Vol.7 No.3 (2023): Journal Scientific MQRInvestigar ISSN: 2588–0659
https://doi.org/10.56048/MQR20225.7.3.2023.1458-1474

Proposal of a software architecture for the development of accessible websites

Propuesta de una arquitectura de software para el desarrollo de sitios web accesibles

Autores:

Palomeque-Zambrano, Eugenio Esteban
UNIVERSIDAD CATÓLICA DE CUENCA
Cuenca – Ecuador
eepalomequez69@est.ucacue.edu.ec
https://orcid.org/0009-0007-1239-3402

Campoverde-Molina, Milton
UNIVERSIDAD CATÓLICA DE CUENCA
Cuenca – Ecuador
mcampoverde@ucacue.edu.ec
https://orcid.org/0000-0001-5647-5150

Citación/como citar este artículo: Palomeque-Zambrano, Eugenio Esteban. Y Campoverde-Molina, Milton. (2023). Propuesta de una arquitectura de software para el desarrollo de sitios web accesibles. MQRInvestigar, 7(3), *1458-1474*.

https://doi.org/10.56048/MQR20225.7.3.2023.*1458-1474*

Fechas de recepción: 01-JUN-2023 aceptación: 21-JUL-2023 publicación: 15-SEP-2023

https://orcid.org/0000-0002-8695-5005
http://mqrinvestigar.com/

Resumen

En los últimos años el consumo de recursos en la web se ha incrementado en gran medida. Sin embargo, las malas prácticas de accesibilidad en los sitios web limitan a muchos usuarios a acceder a su información. Por tal razón, en esta investigación se realiza una arquitectura de software para el desarrollo de sitios web accesibles. En los resultados se analizan los patrones de prácticas de accesibilidad en entornos de trabajo de JavaScript, se abstraen los mismos y se alinean según las directrices de la WCAG. Luego se propone la arquitectura de Software considerando los hallazgos desde el diseño, la generación de contenido, implementación y pruebas de manera iterativa. Esto permite un ahorro de tiempo y recursos al considerar la accesibilidad desde el inicio del desarrollo de un sitio web y no su verificación al final como lista de verificación. Para la validación de la arquitectura se desarrolló un sitio web y se corroboro la accesibilidad haciendo la evaluación con distintas herramientas de evaluación automática de la accesibilidad web. En conclusión, la arquitectura de software propuesta permite la implementación de la accesibilidad web desde el inicio del desarrollo de un sitio web.

Palabras claves: Arquitectura de software, Accesibilidad Web, WCAG.

Abstract

In recent years, resource consumption on the web has increased greatly. However, poor accessibility practices in web sites limit many users to access their information. For this reason, in this research a software architecture for the development of accessible web sites is carried out. In the results, patterns of accessibility practices in JavaScript working environments are analyzed, abstracted and aligned according to WCAG guidelines. Then the Software architecture is proposed considering the findings from design, content generation, implementation and testing in an iterative way. This saves time and resources by considering accessibility from the beginning of the development of a website and not its verification at the end as a checklist. For the validation of the architecture, a web site was developed and the accessibility was corroborated by making the evaluation with different automatic web accessibility evaluation tools. In conclusion, the proposed software architecture allows the implementation of web accessibility from the beginning of the development of a website.

Keywords: Software architecture, Web Accessibility, WCAG.

Introducción

Según el informe mundial de discapacidad de la OMS en conjunto con el Banco Mundial aseguran que más de mil millones de personas poseen algún tipo de discapacidad y aproximadamente 200 millones tienen una discapacidad limitante. Esto afecta la calidad de vida de las personas como sanidad, mayor pobreza, menor status económico y social, bajo rendimiento académico, resultando en obstáculos al acceso a la educación, empleo, incluso a la información. Según la Encuesta Mundial de la Salud alrededor de 785 millones de personas de más de 15 años tienen alguna discapacidad, además del 2,2% de personas tienen dificultades significativas, el 3,8% tienen alguna discapacidad grave, de estos el 5,1% son niños y el 0,7% con discapacidad grave (OMS, 2011). Es común pensar en discapacidad como una limitación o incluso deficiencia, o como un impedimento o algo que imposibilita a las personas para realizar una actividad (María Esther Perez Dalmeda, 2019). Desde un punto de vista médico, una discapacidad se puede traducir como un cuerpo defectuoso o deficiencia que imposibilita de alguna manera a una persona para gozar de una experiencia vital (Ferreira, 2010). También las personas con discapacidad pierden funciones básicas o nunca las llegan a adquirir (Centers, 2022). Otros autores (Marno Retief, 2018) describieron las discapacidades como un problema en el individuo, como un defecto en un sistema en el cual se tiene una anormalidad adquirida de manera hereditaria y patológica, generalmente las personas con discapacidad pasarán mucho tiempo de su vida en rehabilitación o como aprendices con ayuda profesional.

