@dministración electrónica

Observatorio de Accesibilidad

GUÍA DE ACCESIBILIDAD DE APLICACIONES MÓVILES (APPS)

GOBIERNO DE ESPAÑA
MINISTERIO DE HACIENDA Y FUNCIÓN PÚBLICA
SECRETARÍA DE ESTADO DE FUNCIÓN PÚBLICA
SECRETARÍA GENERAL DE ADMINISTRACIÓN DIGITAL

FICHA CATALOGRÁFICA

TÍTULO: Guía de accesibilidad de aplicación móviles (APPS)

Elaboración y coordinación de contenidos:

Publicación elaborada por Juan Aguado Delgado y Francisco Javier Estrada Martínez para la Secretaría General de Administración Digital. Esta guía se ha realizado con el apoyo de la Red de Cooperación ESVI-AL, de la Comunidad Autónoma de Madrid y de la Universidad de Alcalá (UAH).

Diciembre, 2017

Disponible esta publicación en el Portal de Administración Electrónica (PAe):
http://administracionelectronica.gob.es/

Edita:

© Ministerio de Hacienda y Función Pública
Secretaría General Técnica
Subdirección General de Información,
Documentación y Publicaciones
Centro de Publicaciones

Colección: administración electrónica

NIPO: 169-17-202-8

El presente documento cumple con las condiciones de accesibilidad del formato PDF (Portable Document Format).

Se trata de un documento estructurado y etiquetado, provisto de alternativas a todo elemento no textual, marcado de idioma y orden de lectura adecuado.

Para ampliar información sobre la construcción de documentos PDF accesibles puede consultar la guía de accesibilidad en PDFs con Adobe Acrobat 9.0 disponible en el área de documentación del Portal de la Administración Electrónica (PAe)
http://administracionelectronica.gob.es/PAe/accesibilidad/documentacion.

GOBIERNO DE ESPAÑA
MINISTERIO DE HACIENDA Y FUNCIÓN PÚBLICA
CENTRO DE PUBLICACIONES

ÍNDICE

1. INTRODUCCIÓN

El 2 de Diciembre de 2016 se publicó en el Boletín oficial de la Unión Europea la **Directiva** (UE) **2016/2102** del Parlamento Europeo y del Consejo, de 26 de octubre de 2016, sobre la **accesibilidad** de los sitios **web** y **aplicaciones** para dispositivos **móviles** de los organismos del **sector público**. Esta directiva establece los condicionantes, con respecto a su accesibilidad, que deberán cumplir todos los sitios web y aplicaciones móviles del sector público: estatal, regional, local, universitario, etc. incluyendo también entes como centros sanitarios y educativos, bibliotecas, tribunales, etc.

De este modo, a los requisitos de accesibilidad para los portales públicos, que ya se venían aplicando en España desde el Real Decreto 1494/2007, se incorporan los requisitos de accesibilidad a las aplicaciones móviles del sector público de modo que **todas ellas (antiguas o nuevas) deberán ser accesibles a partir del 23 de junio de 2021**.

El momento de inicio de **aplicación** parece lejano, sin embargo, tendrá efecto con **carácter inmediato** no contemplando **ninguna cláusula de excepción** o periodo de transición para aquellas aplicaciones desarrolladas antes de esa fecha. Por lo tanto, para no tener que acometer posteriores recodificaciones de las aplicaciones móviles, se recomienda empezar a aplicar, cuanto antes, los criterios de accesibilidad.

Según las previsiones de la directiva, los requisitos a cumplir son los de la norma **EN 301 549**[1], "Requisitos de accesibilidad de productos y servicios TIC aplicables a la contratación pública en Europa", equivalentes en el apartado **web** a las **WCAG 2.0 de nivel AA**. Sin embargo, en el caso específico de las **aplicaciones móviles**, con directrices aún poco desarrolladas, se ha contemplado la necesidad de generar a nivel europeo unas **Especificaciones Técnicas adicionales**. Así, la Comisión Europea ha preparado el mandato M/554 a las asociaciones europeas de estandarización, para llevar a cabo la actualización de la norma EN 301 549 de modo que se dé una cobertura más completa a las aplicaciones móviles, que debería estar disponible antes del 22 de diciembre de 2018.

En este contexto, teniendo en cuenta los requisitos ya definidos en la EN 301 549, se ha confeccionado el presente documento para **ayudar** a los **desarrolladores/evaluadores** de **aplicaciones móviles** (apps) en materia de **accesibilidad**, máxime considerando que a partir de la aplicación de la directiva mencionada, las empresas que trabajen en/para el sector público tendrán la **obligación** de crear aplicaciones móviles accesibles.

De este modo, utilizando esta guía será posible generar aplicaciones móviles nativas accesibles para los principales **sistemas operativos** del mercado (Android, iOS y/o Windows), así como evaluar el **nivel de accesibilidad** de las mismas, pudiendo detectar y subsanar posibles **barreras** que una app pueda presentar a los usuarios con discapacidad. Además, los contenidos descritos pueden usarse para aplicar **mejora continua** (múltiples iteraciones de evaluación y posterior corrección), lo que facilitará más aún que cualquier usuario (independientemente de sus capacidades) pueda utilizar la aplicación objetivo.

[1] http://www.etsi.org/deliver/etsi_en/301500_301599/301549/01.01.02_60/en_301549v010102p.pdf

Para generar una aplicación móvil accesible, es imprescindible que se consideren los **principios y conceptos** relativos a la accesibilidad tanto durante el **diseño** de dicha aplicación como en el momento de acometer el **desarrollo** del software final. Para lograrlo se requieren conocimientos técnicos concretos para el cumplimiento de todos los requisitos involucrados y la consideración de diferentes **procesos y herramientas**, útiles para acometer evaluaciones de accesibilidad en los momentos clave del ciclo de vida de la aplicación móvil.

Actualmente existen muchas **herramientas** que ayudan a evaluar la accesibilidad de páginas y sitios web, pero apenas hay herramientas para el caso de las **aplicaciones móviles**. Esperamos que esta **guía** sirva como una ayuda adicional en este nuevo ámbito de la accesibilidad en las aplicaciones móviles.

Esta guía es de utilidad tanto para los **desarrolladores**, que pueden hacer comprobaciones durante el diseño y creación de su aplicación móvil, como para las personas encargadas de realizar evaluaciones de accesibilidad sobre la aplicación desarrollada, normalmente **auditores** o **consultores** que velan por el cumplimiento de los requisitos de **accesibilidad**.

A la hora de crear este documento se han tenido en consideración los contenidos previamente publicados en algunas **guías con recomendaciones** sobre accesibilidad para los programadores de apps, como las ofrecidas por el World Wide Web Consortium (W3C) en su web bajo la denominación "Mobile Accessibility", el documento de Santiago Gil titulado "Cómo hacer Apps accesibles", publicado en 2013 en España por CEAPAT-IMSERSO, o la “Metodología para Evaluar la Accesibilidad de Aplicaciones Nativas” confeccionada por ILUNION en el 2015.

La “Guía de accesibilidad de aplicaciones móviles (apps)” surge como resultado de la **colaboración** de la **Universidad de Alcalá (UAH)**, **ILUNION**, el **IMSERSO-CEAPAT** y el **Observatorio de Accesibilidad** del Ministerio de Hacienda y Función Pública del Gobierno de España. Además, se ha contado con el **apoyo** de la **Red ESVI-AL** de Cooperación sobre Accesibilidad en la Educación y Sociedad Virtual, de la que forman parte, entre otros, las asociaciones de personas con discapacidad agrupadas en la Unión Latinoamericana de Ciegos (ULAC) y la Organización Mundial de Personas con Discapacidad (OMPD).

Por último, es importante recalcar que conseguir que las aplicaciones móviles sean más accesibles supone una **transferencia** directa a la **sociedad**, que no solamente favorece a las personas con discapacidad que se descargan y utilizan apps en sus dispositivos, sino a cualquier otro usuario, ya que está demostrado que la accesibilidad **beneficia** a todas las **personas**.

Además, no debe olvidarse que crear aplicaciones accesibles también **ayuda** a las **organizaciones** a satisfacer sus responsabilidades legales, a construirse una imagen corporativa respetable y a ampliar su base de clientes (aproximadamente un 6% de la población española presenta alguna discapacidad, según cifras del IMSERSO[2]).

[2] http://www.imserso.es/imserso_01/documentacion/estadisticas/bd_estatal_pcd/index.htm

2. OBJETIVOS -

Esta guía tiene como objetivo principal ofrecer los **recursos** necesarios para que los desarrolladores puedan **crear aplicaciones móviles accesibles** y/o para que dichas aplicaciones puedan ser **evaluadas** posteriormente (por desarrolladores, consultores, auditores, etc.) para determinar su nivel de accesibilidad.

Como objetivos específicos del documento se plantean los siguientes:

1. - Comprender los **tipos de discapacidad** existentes y su impacto en las características de las aplicaciones desarrolladas para dispositivos móviles.

2. - Localizar aquellas **carencias**, en el proceso de creación o en el producto final, que puedan suponer una reducción de la **accesibilidad** en las aplicaciones móviles.

3. - **Unificar** todos los **recursos** anteriores que sean relevantes, concretándolos en una serie de secciones interrelacionadas que sirvan de referencia para el desarrollo/evaluación de aplicaciones móviles accesibles.

Como puede deducirse de la lectura de los párrafos anteriores, la guía está enfocada para ser de utilidad a **profesionales** de **diversos perfiles**, como desarrolladores, editores de contenido o, en general, a cualquier individuo dedicado a la consecución de los objetivos de accesibilidad de una aplicación móvil en cualquiera de las etapas de su ciclo de vida (desde el momento inicial de su diseño hasta la gestión de contenidos y monitorización durante la fase de producción).

3. CÓMO UTILIZAR LA GUÍA -

La **forma de emplear** el presente documento depende en gran medida del **perfil** del lector, ya que, como se ha comentado anteriormente, los recursos que contiene pueden ser utilizados tanto por desarrolladores como por evaluadores:

- Los **desarrolladores** utilizarán, principalmente, los contenidos del **apartado 5** de la guía (Principios de desarrollo de aplicaciones móviles accesibles), donde conocerán cómo deben trabajar para que sus aplicaciones móviles sean accesibles. Además, contarán con **ejemplos de código** para las distintas plataformas (Android, iOS, Windows) en el anexo correspondiente, con lo que podrán disponer de una referencia a la hora de programar.

- Respecto a los **profesionales dedicados a la evaluación** del nivel de accesibilidad de las aplicaciones móviles, en el **apartado 6** (Validación de accesibilidad de aplicaciones móviles) pueden hallar cómo acometer el análisis de la aplicación móvil fijada como objetivo de la evaluación, complementando dicha información con los datos brindados en el anexo de **requisitos**.

