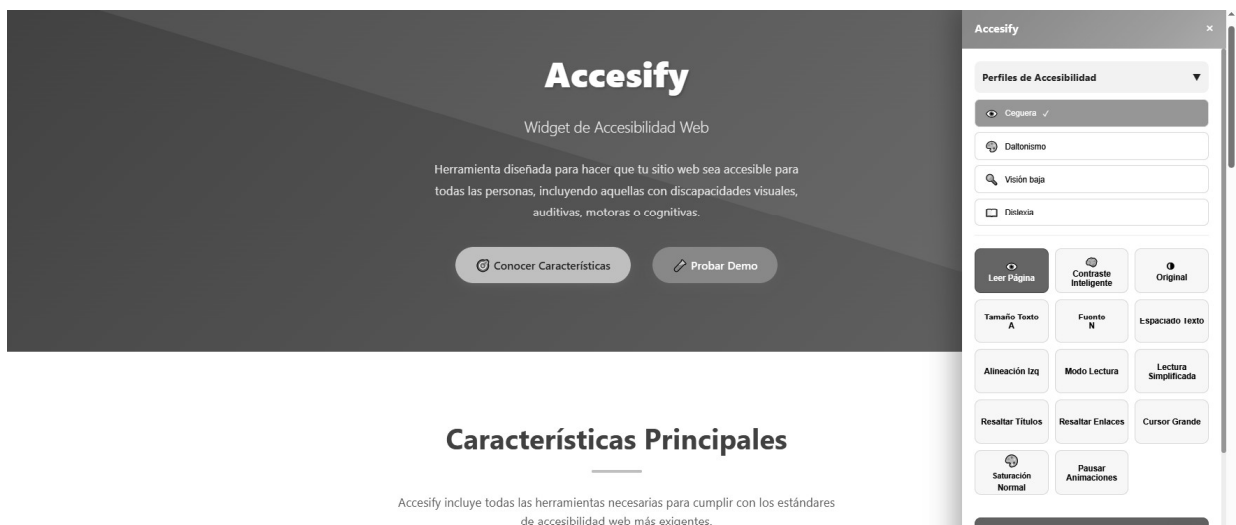


Figura 15. *Página con modo “Perfil Ceguera” activado*

Cada perfil aplica automáticamente combinaciones predefinidas de parámetros y clases CSS, permitiendo una configuración rápida y simplificada del entorno visual sin requerir ajustes manuales individuales.



Figura 16. *Página con modo “Perfil Daltonismo” activado*



Figura 17. *Página con modo “Perfil Visión baja” activado*

Persistencia de preferencias:

Finalmente, el widget incorpora un mecanismo de persistencia de preferencias

mediante almacenamiento local del navegador. A través del uso de almacenamiento en el cliente, las configuraciones seleccionadas por el usuario se conservan entre sesiones, sin necesidad de intervención del servidor ni almacenamiento en base de datos.

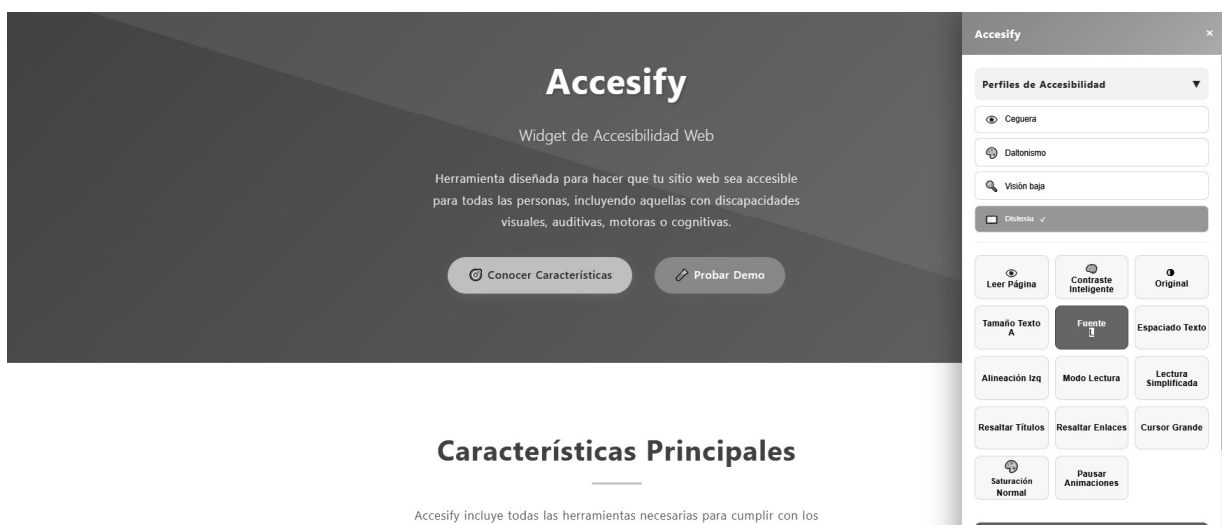


Figura 18. *Página con modo “Perfil Dislexia” activado*

4.2.2 Requerimientos no funcionales

El sistema **Accesify** debe cumplir con una serie de requisitos no funcionales que garantizan su calidad, rendimiento y adecuación al entorno web moderno.

Rendimiento

El widget debe ejecutarse sin generar impactos perceptibles en el tiempo de carga del sitio web anfitrión.

La aplicación de ajustes visuales debe realizarse de manera inmediata, sin requerir recarga de página y sin bloquear el hilo principal de ejecución del navegador.

Compatibilidad

El sistema debe ser compatible con los principales navegadores modernos en sus

versiones actualizadas.

Usabilidad

La interfaz del widget debe ser intuitiva, clara y de fácil interacción, permitiendo al usuario activar o desactivar funcionalidades sin conocimientos técnicos previos.

El diseño debe ser no intrusivo y permitir su posicionamiento configurable en distintas áreas de la pantalla.

Seguridad

El widget no debe alterar ni almacenar información sensible del usuario.

Las configuraciones se almacenan únicamente en el navegador mediante almacenamiento local, evitando la transmisión de datos personales a servidores externos.

Mantenibilidad

El código fuente debe estar estructurado de forma modular, permitiendo futuras extensiones, correcciones o incorporación de nuevas funcionalidades sin afectar la estabilidad del sistema.

Portabilidad

Accesify debe poder integrarse en distintos entornos web sin depender de configuraciones específicas del servidor.

Su distribución como paquete npm permite su reutilización en diferentes proyectos.

Escalabilidad funcional

El sistema debe permitir la incorporación de nuevos perfiles o ajustes de accesibilidad sin requerir rediseño completo de la arquitectura.

Cumplimiento normativo

El desarrollo del widget debe alinearse con las Pautas de Accesibilidad para el Contenido Web WCAG 2.1 establecidas por el W3C, promoviendo el cumplimiento de estándares internacionales de accesibilidad.

4.2.3 Diagrama de caso de uso

A continuación, se presenta el diagrama de casos de uso del sistema **Accesify**, el cual permite identificar al actor involucrado y las principales funcionalidades ofrecidas por el widget de accesibilidad web. Se tiene en cuenta un solo actor principal: el usuario del sitio web, quien interactúa con el widget para personalizar la experiencia de navegación, además este mismo es responsable de la integración, configuración y mantenimiento del widget.

El actor principal proporciona una visión general del alcance funcional del sistema y sirve como base para el análisis de su diseño y comportamiento.

