https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 111 - 128

Desarrollo e implementación de un *widget* de accesibilidad web de código abierto para entornos institucionales

Información del reporte:

Para citar este reporte técnico:

José García, O. y Salazar Licea, L. A. (2025). Desarrollo e implementación de un widget de accesibilidad web de código abierto para entornos institucionales. *Cuadernos Técnicos Universitarios de la DGTIC, 3* (4) páginas (111 - 128). https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136

Oscar José García
Escuela Nacional de Estudios Superiores
Unidad Juriquilla
Universidad Nacional Autónoma de México
oscarjg@unam.mx
ORCID: 0009-0009-5272-870X

Luis Antonio Salazar Licea
Escuela Nacional de Estudios Superiores
Unidad Juriquilla
Universidad Nacional Autónoma de México
LSalazarLicea@unam.mx
ORCID: 0000-0003-3297-3416

Resumen

La accesibilidad web garantiza que los usuarios puedan interactuar con los sistemas en condiciones de inclusión, permitiendo a las personas percibir, entender, navegar e interactuar con un producto web. Ante la falta de una herramienta de código abierto que pudiera ser añadida a los sistemas de la Escuela Nacional de Estudios Superiores Unidad Juriquilla, se desarrolló una solución propia que satisface esta necesidad. Se usaron los lineamientos en temas de accesibilidad y visibilidad establecidos por la Universidad Nacional Autónoma de México y la *Web Content Accessibility Guidelines* para el proceso de diseño y elaboración de la herramienta. La metodología que se utilizó como base para la elaboración de la solución fue cascada (*waterfall*), abarcando desde la planeación hasta la operación y mantenimiento de la herramienta. Las pruebas cualitativas y cuantitativas demostraron su viabilidad como una alternativa a las soluciones comerciales, debido a que se pudieron replicar funcionalidades de productos

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 112 - 128

de pago. Esta herramienta inicialmente se incorporó en un micrositio de divulgación de la dependencia y, al ser tecnológicamente agnóstica respecto a lenguajes y marcos de desarrollo, se integró al resto de los sistemas internos. Se implementaron las funcionalidades principales orientadas a mejorar la experiencia del usuario, entre ellas: ajuste de contraste, modificación del tamaño de texto, realce de enlaces, optimización de visibilidad, control de espaciado vertical, alineación del texto y guía de lectura. La implementación de un lector de pantalla quedó fuera del alcance debido a las limitaciones de soporte nativo de algunos los navegadores web; no obstante, se contempla su inclusión en futuras versiones junto con las mejoras que se recopilen de retroalimentaciones de los usuarios finales.

Palabras clave:

Asistencia para navegación, experiencia inclusiva, componente de inclusión digital.

Abstract

Web accessibility guarantees that users can interact with systems under inclusive conditions, allowing them to perceive, understand, navigate and interact with a web product. In the absence of an open-source tool that could be added to the Escuela Nacional de Estudios Superiores Unidad Juriquilla systems, a custom solution that satisfies this necessity was developed. Accessibility and visibility guidelines established by the Universidad Nacional Autónoma de México and the Web Content Accessibility Guidelines were used for the tool design and elaboration process. Waterfall methodology was used as the basis for the development of the solution, covering from planning to operation and maintenance of the tool. The qualitative and quantitative tests demonstrated its viability as an alternative to commercial solutions because they could replicate functionalities of paid products. This tool was initially incorporated in a dissemination microsite of the dependency and, being technologically agnostic with regard to languages and development frameworks, it was integrated into the rest of the internal systems. The main functionalities aimed at improving the user's experience were implemented, among them: contrast adjustment, text size modification, link highlight, visibility optimization, vertical spacing control, text alignment and reading guide. The implementation of a screen reader was out of reach due to native support limitations of some web browsers; nevertheless, its inclusion is contemplated in future versions along with the improvements that can be collected from final users feedback.

Keywords:

Navigation assistance, inclusive experience, digital inclusion component.

1. INTRODUCCIÓN

La accesibilidad, en el contexto web, es un conjunto de acciones y medidas que tienen como objetivo garantizar la inclusión e igualar las condiciones en las que se consume un producto. En este trabajo, se abordó desde tres puntos de vista interconectados: 1. El técnico y de mercado, que busca mejorar la experiencia de todos los usuarios para aumentar el alcance y la usabilidad (Raymond, M. A., et al, 2024); 2. El de cumplimiento legal, que en México responde a la Ley General para la Inclusión de las Personas con Discapacidad (LGIPD); y 3. La perspectiva humanística y de derechos humanos. Este último enfoque enmarca la accesibilidad no como una característica opcional, sino como un pilar de la equidad en la era digital. El *World Wide Web Consortium* (W3C) reafirma que la misión de la web es ser universal y

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 113 - 128

operable para todas las personas, independientemente de sus capacidades. Desde esta visión, la falta de accesibilidad no es un fallo técnico, sino una barrera que limita la participación plena en la sociedad y el acceso a derechos fundamentales como la educación, el empleo y la cultura (W3C, 2023). Esta perspectiva sitúa el desarrollo de herramientas accesibles no sólo como un ideal o un mandato legal, sino como un acto de responsabilidad social.

De forma institucional, la Universidad Nacional Autónoma de México (UNAM), a través de la Dirección General de Cómputo y de Tecnologías de Información y Comunicación (DGTIC), ha impulsado diversas acciones para el cumplimiento de estos lineamientos, entre las que destacan la emisión de guías de buenas prácticas para la accesibilidad, pautas de accesibilidad para sitios web institucionales, páginas enteras al respecto (https://www.visibilidadweb.unam.mx/buenas-practicas/accesibilidad) y eventos anuales como las Jornadas de Visibilidad Web. Además, la misma DGTIC realiza evaluaciones, mejoras y servicios de accesibilidad en sitios web institucionales mediante la aplicación de estándares y guías como las Pautas de Accesibilidad al Contenido en la Web (WCAG), con el compromiso de asegurar que dichos entornos sean utilizables independientemente de los conocimientos, capacidades personales y características técnicas del equipo utilizado (Moguel Pedraza, F. I., 2024).