Sin embargo, más allá del perfil profesional, hay que destacar que todas las secciones del documento son relevantes y de lectura recomendada, pues permiten fijar el **marco contextual** necesario para comprender la accesibilidad de las aplicaciones móviles y albergan **información de gran importancia**, como sucede con los perfiles de usuario, las herramientas de evaluación (útiles en fase de desarrollo y/o validación) y los productos de apoyo.

Además, no hay que olvidar tampoco que los **desarrolladores** deberían de realizar **evaluaciones** de sus propios productos y prototipos durante el proceso de creación del software, e incluso una vez terminado, para detectar y subsanar posibles fallos de accesibilidad que hayan podido pasar por alto.

Para poder seguir esta guía es recomendable **tener conocimientos previos de las pautas** y criterios de conformidad de las **WCAG 2.0** (en los que se basa en gran medida la norma europea EN 301 549), de los principios del **diseño** accesible, de los **productos de apoyo** existentes y de cómo utilizan los dispositivos y aplicaciones móviles las personas con discapacidad.

No obstante, en la guía se llevará a cabo un breve repaso de algunos **conceptos fundamentales** de las áreas anteriormente mencionadas, esenciales para poder desarrollar y/o validar aplicaciones móviles accesibles de forma rigurosa.

4. INTRODUCCIÓN A LA ACCESIBILIDAD DE APLICACIONES MÓVILES (APPS)

Considerando la **definición** de **accesibilidad** contenida en la norma EN 301 549 y la ofrecida por la Ley orgánica española 51/2003 (del 2 de diciembre de 2003), se puede establecer como la *condición que deben cumplir los entornos, instalaciones, productos, sistemas y/o servicios para que sean comprensibles, utilizables y practicables por todas las personas de la sociedad, independientemente de sus características y capacidades, para conseguir una meta específica en un contexto de uso específico*.

Una aplicación móvil es un tipo muy concreto de **software**, que según las tecnologías implicadas puede ser una aplicación **nativa**, una aplicación **web** o una aplicación **híbrida**.

De cualquier modo, para poder tener una idea adecuada respecto a la accesibilidad de las aplicaciones móviles, es conveniente comentar previamente cuales son las **propiedades** más relevantes que caracterizan a este tipo de software:

- Cuentan con un **diseño** especialmente pensado para ser utilizadas en dispositivos móviles (teléfonos inteligentes o tablets), con un acceso predominante mediante **pantalla táctil**.

- Se descargan de una **plataforma de distribución** gestionada por la empresa responsable del sistema operativo o por el fabricante del dispositivo, lo que suele implicar cierto nivel de **calidad** en el desarrollo y mayor **fiabilidad** y **seguridad** en el proceso de descarga e instalación (tanto en apps comerciales como en apps gratuitas).

- El proceso de **instalación** y las **actualizaciones** son sencillas, sin requerir la intervención del usuario. Posteriormente, se suele completar su configuración y personalización.

- Normalmente tienen un **tamaño reducido**, para adecuarse a las limitaciones técnicas de los dispositivos, aunque los terminales cada vez son más potentes y plantean menos limitaciones a este respecto.

- Al utilizarse en dispositivos personales, ya que un terminal móvil no suele compartirse con otras personas, los sistemas operativos **no requieren identificación** para garantizar la privacidad respecto a otros usuarios ni para personalizar el entorno de trabajo.

- Tienen una función de **herramienta de comunicación** más allá de la que tenían las aplicaciones convencionales de escritorio, por lo que las empresas y organizaciones distribuyen sus propias apps como servicios adicionales al consumidor o como soportes publicitarios.

Como se ha indicado, las aplicaciones móviles se ejecutan en **dispositivos móviles**, los cuales se han convertido en un fenómeno de **consumo** destacado en la sociedad actual, entre

otros motivos, por contar con **funcionalidades** muy demandadas (portabilidad, acceso táctil y simplicidad) y por la **necesidad** de las personas de estar permanentemente conectadas (correo electrónico, redes sociales, etc.).

Por supuesto, las personas con discapacidad quieren ser partícipes igualmente de este **fenómeno sociológico**, pero encuentran mayores dificultades en aquellos casos en los que no se respetan los principios ni se disponen los mecanismos oportunos para satisfacer un nivel adecuado de **accesibilidad**. Al tratarse de una tecnología de gran popularidad, el impacto resultante de la **exclusión** de dichos usuarios es más acusado, estigmatizándolos en gran medida en su día a día y agravando aún más su situación.

Desafortunadamente, existe la creencia de que las personas con **discapacidad**, especialmente aquellas con problemas visuales, no son capaces de utilizar los **dispositivos móviles** y las **aplicaciones móviles** que contienen. Sin embargo, cada vez hay más personas con discapacidad que utilizan estos dispositivos, incluyendo los terminales de última generación que se controlan a través de pantallas táctiles. Así, gracias a los dispositivos y aplicaciones móviles (siempre que sean accesibles) los usuarios con estos perfiles pueden **acceder** a las Tecnologías de la Información y las Comunicaciones (**TIC**), lo que les aporta mayor inclusión social e independencia.

El problema es que los **desarrolladores** de aplicaciones móviles no suelen estar familiarizados con las **necesidades** de los **usuarios** con discapacidad, lo que normalmente se traduce en **barreras** de accesibilidad presentes en sus productos, que impiden que puedan ser utilizados en igualdad de condiciones por todas las personas.

Además, no hay que olvidar que la accesibilidad afecta tanto a los propios dispositivos como al diseño y características de las aplicaciones móviles, siendo hoy en día una **asignatura pendiente**. Si se logra revertir esta situación, generando productos y servicios accesibles, se alcanzaría una **normalización** deseable y una mayor **integración**, ya que las personas con discapacidad siguen las mismas tendencias y quieren los mismos productos que el resto de la población, prestando gran atención a las prestaciones y al diseño de las aplicaciones y huyendo de apps especiales o “adaptadas”.

Para lograr este objetivo es necesario aplicar ciertas **prácticas** durante el desarrollo y cumplir con unos **criterios de conformidad** específicos (todo ello tratado en la presente guía), siendo necesario que los desarrolladores de las aplicaciones móviles tengan **conocimientos sólidos** sobre los requisitos de accesibilidad de las aplicaciones móviles, así como de la diversidad existente en las necesidades de los diferentes perfiles de la comunidad de usuarios, para poder ofrecer a todos ellos una **experiencia** de uso **similar**.

Considerando todo lo anterior, en esencia, una **aplicación móvil** puede ser considerada **accesible** cuando **cualquier usuario**, independientemente de su diversidad funcional, puede utilizarla en su dispositivo móvil satisfactoriamente con su sistema de acceso habitual.

4.1. - PRINCIPALES DIFICULTADES PARA PERSONAS CON DISCAPACIDAD AL USAR APPS

Las personas con discapacidad pueden presentar una amplia gama de **habilidades** y **necesidades** muy **heterogéneas**, dependiendo de la naturaleza de la discapacidad y del grado de afectación.

Para que una aplicación sea accesible, ésta debe cubrir todas las necesidades que un usuario pueda requerir, desde aquellos con dificultades **sensoriales** a los que presentan problemas **cognitivos** o **motores**. Por tanto, las aplicaciones deben satisfacer los requisitos propios de **perfiles** de lo más **diversos**, incluyendo el de aquellas personas que pueden experimentar ciertos procesos adversos, como convulsiones, al hacer uso de las mismas.

A continuación se describen varios perfiles de discapacidad que pueden encontrarse en los usuarios tipo de una aplicación móvil. El objetivo es que el desarrollador conozca las **barreras de accesibilidad** que estos usuarios pueden encontrar, de modo que se comprendan posteriormente las soluciones aportadas a cada problemática.

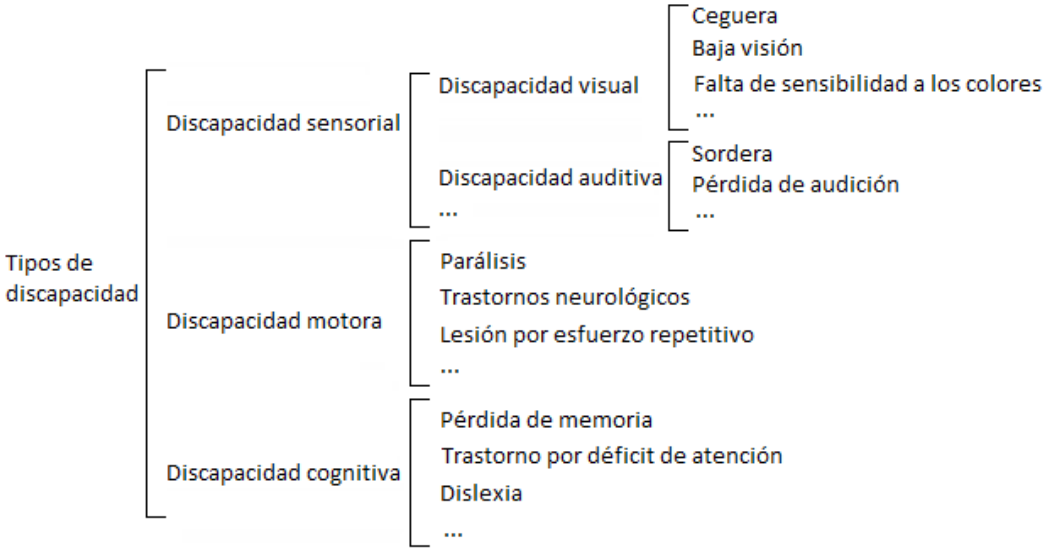

***Figura 1**. Clasificación de las principales formas de discapacidad*

4.1.1. Discapacidad sensorial

Los usuarios con **discapacidad sensorial** presentan dificultades o imposibilidad para percibir la información de **salida** ofrecida por el dispositivo utilizado, a través de uno o varios canales o **sentidos**. No obstante, esta falta de información también puede influir en la **entrada**, dependiendo del caso.

Las personas **invidentes** no pueden acceder al canal **visual**, mientras que aquellas que presentan dificultades en la visión, ya sea baja visión, daltonismo o cualquier otro trastorno relacionado con la vista, pueden hacerlo siempre y cuando se cumplan ciertas **condiciones**.

Ofrecer la información de forma **alternativa**, mediante el canal **sonoro** o **háptico**[3], puede ser una buena **solución** para estos casos, aunque también se pueden adoptar medidas para que el canal visual sea apropiado si el usuario cuenta con alguna capacidad visual. El tamaño del **texto** y los elementos, así como el **contraste**, son fundamentales para garantizar la accesibilidad para estos perfiles concretos de discapacidad visual.

Otro grupo afectado por la discapacidad sensorial es el de las personas con discapacidad **auditiva**. En este caso, los usuarios no pueden captar información a través del **oído**, o lo hacen con dificultad. Esto complica el hecho de que puedan mantener una conversación hablada, escuchen mensajes, utilicen componentes multimedia (como audios o vídeos) o reciban notificaciones sonoras. Por consiguiente, las aplicaciones deben plantear **alternativas visuales** o **hápticas** para transmitir la información, en lugar de utilizar el canal auditivo. Además, en muchos casos los problemas auditivos están relacionados con problemas del **habla**, por lo que la entrada de voz no es un medio de interacción adecuado.