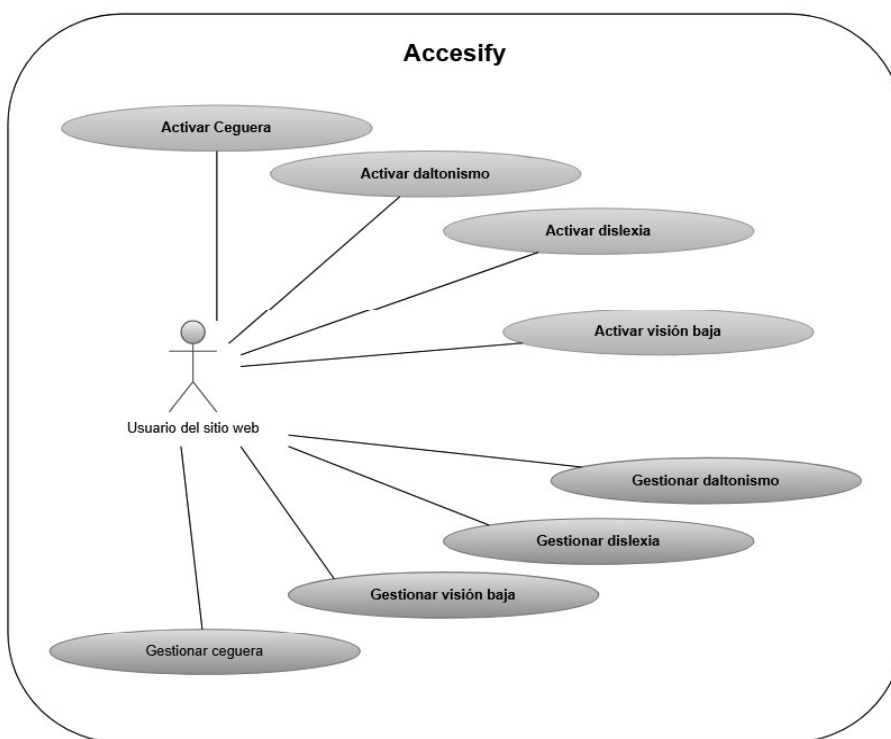


Figura 19. Diagrama de caso de uso

4.2.4 Diagrama de contexto

El diagrama de contexto presenta una visión general del widget **Accesify** y su interacción con los distintos actores y elementos externos. En él se identifican el usuario final, el sitio web donde se integra el widget, el navegador web como entorno de ejecución y el marco normativo que guía su funcionamiento. Este diagrama permite comprender el alcance del sistema sin detallar su implementación interna.

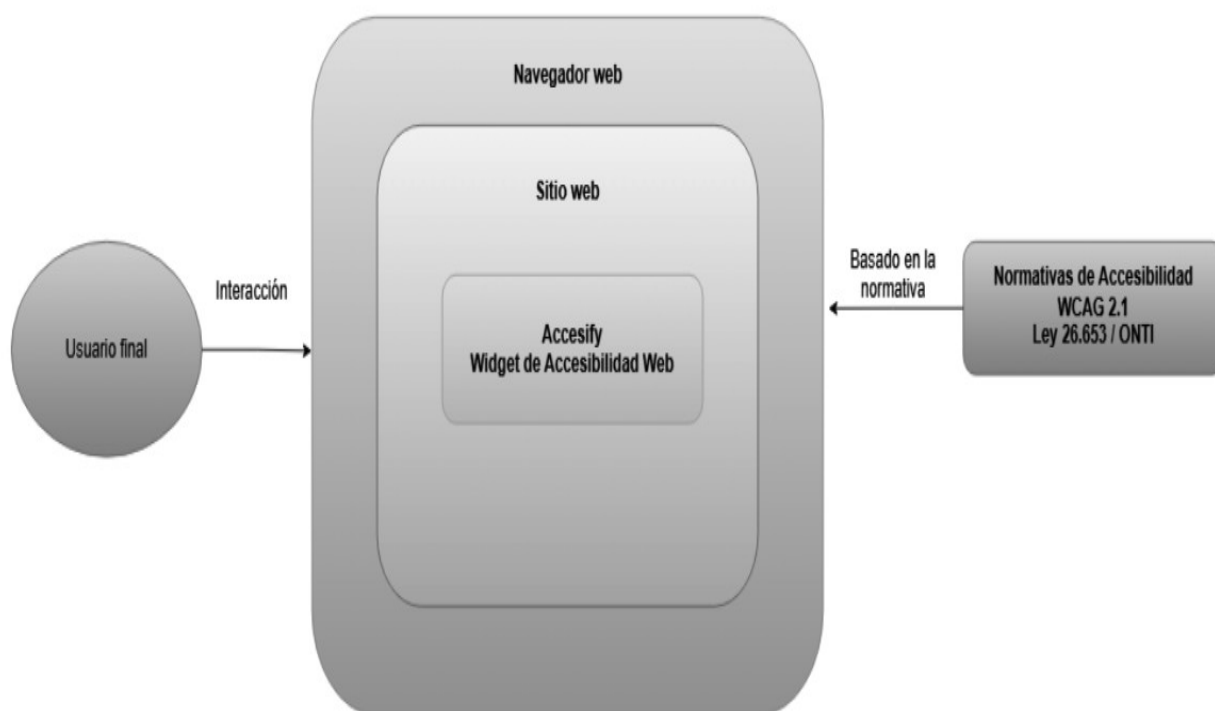


Figura 20. *Diagrama de caso de contexto*

4.2.5 Diagrama de funcionalidad del sistema

El diagrama de funcionalidad representa las principales capacidades del sistema **Accesify**, mostrando las funcionalidades de accesibilidad ofrecidas al usuario y su interacción con el sitio web donde el widget se integra. Este diagrama permite comprender qué acciones puede realizar el sistema sin detallar su implementación técnica.

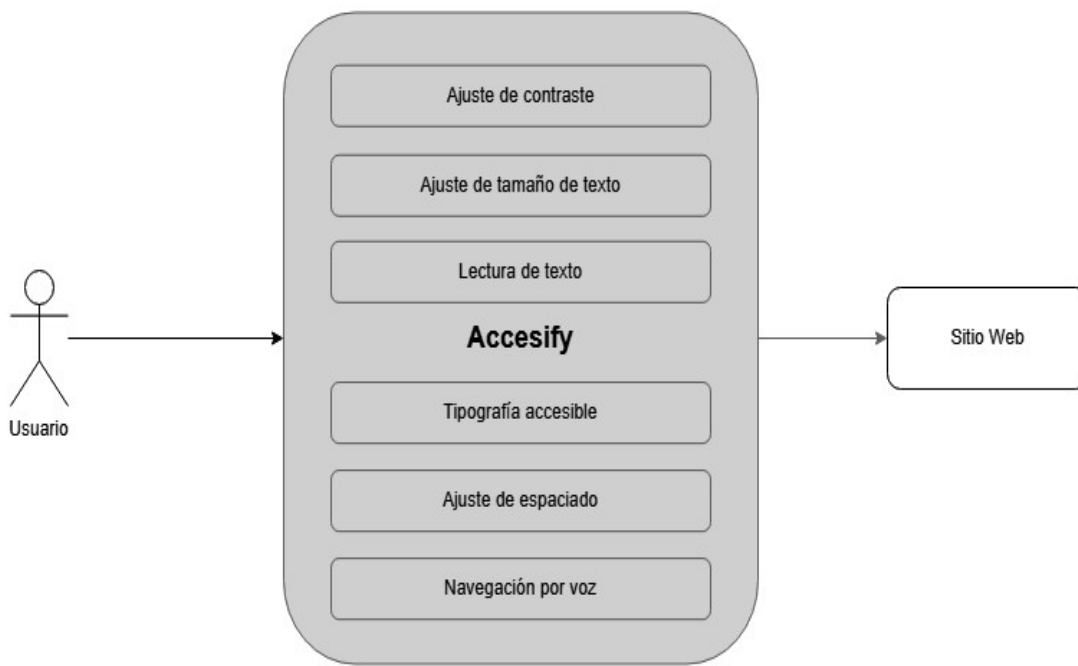


Figura 21. *Diagrama de caso de funcionalidad*

4.3 Tecnologías

El desarrollo de la herramienta **Accesify** se realizó utilizando un conjunto de tecnologías del ecosistema web moderno, enfocadas principalmente en la construcción de interfaces accesibles y fáciles de usar. La elección de estas tecnologías se basó en criterios como la facilidad de mantenimiento, compatibilidad con distintos navegadores y la posibilidad de ampliar las funcionalidades del sistema en el futuro.