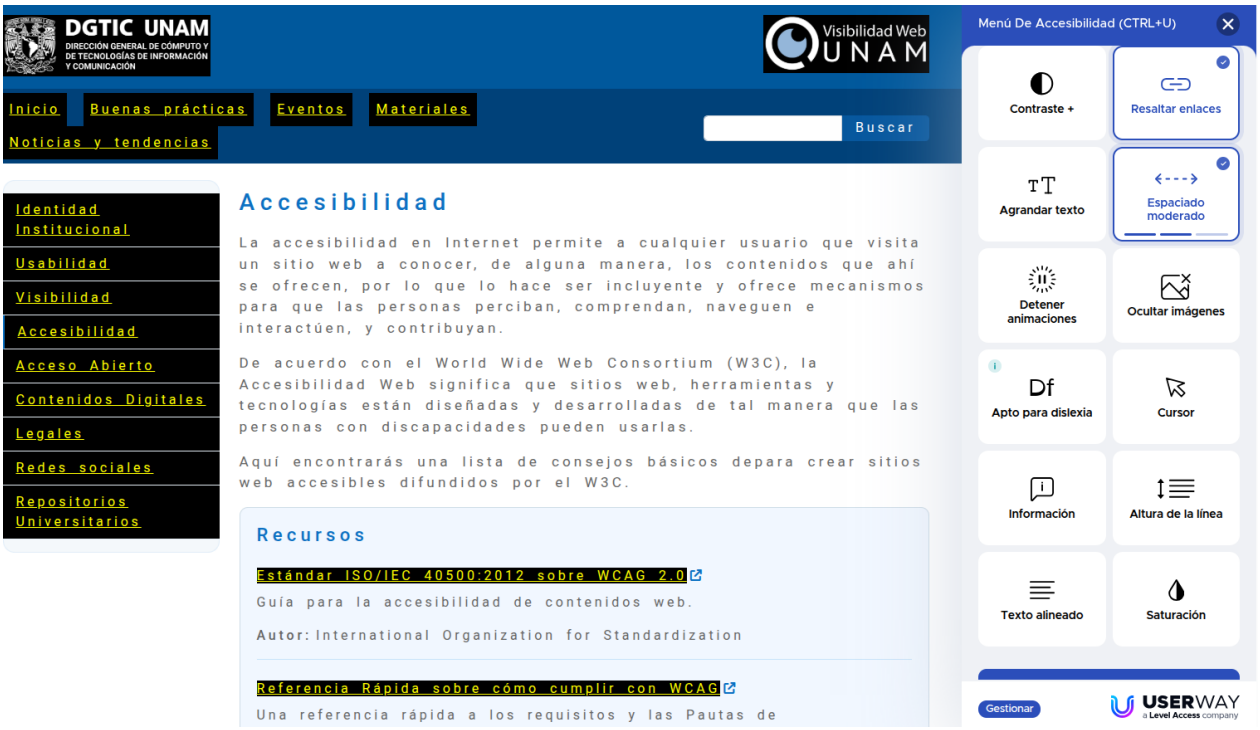
En el contexto de la universidad y debido a la amplia variedad de tecnologías, lenguajes de programación y marcos de desarrollo (*framework*) utilizados para la creación de páginas y sistemas web en las diferentes dependencias de la UNAM, resulta complejo implementar una solución única de accesibilidad. No obstante, una estrategia eficaz y tecnológicamente agnóstica para coadyuvar en la implementación de acciones y medidas de accesibilidad en los sitios web consiste en la incorporación de herramientas digitales, comúnmente denominadas *widgets*, las cuales permiten implementar de manera automatizada ciertas funcionalidades orientadas a mejorar la experiencia de navegación para personas con discapacidad.

La Figura 1 muestra que este tipo de soluciones ya son utilizadas en la universidad, en específico, la solución de la empresa UserWay, que tiene costos que inician desde los 490 euros para sitios pequeños, 1490 euros para sitios medianos y costos personalizables para sistemas más grandes (https://userway.org/es/precios/?tab=widgetdeaccesibilidad).

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 114 - 128

Figura 1

Herramienta de accesibilidad UserWay



Nota. Ejemplo de página web institucional complementada por el widget de accesibilidad desarrollado por UserWay. Tomado de: https://www.visibilidadweb.unam.mx/buenas-practicas/accesibilidad

Para validar la necesidad de una solución propia, se realizó un análisis del estado de la cuestión de las herramientas de accesibilidad web disponibles en el mercado. Soluciones comerciales, como AudioEye y EqualWeb, ofrecen *widgets* robustos, basados en inteligencia artificial, que automatizan la corrección de errores y garantizan el cumplimiento de normativas como la ADA y la WCAG, pero operan bajo un modelo de suscripción mensual que puede superar los 49 dólares para sitios pequeños y aumentar considerablemente según el tráfico y el tamaño de la página (AudioEye, 2025; EqualWeb, 2025). Otras plataformas, como Accessiblyapp, ofrecen planes más asequibles, desde 20 dólares al mes, pero con funcionalidades avanzadas, como la personalización de la marca y el uso de IA para etiquetas alt, reservadas para los niveles de pago superiores (Accessiblyapp, 2025).

Por otro lado, existen alternativas de código abierto como Sienna Accessibility Widget, que se distribuye gratuitamente bajo una licencia MIT y se enfoca en la simplicidad de instalación (Sienna Accessibility, s.f.). Sin embargo, estas soluciones suelen depender del soporte de la comunidad, lo que puede no ser ideal para un entorno institucional que requiere garantías de mantenimiento y una adaptación específica a sus políticas. Tras este análisis, se concluyó que el desarrollo de una herramienta propia, de código abierto y respaldada institucionalmente, representaba la estrategia más sostenible, ya que ofrece control total sobre el código, elimina los costos de licenciamiento recurrentes y permite una personalización alineada con las necesidades específicas de la UNAM.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 115 - 128

Por lo tanto, en este reporte técnico, se tuvo como objetivo describir el proceso de diseño, desarrollo e implementación de un *widget* orientado a mejorar la accesibilidad en entornos web, así como su integración en diversos sitios y plataformas institucionales, independientemente del lenguaje de programación, marco de desarrollo o herramientas utilizadas en su construcción.

2. DESARROLLO TÉCNICO

La metodología que se utilizó para la elaboración de esta herramienta se basó en el ciclo de vida del software mencionado por Sommerville (2016), expuesto en la Figura 2, para la obtención de una primera versión de la herramienta.

Figura 2

Metodología del Ciclo de vida del software

Nota. Metodología del ciclo tomada de Sommerville, I. (2016). Software Engineering (10.ª ed.). Pearson

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 116 - 128

De este modo, se pudo priorizar la creación de un producto inicial que contuviera las funcionalidades esenciales y así poder validar si la solución pudiera contribuir y satisfacer las necesidades de accesibilidad en un entorno real, a la vez que facilitara la retroalimentación de los usuarios finales.

2.1 ANÁLISIS Y DEFINICIÓN DE REQUERIMIENTOS

Con base en el objetivo previamente definido, se realizó el análisis y definición de requisitos del *widget*, donde se adoptaron las Pautas de Accesibilidad al Contenido Web (WCAG 2.1) establecidas por el *World Wide Web Consortium* (W3C).