También existen formas de discapacidad que afectan al sentido del **tacto**, evitando que el usuario pueda sentir los estímulos mecánicos, como la vibración del dispositivo. Aunque este canal no es tan importante a la hora de transmitir información, hay que tener en cuenta que cualquier notificación que se haga llegar por esta vía debe contar con **alternativas**.

4.1.2. Discapacidad motora

Otro de los grandes grupos de usuarios que encuentran barreras a la hora de interactuar con los dispositivos móviles es el de aquellos que presentan alguna clase de discapacidad que les impide o dificulta el **movimiento**, la aplicación de **fuerza** o el uso de varios **gestos** simultáneos.

En esta categoría existe una enorme variedad de casuísticas, desde los casos más severos, que implican la **incapacidad** de **mover** los brazos o las manos (o directamente la ausencia de éstos), hasta aquellos en los que el **movimiento** está **limitado** o entorpecido, de forma que no se pueden realizar **gestos** complicados (o varios simultáneamente) o aplicar fuerza.

Como puede apreciarse, los perfiles de **discapacidad motora** presentan dificultad sobretodo en relación al canal de **entrada**, ya que los dispositivos móviles se centran en la interacción táctil con la pantalla o en la pulsación de teclas. Para poner a disposición de estos usuarios las aplicaciones móviles, deben ofrecerse **alternativas** de entrada más **simples**, en caso de que sean complejas, e incluso abogar por la interacción mediante comandos de **voz**.

4.1.3. Discapacidad cognitiva

Por último tenemos el grupo de usuarios que presentan **discapacidad cognitiva**. Quizás éste sea el perfil más **heterogéneo**, pues, a pesar de que se disponen de clasificaciones aceptadas para las diversas patologías que la provocan, el grado de severidad varía mucho de unos

[3] El término *háptico* se refiere a la ciencia del tacto (háptica), análoga a la acústica en relación al oído y a la óptica en relación a la vista

individuos a otros. No obstante, en general, estos usuarios presentarán dificultades a la hora de **comprender** o **aprender** el funcionamiento de la aplicación.

La principal herramienta que podemos utilizar para lidiar con estas barreras es la **simplicidad** en el manejo. Cuanto más sencilla, **intuitiva**, consistente y **uniforme** sea una aplicación móvil a lo largo de todas sus vistas, más fácil les será de utilizar a las personas afectadas por esta clase de discapacidad. Por tanto, hay que evitar términos enrevesados, instrucciones complejas o navegaciones excesivamente largas y retorcidas.

4.2. PERFILES DE USUARIO CON DISCAPACIDAD

La norma europea EN 301 549 se refiere a los diferentes **perfiles** de usuario en el apartado titulado “Prestación funcional”, con un enfoque descriptivo no normativo por el cual se definen las posibles **necesidades** de los usuarios. A este respecto, en esencia, al desarrollar una aplicación móvil se debe perseguir que cualquier **interacción** que deba ser completada durante su ejecución pueda ser realizada mediante un **método válido** para el usuario, ofreciendo **alternativas** viables para todos los posibles perfiles (por ejemplo: inserción de texto tanto mediante teclado virtual como mediante control por voz).

Hay que tener en cuenta que existen toda clase de **necesidades** a la hora de interactuar con la **interfaz** de los dispositivos móviles, sabiendo que los sistemas operativos (**SSOO**) ofrecen características y **servicios** de lo más diversos para facilitar su uso lo máximo posible (lectores de pantalla, retroalimentación háptica, navegación por gestos, magnificación de pantalla, etc.).

Respecto a la **interfaz de usuario** de las aplicaciones, suele involucrar varios tipos de componentes (vistas y controles) que deben albergar la **información** pertinente para que los servicios de **accesibilidad** del SO y los productos de apoyo (software o hardware) puedan interactuar correctamente. Sin embargo, para garantizar un mayor nivel de **accesibilidad** y **usabilidad** de la aplicación, también deben considerarse muchos aspectos de **diseño** que pueden afectar a estas características, como la redacción de los mensajes de ayuda y/o de la documentación, la organización de los elementos de la interfaz u otros aspectos gráficos (ej: contraste entre texto y fondo).

De cualquier forma, para comprender con **mayor detalle** las posibles necesidades dependiendo del **perfil** del usuario, es conveniente revisar la sección oportuna del apartado ya mencionado (“Prestación funcional”) de la **norma** europea EN 301 549.

4.3. LA NORMA EN 301 549

A grandes rasgos, tanto la **estructuración** de la norma **EN 301 549** como la posterior forma de **evaluación** de la accesibilidad considerando todos sus contenidos, han sido articuladas en base a las **características** concretas del **producto TIC** en vez de en función de las tipologías clásicas del software. Así, se valora, por ejemplo, que el producto implique comunicación bidireccional, que involucre un sistema de reproducción de vídeo, que tenga hardware o características web… en lugar de considerarlo en su totalidad como una aplicación de escritorio, una aplicación móvil, una página web, etc. Este **enfoque** es más adecuado hoy en día, donde la **versatilidad** de los productos software es tan amplia que se están diluyendo las fronteras entre sistemas tradicionales.

Sabiendo esto, es fácil entender que **cada apartado** de la norma guarda relación con una **característica** específica de los productos TIC:

- En los **capítulos 0 al 4** se fijan los **conceptos básicos** para comprender la norma, con una breve introducción, el objeto y campo de aplicación, algunas referencias, definiciones y abreviaturas y una sección en la que se definen los posibles perfiles del usuario en base a sus capacidades (apartado de prestación funcional).

- En el **capítulo 5** se brindan **requisitos genéricos**, que son criterios de conformidad que deben ser considerados en cualquier caso, independientemente del producto a desarrollar/analizar.

- En el **capítulo 6** se detallan los requisitos aplicables a productos **TIC con comunicación bidireccional por voz**, por lo que sólo se tendrán en cuenta en las situaciones en las que la aplicación permita comunicar a dos personas de esta manera.

- El **capítulo 7** alberga los criterios de conformidad aplicables a productos **TIC con capacidades de vídeo**, así que están pensados para aquellos productos que permiten reproducir contenidos audiovisuales (contemplando incluso los subtítulos y la audiodescripción).

- En el **capítulo 8** se describen los requisitos que deben tenerse en cuenta en caso de que el producto implique **Hardware**.

- El **capítulo 9** contiene las pautas que afectan a las páginas **Web**, ampliamente basadas en los requisitos del nivel AA de WCAG 2.0.

- El **capítulo 10** alberga los criterios de conformidad para **documentos no web**, de modo que debe ser tenido en cuenta en los casos en los que el producto TIC sea un documento que no se corresponda con una página web, ni esté incrustado en una (o bien esté incrustado pero no sea presentado al usuario).

- El **capítulo 11** se refiere al **software** en general (software de plataforma, software con interfaces de usuario, herramientas de autor y productos de apoyo), por lo que es uno de los apartados más amplios del documento. Se trata de la **sección principal** en la que debe basarse cualquier **desarrollo/evaluación** de una **aplicación móvil**, independientemente de sus características (que luego pueden suponer que se tenga en consideración requisitos adicionales). En este apartado, además de concretar sus propios requisitos, se solicita de **manera indirecta** el cumplimiento de **otros apartados** de la norma, con requisitos de la sección 9 en caso de que el software tenga contenidos web, de la sección 10 si la aplicación contiene documentos no web, o de la sección 5 para casos de funcionalidad cerrada, entre otros. En este capítulo también se tiene en cuenta el uso de la accesibilidad documentado y las preferencias del usuario.

- En el **capítulo 12** se detallan los criterios de conformidad concernientes a la **documentación y los servicios de apoyo**, en los que se especifica el modo en que deben definirse estos recursos relacionados con el producto TIC para ser accesibles.

- El **capítulo 13** contiene los requisitos aplicables a **TIC que proporciona el acceso a un servicio de intermediación o emergencia**, útil en aquellos casos en los que el producto TIC permite a los usuarios de diferentes modos de comunicación (textos, signos, lectura de labios, voz) interactuar de forma remota a través de una TIC con comunicación bidireccional, gracias a un proceso de conversión entre los distintos modos de comunicación.

- Además se ofrecen varios **anexos** (tanto informativos como normativos) para completar la información del resto de apartados.

En lo que respecta a la **Directiva** (UE) 2016/2102 del Parlamento Europeo y del Consejo, de 26 de octubre de 2016, sobre la accesibilidad de los sitios web y aplicaciones para dispositivos móviles de los organismos del sector público, ésta **obliga** a los desarrolladores a cumplir los **requisitos** referenciados en el capítulo 9 en el caso de desarrollar **páginas web** (o aplicaciones web, incluidas las aplicaciones híbridas para móvil) y a cumplir los criterios de conformidad del capítulo 11 si el producto final es una **aplicación móvil** nativa.

Por tanto, en relación al desarrollo de **aplicaciones móviles** (que es lo que nos ocupa en esta guía), aunque es recomendable considerar todos los requisitos aplicables en base a las características que tenga la aplicación a desarrollar, tal y como se ha descrito más arriba, por **ley** únicamente habría que satisfacer lo estipulado en el **capítulo 11** de **EN 301 549**. Esto supone la consideración tanto de los requisitos explícitamente contenidos en el apartado 11 como aquellos a los que, indirectamente, se haga alusión.

En cualquier caso, esta **situación** puede **cambiar** en virtud de los trabajos actualmente activos para actualización de la **norma** EN 301 549, la definición de un posible **estándar** armonizado para la directiva y la posibilidad de definición de **especificaciones técnicas** concretas para Apps.

5. PRINCIPIOS DE DESARROLLO DE APLICACIONES MÓVILES ACCESIBLES

Para que una **aplicación móvil** sea **accesible** es necesario que cuente con unas **características** concretas y satisfaga ciertas directrices, que deben ser consideradas desde el mismo momento en que comience el desarrollo.

Así, no se debe esperar a que el producto esté terminado para someterlo a **evaluación** y considerar los **mecanismos** que garanticen un nivel adecuado de accesibilidad, sino que hay que acometer estas acciones en **todas las fases de desarrollo**, desde el diseño inicial hasta el mantenimiento, pasando por la codificación.

De hecho, una de las **razones** por las que muchas de las aplicaciones móviles que salen al mercado **no satisfacen** los **requisitos** de accesibilidad es justamente la falta de consideración desde las etapas iniciales, preocupándose únicamente por estas cuestiones cuando el desarrollo está **terminado** o, en el mejor de los casos, muy avanzado.