La aplicación se desarrolló utilizando Laravel 8 como framework web, aprovechando su sistema de vistas para organizar el proyecto, sin necesidad de implementar un backend complejo ni manejar bases de datos.

Para la construcción de la interfaz se utilizó Blade, el motor de plantillas de Laravel, que permite mezclar código con HTML de forma clara, facilitando la reutilización de partes de la interfaz mediante layouts y secciones.

La estructura de la interfaz se armó utilizando HTML siguiendo buenas prácticas, permitiendo que sea compatible con herramientas de accesibilidad como lectores de pantalla y que también se pueda navegar usando el teclado. Los estilos se definieron con CSS, teniendo en cuenta aspectos como el contraste, la legibilidad y la adaptación a distintos tamaños de pantalla, siguiendo las recomendaciones de accesibilidad WCAG 2.1.

Por último, se utilizó JavaScript para manejar el comportamiento de la página, permitiendo que los cambios se apliquen de forma inmediata sin recargar, logrando una experiencia más fluida para el usuario.

4.3.1 Lenguajes de desarrollo

Los lenguajes seleccionados para el desarrollo de **Accesify** fueron PHP, JavaScript y CSS, utilizados de manera complementaria para construir una solución web robusta y orientada a la experiencia del usuario.

PHP fue empleado en el desarrollo del lado del servidor. Su madurez, amplia adopción en entornos web y compatibilidad con arquitecturas modernas justificaron su elección. Este lenguaje permite gestionar la lógica de negocio, el control de flujo de la aplicación y la interacción con la base de datos de manera estructurada y mantenible.

JavaScript fue utilizado en el desarrollo del lado del cliente, aportando dinamismo e interactividad a la interfaz. Su incorporación resulta especialmente relevante en una herramienta de accesibilidad, donde la respuesta inmediata ante las acciones del usuario constituye un factor determinante en la experiencia de uso.

CSS (Cascading Style Sheets) se empleó para definir la presentación visual del sistema, permitiendo separar la estructura y la lógica de la aplicación de los aspectos estéticos. Esta separación favorece un código más ordenado, reutilizable y fácil de mantener.

En el contexto específico de la accesibilidad web, el uso de CSS adquiere especial relevancia, ya que posibilita la adaptación dinámica de estilos relacionados con contraste, tamaño tipográfico, espaciado y disposición de elementos en pantalla. Asimismo, mediante la aplicación de principios de diseño responsivo, se aseguró la correcta visualización y funcionamiento del sistema en distintos dispositivos y resoluciones.

4.3.2 Framework

El proyecto se desarrolló utilizando Laravel 8 como framework web, aprovechando su estructura organizada y su implementación del patrón arquitectónico Modelo–Vista–Controlador (MVC). Esta herramienta permitió establecer una base sólida para la construcción de la aplicación, aun cuando el desarrollo no requirió la implementación de un backend completo ni la gestión de datos persistentes.

La elección al decidir usar Laravel se fundamenta en su capacidad para estructurar el código de manera clara y escalable, facilitando la separación de responsabilidades entre los distintos componentes del sistema. Asimismo, el framework ofrece mecanismos de seguridad integrados y herramientas que favorecen la mantenibilidad del proyecto, incluso en aplicaciones centradas principalmente en la interfaz de usuario.

Si bien en este trabajo el enfoque se orienta al desarrollo del frontend y al widget de accesibilidad, la utilización de Laravel permite sentar las bases para una futura expansión del sistema, posibilitando la incorporación de funcionalidades adicionales que requieran lógica del lado del servidor o persistencia de datos.

En este sentido, el uso del framework no sólo contribuye a la organización del proyecto, sino que también garantiza su adaptabilidad y escalabilidad en escenarios

de mayor complejidad.

4.4 Fase de desarrollo

Una vez seleccionadas las tecnologías, se definieron las estructuras que conforman el frontend del sistema, con el objetivo de implementar las funcionalidades previstas de manera ordenada y eficiente. A partir de la base proporcionada por Laravel 8, se organizaron los componentes de la aplicación siguiendo una estructura clara y modular, permitiendo desarrollar el widget de accesibilidad de forma desacoplada e independiente de la lógica del servidor.

Esta organización favorece la trazabilidad del desarrollo, permitiendo identificar de manera clara la relación entre los distintos componentes del sistema y las funcionalidades implementadas. Asimismo, la estructura modular adoptada contribuye a la escalabilidad del proyecto, ya que facilita la incorporación de nuevas funcionalidades o la adaptación del sistema a diferentes contextos de uso.

De este modo, se logró un desarrollo ágil centrado en la interfaz de usuario, manteniendo al mismo tiempo una organización del código que favorece la mantenibilidad y la posibilidad de expandir el sistema en futuras versiones.

4.4.1 Arquitectura MVC del sistema

Durante la fase de desarrollo se analizó la necesidad de contar con una estructura de proyecto clara y mantenible, que permitiera organizar de forma adecuada las distintas responsabilidades del sistema. Para ello, se adoptó una arquitectura basada en el patrón Modelo–Vista–Controlador (MVC), el cual facilita la separación entre la lógica de negocio, la presentación y el control de la aplicación.

La estructura del proyecto se organizó de la siguiente manera:

- **Modelo:** encargado de representar y gestionar los datos del sistema.
- **Vista:** dedicada a la presentación de la interfaz de usuario, utilizando plantillas dinámicas que permiten mantener separada la lógica de la visualización.
- **Controlador:** responsable de procesar las solicitudes, coordinar la lógica de la aplicación y determinar la respuesta adecuada.

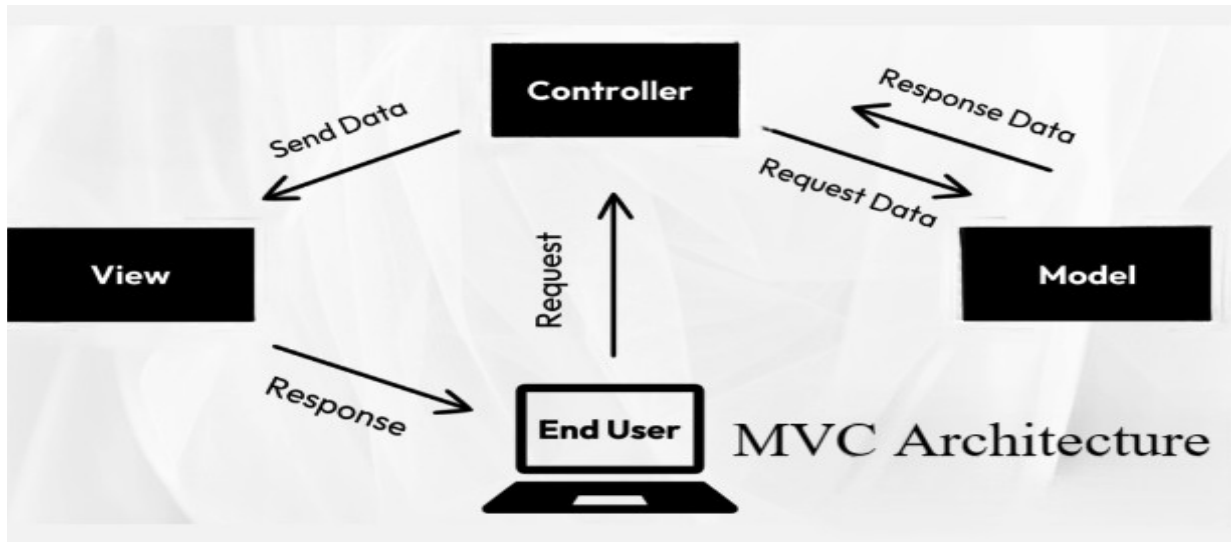


Figura 22. *Fundamentos de Laravel*

Este enfoque permitió mantener el código desacoplado. Además, la utilización de una estructura predefinida redujo la complejidad inicial del desarrollo y permitió concentrar el esfuerzo en la implementación de las funcionalidades de accesibilidad propias de la herramienta.