Considerando que las discapacidades están muy latentes y el uso del internet se ha incrementado enormemente a partir de la pandemia del COVID-19, los sitios web no son desarrollados usando estándares de accesibilidad web. Como se demuestra en (Mondaq, 2022) las demandas por accesibilidad de sitios web ha ido en aumento, en el año 2017 se encontraron 814 demandas, para el año 2018 se registraron 2258 demandas, representando un incremento del 177%. En el año 2021 a pesar de la pandemia por COVID-19 se obtuvo un incremento del 14% con 2895 demandas. Por lo tanto, las tecnologías deben adaptarse para llegar a un número mayor de personas, una de estas tecnologías son los sitios web. Hoy en día ya no existen solo sitios web informativos si no también plataformas complejas que ayudan a la productividad empresarial, educación, médica y mucho más. Por lo que se requieren no solo lineamientos ya dispuestos por la Web Content Accesibility Guidelines (WCAG) o listas de verificación, sino una arquitectura en tiempo de desarrollo para la implementación correcta y oportuna de la accesibilidad en sitios web. Razón por la cual esta investigación tiene como objetivo crear una arquitectura para el desarrollo de sitios web accesibles. Para ello, se analizan diferentes entornos de trabajo basados en JavaScript para establecer un patrón común de prácticas de accesibilidad con el fin de proponer una arquitectura que permita de forma integral el desarrollo de aplicaciones web desde la concepción hasta la producción de forma iterativa, así como validarla con el desarrollo de una aplicación web.

Este documento está dividido en las siguientes secciones, la sección II presenta los principales conceptos sobre discapacidad y accesibilidad en la web. La sección III presenta la revisión de la literatura de acuerdo a arquitecturas de software referente a la accesibilidad. La sección IV se presenta la metodología. La sección V presenta los resultados del desarrollo de la arquitectura. La sección VI presenta la discusión. En la sección VII finalmente se presentan las conclusiones del trabajo realizado.

Desarrollo

Conceptos Relacionados

Arquitectura de software

La arquitectura de software representa la estructura a alto nivel de un software. La arquitectura de software generalmente es una decisión clave, debido a que esta debe satisfacer las necesidades del sistema, tomando en cuenta los requerimientos funcionales y no funcionales del mismo (Fangchao Tian, 2022). El éxito de un artefacto de software depende mucho de su diseño o su arquitectura (H. Cervantes, 2016). Pueden existir varias definiciones de arquitectura de software, pero según Bass "La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar sobre el sistema. Comprende elementos de software, relaciones entre ellos, y propiedades de ambos" (Bass, 2004). Visto de otra manera una arquitectura es una estructura la cual organiza el sistema. Existen varios estilos de arquitecturas, algunos de estos los más utilizados en sitios web educativos son SOA o Arquitecturas Orientadas a los Servicios, Cliente-Servidor, Domain Driven Design (DDD) y Basados en la Estructura (Basados en Componentes, Orientados a Objetos y Arquitectura en Capas) (Milton Campoverde-Molina, 2021).

Accesibilidad Web

La accesibilidad en torno a la Web es el grado en que un sitio web puede interactuar con los usuarios independientemente de sus necesidades, ubicación geográfica, idiomas y aptitudes mentales y físicas (W3C, Accesibility, 2022). Entonces la accesibilidad web es un concepto para que todo tipo de personas tengan acceso a los recursos web, permitiendo una correcta navegación, interacción, etc. Para que un sitio web sea accesible debe seguir las directrices de la WCAG, de lo contrario alguna porción de la población estaría restringida de obtener esa información (Abid Ismail, 2022). Otros autores dicen sobre la accesibilidad web se refiere al diseño del sitio, siendo este entendible y navegable usando distintos tipos de dispositivos (Muzo, 2008). La World Wide Web Consortium (W3C) establece los componentes esenciales de la accesibilidad web, los cuales interactúan entre sí para lograr ser accesibles (W3C, Web Accessibility Initiative WAI, 2022).

Pautas de Accesibilidad para el Contenido Web (WCAG)

Las WCAG son directrices que establecen como desarrollar sitios web accesibles para personas con discapacidad. La W3C ha impuesto varias recomendaciones entorno a la accesibilidad web que se han tornado en estándares, las WCAG que se desarrollan en conjunto con organizaciones y personas alrededor del mundo para generar un estándar a nivel internacional. Las WCAG establecen cuatro principios fundamentales que son: perceptible, operable, entendible y robusto. Además, la WCAG establecen niveles de conformidad como A, AA y AAA, el cual corresponden desde el mínimo aceptable al máximo nivel de conformidad (WCAG, 2023).

JavaScript

JavaScript (JS) es un lenguaje de programación creado por Brendan Eich de Netscape, cuyo principal objetivo es crear dinamismo en sitios web, agregando efectos, visualización, animaciones y más. JS es un lenguaje interpretado por el cual puede ser ejecutado del lado del cliente sin pasos previos, es decir un sitio web que incluya JS se sirve directamente al navegador web y es ejecutado en el mismo, cada navegador web incorpora un motor de JS. Para evitar incompatibilidades Netscape decidió estandarizar el lenguaje por el cual se envió a la European Computer Manufacturers Association (ECMA), el cual creó el comité TC39 para estandarizar el lenguaje JS de manera independiente de cualquier empresa, lo que resultó

en el primer estándar denominado ECMA-262, el cual define el lenguaje ECMAScript, ISO adoptó el estándar ECMA-262 resultando en el estándar ISO-IEC-16262. Por lo tanto, JS es la implementación del estándar ECMAScript. Las versiones actuales de ECMAScript se definen por su año, siendo la última especificación ECMAScript2022 (Pérez, 2009).