Este detalle es de gran relevancia, ya que la **dificultad** y complejidad a la hora de solventar los problemas de accesibilidad **va creciendo** conforme se va **avanzando** más y más en las etapas del desarrollo, de manera que incluir las mejoras de accesibilidad oportunas tendrá **un mayor coste e impacto si se realizan sobre un software terminado** y entrañará menos quebraderos de cabeza y problemas en la gestión de recursos si se hace desde el principio, además de dar lugar a mejores resultados.

Por tanto, completar labores de accesibilidad en el **desarrollo** es fundamental, estableciendo con cuidado el contraste entre el color del fondo y el texto, la paleta de colores, la descripción de los elementos no textuales que sean significativos, los metadatos que serán utilizados por los productos de apoyo, etc. No olvide que **los problemas de accesibilidad se evitan y corrigen de manera más sencilla en las etapas iniciales del desarrollo**.

Es recomendable que se comunique y **registre** debidamente **cómo** se va a **trabajar** la **accesibilidad** en lo relativo a la evaluación temprana y el desarrollo, para que los clientes y desarrolladores lo tengan en cuenta en la **planificación** del proyecto. Así, se mecanizará en gran medida el proceso, se identificarán las técnicas y herramientas a utilizar, se sabrá qué personas serán responsables de las acciones pertinentes y se destinarán los recursos necesarios (personal, tiempo, dinero), evitando sorpresas desagradables.

En esta sección se detallan unas **buenas prácticas** aplicables al desarrollo de aplicaciones móviles accesibles, en la línea de lo comentado anteriormente, con las que los desarrolladores podrán saber **cómo actuar** en cada caso. Además, en otras secciones del presente documento se ofrecen contenidos complementarios, como las pautas para evaluar el nivel de accesibilidad (en sección “Validación de accesibilidad de aplicaciones móviles”) o las herramientas de evaluación útiles tanto en fase de desarrollo como en tiempo de ejecución (en apartado “Herramientas de evaluación de accesibilidad”).

5.1. - BUENAS PRÁCTICAS DE ACCESIBILIDAD APLICABLES AL DESARROLLO DE APPS

Desde la fase inicial del diseño de una aplicación móvil se deben tener en cuenta tanto unos **principios** básicos de **diseño** como unos **requisitos** de **desarrollo** para intentar alcanzar un resultado accesible.

Además, durante el desarrollo hay que considerar las **necesidades** de los **usuarios** con discapacidad y garantizar la compatibilidad con los **servicios de accesibilidad** provistos por los sistemas operativos (SSOO).

Para lograrlo, los SSOO proporcionan la **documentación técnica** necesaria a los desarrolladores, **elementos estándar** para la interfaz de usuario que incorporan información de accesibilidad y **herramientas** de desarrollo que facilitan que los elementos creados sean accesibles, recursos que presentan cambios reseñables entre plataformas debido a la gran fragmentación y diversidad que existe en el mercado de dispositivos móviles.

Todo ello ha sido contemplado a la hora de confeccionar la presente guía, dando lugar al conjunto de **buenas prácticas** de este apartado y siendo el punto de partida para otras secciones que se encuentran más adelante.

Además de los contenidos de las **guías de diseño** y desarrollo de los principales sistemas operativos para plataformas móviles (Android, iOS y Windows), para confeccionar las directrices facilitadas a continuación se han tenido en cuenta los criterios de conformidad de **WCAG 2.0**, así como la adaptación **WCAG2ICT**[4].

NOTA: Si tiene **dudas** específicas sobre el SO que haya tomado como plataforma base, debería complementar la lectura de las pautas de esta sección con la información provista en la/s **guía/s técnica/s** de accesibilidad de la documentación del **SO objetivo**.

5.1.1. Cuál es la responsabilidad del desarrollador

La primera cuestión que hay que plantearse a la hora de empezar a desarrollar una **aplicación móvil accesible** es hasta dónde llega la **responsabilidad** del **desarrollador**. Es fundamental para poder llevar a cabo una valoración objetiva sobre el resultado de su trabajo.

El desarrollador de una aplicación no debería incluir en su cometido la generación de software que permita el **acceso universal** a su producto. Este ámbito es competencia de la **plataforma** para la que se esté desarrollando la aplicación. Sin embargo, podemos encontrarnos con **sistemas operativos** que no ofrezcan las herramientas necesarias para asegurar la accesibilidad. Veremos cómo tratar estos casos en el siguiente apartado.

El desarrollador sí es responsable de que, utilizando las **herramientas de apoyo** que ponga a disposición la plataforma mediante la **capa de accesibilidad**, cualquier usuario pueda acceder a toda la **información** y **funcionalidad** proporcionada por su aplicación. En caso

[4] https://www.w3.org/TR/wcag2ict/

contrario, no podrá considerarse que la aplicación sea accesible. Veremos más acerca de la capa de accesibilidad más adelante.

5.1.2. Plataformas no accesibles

En ocasiones, podemos encontrar que una determinada plataforma móvil no ofrece los **servicios de accesibilidad** que requerimos para poder construir una **aplicación accesible**. Esta ausencia puede ser **total** o **parcial**. En cualquiera de los casos, no podremos considerar que la aplicación desarrollada sea accesible, aunque sea por causa ajena. En estas situaciones, podemos actuar de dos maneras diferentes.

La primera opción y, quizás la más obvia, es **no desarrollar** para dicha plataforma. Esta alternativa debería plantearse principalmente cuando la **ausencia** de **características** de accesibilidad sea tan **grande** que debamos desarrollar por completo una capa de accesibilidad por nosotros mismos.

Por otro lado, podemos **suplir** como desarrolladores las **carencias** del sistema operativo, creando e incluyendo dentro de nuestra aplicación componentes software que proporcionen **mejoras** en la **accesibilidad** de los dispositivos móviles y sus aplicaciones (lectores de pantalla, magnificadores, ampliadores de texto…). Esta opción es la más adecuada cuando las dificultades técnicas que nos ofrece la plataforma no son excesivas y podemos salvarlas invirtiendo un poco más de esfuerzo en la confección de **servicios de apoyo**.

Hay que tener en cuenta que, de elegir la segunda opción, debemos permanecer atentos a las **posibles actualizaciones** de la plataforma respecto a la accesibilidad. Una vez que ésta incluya las características que se han suplido como desarrollador, esta funcionalidad debería desaparecer de nuestra aplicación para **no interferir** con los servicios de accesibilidad nativos de la plataforma, dotando al usuario de una experiencia más consistente.

Dicho esto, hay que mencionar que las **plataformas** móviles más utilizadas (Android, iOS y Windows) poseen muchas **características** de accesibilidad por defecto, de modo que normalmente no se suele requerir un desarrollo específico de estos servicios y utilidades.

5.1.3. La capa de accesibilidad

La capa de accesibilidad es un concepto introducido en la construcción de las **interfaces de usuario** de los sistemas operativos para dotar a las herramientas técnicas de apoyo de la **información** suficiente para que éstas puedan ofrecer **acceso universal** a los distintos perfiles de usuario.

Se trata de una pieza **software**, incluida dentro de las Application Programming Interface (API) del sistema operativo, que permite definir ciertas **propiedades** especiales para los **componentes gráficos** y **métodos** para poder acceder mediante software a las mismas. Esta capa suele venir ya incluida en los componentes gráficos predeterminados que pone cada plataforma a nuestra disposición.

Aunque cada **fabricante** es libre de implementar esta capa de accesibilidad como crea conveniente, el objetivo de todas ellas es común: **permitir** el **acceso** de personas con discapacidad a las **funcionalidades** del sistema y facilitar a los desarrolladores las herramientas apropiadas para que sus aplicaciones puedan ser accesibles.

Por tanto, un **sistema operativo** que no cuente con capa de accesibilidad, no será considerado **accesible**, aunque los desarrolladores pueden incluir suficientes componentes software en sus aplicaciones como para suplir esta carencia. Por otra parte, aunque una plataforma disponga de capa de accesibilidad, no se convierte automáticamente en accesible. Esto dependerá de las **características** que sea capaz de cubrir esta capa.

Por último, hay que destacar que el hecho de disponer de una capa de accesibilidad es sólo una **herramienta** para dotar de **accesibilidad** a una aplicación. Es responsabilidad del **desarrollador** utilizar esta herramienta de la manera adecuada para que, efectivamente, el producto final sea accesible.

A continuación se referencian las capas de accesibilidad presentes en algunas de las principales **plataformas**:

- Android (https://developer.android.com/guide/topics/ui/accessibility/index.html)
- iOS (https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/iPhoneAccessibility/Accessibility_on_iPhone/Accessibility_on_iPhone.html)
- Windows (https://developer.microsoft.com/en-us/windows/accessible-apps)

5.1.4. Directrices para el desarrollo

En esta sección se van a exponer ciertas **directrices**, comunes a todos los **sistemas operativos móviles**, por medio de las cuales se puede proporcionar **accesibilidad** a nuestras aplicaciones. Las directrices se han agrupado de un modo coherente, que permite identificarlas de forma lógica por la relación entre conceptos.

Así, cada apartado brinda una breve **descripción** que permite conocer los aspectos relacionados con dicha categoría que hay que cumplir. Sin embargo, al tener un estilo **conceptual** e independiente de la plataforma, no se ofrecen las instrucciones específicas sobre cómo implementar **soluciones** particulares en cada caso. Para tener una información detallada a este respecto, se recomienda tanto la lectura y utilización de la capa de accesibilidad y documentación del **Sistema Operativo** pertinente como la revisión de los **ejemplos** ofrecidos al final de la presente guía (anexo II).

5.1.4.1. Propiedades de los componentes

Una de las características más típicas de la capa de accesibilidad para las distintas plataformas es la definición de ciertas **propiedades** que permiten el acceso a la información

del **componente gráfico** a las diferentes **tecnologías de apoyo**, como los lectores de pantalla.

El objetivo de estas propiedades es poder **determinar** de manera programática ciertas **características** de los componentes, como su nombre, la acción que desencadenan al ser activados, su estado o su valor, entre otras.

Cada **sistema operativo** utiliza una sintaxis propia para desempeñar esta labor y define ciertas propiedades. No obstante, conceptualmente es bastante similar en todos ellos.

Principales tipos de propiedades

El **nombre**, etiqueta o contenido textual es una propiedad presente en todas las capas de accesibilidad existentes. Permite definir el **texto** que se proporcionará a las herramientas de apoyo cuando el **foco** se sitúe en el elemento. Es usado sobre todo para proporcionar una **alternativa textual** a los elementos gráficos, como las imágenes.

El nombre **no** debe incluir el **tipo de elemento** que se está etiquetando, ya que esto puede conducir a la repetición de palabras como “botón” que las herramientas ya extraen de otras propiedades.

El nombre **no** debe **describir visualmente** el elemento al que representa, sino la **acción** asociada a dicho elemento. Por ejemplo, si contamos con un botón para enviar un correo electrónico y éste es representado por un sobre, el nombre debería ser “enviar” y en ningún caso “sobre”. El único motivo para describir visualmente un elemento sería aquel en el que se utilice un **elemento gráfico** para transmitir información en lugar de representar una acción. Es aconsejable que las **etiquetas** sean **cortas** y concisas.