4.4.2 Organización del proyecto

La estructura física del proyecto sigue la organización estándar propuesta por Laravel, separando claramente el código fuente de los archivos generados para producción.

Los principales directorios utilizados en el desarrollo fueron los siguientes:

- **app/**: contiene la lógica principal de la aplicación, incluyendo modelos y controladores que gestionan el comportamiento del sistema.
- **routes/**: define las rutas de acceso a la aplicación. En particular, el archivo web.php gestiona las rutas orientadas a vistas, mientras que api.php permite definir endpoints para consumo programático.
- **resources/**: alberga los recursos editables del frontend:
 - **resources/js/**: contiene los archivos JavaScript fuente.
 - **resources/css/** o **resources/sass/**: incluye los estilos del sistema.
 - **resources/views/**: almacena las plantillas Blade utilizadas para la generación dinámica de vistas.
 - **components/**: contiene componentes reutilizables, como el widget de accesibilidad.
 - **layouts/**: define estructuras base reutilizables para las vistas.
 - **resources/lang/**: gestiona archivos de internacionalización y mensajes del sistema.
- **public/**: constituye el punto de acceso público de la aplicación. Contiene los archivos compilados (CSS, JavaScript, imágenes) y el archivo index.php, que actúa como punto de entrada del framework.

4.4.3 Empaquetado y bundling con Webpack

Para generar un archivo distribuable multiplataforma se empleó Webpack como herramienta de empaquetado.

La configuración del output permite definir el archivo de salida, su ubicación y el tipo de módulo exportado. Un fragmento representativo de esta configuración se presenta en el Apéndice (Sección 7.1).

La elección del formato UMD (Universal Module Definition) permite que el componente pueda ser utilizado en distintos entornos de ejecución:

- Como módulo CommonJS (Node.js)
- Como módulo AMD
- Como objeto global accesible mediante la etiqueta <script>

El proceso de compilación se realiza mediante comandos definidos en el entorno de desarrollo, los cuales se detallan en el Apéndice (Sección 7.1).

El resultado es un archivo optimizado ubicado en el directorio dist/, el cual expone el objeto global Accesify.

4.4.4 Gestión de dependencias con npm

El componente de accesibilidad fue diseñado como un módulo desacoplado y reutilizable, independiente de la aplicación principal. Para su gestión de dependencias y distribución se utilizó el gestor de paquetes npm, estándar en el ecosistema de Node.js.

Node.js es un entorno de ejecución que permite ejecutar código JavaScript fuera del navegador, construido sobre el motor V8 de Google Chrome. Este entorno proporciona una arquitectura basada en eventos y un modelo de entrada y salida no bloqueante (non-blocking I/O), permitiendo construir aplicaciones escalables y eficientes. Dentro de este ecosistema, npm (Node Package Manager) funciona como el gestor de paquetes oficial, permitiendo instalar, compartir y administrar dependencias utilizadas durante el desarrollo de aplicaciones JavaScript, tal como se describe en su documentación oficial [19].

Inicialización y configuración del paquete

El proyecto fue inicializado mediante comandos específicos del gestor npm, los cuales se detallan en el Apéndice (Sección 7.2).

Este proceso genera el archivo `package.json`, que define los metadatos del paquete (nombre, versión, punto de entrada, scripts y dependencias). Un ejemplo de esta configuración puede consultarse en el Apéndice (Sección 7.2).

Las dependencias necesarias para el empaquetado fueron instaladas como dependencias de desarrollo, tal como se muestra en el Apéndice (Sección 7.2).

Se optó por clasificarlas como dependencias debido a que son utilizadas únicamente durante el proceso de construcción y no en tiempo de ejecución del widget.

4.4.5 Versionado y publicación en npm

El proceso de publicación del paquete se realiza mediante comandos del gestor npm, los cuales se detallan en el Apéndice (Sección 7.3).

Una vez generado el bundle, el paquete fue publicado en el registro público de npm.

Cada nueva versión requiere actualizar el número de versión siguiendo el esquema de versionado semántico (SemVer), procedimiento que también se detalla en el Apéndice (Sección 7.3).

Este mecanismo permite mantener control sobre la evolución del componente y garantizar compatibilidad hacia atrás.

Instalación en un proyecto consumidor

Una vez publicado, el widget puede ser incorporado en cualquier aplicación mediante comandos de instalación, los cuales se presentan en el Apéndice (Sección 7.3).

Su utilización puede realizarse tanto como módulo como mediante inclusión directa en el navegador, según se describe en el Apéndice (Sección 7.3).

Esta versatilidad garantiza su integración en proyectos con distintos modelos de arquitectura.

4.4.6 Automatización de instalación

Adicionalmente, se desarrolló un script de instalación automatizado utilizando Node.js, el cual copia los recursos necesarios al directorio público del proyecto consumidor.

Este procedimiento se ejecuta mediante comandos específicos, los cuales se detallan en el Apéndice (Sección 7.4).

Esta estrategia reduce errores manuales de configuración y facilita la adopción del componente en distintos entornos.

4.4.7 Justificación técnica

La utilización de npm como mecanismo de distribución presenta las siguientes ventajas:

- Gestión centralizada de versiones
- Instalación automatizada de dependencias
- Distribución desacoplada del proyecto principal
- Reutilización en múltiples aplicaciones
- Integración con pipelines de integración continua

El empleo de un bundle en formato UMD complementa esta estrategia, permitiendo compatibilidad tanto con sistemas modernos basados en módulos como con entornos tradicionales basados en scripts globales.

4.4.8 Diseño de interfaz y experiencia de usuario

Interfaces gráficas de la herramienta

Características del panel Flotante Principal

- **Diseño minimalista y no intrusivo:** icono circular o cuadrado con controles expandibles.
- **Posicionamiento configurable:** esquinas de la pantalla (superior/inferior, izquierda/derecha)
- **Funcionalidades a la vista:** el panel flotante principal presenta de forma clara las distintas herramientas de accesibilidad disponibles. Cada funcionalidad se representa mediante botones o íconos intuitivos que

permiten identificar rápidamente su propósito sin requerir explicaciones adicionales. Entre las principales opciones visibles se encuentran los controles para ajustar el contraste de la página, aumentar o disminuir el tamaño del texto, resaltar enlaces y mejorar la visibilidad del cursor.

Asimismo, el panel permite activar diferentes modos de lectura y navegación que facilitan la interacción con el contenido del sitio web. Estas funcionalidades se aplican de manera inmediata sobre la página cargada en el navegador, permitiendo al usuario adaptar la visualización según sus necesidades. La disposición de los controles busca priorizar la claridad y la facilidad de uso, permitiendo que las herramientas sean accesibles incluso para usuarios con poca experiencia en el uso de tecnologías digitales.

- **Sistema de anclaje:** permite reposicionamiento manual por arrastre.

En la figura 23 se observa el widget **Accesify** con el panel de herramientas desplegado.