Figura 3

Relación de funcionalidad web vs discapacidades

FUNCIONALIDAD WEB	DISCAPACIDAD	BENEFICIO	DESCRIPCIÓN
CONTRASTE	Visual (baja visión, daltonismo), cognitiva	Legibilidad	Facilita la lectura y hace distinción entre texto y fondo
RESALTADO DE ENLACES	Visual, cognitiva	Navegación clara	Identificación de hipervínculos
TAMAÑO DE TEXTO	Visual (baja visión), motriz	Adoptabilidad	Ajusta el tamaño del texto y mantiene la funcionalidad.
VISIBILIDAD DE IMÁGENES	Visual (ceguera), cognitiva	Acceso a contenido	Texto alternativo y descripciones
ESPACIADO DE TEXTO	Visual, dislexia, cognitiva	Comprensión	Reducción de fatiga visual.
ALINEACIÓN DE TEXTO	Dislexia, visual	Enfoque visual	Facilita el rastreo visual
MÁSCARA DE LECTURA	Dislexia, Trastorno de Déficit de Atención e Hiperactividad (TDAH), visual	Enfoque visual y concentración	Reducción de carga visual.

La Figura 3 describe estas directrices e incluye la mayoría de las funcionalidades implementadas en soluciones comerciales.

2.2 DISEÑO DEL SISTEMA Y SOFTWARE

También se consideró la implementación de un diseño responsivo para garantizar la compatibilidad con dispositivos móviles con base en el enfoque *Mobile First* (ver Figura 4) (Roth, R. E., at al., 2024).

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 117 - 128

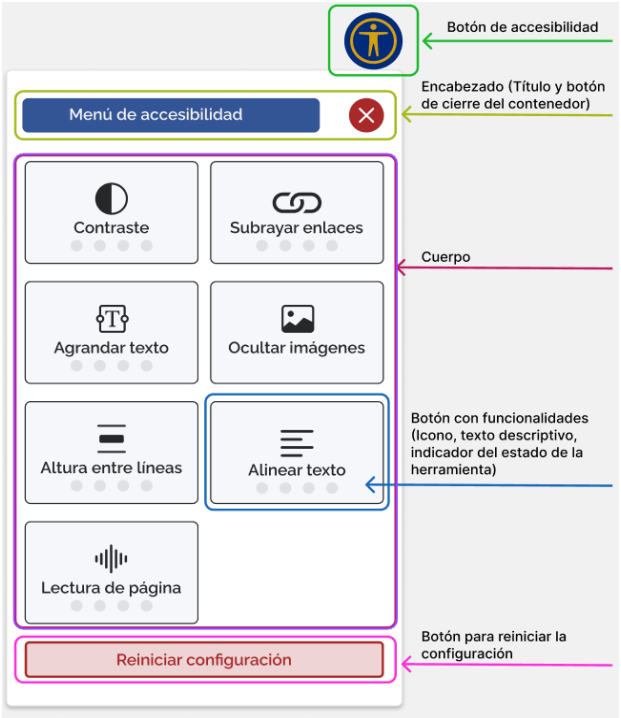
Figura 4

Enfoque de diseño basado primero en móviles y después adaptado a pantallas de mayor tamaño

El maquetado de la interfaz se realizó en Figma con un boceto preliminar que contenía la estructura base, la cual se puede observar en la Figura 5.

Figura 5

Interfaz de usuario diseñada en Figma



https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 118 - 128

Para esta primera iteración del diseño, se priorizó un enfoque de minimalismo funcional (Youvan, D. C., 2024) y usabilidad básica (Beaird, J., Walker, A., & George, J. 202). Se buscó realizar un producto funcional, en un tiempo corto de producción, que fuera entendible y fácil de utilizar. Se definió el contenedor de la herramienta con un ancho de 350 pixeles como valor máximo, por lo que no se necesitaron cambios superiores para su adaptación a dispositivos móviles. En el caso de los íconos, se utilizó inicialmente el *plugin* de Bootstrap Icons que se encuentra disponible en Figma.

Además de los íconos, se puede observar debajo del título de cada funcionalidad el indicador de estado. Por defecto, se decidió que tenga un color distinto el primer círculo, el cual cambiará al siguiente cuando se interactúe sobre él; de este modo, se puede saber en qué estado se encuentra la funcionalidad y cuántas variantes tiene.

2.3 IMPLEMENTACIÓN Y PRUEBAS

La implementación se dividió en 3 partes medulares: la creación de funcionalidades, el desarrollo de la interfaz de usuario y las pruebas.

2.3.1 CREACIÓN DE FUNCIONALIDADES

En la primera etapa, Desarrollo de funcionalidades, se implementó la lógica necesaria para cumplir con los requisitos establecidos. Al ser una herramienta destinada a entornos web, se optó por el uso de JavaScript (JS) como lenguaje de programación base. Como punto importante a destacar, debido a la necesidad de mantener un control total sobre el código fuente, se decidió evitar el uso de componentes externos, librerías y *frameworks* de terceros. Como resultado, no sólo se tuvo una reducción en la dependencia de tecnologías externas, sino también se logró que el *widget* desarrollado tuviera una reducción significativa de tamaño, lo que representa una ventaja sustancial para su integración.

El código JS se estructuró en tres partes principales para facilitar su organización y mantenimiento:

- Funciones con propósitos específicos: Cada una de estas funciones maneja una funcionalidad específica de la herramienta (como el contraste o el tamaño del texto), o son funciones utilitarias que apoyan la lógica, por ejemplo, al cambiar entre las variantes de una funcionalidad.
- Diccionario principal: Ésta es la estructura base de la herramienta, donde se definieron datos clave como el nombre de cada funcionalidad, la función a la que corresponde y las variantes que posee.
- Función principal: Se encarga de iniciar la lógica y manejar el llamado a las demás funciones en el orden establecido para el correcto funcionamiento del *widget*.

En esta etapa, se optó por seguir un desarrollo rápido, enfocado en lograr crear cada una de las funcionalidades básicas descritas en la parte de la definición de requerimientos; de este modo, se consiguió reducir la cantidad de tiempo ocupada para la creación de la primera versión de la herramienta, es decir, se pasó de un tiempo estimado de dos meses a un mes, tomando en consideración que el tiempo no fue asignado exclusivamente a esta actividad. Por lo anterior, aunque la calidad del código y su estructura son importantes, al menos en esta etapa, se decidió que fueran secundarias para realizar una refactorización y optimización del código en una etapa posterior.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 119 - 128

2.3.2 DESARROLLO DE LA INTERFAZ DE USUARIO

Para elaborar la interfaz, se utilizó Tailwind CSS con su enfoque *utility-first*; todas las clases generadas se agregaron directamente en el archivo JS, lo que influyó en el tamaño final del archivo. Debido a esto, se decidió hacer el cambio de enfoque y se utilizó un archivo *Cascading Style Sheets* (CSS) para juntar en un solo lado todos los estilos visuales. Además, se tomó en consideración que no todos los usuarios que implementarían el herramienta en sus sistemas sabrían cómo trabajan los estilos CSS, por lo que podría haber una confrontación de estilos, por ejemplo, entre las clases mt-4 de Bootstrap y Tailwind CSS, sólo por mencionar un caso en específico. En consecuencia, se decidió continuar con el uso de Tailwind CSS, pero haciendo uso de la directiva @apply en clases personalizadas del archivo CSS; de este modo, se mantuvieron los beneficios de este marco de trabajo de CSS y se evitaron posibles inconsistencias.