El **rol** o rasgo de un elemento hace referencia al **tipo de componente** gráfico que se está representando. Aquí se define si el elemento debería ser transmitido como un botón, una lista, un enlace o cualquier otra clase de vista de la interfaz.

El **estado** de un elemento indica en qué **situación** se encuentra. Podemos pensar en casillas de verificación marcadas o no, pero también en botones inhabilitados o en componentes que puedan presentar dos o más estados diferentes. Nótese que, en cualquier caso, **no** es suficiente con **transmitir** esta información de **manera visual**, cambiando el color del elemento, por ejemplo.

El **valor** de un elemento es una propiedad que suele ser más útil en **componentes interactivos de edición**. Esto es debido a que en un texto plano o un botón, el valor coincidirá con el nombre o etiqueta. En cambio, en un **campo** de **formulario** el valor hace referencia al contenido del campo, en lugar de a la etiqueta que indica a qué hace referencia ese campo.

También pueden proporcionarse **descripciones** más extensas para algunos objetos que serán anunciadas si el foco se mantiene en ellos. Esta descripción es **opcional** y sólo debería proporcionarse si la **función** del elemento todavía es ambigua mediante el uso de un nombre o etiqueta. En este caso, una buena convención es utilizar la tercera persona del singular, ya que se está describiendo el objeto.

Si la plataforma lo permite, es necesario **asociar** cada **campo** de un formulario con el **elemento gráfico** correspondiente que lo describa visualmente. En este caso, el nombre o etiqueta del campo serían sustituidos por el texto que contenga dicha etiqueta visual, y no sería necesario proporcionar esta propiedad al campo en sí.

Acceso mediante lenguaje de interfaz

Una de las formas de acceder a estas propiedades suele ser desde el propio **lenguaje** que se utiliza para **definir** la **interfaz gráfica** de las aplicaciones. En la mayoría de los casos, este lenguaje consta de una estructura XML en la que se van declarando los distintos **elementos** con los **atributos** que los definen. Entre estos atributos, pueden encontrarse aquellos que corresponden a las propiedades de la capa de accesibilidad. Cuando sea posible, se recomienda utilizar este método para proporcionar la información necesaria.

Acceso mediante programación

En algunas ocasiones, debemos **cargar** en la **interfaz** componentes de **manera dinámica**. En estos casos, no podemos recurrir al lenguaje de la interfaz para definir sus atributos y, por tanto, tampoco las propiedades de accesibilidad. En estos casos, las plataformas proporcionan el **acceso programático** a estos **atributos** mediante la capa de accesibilidad, tal y como haríamos para determinar el tamaño, color o cualquier otra característica del elemento.

Componentes estándar del sistema

Los **fabricantes** de las distintas plataformas son conscientes de la importancia de la accesibilidad y por eso integran esta capa en los **componentes gráficos predefinidos** dentro de las API de sus sistemas.

En la mayoría de los casos, si utilizamos los componentes que los sistemas operativos ponen a nuestra disposición como desarrolladores, el **trabajo** que deberemos hacer para dotar de accesibilidad a nuestra aplicación será **reducido**. Así, nuestra intervención sólo será necesaria cuando estos elementos no puedan **inferir** las **propiedades** de accesibilidad (atributos) directamente de su contenido, como en aquellos casos en que usemos imágenes o iconos.

Considerando esto, es recomendable **desarrollar** las **aplicaciones móviles** usando **componentes estándar**, ya que la interfaz de usuario será compatible con los servicios de accesibilidad y con los productos de apoyo, garantizando cierto nivel de **accesibilidad** en el resultado final, sin que ello suponga un gran esfuerzo adicional a la hora de desarrollar.

Fundamentalmente, el proceso de desarrollo con componentes estándar implica los siguientes **pasos**:

1. - Añadir **texto descriptivo** a los controles de la interfaz de usuario de la aplicación, utilizando el atributo apropiado en cada caso.

2. - Comprobar que se puede llegar a todos los **elementos** de la interfaz de usuario que pueden aceptar una **interacción** (tocar o escribir) con un controlador **direccional** (ratón de bola; D-pad físico o virtual; navegación por gestos).

3. - Comprobar que los mensajes de **audio** están siempre acompañados por una **alternativa visual** o **háptica**, permitiendo su utilización a los usuarios con problemas de audición.

4. - Comprobar el correcto **funcionamiento** de la aplicación utilizando únicamente los servicios y características de **navegación** accesibles.

Componentes personalizados

No siempre podemos alcanzar toda la funcionalidad que requiere nuestra aplicación mediante el uso de los **componentes gráficos** predefinidos en el sistema. De ser así debemos recurrir a crear los nuestros **propios**, aprovechando las APIs que ya pone éste a nuestra disposición.

En tal caso, será necesario **proporcionar** la mayor parte de la **información** de los **atributos** de los componentes de la interfaz en el proceso de desarrollo, entre otras cosas. Además de definir el tamaño, la forma o el comportamiento de estos elementos de manera programática, la capa de accesibilidad nos permite también transmitir la información que queramos a las herramientas de apoyo. Por tanto, aunque el **trabajo** será algo más **arduo**, también podemos desarrollar aplicaciones accesibles en estos casos.

Normalmente, las plataformas proporcionan la capacidad para determinar si un componente gráfico ha de ser considerado como un **contenedor** o como un elemento individual. De esta manera, podemos construir nuestras interfaces de forma que los usuarios que utilicen **tecnologías de asistencia** puedan interactuar con ellas.

Es importante saber que, al generar una vista o elemento personalizado que contiene otros **elementos** (contenedor) con los que el usuario debe interactuar, hay que hacer que éstos sean **accesibles individualmente** y que el contenedor no proporcione información de accesibilidad, ya que el usuario interactuará con los elementos individuales de su interior.

A la hora de crear **componentes personalizados** para la interfaz de la aplicación, que muestren información en pantalla o que ofrezcan interacción a los usuarios, es necesario:

1. - Garantizar su accesibilidad añadiendo la **información** apropiada a los **atributos** correspondientes.

2. - **Comprobar** que proporcionan la información de accesibilidad necesaria para que los servicios de accesibilidad y los **productos de apoyo** funcionen correctamente.

5.1.4.2. Navegación

La **navegación** es una de las partes principales a tener en cuenta de una aplicación móvil. Es importante que el usuario tenga claro en qué **vista** de la aplicación se encuentra en cada momento y cómo puede **desplazarse** a la vista que desea alcanzar.

Del mismo modo, es importante que la **disposición visual** de los elementos sea **consistente** y tenga sentido, resaltando los elementos más importantes para trabajar con la aplicación, teniendo que desplazarnos haciendo **scroll** lo menos posible para completar cada acción.

Cuando hablamos de accesibilidad, hay que tener en cuenta también ciertos aspectos inherentes a la forma que tiene una persona con discapacidad para **interactuar** con nuestra aplicación. Puede que la **discapacidad** que presente sea motriz o puede que sea incapaz de ver dónde están situados los componentes gráficos de la interfaz. En cualquier caso, el modo de **navegación difiere** sustancialmente del habitual.

El foco del sistema

El foco del sistema es un **elemento virtual** que podemos asimilar como un **puntero** hacia un componente gráfico. En palabras simples, indica qué elemento de la pantalla está enfocado. Gracias a esto, los usuarios pueden recibir la **información** del elemento a través de las **propiedades** de la capa de accesibilidad o interactuar con él.

Mientras que un usuario común desplaza este **foco** directamente mediante pulsaciones en la pantalla táctil o mediante el desplazamiento y posterior clic de un cursor, ciertos **perfiles** de discapacidad utilizan un estilo de **navegación secuencial**. En estos casos, el foco va **saltando** de elemento en elemento mediante algún comando, hasta que se alcanza aquel que se desea activar.

Visibilidad del foco

Una de las características importantes para que una **aplicación** sea **accesible** es la visibilidad del foco. Aunque a menudo se piensa en usuarios invidentes como el principal objetivo de estas medidas, hay que tener en cuenta que también hay **perfiles** motrices que no pueden interactuar correctamente con la pantalla. A éstos les resulta útil que el **foco** sea **visible** en su desplazamiento a lo largo de la **interfaz**.

Aunque esta funcionalidad es más propia de la **capa de accesibilidad** de la plataforma, si ésta no la ofrece, el desarrollador deberá proporcionarla de alguna forma.

Elementos receptores del foco

Una cuestión importante a tener en cuenta es qué elementos deben recibir el foco. La respuesta es que sólo deben recibir el foco aquellos **elementos** que permitan realizar una cierta **funcionalidad** o que sean imprescindibles para **comprender** totalmente la **información** que se muestra en la vista.

De este modo, tanto **elementos interactivos** como **bloques de texto** plano son potenciales receptores del foco, mientras que elementos decorativos como marcos no deberían recibirlo en ningún caso, pues sólo confundirían al usuario.

Orden de navegación

El orden en que el **foco** recorre los diferentes elementos de nuestra interfaz gráfica es muy importante. El foco no se **desplaza** teniendo en cuenta la disposición visual, sino que lo hace teniendo en cuenta el **orden** en que los **elementos** fueron incorporados a ésta, siguiendo el esquema de árbol, bien sea mediante el lenguaje de la interfaz o programáticamente.

Sin embargo, no estamos obligados a incluir los elementos en este orden, ya que esto podría afectar a la **experiencia** visual de la interfaz. La capa de accesibilidad pone a nuestra disposición herramientas que nos permiten determinar el **orden** en el que los elementos serán enfocados, si algunos serán ignorados o si ni siquiera serán accesibles mediante la capa de accesibilidad, lo que puede ser útil a la hora de **eliminar información** innecesaria puramente **decorativa**.

Cambios de contexto

Un cambio de contexto es todo aquel **evento** que desencadena un **cambio** en el **lienzo** de la **aplicación**, ya sea total o parcialmente. La entrada o salida de componentes gráficos, cambios en su contenido o el cambio a otra vista son cambios de contexto.

Como norma general, todo cambio de contexto ha de ser **notificado** al usuario a través de la capa de accesibilidad en caso de que se produzca de **manera automática**.

En ningún caso se debería recargar el lienzo de la página automáticamente sin una acción explícita del usuario. Cuando esto se produce entre vistas diferentes, puede **causar confusión** al no saber qué es lo que ha pasado exactamente, si ha cometido algún error o ha pulsado algo que no debía. Cuando se produce con vistas similares, por ejemplo, para adaptar los campos de un formulario en función de una de las opciones expuestas, el **foco** del sistema **se pierde** y se ha de recorrer de nuevo el camino ya realizado.