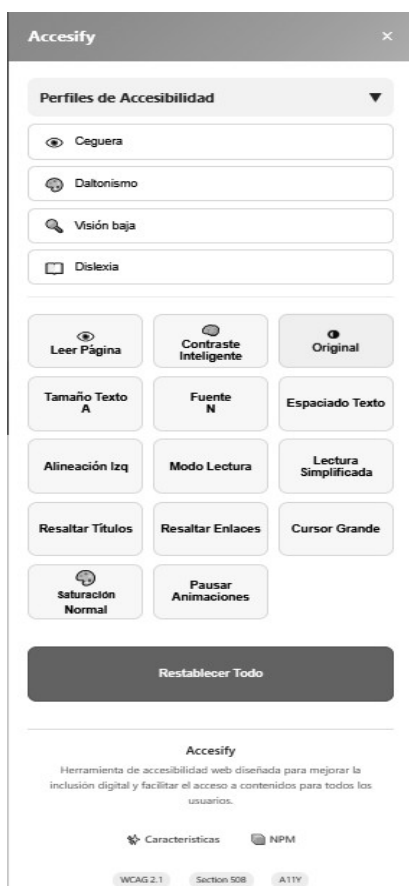


Figura 23. Funcionalidades de **Accesify**.

Propósito de cada Botón:

- **de “Ceguera”:** permite a los usuarios ingresar la URL que desean evaluar para verificar su accesibilidad según los estándares definidos.
- **de “Daltonismo”:** facilita la carga de un archivo TXT que contiene instrucciones detalladas para las acciones personalizadas que se ejecutarán durante la evaluación. Esto permite una evaluación más precisa y específica de la accesibilidad web en escenarios específicos. (imagen 3.4)
- **de “Dislexia”:** inicia el proceso de análisis y evaluación de accesibilidad para la URL proporcionada, utilizando las configuraciones y acciones seleccionadas por el usuario para guiar el proceso.
- **de “Visión baja”:** activa una serie de configuraciones orientadas a la evaluación de la accesibilidad para personas con visión reducida, considerando aspectos como contraste, escalabilidad del texto y claridad visual de los elementos, permitiendo detectar posibles barreras de acceso en la interfaz analizada.

Diseño y Usabilidad:

Para el diseño del encabezado de la página se tuvo en cuenta la usabilidad y la claridad, garantizando que los usuarios puedan interactuar fácilmente con su contenido web y acceder a las herramientas necesarias para personalizar la experiencia de accesibilidad de manera intuitiva y eficiente.

4.4.9 Subir el widget a producción

Accesify fue publicado como paquete en npmjs, una plataforma ampliamente utilizada para distribuir librerías de JavaScript. Esto permite que el widget pueda ser instalado e integrado fácilmente en otros proyectos mediante un gestor de paquetes.

La publicación en NPM facilita la reutilización del componente, permitiendo que otros desarrolladores lo instalen en sus aplicaciones sin necesidad de copiar manualmente el código.

El paquete se encuentra disponible en la siguiente URL:

<https://www.npmjs.com/package/accesify?activeTab=code>

Por otro lado, se evaluó el uso de plataformas de despliegue como Vercel, que ofrecen integración continua y despliegue automático, facilitando la publicación de aplicaciones web. Sin embargo, estas plataformas presentan ciertas limitaciones para este proyecto.

Una de las principales limitaciones es la imposibilidad de contar con un entorno completo de Google Chrome para ejecutar pruebas de accesibilidad automatizadas directamente en el servidor. Este inconveniente puede resolverse parcialmente mediante el uso de herramientas como Puppeteer, que permite controlar navegadores de forma programada.

Debido a estas limitaciones, a futuro se podría considerar el uso de un servidor propio, brindando mayor control sobre el entorno de ejecución y permitiendo optimizar las pruebas de accesibilidad.

Adicionalmente, se publicó una instancia de demostración del widget en un sitio web accesible públicamente, con el objetivo de permitir la visualización e interacción con el componente en un entorno real de uso. Esta página permite observar el funcionamiento del widget, su interfaz y las distintas funcionalidades de accesibilidad implementadas.

La demostración del widget se encuentra disponible en el siguiente enlace:

<http://accesify.page.gd/>

4.4.10 Diseño modular y desacoplamiento del componente

El desacoplamiento del componente se implementa mediante la separación de responsabilidades entre los distintos elementos que lo conforman, diferenciando la lógica de funcionamiento, la presentación visual y los recursos necesarios para su ejecución. Esta organización permite incorporar el widget en un sitio web mediante la inclusión de los archivos generados durante el proceso de compilación, evitando interferencias con el código existente de la aplicación y reduciendo la posibilidad de conflictos con otras dependencias.

Asimismo, la arquitectura modular adoptada favorece la mantenibilidad y evolución del sistema, ya que posibilita realizar modificaciones o incorporar nuevas funcionalidades al componente sin afectar el funcionamiento general del sitio web en el que se encuentra integrado. Este aspecto resulta especialmente relevante en el contexto de herramientas orientadas a la accesibilidad, donde la actualización constante en función de nuevos estándares y recomendaciones constituye un factor importante para garantizar su vigencia y utilidad.

En consecuencia, el diseño modular y el desacoplamiento del componente contribuyen a mejorar la portabilidad, escalabilidad y mantenibilidad del sistema, permitiendo que el widget pueda ser integrado y utilizado en distintos entornos de desarrollo web de manera flexible y eficiente.

4.5 Proceso de pruebas

En esta fase del proyecto, se llevarán a cabo pruebas para garantizar la eficacia y la calidad de la herramienta desarrollada y evaluar su funcionamiento frente a la accesibilidad de diferentes páginas web.

Pruebas de la herramienta

Se realizaron pruebas con la herramienta para validar su funcionalidad. Estas pruebas se realizan con Laravel, a través de distintas pantallas donde se podrá visualizar cada tipo de funcionalidad a la perfección.

Se seleccionó esta forma de testear por varias razones. En primer lugar, **Accesify** se integra directamente en la web diseñada para hacer estas pruebas, esto permite una ejecución de pruebas significativamente más rápida, moviéndose de pantalla en pantalla de forma simple. Además, proporciona una experiencia de desarrollo más amigable, con una configuración inicial más simple y una excelente documentación, ya que todo está desarrollado en la misma plataforma, tanto el widget como la página web de testeo.

Para facilitar la verificación del comportamiento del widget, se desarrolló una página web de pruebas dentro del mismo entorno del proyecto. Esta página contiene distintos tipos de elementos visuales como títulos, párrafos, enlaces, botones, imágenes y bloques de contenido, permitiendo observar cómo afectan las diferentes funcionalidades del widget sobre cada tipo de componente.

Este entorno de prueba permitió evaluar de forma rápida y controlada el funcionamiento de **Accesify**, verificando visualmente los cambios generados por cada opción del panel de accesibilidad.

El objetivo es asegurar que la herramienta sea robusta, confiable y permite hacer adaptaciones instantáneas para que determinados contenidos sean accesibles según necesidades específicas.

Asimismo, la página de pruebas se diseñó con diferentes tipos de contenido y estructuras de interfaz, permitiendo comprobar cómo se comporta el widget frente a distintas situaciones reales de uso, como páginas con gran cantidad de texto,

interfaces con múltiples enlaces o contenido visual con diferentes niveles de contraste.

4.5.1 Pruebas funcionales

Se realizaron pruebas funcionales con el objetivo de verificar el correcto comportamiento de cada una de las funcionalidades del widget Accesify. Estas pruebas consistieron en:

1. Activación individual de cada perfil de accesibilidad.
2. Verificación visual de los cambios en la interfaz.
3. Validación de compatibilidad entre funcionalidades simultáneas.
4. Comprobación del comportamiento en diferentes resoluciones de pantalla.

Las pruebas se ejecutaron en un entorno controlado dentro del proyecto, utilizando distintas vistas de prueba diseñadas específicamente para evaluar contraste, tipografía, escalabilidad y disposición de elementos.

Durante estas pruebas se verificó que cada funcionalidad del widget aplique correctamente los cambios sobre los elementos de la página, como el ajuste de contraste, modificación del tamaño de texto, cambio de tipografías y resaltado de enlaces. También se comprobó que las funcionalidades pudieran activarse y desactivarse sin afectar el funcionamiento general del sitio.

Además, se evaluó el comportamiento del widget cuando varias opciones de accesibilidad se encuentran activas al mismo tiempo, con el objetivo de asegurar que no se generen conflictos entre ellas y que los cambios aplicados se mantengan estables.