2.3.3 PRUEBAS VISUALES

La primera parte de las pruebas corresponde a las visuales, éstas están relacionadas principalmente con el diseño de la interfaz de usuario. Para revisar esta parte, se utilizó la herramienta DevTools del navegador, específicamente en la sección de *Toggle Device Toolbar*, con el propósito de evaluar la adaptabilidad en distintos dispositivos móviles. A través de estas pruebas, se identificaron y corrigieron errores en la visualización de los elementos y se mejoró la estructura del *widget*.

Las pruebas de humo (manual) se realizaron tomando como base las funcionalidades con las que cuenta la herramienta y su interacción. La Figura 6 muestra todas las pruebas realizadas a cada una de las funcionalidades. Hasta el momento, la única herramienta que no cuenta con una variante de uso es la de máscara de lectura, aunque será agregada en una iteración posterior.

Figura 6

Evaluación de funcionalidades implementadas

FUNCIONALIDAD	CARGA SIN ERRORES	SE PUEDE ACTIVAR	CAMBIO ENTRE OPCIONES	FUNCIONA JUNTO A OTRAS	PERSISTE EN EL NAVEGADOR
CONTRASTE	✓	✓	✓	✓	✓
RESALTADO DE ENLACES	✓	✓	✓	✓	✓
TAMAÑO DE TEXTO	✓	✓	✓	✓	✓
VISIBILIDAD DE IMÁGENES	✓	✓	✓	✓	✓
ESPACIADO DE TEXTO	✓	✓	✓	✓	✓
ALINEACIÓN DE TEXTO	✓	✓	✓	✓	✓
MÁSCARA DE LECTURA	✓	✓	Aún no	✓	✓

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 120 - 128

Finalmente, se revisaron los siguientes puntos:

- Se verificó que las funciones se activaran inmediatamente.
- Se verificó que la variante de la función trabajara adecuadamente.
- Se comprobó que el teclado funcionara para navegar por la herramienta
- Se validó el correcto funcionamiento de los botones para reiniciar las funcionalidades, así como la interacción del botón principal para mostrar u ocultar el *widget*.
- Se verificó que el botón de accesibilidad se pudiera mover de lugar dependiendo de su configuración en el *script*.
- Se confirmó que la herramienta no generara errores en la consola del navegador.

2.3.4 INTEGRACIÓN Y PRUEBAS DEL SISTEMA

Como resultado de las etapas previamente descritas, se obtuvo la primera versión del *widget*. Éste se conforma de dos archivos: un .js, que contiene la lógica, y un .css, que define los estilos asociados. Para su integración en un entorno web, el .css debe ser agregado dentro de la etiqueta HTML <head> y el archivo .js antes de finalizar la etiqueta <body>, véase la Figura 7 para más detalles.

Figura 7

Integración del archivo .css y .js dentro de la estructura HTML

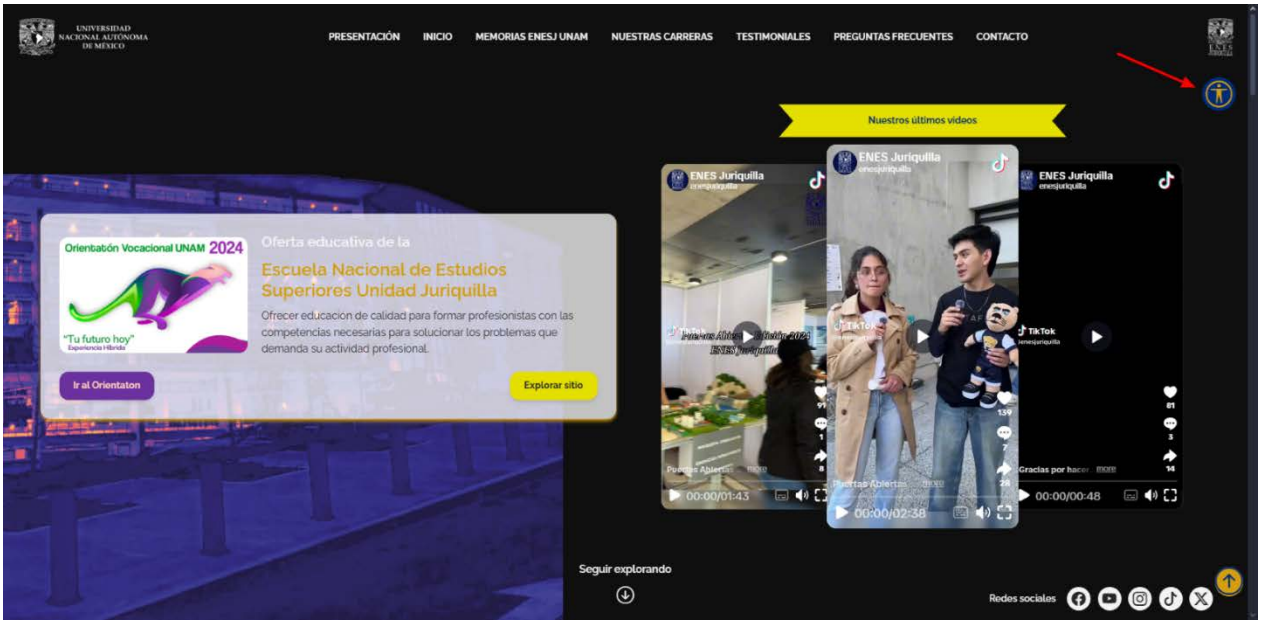
https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 121 - 128

Hasta este punto, las pruebas que se hicieron fueron en ambientes controlados con elementos por defecto, es decir, en páginas web sencillas con elementos HTML estándar como <img>, <a>, <p>, <body>, entre otros. En estos escenarios, los resultados fueron satisfactorios ya que la solución se integró de manera adecuada.

La integración en un ambiente productivo se redujo a cargar en la carpeta *static* de uno de nuestros sitios web los archivos de la herramienta, y se mandaron a llamar dentro del elemento HTML principal con una etiqueta <link> y <script>, donde cada uno contenía la ruta relativa respectiva del archivo. El siguiente paso fue actualizar el contenido del sitio web y comenzar a interactuar para validar las funcionalidades (ver Figura 8).