Pasos de navegación

Una característica importante ya no sólo a nivel de accesibilidad, sino de usabilidad, son los **pasos** que el usuario tiene que llevar a cabo para **alcanzar** la **funcionalidad** que desea. Este número de pasos debería ser el **menor posible**. De otro modo, el usuario puede sentirse desorientado y ralentizado en sus tareas. Como regla general, deberíamos poder llegar a cualquier lugar de nuestra aplicación en **3 pasos**, aunque no ha de tomarse como algo restrictivo si la inclusión de algún paso adicional contribuye a clarificar la información o las acciones.

Navegación por procesos

Un caso especial son aquellas **tareas** que requieren una **navegación** a lo largo de un **proceso**. Podemos englobar en esta categoría asistentes de configuración, cuestionarios con una cantidad más o menos grande de cuestiones… En estos casos, la **navegación** es **secuencial** y el proceso no se termina hasta que se alcanza la **última vista**.

En estas situaciones, es importante incluir en cada vista el **progreso** que el usuario ha **completado** ya, sea en forma porcentual o de número de pasos.

De igual modo, es imprescindible que el usuario pueda **navegar** adelante y atrás para **corregir** cualquier dato introducido con anterioridad.

5.1.4.3. Diseño de la interfaz

El diseño de la **interfaz gráfica** es un punto importante para la accesibilidad. Es vital tanto para personas con algún tipo de visión reducida como para aquellas con discapacidades cognitivas o motrices. Un **diseño limpio**, **claro** y **consistente** puede ayudar a estas personas a utilizar la aplicación.

Tamaño

En primer lugar, debemos tener en cuenta el **área interactiva** de los **elementos**, dado que estamos trabajando con dispositivos hápticos. Superficies muy reducidas pueden conducir a **errores** o la imposibilidad directa para acceder a ciertas funcionalidades. El área útil de interacción debe ser de aproximadamente **9 mm** de lado, pudiendo ser mayor si se desea. En esta superficie hay que tener en cuenta también el **área de relleno** que rodea al elemento, que si bien no pertenece al mismo desde el aspecto visual, sí que sirve como **zona de interacción** con el mismo.

Del mismo modo, la **separación** entre las distintas áreas interactivas debería ser de al menos **1 mm** para evitar la activación involuntaria de funciones.

Hay que tener en cuenta además que los elementos deberán poseer un **tamaño** suficiente para albergar en ellos el **texto** internacionalizado. Aunque en un idioma el texto pueda caber en el contenedor, es posible que en otro se alargue y no quepa.

Respecto al **tamaño del texto**, hay que dar la posibilidad de **agrandarlo** en caso de que el usuario lo necesite. Esta funcionalidad suele venir incorporada por la propia capa de accesibilidad de la plataforma en cuestión. La misión del desarrollador en este punto es no interferir con ella y utilizar las **herramientas** que el sistema pone a su disposición. En caso de que no esté disponible esta funcionalidad de manera nativa, se deberá implementar por parte del desarrollador.

Contraste

El **contraste** del **texto**, ya sea plano o embebido en una imagen, debe tener una **relación mínima** para que los **usuarios** con algún tipo de dificultad visual puedan **leerlo** con claridad. Este contraste es dependiente también del tamaño del texto en cuestión. Podemos obtener esta relación de contraste utilizando algunas de las herramientas que se verán más adelante.

Para el **texto pequeño** (inferior a 18 pt con texto normal o a 14 pt con texto en negrita), el contraste deberá situarse en una relación mínima de **4.5:1**.

Para el **texto grande** (18 pt con texto normal o 14 pt con texto en negrita), el contraste deberá situarse en una relación mínima de **3:1**.

Además, algunas plataformas proporcionan la funcionalidad de **alto contraste** a través de su capa de accesibilidad. Se deberá velar para **no interferir** con esta capacidad en función del sistema operativo en cada caso.

Destellos

Los **destellos** o parpadeos en la pantalla pueden producir **convulsiones** o ataques en usuarios sensibles a esta problemática. Para evitarlo, se ha de procurar que ningún elemento destelle o parpadee más de **3 veces por segundo**. A esto se lo conoce como la regla de los 3 destellos.

Además, es aconsejable **evitar** que elementos que ocupen **regiones grandes** o **centrales** de la pantalla destellen o parpadeen.

Lenguaje

El lenguaje es importante especialmente para aquellos usuarios con discapacidad cognitiva. Es el principal medio de **transmisión** de la **información** y, por tanto, el responsable de la interacción entre el usuario y la aplicación.

Es importante que la aplicación esté convenientemente localizada para los distintos **idiomas** de los **usuarios** objetivo.

Los **textos** deberán ser lo más **claros** y **concisos** que sea posible. Las fórmulas rebuscadas pueden causar confusión en el usuario o que éste no sepa identificar adecuadamente la función que está consultando.

Además, todos los textos deben ser **correctos** ortográfica y sintácticamente, haciendo uso de un tipo de **fuente** no problemática para casos de dislexia.

Iconos

Los iconos son una forma visualmente atractiva de representar la **función** de un **elemento** de la interfaz. Sin embargo, no siempre son perfectamente **reconocibles**. Algunos perfiles con dificultades cognitivas o visuales pueden encontrarlos confusos.

Salvo algunos iconos universalmente reconocidos, como pueden ser los típicos de un reproductor multimedia (play, stop, etc.), se debería proporcionar una **alternativa textual** a los **iconos** que utilicemos en nuestra aplicación. El usuario debería poder elegir si desea utilizar los iconos como representación de los elementos de la interfaz o prefiere el texto plano.

En algunos casos puede ser **difícil** sustituir un icono por un texto, ya que es posible que éste no quepa en el **espacio** dispuesto. Es posible utilizar **alternativas** como mostrar tooltips al mantener pulsado el elemento, de forma que el usuario pueda saber a ciencia cierta la función que aporta el componente.

Consistencia

La **consistencia** de la **apariencia** entre las distintas vistas de la **aplicación** puede ayudar a que los usuarios naveguen de forma más cómoda y **eficiente** por ella. No sólo eso, sino que esto ayuda además a que aquellos usuarios con dificultades cognitivas sean capaces de asimilar mejor la información.

No existe una regla clara al respecto. Simplemente, procura que las **vistas** de la aplicación sean **consistentes** entre sí. Si tiene una barra de herramientas, procura que ésta aparezca siempre en la misma zona de la pantalla. Considera que los **colores** elegidos sean **homogéneos** en todo el diseño y que la fuente tipográfica también lo sea.

Además, es una buena práctica en este caso seguir las **guías de estilo** de cada plataforma. De esta manera, la aplicación presentará un aspecto homogéneo respecto al resto del **sistema operativo** y la curva de aprendizaje será mínima.

5.1.4.4. Ayuda de entrada

La **entrada** de **datos** por parte del usuario es una parte fundamental en muchas aplicaciones. Desde el **formulario** más básico al más complejo, podemos identificar ciertas acciones que repetimos constantemente en nuestro día a día.

No obstante, ciertos **perfiles** de usuarios pueden sufrir algunas **dificultades**. Éstas van desde tener un ritmo más lento de escritura hasta la imposibilidad total de introducir textos mediante el teclado o la pantalla táctil.

Podemos aprovechar las **herramientas** de las distintas plataformas para suplir estas discapacidades y permitir así el **acceso universal** a nuestra aplicación, tales como el dictado por voz.

Autocompletado

El autocompletado es una de las funciones más importantes con respecto a la ayuda de entrada. Permite a los usuarios **escribir** una mayor cantidad de **texto** sin tanto **esfuerzo**. El ejemplo típico es el corrector de los diferentes teclados que podemos encontrar en los dispositivos, que trata de **predecir** las **palabras** que queremos escribir. Por tanto, podríamos pensar que esta funcionalidad no nos corresponde tanto a nosotros como desarrolladores de la aplicación.

No obstante, hay que decir que el autocompletado también es **útil** cuando se trata de rellenar **campos** con un **contenido** muy **determinado** y que podemos tener ya almacenado. Entrarían en esta categoría e-mail, número de teléfono, claves alfanuméricas como los documentos de identidad… Siempre que se pueda, se debería proporcionar esta ayuda a los usuarios para facilitarles rellenar los formularios.

Portapapeles

La función de cortado, copiado y pegado de texto es muy útil y ampliamente utilizada. **Reduce** considerablemente el **esfuerzo** que se tiene que realizar a la hora de rellenar ciertos **formularios** o de trabajar con **documentos**.

Los **sistemas operativos** ya integran esta **funcionalidad** de portapapeles. La única responsabilidad que debe asumir el desarrollador en este punto es **no interferir** con las opciones del sistema dentro de los componentes gráficos que diseñe, de forma que los usuarios puedan aplicar dichas funciones con libertad.

Entrada por voz

Para las personas que les resulta muy difícil la **introducción** de **texto** mediante un teclado, ya sea físico o virtual, esta función es vital. Como en casos anteriores, esta capacidad viene dada por el **sistema operativo** y la única responsabilidad del desarrollador es **no interferir** con ella.

Restricciones correctas

Las diferentes plataformas permiten **restringir** de una u otra forma los **caracteres** que pueden introducirse en un determinado **campo** de formulario. Es importante que estas restricciones sean correctas y lo más refinadas posible. Esto ayudará a la detección de **errores** y también restringirá los caracteres que el usuario puede introducir, de modo que su inserción será más **intuitiva**.

Mensajes de error concretos

Es importante que los **mensajes** de **error** que reciba el usuario cuando la entrada no sea correcta sean **precisos**. No es lo mismo decir que hay un error en un formulario que concretar en qué **campo** se ha producido el problema. Del mismo modo, es sustancialmente diferente indicar sólo que un campo es incorrecto que especificar cuál es el **problema** en concreto.

Para poder hacer esto, es importante **restringir** convenientemente la **entrada** y realizar un **análisis** correcto de la **información** introducida, de forma que seamos capaces de discriminar cuál ha sido exactamente el punto en el que el usuario ha errado.

5.1.4.5. Adaptabilidad temporal

En ciertas ocasiones podemos presentar **contenidos** o **funciones temporalmente dependientes**. En esta clasificación se engloba cualquier elemento que desaparezca pasado un cierto tiempo o alguna funcionalidad que solicite una acción del usuario antes de que transcurra un determinado lapso.

Este tipo de **interacción** puede ser **inaccesible** para ciertos **perfiles**, ya que algunos usuarios pueden no encontrar el componente que deben activar a tiempo, les pase desapercibido el contenido volátil antes de poder leerlo o requieran de más tiempo para poder llevar la acción solicitada a cabo. En cualquiera de los casos, esto se convierte en una **barrera** difícilmente salvable.

Siempre que incluyamos comportamientos dependientes del **tiempo** en nuestra aplicación debemos considerar que el plazo determinado sea **suficiente** para completar la tarea. Además, debemos dar la opción de **ampliar** este **plazo** manualmente para aquellos usuarios que necesiten más tiempo, hasta un límite de 24 horas.