4.5.2 Casos de prueba

Con el fin de validar el funcionamiento del widget Accesify, se definieron distintos casos de prueba que permiten verificar el comportamiento de sus funcionalidades en diferentes escenarios de uso.

Caso de prueba 1: Activación del modo de alto contraste

- **Precondiciones:**

El sitio web se encuentra cargado en el navegador y el widget Accesify está visible en la interfaz.

- **Pasos:**

1. Abrir el panel del widget.
2. Seleccionar la opción de alto contraste.

- **Resultado esperado:**

La interfaz del sitio modifica su esquema de colores, aumentando el contraste entre el texto y el fondo para mejorar la legibilidad.

Caso de prueba 2: Aumento del tamaño de texto

- **Precondiciones:**

El widget Accesify se encuentra activo en la página.

- **Pasos:**

1. Abrir el panel del widget.
2. Seleccionar la opción aumentar tamaño de texto.

- **Resultado esperado:**

El tamaño del texto del contenido principal aumenta sin romper la estructura de la página ni afectar la disposición de los elementos.

Caso de prueba 3: Activación simultánea de funcionalidades

- **Precondiciones:**

El widget Accesify se encuentra cargado y funcionando correctamente.

- **Pasos:**

1. Activar el modo de alto contraste.
2. Activar el aumento del tamaño de texto.
3. Activar el resaltado de enlaces.

- **Resultado esperado:**

Las funcionalidades se aplican de manera simultánea sin generar conflictos visuales ni afectar el funcionamiento del sitio.

Caso de prueba 4: Compatibilidad con diferentes resoluciones

- **Precondiciones:**

El sitio web se encuentra cargado y el widget Accesify está disponible.

- **Pasos:**

1. Abrir la página en distintas resoluciones de pantalla (escritorio, tablet y móvil).
2. Activar diferentes funcionalidades del widget.

- **Resultado esperado:**

El widget mantiene su posición y funcionamiento correcto en todas las resoluciones evaluadas.

4.5.3 Pruebas de integración

Las pruebas de integración se realizaron incorporando el widget en una aplicación web de prueba desarrollada en Laravel. Para ello se compilaron los archivos JavaScript y CSS mediante Laravel Mix y se verificó su correcta generación dentro del directorio **public**. Posteriormente se incluyeron los archivos compilados en la plantilla principal de la aplicación y se accedió a distintas páginas del sitio para comprobar que el widget se cargara correctamente en el navegador.

Durante las pruebas se utilizó la consola de desarrollo del navegador para revisar la correcta carga de los scripts, la ausencia de errores en JavaScript y la aplicación dinámica de las clases CSS sobre el DOM.

Se verificó la correcta integración entre:

- Frontend JavaScript.
- Archivos compilados mediante Laravel Mix.
- Sistema de carga de assets desde el directorio **public/**.

Estas pruebas permitieron comprobar que el widget pueda cargarse correctamente dentro de la aplicación web sin interferir con el funcionamiento normal del sitio. También se constató que los archivos de estilos y scripts necesarios para el funcionamiento del widget se carguen correctamente al momento de renderizar la página.

Adicionalmente, se comprobó que la inclusión del componente dentro de distintas vistas de Laravel no genera conflictos con otros scripts o estilos existentes en la aplicación.

4.5.4 Pruebas de usabilidad

Para evaluar la usabilidad del widget se realizaron pruebas en distintas páginas web de prueba donde se integró Accesify. Durante estas pruebas se interactuó con el panel de accesibilidad simulando el uso que podría realizar un usuario al navegar por el sitio.

Las pruebas se llevaron a cabo siguiendo los siguientes pasos:

1. Acceder a una página web con el widget integrado.
2. Identificar visualmente el panel flotante de accesibilidad.
3. Activar y desactivar las distintas funciones del widget.
4. Observar los cambios generados en la interfaz del sitio.
5. Verificar que el panel no interfiera con el contenido principal de la página.

Durante este proceso se buscó comprobar que el widget resulte intuitivo para el usuario y que las opciones de accesibilidad puedan encontrarse y utilizarse fácilmente. También se evaluó la claridad de los iconos y textos utilizados en cada opción del widget, asegurando que los usuarios comprendan de forma intuitiva la función de cada herramienta disponible.

Se evaluó:

- Claridad visual del panel flotante.
- Facilidad de activación y desactivación de las funciones.
- Comprensibilidad de los iconos.
- No interferencia con el contenido principal.

5 Conclusiones

5.1 Resumen final

La accesibilidad web se ha consolidado como un aspecto fundamental en el desarrollo de tecnologías digitales, especialmente en un contexto donde Internet constituye uno de los principales medios de acceso a la información, los servicios y la educación. Sin embargo, a pesar de los avances tecnológicos, aún existen numerosas barreras que dificultan el acceso pleno a los contenidos web para personas con diferentes tipos de discapacidades, evidenciando la necesidad de continuar promoviendo prácticas de desarrollo inclusivas.

A lo largo de este trabajo se abordó la problemática de la accesibilidad web, analizando su evolución, su marco normativo y los estándares internacionales establecidos por el World Wide Web Consortium (W3C) a través de las Pautas de Accesibilidad para el Contenido Web (WCAG). Asimismo, se consideró el contexto normativo de la República Argentina, particularmente la Disposición ONTI 6/2019, que establece criterios y recomendaciones para garantizar que los sitios web de organismos públicos sean accesibles para todos los ciudadanos.

En este marco, se desarrolló **Accesify**, un widget de accesibilidad web diseñado para facilitar la adaptación de los sitios web a diferentes necesidades de los usuarios. Esta herramienta permite incorporar funcionalidades orientadas a mejorar la experiencia de navegación, como ajustes de contraste, modificaciones de visualización y otras opciones destinadas a reducir barreras de acceso para personas con discapacidades visuales o dificultades de lectura.

Una de las principales características del widget desarrollado es su enfoque modular y reutilizable, ya que permite integrarlo fácilmente en diferentes aplicaciones web. Además, su implementación busca simplificar la adopción de prácticas de accesibilidad por parte de desarrolladores y organizaciones, contribuyendo así al cumplimiento de los estándares de accesibilidad y a la promoción de un entorno

digital más inclusivo.

Finalmente, este trabajo no sólo permitió profundizar en los fundamentos teóricos y normativos de la accesibilidad web, sino también aplicar dichos conocimientos en el desarrollo de una solución concreta orientada a mejorar la accesibilidad de los contenidos digitales. En este sentido, el proyecto busca aportar una herramienta que contribuya a la construcción de una Web más accesible, equitativa e inclusiva, donde todas las personas puedan acceder a la información y a los servicios digitales sin enfrentar ningún tipo de barreras tecnológicas.

5.2 Futuras investigaciones

Este trabajo final constituye un primer paso hacia el desarrollo de herramientas orientadas a mejorar la accesibilidad en entornos web. A partir de los resultados obtenidos, pueden identificarse diversas líneas de investigación que podrían profundizar y ampliar el alcance de la propuesta presentada.

Una posible línea de trabajo consiste en analizar la incorporación de funcionalidades orientadas a **otras formas de discapacidad**, como la discapacidad auditiva. En este sentido, podrían explorarse herramientas que faciliten la accesibilidad de contenidos multimedia mediante la generación automática de subtítulos, transcripciones de audio o sistemas de interpretación en lengua de señas.

Otra línea de investigación podría centrarse en la **extensión de las funcionalidades del widget de accesibilidad desarrollado**, incorporando nuevas opciones de configuración orientadas a diferentes necesidades de los usuarios. Entre ellas podrían incluirse configuraciones avanzadas de lectura, mejoras en el reconocimiento de voz o mecanismos de asistencia basados en inteligencia artificial.