Figura 8

Micrositio divulgación ENES Juriquilla – Botón de la herramienta de accesibilidad



Nota. En la figura, se observa la implementación de la herramienta en uno de los sitios web de la ENES Juriquilla, por lo que la incrustación fue satisfactoria debido a que se integró correctamente en el sitio sin realizar alguna otra acción. Fuente:https://divulgacion.enesjuriquilla.unam.mx

Al dar clic sobre el botón de accesibilidad, se despliega la herramienta (ver Figura 9). Las pruebas realizadas fueron las mismas descritas en el apartado

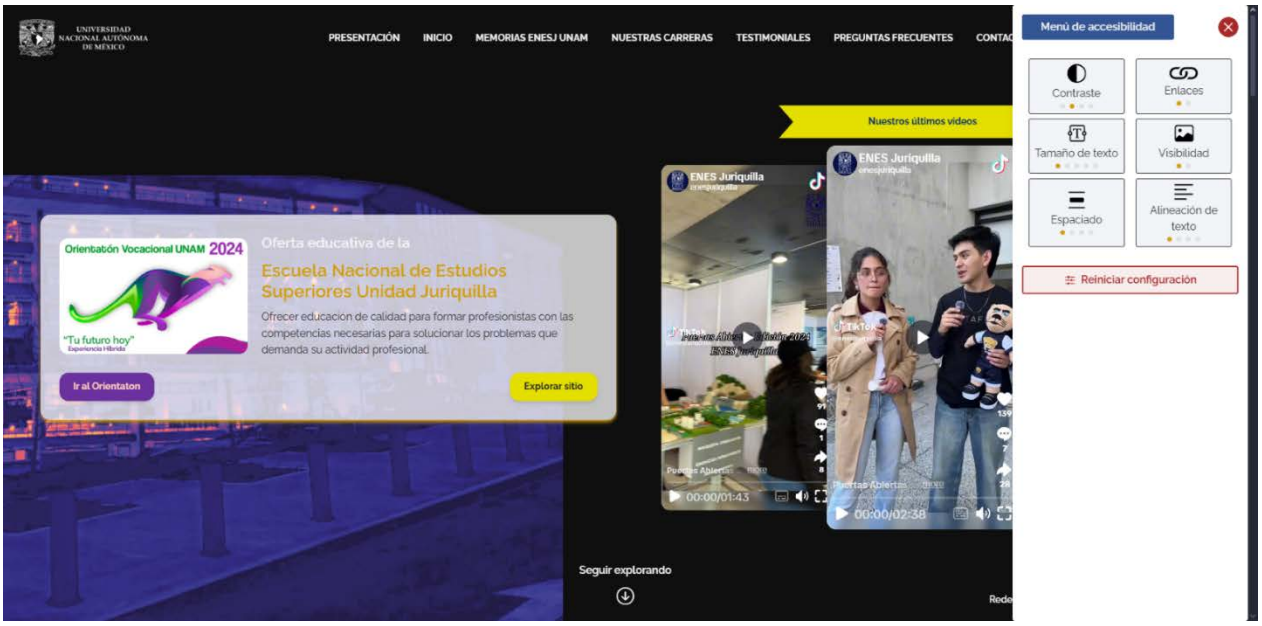
2.3.5 PRUEBAS DE FUNCIONALIDAD

En la Figura 9, se observan las funcionalidades integradas en esta primera versión del *widget*.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 122 - 128

Figura 9

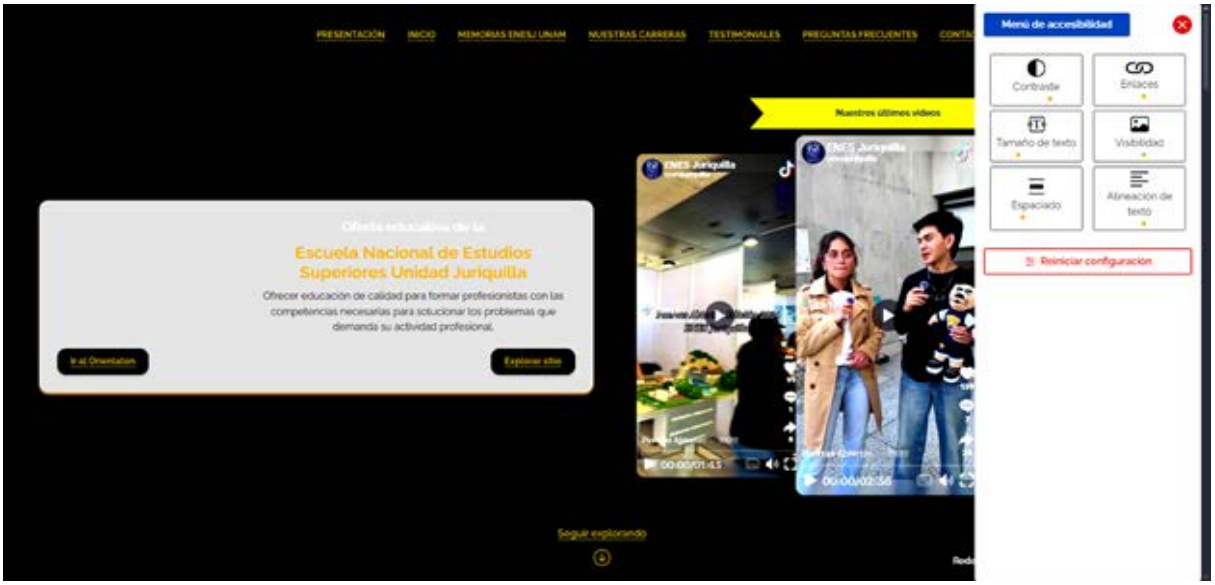
Micrositio divulgación ENES Juriquilla – herramienta de accesibilidad abierta



Al activar alguna de las opciones, se genera un cambio visible en la vista del entorno web, como se puede observar en la Figura 10.

Figura 10

Micrositio divulgación ENES Juriquilla – Utilización de la herramienta de accesibilidad



https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 123 - 128

En la figura anterior, se puede observar la herramienta integrada y en funcionamiento. En este caso, como ejemplo, se habilitó la opción de contraste, haciendo que la página web visible se vea con tonos fuertes, se resaltaron los enlaces, se ocultaron las imágenes y se alineó al centro el texto. Visualmente, en el contenedor de funcionalidades, se puede observar que la opción utilizada está marcada de un color distintivo. Para reiniciar las opciones, existen dos maneras: la primera es dar clic a cada funcionalidad hasta llegar a la opción por defecto y la segunda es dar clic en el botón reiniciar configuración para que todas las funcionalidades regresen a su opción base.

2.4 OPERACIÓN Y MANTENIMIENTO

Una vez probada la herramienta en un sitio web institucional, se decidió liberar esa versión del *widget* en los demás entornos web de la dependencia, todo con el propósito de encontrar errores y revisar que las funcionalidades trabajen adecuadamente en diferentes navegadores y junto a las diferentes tecnologías con las que hayan sido desarrolladas.

Como parte de la mejora continua, tras el despliegue y operación, se inició la optimización de la herramienta. El código JS inicial tuvo un total de 430 líneas de código, mientras que el archivo CSS tuvo 227 líneas de código. Si bien las líneas de código (*Lines of Code*, o LOC por sus siglas en inglés) son una métrica común para medir el tamaño de un programa, no reflejan necesariamente su calidad o eficiencia. En el contexto del desarrollo de *software*, un código conciso y refactorizado, a menudo, es más valioso que uno extenso. En este caso específico, el tamaño inicial del código resultante se consideró como un punto de partida para evaluar el impacto de las mejoras futuras.