Además, es recomendable **no incluir información** crítica en **mensajes volátiles** si ésta no puede ser consultada por ningún otro medio. Este tipo de contenido debe ser tomado como apoyo en dichos casos y no como una plataforma principal para la comunicación con el usuario.

Por supuesto, los **controles principales** de la aplicación **nunca** deberían ser **ocultados** automáticamente transcurrido un cierto tiempo. Además, se debe dar la posibilidad de **restaurar** fácilmente aquellos **controles** secundarios que se hayan ocultado.

5.1.4.6. Alternativas de entrada

Un dispositivo móvil cuenta con una amplia gama de **posibilidades** a la hora de poder **interactuar** con él. La más extendida es la **pantalla táctil**, pero éste puede disponer también de **botones**, **teclados** u otros dispositivos físicos. Además, cuenta con una gran variedad de **sensores**, tales como acelerómetros, giroscopios, sensores de luz… Todos ellos están a disposición del desarrollador para permitir la interacción con el usuario.

Sin embargo, no todos los **métodos de entrada** son aptos para cualquier **perfil** de usuario. Por ejemplo, será complicado que una persona con problemas motrices pueda agitar el terminal o realizar un gesto complejo sobre la pantalla.

Éste es el motivo por el que es necesario ofrecer varias **alternativas** de **entrada** para nuestra aplicación, procurando **cubrir** las **posibilidades** a las que nuestros usuarios puedan necesitar hacer frente.

No interferir con los métodos de entrada predeterminados

Cada **sistema operativo** cuenta ya con una serie de **atajos** o gestos de entrada que permiten realizar ciertas **funciones** de forma rápida y sencilla. Éstas pueden ir desde desplazar el foco del sistema, desplazarse por una lista o cambiar de página, hasta acceder a la bandeja de notificaciones o los ajustes.

Cuando establezcamos una determinada manera de interactuar con nuestra aplicación, debemos tener en cuenta el comportamiento predeterminado del resto del sistema. En especial, debemos tener cuidado con **no reutilizar gestos** o atajos que ya estén **asignados** para la realización de **otras acciones**.

Indicaciones no dependientes de la entrada

Dado que los usuarios pueden utilizar **diversos métodos** de **entrada** para interactuar con el dispositivo, no podemos establecer uno como método preferente. Por ello, se deben evitar indicaciones explícitas a un dispositivo de entrada en concreto, como “pulsa enter” o “toca dos veces la pantalla”. En lugar de eso, se ha de optar por **indicaciones** más **abstractas** como “activar el elemento”.

Utilizable por cualquier método de entrada soportado por el sistema

La aplicación no debe ser dependiente de un determinado método de entrada a menos que sea absolutamente imprescindible. Debe **soportar** la **navegación** mediante cualquiera que sea el **dispositivo** empleado, siempre que esté soportado por la plataforma móvil.

Podemos considerar **excepciones** aquellos casos en los que sea indispensable que la **entrada** se realice de una cierta forma. Por ejemplo, si la funcionalidad de la aplicación se basa en medir el desplazamiento del usuario mediante el acelerómetro y sólo puede

cumplir con su cometido de esa manera, no hay ninguna otra forma de lograr el mismo resultado.

Alternativa para gestos complejos

Como ya se ha mencionado, los usuarios con ciertos **perfiles** no pueden realizar algunos **gestos complejos**, como agitar el terminal, deslizar los dedos por la pantalla… Siempre se ha de poder ejecutar la misma acción mediante una **alternativa** que no requiera de estos gestos.

Alternativa para dobles pulsaciones

Pulsar dos veces seguidas un elemento para efectuar una determinada acción es algo común, pero puede suponer una **barrera** para ciertos **usuarios** que no sean capaces de realizar las pulsaciones en el tiempo preciso. Ampliar este tiempo tampoco es una opción válida, ya que eso puede interferir con otras acciones y entorpecer la experiencia de navegación. Por tanto, se ha de ofrecer una **alternativa** para realizar dichas acciones que no requiera de una **doble pulsación**.

Alternativa para pulsaciones mantenidas

Igual que en el caso anterior, una **pulsación mantenida** puede no ser apta para ciertos perfiles de usuarios con discapacidad. Es necesario ofrecer una **alternativa** que no requiera de una pulsación mantenida para realizar la misma acción.

Alternativa para combinaciones de teclas

Algunos usuarios son incapaces de pulsar más de una tecla a la vez en un teclado. Para cubrir este propósito, los **sistemas operativos** cuentan con opciones para poder realizar las **combinaciones** de **teclado** de manera secuencial.

Si el sistema operativo no cuenta con esta posibilidad, deberíamos **evitar** utilizar combinaciones de teclas para realizar cualquier acción.

Asistentes de voz

Los asistentes de voz son **herramientas** poderosas basadas en la inteligencia artificial. Su modo de funcionamiento básico consiste en el procesamiento del **lenguaje natural** o de **comandos** para realizar diferentes **acciones** o consultas en el sistema.

La mayoría de estos asistentes disponen de API para extender su funcionalidad e integrarlos con nuestras aplicaciones. Utilizar estos asistentes como **método de interacción** puede suponer también una **mejora** de **accesibilidad** bastante grande.

5.1.4.7. Alternativas de salida

Del mismo modo que los dispositivos móviles cuentan con una gran variedad de canales de entrada, la salida es igualmente diversa. El principal **canal** del que disponemos es el **visual**, pero también contamos con el **auditivo** y hasta el **háptico**.

Sin embargo, algunos **perfiles** de discapacidad no tienen acceso a ciertos canales. Si queremos que nuestra aplicación sea accesible, debemos hacer que la **información** sea **redundante** en más de un canal simultáneamente.

Subtítulos

Para los perfiles con discapacidad auditiva es imprescindible incluir **subtítulos** en cualquier **contenido** que utilice el canal **sonoro** para transmitir información relevante. Los subtítulos deben estar **sincronizados** con dicho contenido.

Además, es importante procurar que los subtítulos sean lo más **legibles** posible. Por ello, en lugar de incluirlos dentro de la imagen del vídeo, por ejemplo, es mejor colocarlos en un espacio aparte, como una banda negra bajo éste. Además, el texto debe tener un **contraste suficiente**.

Audiodescripción

Para usuarios con discapacidad visual severa es importante recibir de alguna manera la **información** que se transmite de manera **visual** en el contenido, como puede ser en los vídeos, por ejemplo. En estos casos, se debe incluir una pista de **audiodescripción** junto al contenido multimedia que **describa** los **sucesos** relevantes que ocurren en éste y que sólo pueden ser comprendidos mediante el canal visual. Esta pista deberá estar **sincronizada** con el contenido.

Es importante hacer constar que la **información** suministrada mediante audiodescripción debe ser suficiente para **comprender** lo que esté sucediendo en el vídeo, pero **no** tanta como para **abrumar** o confundir al usuario. En este sentido, se recomienda utilizar frases cortas y concisas, que describan bien la acción, pero sin entrar en detalles minuciosos que no aporten demasiado a la comprensión.

Uso del color

En muchas ocasiones se utiliza el **color** como único medio para **representar** cualquier tipo de **meta-información**. Por ejemplo, si un texto está en rojo, se asocia a que se trata de la salida correspondiente a un error, mientras que si es amarillo se pretende indicar alguna clase de alerta no crítica.

No obstante, no es suficiente con utilizar este **sistema cromático** para transmitir dicha meta-información. Algunos **perfiles** no pueden diferenciar en absoluto los colores usados, mientras que otros pueden confundirlos o, incluso, **no** llegar a **comprender** en absoluto su significado.

Por eso, además de usar el código cromático como medio de transmisión de información, se debería utilizar una **alternativa** que pudiera ser claramente identificada por todos estos perfiles. Ésta puede ser una pequeña **etiqueta**, un **icono** o similar, siempre correctamente dispuesto para hacerlo accesible.

Alternativa visual

En ocasiones transmitimos información al usuario mediante una **señal visual**, como un destello, el cambio de color de un elemento… No obstante, esto no es suficiente para **notificar** a aquellos usuarios incapaces de apreciar el **cambio** visualmente, por lo que se hace necesario utilizar un **canal alternativo** para hacer llegar esta información.

Alternativa auditiva

Igual que en el caso anterior, se pueden **notificar** ciertos cambios o **eventos** al usuario mediante el **canal auditivo**, algo bastante común en las aplicaciones modernas. No obstante, existen perfiles para los que este medio es inviable, de modo que se hace necesario utilizar un **canal alternativo** a través del que se transmita la misma información.

Alternativa háptica

Este caso es un tanto diferente a los dos anteriores. Si bien podríamos considerar la existencia de **perfiles** de usuario incapaces de detectar este tipo de notificaciones, como las **vibraciones**, también debemos contemplar la existencia de **dispositivos** que no cuenten con dicha funcionalidad.

Podemos pensar que, hoy en día, todos los terminales móviles tienen vibración incorporada. No obstante, ciertos dispositivos como las **tablets** o los **convertibles** pueden no disponer de esta característica. Estos terminales tienden a utilizar las **mismas plataformas** que sus homólogos de telefonía, de modo que las aplicaciones deben estar preparadas para funcionar en ambos medios, ofreciendo las notificaciones pertinentes por **canales alternativos**.

5.1.4.8. Elementos molestos innecesarios

Como desarrolladores no estamos limitados a la **funcionalidad esencial** de una **aplicación**, sino que podemos **decorarla** como mejor creamos oportuno e incluso introducir **elementos** que pueden ser **ajenos** a la misma. Ejemplos de ello son animaciones, anuncios o sonidos de fondo, como un hilo musical.

Algunos de estos **elementos** pueden ser un tanto **intrusivos** y, hasta cierto punto, suponer una barrera de accesibilidad. Pensemos por un momento en la situación que se produce cuando un usuario de lector de pantalla accede a una aplicación en la que suena una música de fondo a un volumen elevado, que opaca la voz del sintetizador. En este contexto, el usuario tendría serias **dificultades** para **interaccionar** con la aplicación, si no directamente imposibilidad para hacerlo.

De la misma forma, un contenido que se desplaza de un lado a otro, aparece y desaparece, o cualquier otro efecto dinámico, puede poner en serios aprietos a algunos **perfiles** de usuarios. Puede que no les **impida** completamente el **acceso**, pero **dificulta** la interacción, puede confundirles y, cuanto menos, estorbarles.

Se debe procurar **reducir** al mínimo todo este tipo de **elementos** potencialmente **molestos**. En caso de que se desee introducirlos o su aparición sea parte de la experiencia de la

aplicación, se debe dar la **opción** de **pausarlos**, **detenerlos** u **ocultarlos** de una forma sencilla, que no requiera de una navegación intrincada a lo largo de la aplicación.

5.1.4.9. Documentación y ayuda

En muchas ocasiones se tiende a pensar que la **aplicación** es únicamente el **software** que se ejecuta en la plataforma correspondiente. Sin embargo, dentro de este concepto se engloba también cualquier tipo de **documentación** de ayuda o **soporte al usuario** que se pueda proporcionar.