Trabajos Relacionados

En el año 2020 (Milton Campoverde-Molina, 2021), se realizó una revisión de la literatura con 23 estudios más significativos en educación e ingeniería de software. Este documento destacó que los sitios web educativos desarrollados bajo distintos tipos de arquitecturas como SOA, DDD, Cliente-Servidor entre otras, pero se concluyó que ninguna de estas arquitecturas toma en cuenta la accesibilidad y no favorecen del todo a la enseñanza colaborativa en línea, en el mismo se recomienda la creación de una arquitectura que incluya las normas, reglamentos, leyes y estándares de calidad que den cabida a la inclusión educativa o al acceso universal.

En el año 2017 (S. Angelov, 2017), se realizó un estudio de las arquitecturas de software en el ámbito de la enseñanza utilizando métodos ágiles. Esto fue realizado debido a que la enseñanza de las arquitecturas de software no se manejaba correctamente en la literatura o en la práctica solo se centraban según los requerimientos de las partes interesadas y no consideraban la accesibilidad. Una vez más se nota la falencia de no incluir los estándares de accesibilidad en la práctica o enseñanza en las arquitecturas de software. En los resultados se mencionó que en las arquitecturas de software ni siquiera se considera la accesibilidad como un componente a tomar en cuenta.

En el año 2015 (Peng Lu, 2015), se propuso una arquitectura para un servicio web orientado al aprendizaje electrónico basado en componentes y una capa de dominio. Esto técnicamente puede ser una solución a nivel de desarrollo, pero una vez más no se tomó en cuenta la accesibilidad ni la aplicación de los estándares en ninguno de sus componentes a alto nivel. En la conclusión el autor determinó que se debe profundizar en la investigación debido a la evolución de la educación, el trabajo colaborativo y los usuarios con diferentes capacidades.

En el año 2012 (Hongyu Pei Breivold, 2012), realizó una revisión de las arquitecturas de software y su evolución encontrando que existen varios factores que pueden cambiar debido a los nuevos requerimientos del negocio para ajustarse al mercado, migraciones o administración de proyectos. Estos factores, se dividieron en categorías como consideraciones de calidad de diseño, calidad de arquitectura, valor económico y técnicas de modelado. Sin embargo, estas categorías no cubren el amplio espectro de la evolución del software y su arquitectura, siendo la accesibilidad el factor menos tomado en cuenta, siendo los propietarios del sistema los que tratan la hoja de ruta de la arquitectura o evolución.

Material y métodos

A continuación, se presentan los pasos que se siguieron en esta investigación:

1. Análisis del diseño de la interfaz gráfica. En esta fase se determinarán los roles de SCRUM, se presentan los requerimientos de los usuarios y se especifica las tareas para cada requerimiento denominado Sprint Backlog (pila de tareas), con su estimación y prioridad.
2. Evaluación de marcos de trabajo de JavaScript. En el segundo paso se procederá con una revisión y evaluación de las distintas arquitecturas más utilizadas en el desarrollo de sitios web, así como también de las diferentes tecnologías, librerías y entornos de

trabajo actuales para el desarrollo de interfaces gráficas de usuarios para sitios web. Se pretende establecer una relación entre la arquitectura un utilizada, su estilo y patrones de diseño y establecer la relación de estos con el código y el principio de operabilidad de la WCAG.

3. Desarrollo de la arquitectura. De la abstracción de los análisis previos, se establecerá una arquitectura procedimental con elementos mínimos, desde el diseño conceptual de la aplicación hasta la puesta en producción que satisfaga los principios WCAG y se pueda desarrollar una aplicación accesible.
4. Pruebas de la arquitectura. En este paso de procederá con las pruebas de un sitio web que aplique la arquitectura propuesta con herramientas automatizadas y de ser posible con personas que padezcan discapacidad, esto asegura que la aplicación garantice su accesibilidad.

Resultados

Análisis del diseño de la interfaz gráfica

Para la evaluación de una interfaz gráfica se analizaron sitios web que implementan correctamente los principios de las WCAG. Con la finalidad de analizar cómo se implementaron los principios de percepción y entendimiento y como fueron estructurados en los sitios web. De esa abstracción se obtuvo un denominador común para el desarrollo de la arquitectura. Se analizaron más de 20 sitios web que siguen los lineamientos WCAG, todos estos diseños aplican correctamente los principios de percepción y entendimiento.

Con el análisis se encontró que utilizan colores con alto contraste por defecto, una única fuente, un diseño simple y entendible, separan correctamente cada sección, cada sección contiene un título, un párrafo corto y claro con lenguaje fácil de entender, separación semántica de secciones con espacios en blanco, separación tonal con el fondo y una sola tipografía. El uso de estos parámetros hace que los sitios sean fáciles de escanear con la vista, de entender y tienen una estructura predecible similar en el resto de páginas. También se encontró que algunos de estos sitios web al mostrar el menú principal en todas sus páginas, al navegar a otra página, muestran el link de saltar contenido, el cual permite dirigirse directamente al contenido principal de las misma, sin necesidad de recorrer el menú principal u otras secciones.