Asimismo, futuras investigaciones podrían analizar la **integración del widget con diferentes plataformas y frameworks web**, evaluando su compatibilidad con gestores de contenido (CMS) y otros entornos de desarrollo. Este tipo de estudios permitiría facilitar su implementación en distintos tipos de sitios web y contextos tecnológicos.

6 Anexo

6.1 Guía de acrónimos y términos

- **Backend:** es parte de una aplicación web que funciona del lado del servidor. Se encarga de procesar la lógica del sistema, manejar los datos y responder a las solicitudes que realiza el usuario desde la interfaz. Generalmente incluye la conexión con bases de datos, la validación de información y la ejecución de operaciones necesarias para el funcionamiento de la aplicación.
- **Click:** en el contexto de la interacción con una interfaz de usuario, un "click" es una acción realizada por un usuario al presionar y soltar el botón del ratón (en una computadora) o tocar la pantalla (en dispositivos táctiles). Un clic se utiliza comúnmente para activar elementos interactivos como botones, enlaces y controles de formulario en aplicaciones y páginas web.
- **CSS:** cascading Style Sheets es un lenguaje de hojas de estilo utilizado para definir el aspecto y el formato visual de documentos HTML y XML. CSS permite a los desarrolladores web controlar la presentación de contenido web, incluyendo la tipografía, los colores, los márgenes, el espaciado, el diseño de la página y otros aspectos visuales. Con CSS, los desarrolladores pueden separar la estructura del contenido de su presentación visual, facilitando la creación de sitios web bien diseñados y mantenibles.
- **DOM:** el Document Object Model es una representación programática de la estructura y el contenido de un documento HTML o XML. El DOM organiza el contenido del documento en una estructura de árbol de nodos, donde cada nodo representa un elemento, atributo o texto dentro del documento. El DOM proporciona una interfaz que permite a los desarrolladores acceder, modificar y manipular dinámicamente el contenido, la estructura y el estilo de una página web utilizando código JavaScript u otros lenguajes de programación.
- **Frontend:** el "Frontend" se refiere a la parte de un sistema informático o una aplicación web que se encarga de la presentación y la interacción con el

usuario final. En el contexto de una aplicación web, el frontend incluye la interfaz de usuario, el diseño visual, la experiencia del usuario y cualquier otra funcionalidad que se ejecute en el navegador del cliente. El frontend se diferencia del backend, que se encarga de procesar los datos y realizar operaciones lógicas y de negocio en el servidor.

- **GitHub:** es una plataforma utilizada para alojar proyectos de software y gestionar el control de versiones del código. Se basa en Git, un sistema que permite registrar y organizar los cambios realizados en los archivos a lo largo del tiempo, evitando la pérdida de información y facilitando el trabajo ordenado sobre un mismo proyecto.

Esta herramienta es ampliamente utilizada en el desarrollo de software debido a las siguientes ventajas:

Control de versiones: permite llevar un registro de las distintas modificaciones realizadas sobre el código y volver a versiones anteriores en caso de ser necesario.

Trabajo colaborativo: facilita el desarrollo conjunto entre varias personas, permitiendo que cada una aporte cambios sin interferir con el trabajo de los demás.

Visibilidad del proyecto: posibilita publicar y compartir proyectos, resultando útil para mostrar trabajos realizados o permitir que otros los revisen y reutilicen.

Integración y despliegue: muchas herramientas y servicios pueden conectarse directamente con GitHub para automatizar procesos de prueba e implementación.

El proyecto final desarrollado en este trabajo se encuentra disponible en un repositorio público de GitHub, permite su consulta, descarga y reutilización:

URL: <https://github.com/facuferrara/accesify>

- **Google Chrome:** es un navegador web desarrollado por Google. Es uno de los navegadores más populares y ampliamente utilizados en todo el mundo. Google Chrome ofrece una interfaz de usuario intuitiva, un rendimiento rápido, una amplia compatibilidad con estándares web modernos y una amplia gama de funciones y extensiones que mejoran la experiencia de navegación para los

usuarios. Google Chrome está disponible para múltiples plataformas, incluyendo Windows, macOS, Linux, Android e iOS.

- **HTML:** HyperText Markup Language es el lenguaje estándar utilizado para crear y diseñar páginas web y aplicaciones web. HTML utiliza etiquetas y elementos para estructurar y organizar el contenido de una página web, incluyendo texto, imágenes, enlaces, formularios y otros elementos interactivos. HTML proporciona una estructura básica para el contenido web, que luego se estiliza y se mejora con CSS (Cascading Style Sheets) y se hace interactivo con JavaScript.
- **HTTP:** HyperText Transfer Protocol es un protocolo de comunicación utilizado para transferir datos entre un cliente (como un navegador web) y un servidor web en Internet. HTTP define la forma en que se envían y reciben solicitudes de recursos web (como páginas HTML, imágenes, archivos CSS y JavaScript) y cómo se envían las respuestas del servidor al cliente. HTTP es el protocolo fundamental que permite la comunicación entre navegadores web y servidores web, y es la base de la World Wide Web.
- **JavaScript:** es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, utilizado principalmente en el desarrollo web para agregar interactividad y dinamismo a las páginas web. JavaScript se ejecuta en el navegador del cliente y permite a los desarrolladores realizar acciones como manipular el contenido HTML, responder a eventos del usuario, realizar validaciones de formulario, crear efectos visuales y animaciones, realizar solicitudes a servidores web (a través de AJAX) y mucho más. JavaScript es un componente fundamental del desarrollo frontend y es compatible con todos los navegadores web modernos.
- **Laravel:** es un framework de código abierto para el desarrollo de aplicaciones web en PHP, que sigue el patrón de arquitectura Modelo–Vista–Controlador (MVC). Su objetivo principal es simplificar tareas comunes del desarrollo web, promoviendo buenas prácticas, organización del código y escalabilidad del sistema.
- **Mozilla:** es una organización sin ánimo de lucro que se dedica a promover un

Internet saludable y abierto. Es conocida principalmente por su navegador web Firefox, que es uno de los navegadores web más populares y utilizados en todo el mundo. Además del navegador Firefox, Mozilla desarrolla y promueve una variedad de productos y proyectos relacionados con la web, incluyendo el motor de renderizado Gecko, el lenguaje de programación Rust, la fundación Mozilla Foundation y la iniciativa Mozilla Developer Network (MDN), que proporciona recursos y documentación para desarrolladores web.

- **Script:** es un programa o conjunto de instrucciones escritas en un lenguaje de programación que realiza una tarea específica en un sistema informático o una aplicación de software. Los scripts son utilizados para automatizar tareas, procesar datos, realizar cálculos, interactuar con el usuario, y más. En el contexto de la web, los scripts son comúnmente utilizados en páginas web y aplicaciones web para agregar interactividad, dinamismo y funcionalidad a la experiencia del usuario, mediante el uso de lenguajes de programación como JavaScript.
- **Seo (Search Engine Optimization):** es una optimización que utilizan los motores de búsqueda, de esta forma pueden rastrear, indexar y procesar el sitio.
- **Sistema Operativo:** un "Sistema Operativo" es un software que gestiona los recursos de hardware y proporciona servicios básicos para los programas de aplicación en una computadora. El sistema operativo controla el funcionamiento del hardware, coordina las actividades de los programas, gestiona el almacenamiento de datos, facilita el acceso a los dispositivos periféricos, y proporciona una interfaz de usuario para interactuar con la computadora. Algunos ejemplos de sistemas operativos populares incluyen Microsoft Windows, macOS, Linux, Android y iOS.
- **Software:** el "Software" es un conjunto de programas, aplicaciones y datos que se utilizan para realizar tareas específicas en un sistema informático. El software puede ser categorizado en diferentes tipos, como software de sistema, que incluye sistemas operativos y utilidades de software, y software de aplicación, que incluye programas diseñados para realizar tareas

específicas, como procesamiento de texto, edición de imágenes, navegación web, reproducción multimedia, y más. El software es esencial para el funcionamiento de computadoras, dispositivos móviles y otros dispositivos electrónicos.