Posteriormente, se tomaron las funcionalidades y se reestructuró el código para hacerlo más adaptable al momento de agregar nuevas funciones. Este proceso, conocido en la ingeniería de software como refactorización, se enfocó en la separación de responsabilidades, la eliminación de redundancias y la mejora de la legibilidad, manteniendo siempre el comportamiento funcional del *widget*. El objetivo fue, como sugiere Martin Fowler (2018), mejorar la estructura interna del software sin alterar su comportamiento externo. Como resultado, el archivo JS se redujo significativamente de 31 a 25 kilobytes; por su parte, el archivo CSS se redujo mínimamente de 8 a 7 kilobytes, lo que demostró la eficiencia del proceso de optimización.

Como última etapa de la refactorización del código, se optó por emplear *Google Closure Compiler* como herramienta de compilación y minificación de código JS. De acuerdo con Antal et al. (2024), en su estudio comparativo de herramientas utilizadas en *call graph*, sólo *Google Closure Compiler* y TAJS fueron capaces de analizar correctamente JavaScript moderno en múltiples archivos. Esto representa una ventaja en el contexto de este trabajo, ya que, en futuras actualizaciones, la herramienta seguirá siendo funcional.

A pesar de las bondades que ofrece la herramienta *Google Closure Compiler*, se debe tener especial cuidado al momento de minificar, pues *Closure* tiene 2 opciones principales y, según se elija entre *SIMPLE_OPTIMIZATIONS* o *ADVANCED_OPTIMIZATIONS,* el resultado de la minificación puede ser o más simple o más extremo Para el caso del archivo JS, cuando se utilizó *ADVANCED_OPTIMIZATIONS,* esto provocó una serie de errores, por lo que se decidió utilizar *SIMPLE_OPTIMIZATIONS*. La diferencia en el tamaño del archivo JS resultante no varió demasiado, se mantuvo en 9 kilobytes estables; sin embargo, para reducir ligeramente el tamaño del archivo, se utilizó un *plugin* de Visual Studio Code llamado MinifyAll, el cual redujo el tamaño del archivo a 8 kilobytes, lo que se consideró adecuado como tamaño final.

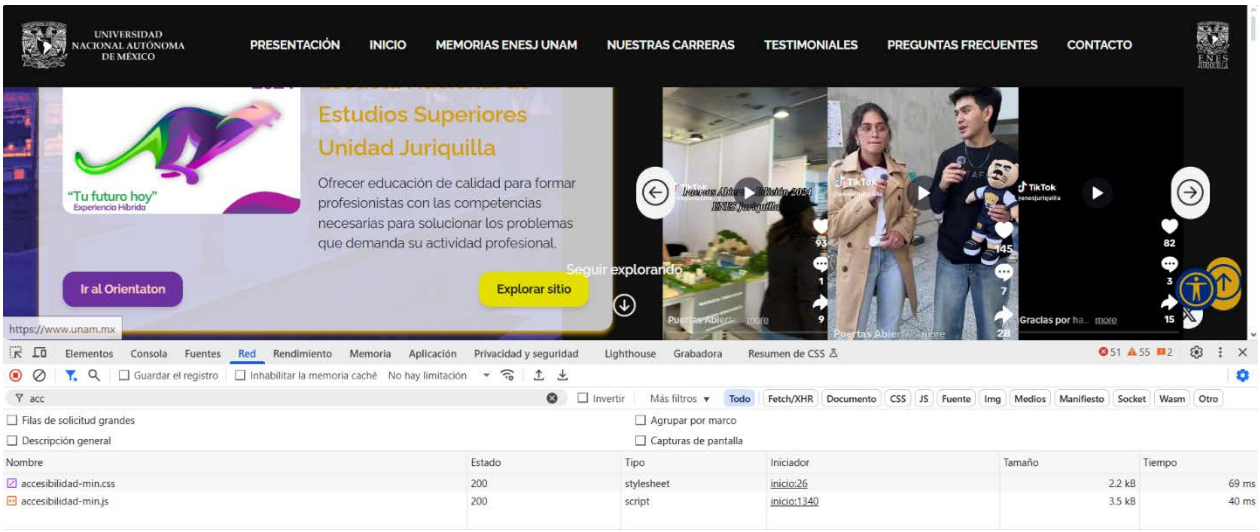
https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 124 - 128

3. RESULTADOS

La solución se presenta en archivos CSS y JS minificados que pueden ser añadidos en cualquier entorno web que permita insertar etiquetas <link> y <script>. El peso total del *widget* de accesibilidad fue de aproximadamente 15 kilobytes. Al hacer la revisión del rendimiento en DevTools, se observó que, si la herramienta es servida mediante un servidor Apache, el tamaño total llega a bajar hasta los 5.7 kilobytes con un tiempo de carga de apenas 109ms; esto se debe a la tecnología de compresión Gzip, función adicional de Apache. Lo anterior se puede observar en la Figura 11.

Figura 11

Función adicional de Apache, Gzip



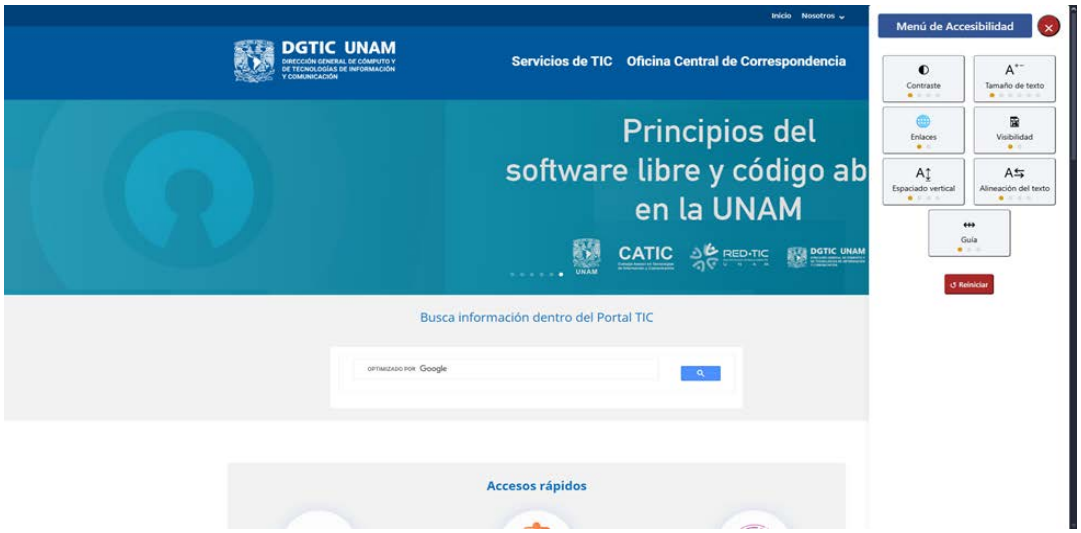
Cada kilobyte de un sitio web debe ser descargado por el navegador del usuario, por lo que un menor peso total de la página se traduce directamente en un menor tiempo de carga. Mientras que el tamaño promedio de una página web en 2025 supera los 2 megabytes (más de 2,000 kilobytes), una herramienta que pesa menos de 15 kilobytes representa una carga adicional mínima, lo que asegura que su implementación no degrade el rendimiento del sitio (HTTP Archive, 2025). Teniendo esto presente, se puede considerar a ésta como una herramienta ligera que cuenta con las funcionalidades más comunes en el mercado. Hasta este momento, no se ha pensado en recabar datos estadísticos, ya que se entra en un tema donde se tendrían que definir los términos de privacidad sobre la recopilación de información y donde también habría que cambiar ciertas partes de la lógica de la herramienta.