Además, los sitios web analizados usan correctamente la paleta de colores, para favorecer a usuarios con deficiencia en la visión, se recomienda utilizar herramientas para seleccionar paletas de colores accesibles. También, los sitios web con respecto al color utilizan por defecto colores con alto contraste evitando así la programación para la selección de alto contraste. Se encontró también que no tienen ayudas para personalizar la tipografía, como cambiar su tamaño, pero implementa correctamente la adaptación al cambio de tamaño del explorador web por lo que al aplicar un zoom de 200% los sitios mantienen su estructura al adaptarse correctamente, por lo que la implementación del cambio de tamaño de tipografía puede no ser necesario.

Evaluación de marcos de trabajo de JavaScript

Se analizaron diferentes librerías y entornos de trabajo de JavaScript o TypeScript más populares como: React, Vue.js, Next.js, Nuxt.js, Svelte, Ember, Backbone.js, todas estas están orientadas al desarrollo acelerado de sitios web reactivos, ligeros y con las mejores

prácticas, entre estas la accesibilidad. La documentación de estas librerías coincide en el término a11y para dirigirse a la accesibilidad y consideran necesaria su correcta aplicación para permitir que las tecnologías de asistencia puedan interpretar correctamente el sitio web. Algunas librerías como React y Vue.js recomiendan en primera instancia técnicas estándar de HTML como el uso correcto de los encabezados, ejemplo h1. En Vue.js recomiendan inicialmente que el diseño mismo permita una implementación accesible y que la estructura del contenido debe ser la adecuada. Además, la mayoría de estos entornos de trabajo recomiendan la aplicación de los estándares WCAG en mayor o menor alcance. Otro punto a destacar, es establecer puntos de referencias a través de los roles ARIA, esto proporciona un acceso programático a ciertas secciones utilizando tecnologías de asistencia, esto permite navegar directamente a una sección y omitir cierto contenido.

La mayoría de entornos de trabajo recomiendan el uso de atributos WAI-ARIA para construir widgets más complejos. Otros entornos mencionan que se debe tomar precauciones al extender o aumentar elementos nativos de HTML, puesto que puede sobrescribir funcionalidades bien soportadas por los navegadores. También recomiendan utilizar contenedores para estos elementos nativos para agregar instrucciones ARIA y no reemplazarlo con widgets incompatibles.

Entre las coincidencias que se encontraron, es que todos los entornos de trabajo respetan los 4 principios rectores principales de la WCAG 2.1 que son: Perceptible, Operable, Comprensible y Robusto, también conocido como POUR por sus siglas en inglés. Otra coincidencia fue la aplicación de una correcta semántica del código HTML, la documentación de estos entornos de trabajo manifiesta que el uso de las técnicas HTML y su semántica es la base de la accesibilidad, pues estos refuerzan el significado de la información que cada etiqueta contiene, no se garantiza la accesibilidad, pero contribuye a un gran porcentaje de la misma. Otras formas semánticas también se aplican en los formularios, estos tienen diferentes etiquetas como: <form>, <label>, <input>, <textarea>, <button> entre otras, se debe utilizar correctamente estas etiqetas junto con atributos que agregan información a los mismos, por ejemplo, el uso del autocomplete, for en las etiquetas label y id en las etiquetas input. Otra coincidencia que se encontró es la recomendación de uso de elementos nativos del HTML, puesto que estos tienen interacciones estandarizadas totalmente compatibles con la accesibilidad, en lo posible se debe utilizar estos componentes antes que re implementarlos con widgets complejos e incompletos, un ejemplo de esto sería utilizar directamente la etiqueta <button> en un formulario antes que crear un componente que simule su comportamiento o en su defecto el componente creado debe ser un contenedor de la etiqueta apropiada.

Para la construcción de widgets más complejos con JavaScript con técnicas accesibles, se encontró que estos entornos de trabajo, utilizan la Iniciativa de Accesibilidad Web - Aplicaciones de Internet Enriquecidas y Accesibles (WAI-ARIA por sus siglas en inglés), WAI-ARIA es muy útil con el contenido dinámico y controles avanzados en los que ciertos usuarios dependen de lectores de pantalla o que no puedan utilizar el mouse. WAI-ARIA ayuda a la accesibilidad mediante roles para identificar los tipos de widget, roles de estructura, propiedades de estado, además proporciona navegación por teclado para los objetos y eventos web como enfoques correctos de objetos. Ciertos entornos de trabajo recomiendan usar esta técnica con puntos de referencia para poder tener acceso programático a los mismos, las tecnologías de asistencia utilizan estos roles de ARIA para navegar entre secciones de manera rápida y omitir ciertos contenidos, se deben utilizar roles ARIA siempre que sea posible.