- **URL:** Uniform Resource Locator es una cadena de caracteres que se utiliza para identificar la ubicación de un recurso en Internet. Una URL generalmente consta de varios componentes, incluyendo el esquema (como "http://" o "https://"), el nombre de dominio (como "example.com"), el camino (la ruta al recurso en el servidor), y opcionalmente, parámetros de consulta y fragmentos. Las URL se utilizan comúnmente para especificar la dirección de una página web, un archivo, una imagen, un video u otro recurso en la web.
- **W3C:** World Wide Web Consortium es una organización internacional que desarrolla estándares y especificaciones para la World Wide Web, con el objetivo de promover la interoperabilidad y la evolución de la web. El W3C fue fundado en 1994 por Tim Berners-Lee, el inventor de la web, y está compuesto por un consorcio de organizaciones y miembros individuales que colaboran en el desarrollo de estándares abiertos para la web. El W3C es responsable de especificaciones como HTML, CSS, XML, SVG, HTTP, y las Web Content Accessibility Guidelines (WCAG), entre otros.
- **WCAG:** Web Content Accessibility Guidelines son un conjunto de directrices desarrolladas por el W3C que proporcionan recomendaciones para hacer que el contenido web sea más accesible para personas con discapacidades. Las WCAG establecen principios generales de accesibilidad, pautas específicas y criterios de éxito que los desarrolladores pueden seguir para mejorar la accesibilidad de sus sitios web y aplicaciones web. Las directrices WCAG abordan aspectos como la percepción, la operabilidad, la comprensión y la robustez del contenido web para garantizar que sea usable por todos los usuarios, independientemente de sus habilidades o limitaciones.

7 Apéndice

7.1 Configuración de Webpack

A continuación se presenta la configuración utilizada para el empaquetado del componente mediante Webpack.

Archivo: webpack.config.js

```
const path = require('path');

module.exports = {
  entry: './public/js/accesibilidad.js',
  output: {
    filename: 'accesify.js',
    path: path.resolve(__dirname, 'dist'),
    library: 'Accesify',
    libraryTarget: 'umd',
    globalObject: 'this'
  },
  mode: 'production'
};
```

La elección del formato **UMD (Universal Module Definition)** permite que el componente pueda ser utilizado en distintos entornos de ejecución:

- Como módulo **CommonJS** (Node.js)
- Como módulo **AMD**
- Como objeto global accesible mediante la etiqueta **<script>**

Comando de compilación:

npm run build

El resultado del proceso es un archivo generado en el directorio dist/, listo para su distribución e integración en aplicaciones consumidoras.

7.2 Inicialización y configuración del paquete npm

Inicialización del proyecto:

npm init -y

Este comando genera el archivo package.json, el cual define los metadatos del paquete.

Ejemplo de configuración:

```
{  
  
  "name": "accesify",  
  
  "description": "Widget de accesibilidad WEB auto-inicializable",  
  
  "main": "public/js/accesibilidad.js",  
  
  "scripts": {  
  
    "test": "echo \"Error: no test specified\" && exit 1"  
  
  }  
}
```

Instalación de dependencias de desarrollo:

```
npm install webpack webpack-cli --save-dev
```

7.3 Comandos de publicación y versionado

Publicación del paquete:

```
npm login
```

```
npm publish --access public
```

Versionado

```
npm version patch
```

```
npm publish
```

Instalación en un proyecto consumidor

```
npm install accesify
```

Uso como módulo:

```
Import Accesify from "accesify-widget";
```

Uso como script global:

```
<script src="node_modules/accesify-widget/dist/accesify.umd.js"></script>
```

7.4 Script de instalación automatizada

El proceso de instalación del widget se realiza mediante la inclusión del componente dentro del proyecto consumidor.

Ejemplo de integración en una vista Blade:

```
@include('components.accesify-widget');
```

Este componente se encarga de cargar automáticamente los recursos necesarios del widget en la aplicación.

8 Bibliografía

[1] W3C. World Wide Web Consortium.

Disponible en:

<https://www.w3.org/WAI/fundamentals/accessibility-intro>

Último acceso: 10 de abril de 2026.

[2] INAP. Accesibilidad Web.

Disponible en:

<https://capacitacion.inap.gob.ar/actividad/accesibilidad-web-introduccion-y-pautas/>

Último acceso: 10 de abril de 2026.

[3] ONU. Noticias ONU

Disponible en: <https://news.un.org/es/story/2006/12/1093071>

Último acceso: 10 de abril de 2026.

[4] LEY DE ACCESIBILIDAD DE LA INFORMACIÓN EN LAS PÁGINAS WEB

Disponible en:

<https://servicios.infoleg.gob.ar/infolegInternet/anexos/175000-179999/175694/norma.htm>

Último acceso: 10 de abril de 2026.

[5] Mique. El diseño web adaptado a personas con discapacidad

Disponible en: <https://www.mique.es/disenio-web-adaptado/>

Último acceso: 10 de abril de 2026.

[6] Visual | Iniciativa de Accesibilidad Web (WAI) | W3C

Disponible en: <https://www.w3.org/WAI/people-use-web/abilities-barriers/visual/>

Último acceso: 10 de abril de 2026.

[7] Diccionario de la lengua española.

Disponible en: <https://dle.rae.es/discapacidad>

Último acceso: 1 de marzo de 2026.

[8] Anyela Lorena Orozco Moreno, 2020.

Disponible en:

<https://repositorio.unal.edu.co/bitstream/handle/unal/78510/30238250.2020.pdf?sequence=4&isAllowed=y>

Último acceso: 10 de marzo de 2026.

[9] Organización Mundial de la salud, 2025.

Disponible en:

<https://www.who.int/es/news-room/fact-sheets/detail/blindness-and-visual-impairment>

Último acceso: 12 de febrero de 2026.

[10] Naciones Unidas, 2008.

Disponible en:

<https://www.un.org/development/desa/disabilities/convention-on-the-rights-of-persons-with-disabilities.html>

Último acceso: 12 de enero de 2026.

[11] Userway, 2015.

Disponible en: <https://userway.org/es/blog/lector-de-pantalla/>

Último acceso: 10 de abril de 2026.

[12] Microsoft – Usar la narración de texto en Windows 10 y 11.

Disponible en:

<https://support.microsoft.com/es-es/topic/usar-un-lector-de-pantalla-para-explorar-y-navegar-por-microsoft-store-0d949337-7c9f-4215-8f33-a6735731a5fe>

Último acceso: 10 de abril de 2026.

[13] OpenDyslexic – Tipografía de código abierto para mejorar la lectura en dislexia.

Disponible en: <https://opendyslexic.org/>

Último acceso: 10 de abril de 2026.

[14] Pautas de accesibilidad para el contenido de la web (PDF).

Disponible en:

<https://www.argentina.gob.ar/sites/default/files/pautas-de-accesibilidad-para-el-contenido-de-la-web.pdf>

Último acceso: 10 de abril de 2026.

[15] Accedeme, 2022: Resolución 479 / 2009-,2010

Disponible en: <https://www.argentina.gob.ar/normativa/nacional/ley-26653-175891>

Último acceso: 10 de abril de 2026.

[16] WCAG22, 2023.

Disponible en: <https://www.w3.org/TR/WCAG22/#resize-text>

Último acceso: 10 de abril de 2026.

[17] ADA, 18 de Marzo del 2022.

Disponible en: <https://www.ada.gov/resources/web-guidance/>

Último acceso: 10 de abril de 2026.

[18] Boletín Oficial de la República de Argentina, 2019.

Disponible en: <https://www.boletinoficial.gob.ar/detalleAviso/primera/206723/20190515>

Último acceso: 10 de abril de 2026.

[19] Node.js Foundation. (s.f.). About Node.js.

Disponible en: <https://nodejs.org/en/about>

Último acceso: 10 de abril de 2026.