Finalmente, para demostrar su utilidad, se decidió insertar el código generado en una página de la DGTIC UNAM. En la Figura 12, se puede observar que se ha integrado bastante bien a pesar de que se desconoce la forma en la que está estructurada la página, la forma de manejar los estilos CSS y hasta las tecnologías que se utilizaron para la creación del sitio.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 125 - 128

Figura 12

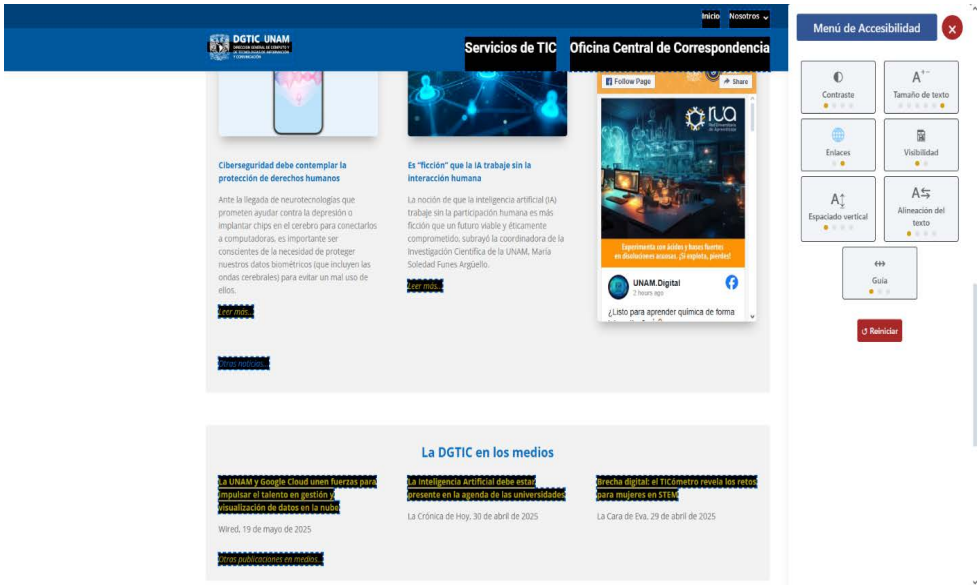
Integración de la herramienta de accesibilidad al sitio institucional DGTIC UNAM



En la integración que se realizó al sitio de la DGTIC, se comprobó que las funcionalidades de la solución trabajan adecuadamente y, a simple vista, no existe una gran diferencia de la que ofrecería una solución comercial. A continuación, se presentan las figuras 13, 14 y 15, en donde se puede observar el funcionamiento de la herramienta.

Figura 13

Herramienta de accesibilidad - activación de la funcionalidad: tamaño de texto y enlaces en el sitio de la DGTIC



https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 126 - 128

Figura 14

Herramienta de accesibilidad - activación de la funcionalidad: contraste, tamaño de texto, enlaces, visibilidad de imágenes, espaciado vertical y alineación del texto en el sitio web de la DGTIC

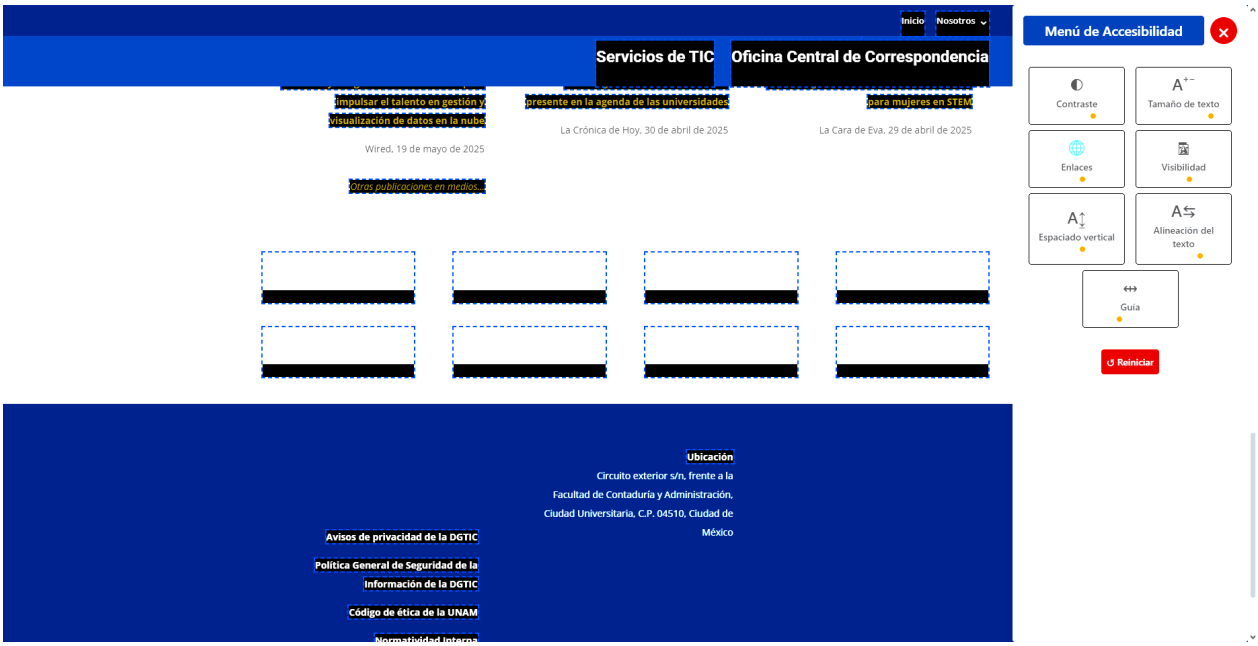
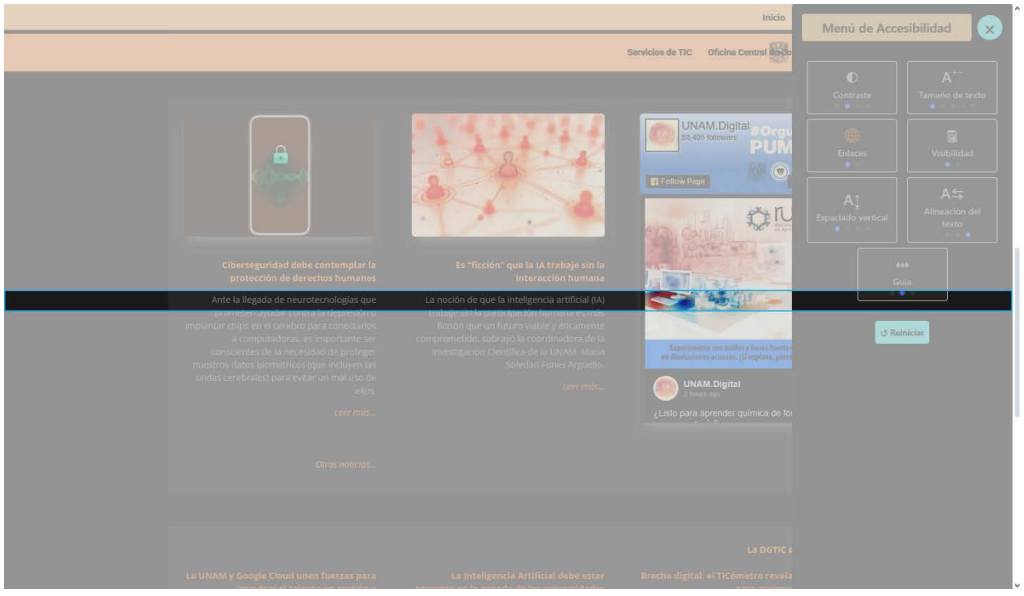