Para los controles de foco, es decir el elemento del Modelo de Objetos del Documento DOM por sus siglas en inglés, que está actualmente activo también se puede utilizar ARIA, además de ayudas visuales mediante CSS, además de utilizar atributos de HTML como tabindex, se debe asegurar que utilizar el teclado o el mouse genere la misma experiencia.
De la notificación de errores, algunos entornos coinciden en que deben mostrar o listar los errores adecuadamente y deben ser accesibles a todos los usuarios, según WCAG se pueden utilizar etiquetas prominentes como h1, title, cuadros de dialogo usando ARIA con el rol adecuado, mediante inputs con el atributo describedby de ARIA. De la configuración correcta del idioma, debido a que algunos lectores de pantalla lo utilizan para seleccionar la voz correcta. Además del título del documento, esto da a los usuarios no videntes contexto del sitio actual. La Tabla 1 muestra una comparación de las recomendaciones de aplicación de los estándares de accesibilidad según cada marco de trabajo:

Tabla 1.
Aplicación de accesibilidad según marco de trabajo JavaScript.

Marco de trabajo	Recomienda semántica HTML	Recomienda estructura	Etiquetas Nativas	WAI-ARIA	Atajos de teclado	Saltar a contenido	Asistencia a lectores de pantalla
Vue1	Si	Si	Si	Si	Si	Si	Si
React2	Si	Si	No	Si	Si	No	No
Svelte3	Si	No	Si	Si	Si	No	No
Backbone4	No	No	Si	No	No	No	No
Nuxtjs5	Si	Si	Si	Si	Si	No	Si
Nextjs6	Si	Si	No	Si	Si	No	Si

La Tabla 1 nos presenta que el 83% de los entornos de trabajo recomienda una correcta semántica HTML para mantener la accesibilidad, además el 66% usa etiquetas HTML nativas que es lo recomendable puesto que las mismas aseguran la accesibilidad por defecto. Solo el 16% toman en cuenta un vínculo para saltar al contenido de la página y el 50% favorece por defecto a los lectores de pantalla.
Del análisis de estos entornos o librerías, se encontró que la mayoría de estos solo establecen pautas y recomendaciones en base a su propio criterio, aplicando los principios WCAG en mayor o menor medida. Estas son solo recomendaciones, para algunos entornos de trabajo no es obligatorio el uso de las mismas por lo que el programador es libre de obedecerlas o no, resultando en una aplicación con falencias en accesibilidad.

Desarrollo de la arquitectura

El principal problema al diseñar e implementar un sitio web es que el personal técnico no es necesariamente experto en accesibilidad, es más, su conocimiento en principios y pautas de la WCAG puede ser nulo. Esto deriva a aplicativos muy dinámicos, con un buen desempeño y robustos, pero con mala puntuación en accesibilidad. Por lo que se propone la siguiente arquitectura para el diseño e implementación de sitios web accesibles. La Figura 1 representa

[1] https://vuejs.org/
[2] https://react.dev/
[3] https://svelte.dev/
[4] https://backbonejs.org/
[5] https://nuxt.com/
[6] https://nextjs.org/

la arquitectura en alto nivel, esta se distingue de varias listas de chequeo disponibles en línea, debido a que estas son una revisión después de la implementación de la aplicación, y también se distingue de algunos gráficos que muestran los criterios WCAG, pero son poco explicativos.

Figura 1.

Diagrama de arquitectura de alto nivel de desarrollo con accesibilidad.

Fuente: Elaboración propia.

Esta arquitectura es independiente de la metodología a usarse, puede ser viable con metodologías ágiles como SCRUM, prototipado, entre otras. A nivel de código, esta utiliza una arquitectura en capas, la capa a nivel de frontend es la inherente y sujeta de aplicación de los principios de accesibilidad por lo que las capas del lado del backend se descartan. En el frontend se recomienda en una metodología basada en componentes conocida como diseño atómico, la organización de directorios en un proyecto deriva en una arquitectura estructural. Según Brad Frost, el diseño atómico es una metodología compuesta por cinco etapas distintas que trabajan juntas para crear sistemas de diseño de interfaces de forma más deliberada y jerárquica, estas etapas son: átomos, moléculas, organismos, plantillas y páginas (Frost, 2022). El diseño atómico nos permite crear componentes básicos conocidos como átomos, un ejemplo de átomo es una etiqueta button de html, pero está diseñado de tal manera que consta con todas las clases, atributos y propiedades que el elemento en sus diferentes estados pueda tener según la guía de diseño, esto resulta muy útil debido a que evita inconsistencias en el desarrollo, siendo este componente reutilizable en todo el sistema. Estos átomos son la base para crear moléculas, que son la unión de átomos, las moléculas forman organismos que para efectos de diseño se pueden ver como secciones y la unión de secciones forman páginas, aquí se puede ver el beneficio del diseño atómico, un componente atómico bien diseñado permite su reutilización, reduce inconsistencias, menos componentes repetidos, mayor performance y para efectos de accesibilidad se evitaran componentes que no tengan una correcta configuración según los criterios WCAG.

Fase de diseño

El primer punto en la propuesta de arquitectura es el diseño, este debe obedecer a los principios de WCAG de percepción y entendimiento. Se requiere conocer el tipo de usuario objetivo, este análisis es importante puesto que en función del tipo de usuario se establece el contexto de la aplicación, puesto que distintos usuarios pueden tener ciertos tipos de discapacidad, una de ellas intelectual, o diferencias de cultura o de lenguaje. Es recomendable favorecer un diseño limpio, sin agentes distractores evitando secciones u organismos con moléculas que no son necesarias en dicho contexto, simplificando siempre el diseño. Se debe tener en cuenta una paleta de colores que favorezcan el contraste por defecto, el contraste mínimo recomendado por WCAG entre texto y fondo debe ser de una diferencia de luminancia de 3:1. De las secciones, estas deben tener solo información relevante y concisa sobre un tema específico, además deben ser estas predecibles así como otras estructuras de la aplicación, siendo estas estructuras: menú, barra lateral, encabezado, pie de página y

secciones, de esta manera la navegación por el sitio resulta familiar y el usuario sabrá en donde se encuentra y donde encontrar información en todo momento. En la Figura 2 se muestra la arquitectura propuesta con los puntos más relevantes de cada sección.

Figura 2

Puntos por sección de la propuesta de arquitectura.

Fuente: Elaboración propia.

Contenido

El contenido se basa también según el contexto del usuario, la compresión de lectura entre un adulto, un niño, una persona con discapacidad intelectual difieren enormemente, es por eso que el contenido debe ser evaluado correctamente según el público objetivo, este debe tener una tipografía legible, debe contener espacios semánticos, para separar secciones o contenidos con temas diferentes. Cada sección debe tener párrafos cortos y concisos aplicando un lenguaje simple, fácil de entender, si se encuentran palabras técnicas se debe dar una descripción, además de expandir las abreviaturas. Las secciones deben tener una secuencia de contenido con sentido.

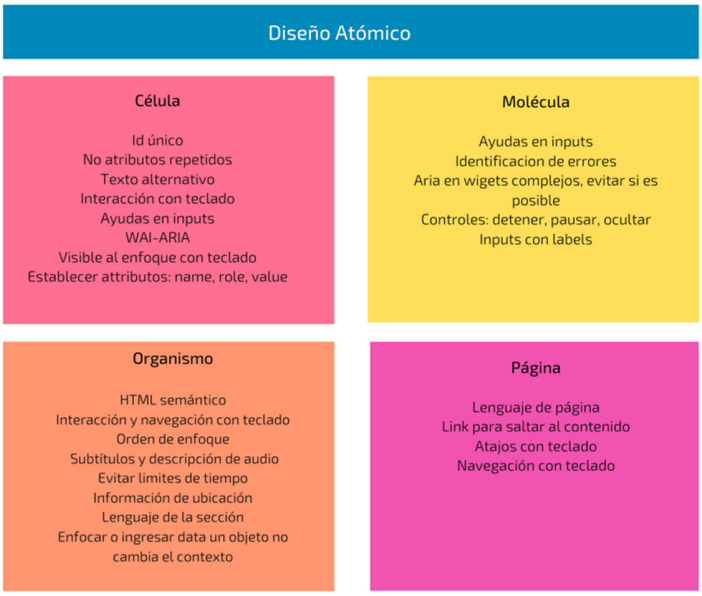
Implementación

El patrón estructural a utilizarse en la capa de frontend será el que establece la metodología de Diseño Atómico. En cada componente se pueden aplicar técnicas que siguen las directivas WCAG. La estructura de implementación destaca los directorios: cells, molecules, organisms, layouts y pages, por buenas prácticas sus nombres están en inglés.

En la Figura 3 se muestra la arquitectura propuesta en la capa de implementación con los componentes mínimos de cada sección.

Figura 3.

Arquitectura propuesta sección de implementación.



Fuente: Elaboración propia.

Células

Las células son componentes más elementales como una etiqueta button en el cual se debe configurar todos los atributos, propiedades y posibles estados. Dependiendo del componente se debe configurar un id único, atributos necesarios no repetidos como name, role value, texto alternativo de ser necesario, interacción y visibilidad al enfoque con teclado, de ser un input deben constar con etiquetas label y ayudas necesarias, de ser un widget complejo agregar atributos WAI-ARIA, de ser posible estos tipos de widget deben evitarse. En la Figura 4, se muestra la implementación de un componente de botón en el cual se aplican todas las configuraciones necesarias como técnicas WCAG. La ventaja de esta técnica es que el componente es reutilizable lo que reduce las inconsistencias e incrementa la valoración de accesibilidad.

Figura 4.

Componente de botón accesible.

```
export const Button: FC<Props> = ({  6+ usages
    url = '#',
    className = '',
    children : string | ... ,
    onClick : ... ,
    ariaLabel = '',
}) => {
    return (
        <Link
            href={url}
            onClick={onClick}
            className={`fade-in relative text-white bg-gradient-to-r f
            ${className}`}
            aria-label={ariaLabel}
        >
            {children}
        </Link>
    )
```

Fuente: Elaboración propia.

Molécula

La molécula representa un componente conformado mediante células, para seguir con los lineamientos WCAG se debe tener cuenta las ayudas en inputs, inputs con labels, identificación de errores, atributos WAI-ARIA, de ser necesario controles como: detener, pausar y ocultar.

Organismo

Un organismo está conformado por moléculas, a la vista de un usuario, representaría a una sección, para que un organismo siga los lineamientos WCAG debe tener en cuenta el html semántico que es una de las principales recomendaciones.