Figura 15

Herramienta de accesibilidad - activación de la funcionalidad: contraste, alineación del texto y guía – máscara de lectura en el sitio web de la DGTIC



https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 127 - 128

Si bien la solución generada en la dependencia aún no cuenta con ciertas características que destacan en herramientas de accesibilidad comerciales como *UserWay*, misma que posee características adicionales como espaciado entre letras, apto para dislexia, cambiar tipo de cursor, saturación, lector, etc., aún con lo anterior, se tiene un acercamiento inicial a las funcionalidades más usadas y, a futuro, se podrían tomar en consideración para ser agregadas en una posterior iteración.

4. CONCLUSIONES

El proyecto logró el diseño, desarrollo y la implementación exitosa de una herramienta web de accesibilidad pensada inicialmente para su uso a nivel institucional. El producto resultante cumplió las expectativas iniciales al demostrar su viabilidad como una alternativa a las soluciones comerciales, lo que lo convierte en un punto de apoyo valioso para aquellas plataformas y sitios web institucionales que buscan mejorar la experiencia de usuario. Cabe aclarar que este apoyo se centra en la capa de presentación que manipula el usuario y que la herramienta no está diseñada para modificar aspectos de fondo como el SEO o el rendimiento del sitio, los cuales requieren intervenciones a nivel de código.

A través del proceso descrito, se pudo observar que el uso de metodologías de desarrollo como la cascada también permite obtener resultados funcionales y optimizados en un tiempo reducido.

Como parte de la mejora continua, se han identificado las siguientes áreas de oportunidad para futuras iteraciones de la herramienta:

- Implementación de un lector de pantalla nativo, lo que requiere una evaluación de la compatibilidad en los diferentes navegadores web.
- Mejoras en el rendimiento visual, en especial con la funcionalidad “alinear texto” y “máscara de lectura”.
- Mejoras en la experiencia de usuario, como el reinicio del indicador de estado a su valor predeterminado.
- Implementación de las funcionalidades guía de lectura, espaciado entre letras y tipo de texto para personas con dislexia.

Es necesario dar mantenimiento a esta herramienta a la vez que se revisan los temas de mejora continua; de esta manera, se demuestra el compromiso del uso y aprovechamiento de las TIC en la UNAM y su aplicación en temas de accesibilidad.

REFERENCIAS

Accessiblyapp. (2025). *Leading Accessibility Widget for ADA & WCAG Compliance*. Accessibly. https://accessiblyapp.com/

AudioEye. (2025). *Plans & Pricing*. AudioEye®. https://www.audioeye.com/plans-and-pricing/

Beaird, J., Walker, A., & George, J. (2020). *The principles of beautiful web design*. SitePoint Pty Ltd.

Congreso de la Unión. (2024). *Ley General para la Inclusión de las Personas con Discapacidad*, artículo 32. *Diario Oficial de la Federación*, 14 de junio de 2024. Recuperado el 9 de junio de 2025, de https://

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.136 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 128 - 128

www.diputados.gob.mx/LeyesBiblio/pdf/LGIPD.pdf

EqualWeb. (2025). *Affordable web accessibility solutions pricing*. EqualWeb. https://www.equalweb.com/10842/8592/pricing

Fowler, M., & Beck, K. (2018). *Refactoring: Improving the Design of Existing Code* (2.ª ed.). Addison-Wesley Professional.

HTTP Archive. (2025). *State of the Web - Page Weight*. HTTP Archive. https://httparchive.org/reports/page-weight

Moguel Pedraza, F. I. (2024). Recomendaciones técnicas para implementar y asegurar la accesibilidad web en los sitios institucionales. *Cuadernos Técnicos Universitarios de la DGTIC*, 2 (2). https://doi.org/10.22201/dgtic.ctud.2024.2.2.50

Raymond, M. A., Smith, H., Carlson, L., & Gupta, A. (2024). An Examination of Digital Accessibility Within Social Media Platforms: Problems for Vulnerable Consumers and Policy Implications. *Journal of Advertising Research*, 64(4), 430–450. https://doi.org/10.2501/JAR-2024-026

Roth, R. E., Çöltekin, A., Delazari, L., Denney, B., Mendonça, A., Ricker, B. A., Wu, M. (2024). Making maps & visualizations for mobile devices: A research agenda for mobile-first and responsive cartographic design. Journal of Location Based Services, 18(4), 408–478. https://doi.org/10.1080/17489725.2023.2251423

Sienna Accessibility. (s.f.). *Sienna Accessibility Widget*. Sienna Accessibility. https://accessibility-widget.pages.dev/

Sommerville, I. (2016). *Software Engineering* (10.ª ed.). Pearson.

W3C. (2023). *Introduction to Web Accessibility*. Web Accessibility Initiative (WAI). https://www.w3.org/WAI/fundamentals/accessibility-intro/

World Wide Web Consortium. (2018). *Web Content Accessibility Guidelines (WCAG) 2.1*. W3C. Recuperado el 6 de marzo de 2025, de https://www.w3.org/TR/WCAG21/

Youvan, D. C. (2024). The Essence of Less: Exploring Abstract and Minimalist Concepts Across Art, *Design, Technology, and Philosophy*.