Página

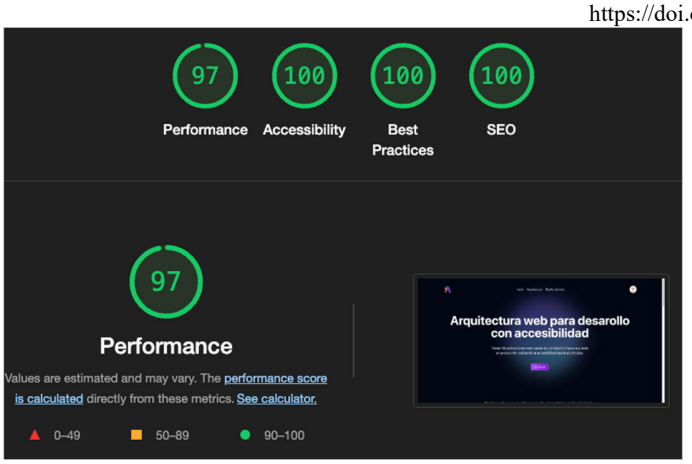
Una página es el resultado de la unión de uno o varios organismos, se debe configurar el lenguaje de página para ayudar a software con voz sintética, al ingresar a una página debe estar disponible un botón de saltar a contenido, el cual evita recorrer ciertas estructuras con el teclado. De implementar atajos con teclado, estos se deben configurar a nivel de página para evitar conflictos. En la página se debe asegurar la navegación con teclado, el enfoque debe recorrer cada sección en orden.

Pruebas

Para la revisión de la implementación el primer paso es la revisión con la aplicación Lighthouse del explorador Google Chrome, existen aplicaciones como Tenon que permiten analizar en línea, en esta herramienta, nuestra propuesta no presenta errores, plugins o mediante API, otras alternativas pueden ser: Wave para nuestro caso dio resultados con cero errores y cero errores de contraste, AChecker que resultó con cero errores, Colour Contrast Analyzer con una puntuación de 5:1, W3C Markup Validator Service para nuestro caso no presentó errores, Adobe Color. En la Figura 5 se presenta un análisis del proyecto implementado con la propuesta de la arquitectura mediante la herramienta Lighthosue.

Figura 5.

Análisis del sitio web implementado la arquitectura de accesibilidad propuesta.



Fuente: Lighthouse.

Como se había descrito anteriormente la semántica es esencial, se analizó la misma con la herramienta Html Checker Validator de W3C en el cual no se presentaron errores. No se puede confiar cien por ciento en una sola herramienta, se recomienda analizar la aplicación con varias alternativas. Otra de las pruebas que no siempre son posibles, pero es una de las más importantes es mediante personal con discapacidad, el usuario dependerá del contexto y al público objetivo, según el usuario se podrán ejecutar distintas pruebas como navegación mediante teclado, pruebas de contraste de color, según Wave, la propuesta obtuvo un resultado de 8.59:1, el cual satisface el nivel AAA de la WCAG, otras pruebas pueden ser pruebas con voz sintética, pruebas de comprensión y desactivar los estilos CSS para una prueba de estructura. En la Tabla 2 se recopila información sobre las pruebas de accesibilidad obtenidas con diferentes herramientas.

Tabla 2.

Evaluación de accesibilidad.

Herramienta	Nivel	Puntuación general	Errores generales	Errores de contraste	WAI-ARIA	Estructura semántica
Access Monitor[7]	AAA	9.9	2	1	Si	Si
WAVE[8]	AAA	-	0	0	Si	Si
Lighthouse[9]	AAA	100	0	0	Si	Si
Achecker[10]	AA	-	0	0	Si	Si
ARC Toolkit[11]	AA	-	0	0	Si	Si
HTMLChecker[12]	AA	-	0	-	Si	Si
AccessScan[13]	AAA	95	1	0	Si	Si
SiteImprove[14]	AAA	95	5	0	Si	Si

[7] https://accessmonitor.acessibilidade.gov.pt/
[8] https://wave.webaim.org/
[9] https://n9.cl/wfyn
[10] https://achecker.achecks.ca/checker/index.php
[11] https://www.tpgi.com/arc-platform/arc-toolkit/
[12] https://validator.w3.org/
[13] https://accessibe.com/accessscan
[14] https://n9.cl/0hlz2

Tenon15	AA	-	0	0	No	No
SiteSort16	AAA	-	1	0	Si	Si
DynoMapper17	AA	-	0	0	No	No
Rocket Validator18	AA	-	0	0	No	No

No todas las herramientas de pruebas de accesibilidad presentan un coeficiente, algunas de ellas muestran la cantidad de errores o advertencias. Entre las herramientas que presentan un puntaje se obtiene una media de 97.25% de ajuste a los estándares WCAG 2.1 nivel AAA. El 50% es capaz de analizar con nivel AAA. En el 33% de las herramientas presentan errores generales y solo en el 8% presenta errores de contraste.

Discusión

Hoy en día en el cual el trabajo remoto ha demostrado en varios casos ser más productivo, el uso de aplicaciones basadas en la web se ha incrementado, siendo necesario que estos aplicativos sean accesibles para la mayoría de sus usuarios. Como concluyó (Milton Campoverde-Molina, 2021), es necesario una propuesta de arquitectura guía para un desarrollo con accesibilidad, también se muestra que varias aplicaciones analizadas no aplican las directrices WCAG. Otros trabajos en línea como Project A11Y solo presentan listas de verificación (ProjectA11Y, 2023), a diferencia, esta propuesta es una arquitectura desde la concepción del proyecto, su diseño, evaluación de contenido y guía en tiempo de desarrollo para alcanzar niveles de conformidad según WCAG, para el desarrollo de un sitio web accesible, y no como en los casos anteriores, en el cual el desarrollo es primero y las verificaciones después.

Conclusiones

El resultado de esta investigación era proponer una arquitectura para desarrollar sitios web accesibles. El resultado principal fue la creación de una arquitectura de lado del frontend, apoyada en la arquitectura estructural denominada como Diseño Atómico. El diseño atómico permite la creación de componentes evitando la repetición de código y favoreciendo la consistencia, siendo ideal para nuestra propuesta. La arquitectura desarrollada se basa en cuatro pilares fundamentales: diseño, contenido, implementación y pruebas. En cada fase se elabora un análisis del público objetivo para así favorecer las directrices WCAG.
Luego establecer esta arquitectura, se verificó la misma desarrollando un sitio web siguiendo sus lineamientos, en el cual sus resultados fueron más que alentadores, fácilmente obteniendo puntuaciones muy altas y un nivel de conformidad de al menos AA en varias herramientas de análisis de accesibilidad, en algunas herramientas se presentaban advertencias, pero en ninguna, errores como tal. Se recomienda la aplicación de la arquitectura para alcanzar la mayor cantidad de usuarios posibles, acatar regulaciones e introducir a los diseñadores,

[15] https://tenon.io/
[16] https://www.powermapper.com/
[17] https://n9.cl/dl320e
[18] https://rocketvalidator.com/

generadores de contenido y desarrolladores a ser conscientes de la accesibilidad y que es posible aplicarlas siguiendo lineamientos básicos.

Referencias bibliográficas

Abid Ismail, K. K. (2022). Web accessibility investigation and identification of major issues of higher education websites with statistical measures: A case study of college websites. *Journal of King Saud University – Computer and Information Sciences, 34*, 901-911.

Bass, C. K. (2004). *Software Achitecture in Practice.* Boston: Pearson Education.

Centers, M. h. (2022). *Mentalhelp.net*. Retrieved 03 23, 2022, from https://www-mentalhelp-net.translate.goog/disabilities/?_x_tr_sl=auto&_x_tr_tl=es&_x_tr_hl=es

Fangchao Tian, P. L. (2022). Relationships between software architecture and source code in practice: An exploratory survey and interview. *Information and Software Technology, 141*.

Ferreira. (2010). De la minus-valía a la diversidad funcional: un nuevo marco teórico-metodológico. *Política y Sociedad, 47*(1), 45-65.

Frost, B. (2022). *Atomic Design.* Pittsburgh: Brad Frost.

H. Cervantes, P. V. (2016). *Arquitectura de Software conceptos y ciclo de desarrollo.* Sant Fe: Cengage Learning Editores.

Hongyu Pei Breivold, I. C. (2012). A systematic review of software architecture evolution research. *Information and Software Technology*, 54, 16–40.

María Esther Perez Dalmeda, G. C. (2019). Theoretical models of disability: tracing the historical development of disability concept in last five decades. *Revista Española de Discapacidad, 7*, 7'27.

Marno Retief, R. L. (2018). Models of disability: A brief overview. *Theological Studies*.

Milton Campoverde-Molina, S. L.-M. (2021). Systematic literature review on software architecture of educational websites. *The Institution of Engineering and Technology*.

Mondaq. (2022). *Las demandas federales por accesibilidad de sitios web aumentaron en 2021 a pesar de la pausa pandémica de mitad de año.* Retrieved 06 21, 2023, from https://n9.cl/fcbv7

Muzo, A. (2008). Análisis Diseño e Implementación De Una Alternativa De Solución Para Mejorar La Navegación Web Para Personas Discapacitadas.

OMS. (2011). *Informe mundial sobre la discapacidad.* Retrieved 06 21, 2023, from http://www.who.int/disabilities/world_report/2011/summary_es.pdf

Peng Lu, X. C. (2015). E-learning-Oriented Software Architecture Design and Case Study. *iJET*, vol. 10, Issue 5.

Pérez, J. E. (2009). *Introducción a JavaScript*. Retrieved 06 21, 2023, from https://n9.cl/neyyv

ProjectA11Y. (2023, 0628). *A11Y Project*. Retrieved from https://www.a11yproject.com/checklist/

S. Angelov, P. d. (2017). Designing and applying an approach to software architecting in agile projects in education. *The Journal of Systems and Software*, 127, 78 – 90.

W3C. (2022). *Accesibility*. Retrieved 03 19, 2022, from https://www.w3.org/standards/webdesign/accessibility

W3C. (2022). *Web Accessibility Initiative WAI*. Retrieved 06 21, 2023, from https://n9.cl/2tudpq

WCAG. (2023). *Web Accessibility Initiative WAI*. Retrieved 06 21, 2023, from https://n9.cl/ezxq4.

Conflicto de intereses:
Los autores declaran que no existe conflicto de interés posible.
Financiamiento:
No existió asistencia financiera de partes externas al presente artículo.
Agradecimiento:
N/A
Nota:
El artículo no es producto de una publicación anterior.