Así, es importante que la documentación de ayuda sea accesible. En concreto, si alguna **funcionalidad** de la aplicación presenta **particularidades** para permitir el acceso universal, éstas deberían estar recogidas en la **ayuda de la aplicación** con el proceso detallado.

Del mismo modo, cualquier sistema de **atención al usuario**, ya sea online, telefónico o presencial, debería ser **accesible**.

Además, es recomendable hacer estudios sobre la **satisfacción** de los usuarios con la aplicación, fijándose especialmente en los **perfiles** con discapacidad, y disponer de un canal de **comunicación eficaz** para la notificación de fallos de accesibilidad en nuestra aplicación, con tal de resolverlos lo antes posible.

5.2. CHECKLIST DE COMPROBACIÓN DE PAUTAS DE DESARROLLO

La siguiente tabla es una **lista de comprobación** del cumplimiento de las **directrices** o pautas de desarrollo indicadas en el apartado anterior. A través de ella el desarrollador puede hacer un **chequeo rápido** para saber hasta qué punto se han tenido en cuenta dichas pautas.

El modo de empleo de la tabla es muy sencillo. Para **cada pauta**, debe indicarse si se **cumple**, no se cumple o no es aplicable e incluir un **comentario** explicativo claro. Este recurso se presta a un **uso iterativo**, de modo que sería posible llevar a cabo múltiples chequeos y correcciones de manera reiterada, con lo que es posible completar una **mejora continua** sobre la aplicación en desarrollo.

Pauta	Cumplimiento (S/N/NA)	Comentarios
➢ **Propiedades de los componentes**		
• **Principales tipos de propiedades**		
• **Acceso mediante lenguaje de interfaz**		
• **Acceso mediante programación**		
• **Componentes estándar del sistema**		
• **Componentes personalizados**		

Pauta	Cumplimiento (S/N/NA)	Comentarios
➢ **Navegación**		
• **El foco del sistema**		
o **Visibilidad del foco**		
o **Elementos receptores del foco**		
o **Orden de navegación**		
• **Cambios de contexto**		
• **Pasos de navegación**		
• **Navegación por procesos**		
➢ **Diseño de la interfaz**		
• **Tamaño**		
• **Contraste**		
• **Destellos**		
• **Lenguaje**		
• **Iconos**		
• **Consistencia**		
➢ **Ayuda de entrada**		
• **Autocompletado**		
• **Portapapeles**		
• **Entrada por voz**		
• **Restricciones correctas**		
• **Mensajes de error concretos**		
➢ **Adaptabilidad temporal**		
➢ **Alternativas de entrada**		
• **No interferir con los métodos de entrada predeterminados**		
• **Indicaciones no dependientes de la entrada**		
• **Utilizable por cualquier método de entrada soportado por el sistema**		
• **Alternativa para gestos complejos**		
• **Alternativa para dobles pulsaciones**		
• **Alternativa para pulsaciones mantenidas**		
• **Alternativa para combinaciones de teclas**		
• **Asistentes de voz**		

Pauta	Cumplimiento (S/N/NA)	Comentarios
➢ **Alternativas de salida**		
• **Subtítulos**		
• **Audiodescripción**		
• **Uso del color**		
• **Alternativa visual**		
• **Alternativa auditiva**		
• **Alternativa háptica**		
➢ **Elementos molestos innecesarios**		
➢ **Documentación y ayuda**		

5.3. CONTENIDOS DE TERCERAS PARTES

En ocasiones las aplicaciones móviles albergan o dependen de contenidos, servicios o tecnologías de terceras partes (mapas incrustados, documentos en PDF, etc.) que escapan al **control del desarrollador** desde un punto de vista creativo, por lo que son utilizados dentro de la aplicación móvil, pero no son confeccionados por el personal como sí sucede con el resto de componentes.

Para que la aplicación móvil desarrollada sea accesible, **todos sus elementos** deben satisfacer los **requisitos** oportunos, incluyendo aquellos que sean de terceras partes. Por este motivo, es importante someter a esos componentes a las mismas comprobaciones que los demás siempre que se pueda, realizando un **seguimiento continuo** por si se producen cambios reseñables en sus características y **subsanando** en la medida de lo posible cualquier desviación evidenciada. Para corregirlos, dado que no se cuenta con control suficiente sobre los elementos de terceras partes, es posible que haya que elegir **alternativas accesibles** para los mismos o, en caso de que no sea viable, poner los fallos de accesibilidad revelados en conocimiento de los propietarios de la tecnología externa en cuestión e indicar claramente en la aplicación móvil que ese componente no es accesible.

Considerando la imposibilidad de actuar directamente sobre los contenidos de terceras partes, se recomienda **evitar** la incorporación de dichos recursos en las aplicaciones móviles desarrolladas. En caso de que no haya más remedio que incluirlos, se debe **verificar** previamente su nivel de accesibilidad antes de integrarlos en la aplicación.

5.4. MONITORIZACIÓN

Cuando se genera una aplicación móvil debe evaluarse la idoneidad de todos sus contenidos, no sólo a nivel funcional, sino también al respecto del nivel de **accesibilidad** de los mismos. Sin embargo, además de estas comprobaciones iniciales, cuando se efectúan **cambios** importantes sobre la aplicación, incorporando nuevos contenidos o acometiendo modificaciones de peso sobre cualquiera de sus componentes, deben ejecutarse **pruebas**

adicionales para asegurar que tanto la accesibilidad como la calidad del producto no se han visto comprometidas. Desafortunadamente, no siempre se realizan estas comprobaciones, por motivos de lo más diversos, como actualizaciones contrarreloj o externalización del mantenimiento, involucrando un personal que no sigue los procedimientos establecidos y/o no dispone de la formación adecuada.

Se recomienda implantar una política de **monitorización** que permita recolectar datos, identificar problemas en el futuro y medir una serie de indicadores relacionados con la calidad de la aplicación móvil, el nivel de accesibilidad, el cumplimiento de estándares, etc. Así, se debe considerar el **seguimiento** de los productos a lo largo de todo su **ciclo de vida**, no sólo en las etapas iniciales, repitiendo todo el **proceso de revisión** en caso de que la aplicación, una vez lanzada al mercado, sufra cambios de calado que afecten a la interacción con respecto al usuario.

El análisis de los datos obtenidos permitirá **conocer el estado y la evolución** de las aplicaciones, si se están cumpliendo los objetivos técnicos de accesibilidad establecidos o los objetivos generales del producto, además de facilitar la aplicación de las acciones correctivas pertinentes para solucionar posibles desviaciones. De este modo, la monitorización permitirá verificar que las aplicaciones móviles **no se degradan a lo largo del tiempo** como consecuencia de la adhesión de nuevas características, contenidos o funcionalidades.

Para conseguir que una aplicación móvil mantenga el mismo nivel de accesibilidad en el tiempo se deben realizar las siguientes acciones:

1. - Concretar qué personas serán las **responsables** del proceso de monitorización, creando **procedimientos** que permitan resolver rápidamente las desviaciones y problemas que sean revelados.

2. - Obtener las **herramientas** (HW/SW) que se estimen necesarias para acometer la monitorización de la mejor manera posible.

3. - Identificar claramente los **requisitos** que deben satisfacerse y el **ámbito** o alcance de la evaluación dentro de la aplicación móvil.

4. - Especificar el **método** y la **frecuencia** de las evaluaciones, definiendo de manera detallada cómo registrar la información recabada para tener una **documentación** rigurosa (fecha, requisitos, tecnologías empleadas, resultados, etc.).

5. - Analizar los **nuevos componentes** que sean incorporados a la aplicación móvil tras haber sido lanzada al mercado.

Opcionalmente, se pueden incorporar a la aplicación móvil los mecanismos oportunos para **obtener información** directa de los **usuarios** (*feedback*), incluyendo aquellos que presenten alguna forma de discapacidad. Estos datos, junto con los que puedan obtenerse mediante encuestas o formularios, permitirán conocer hasta qué punto los usuarios pueden usar el producto y ayudarán a saber cuál es el nivel de satisfacción alcanzado.

Además, se recomienda emplear **métodos complementarios** durante la monitorización para optimizar el proceso. Así, se pueden combinar los eventos y resultados estadísticos obtenidos con herramientas de **evaluación automática** con los resultados cualitativos y detallados alcanzados mediante **análisis manuales**, con la fiabilidad extra de contar con la opinión de expertos en la materia.

No hay que olvidar que algunos **requisitos** de accesibilidad no pueden ser comprobados mediante herramientas automáticas, requiriendo siempre de una **revisión manual**, o deben ser chequeados posteriormente por personal cualificado para evitar falsos positivos.

Por supuesto, los procesos de **evaluación** de conformidad respecto al nivel de accesibilidad a los que se hace referencia pueden ser realizados tanto por profesionales que se encuentren dentro de la **empresa** que llevó a cabo el desarrollo como por **auditores externos** que analicen el producto periódicamente.

6. VALIDACIÓN DE ACCESIBILIDAD DE APLICACIONES MÓVILES -

Tomando como base la norma europea EN 301 549 es posible **evaluar** la conformidad de una aplicación móvil en materia de **accesibilidad**, ya que en dicho documento se definen qué requisitos están involucrados en la evaluación y cómo analizarlos.

En este apartado se detallan las características de la **validación automática** y la **validación manual**, ofreciendo las principales diferencias entre ellas, se facilita una **tabla de equivalencias** entre conjuntos de requisitos de accesibilidad (nivel AA de WCAG 2.0 y apartado relativo a aplicaciones móviles de norma EN 301 549) y se establece cómo realizar el **proceso** de validación del nivel de accesibilidad de las aplicaciones móviles.

6.1. VALIDACIÓN AUTOMÁTICA VS VALIDACIÓN MANUAL

A la hora de evaluar la accesibilidad de una aplicación, se suelen emplear **herramientas automáticas** con las que es posible identificar fácilmente algunos de los problemas de accesibilidad que presenta. Sin embargo, los resultados de este tipo de herramientas involucran normalmente tanto **falsos negativos** como **falsos positivos**, puesto que no son capaces de detectar todos los fallos existentes (algunos de ellos no pueden procesarse automáticamente) y porque algunos de los errores revelados no son realmente fallos.

Así, por ejemplo, una herramienta de evaluación automática puede detectar si una imagen tiene **texto alternativo**, pero no puede interpretar si dicho texto es **representativo** con respecto al contenido de la imagen.

Esta ausencia de **fiabilidad** implica que deben acometerse **análisis manuales** complementarios, en los que el evaluador experto no sólo debe llevar a cabo un examen completo en base a los **requisitos** pertinentes, sino que además debe revisar la totalidad de **fallos** detectados por las herramientas de análisis automático.

Por tanto, las herramientas automáticas servirán de ayuda en el proceso de evaluación, pero su utilización nunca debe ser entendida como un análisis completo.

Además, **cada herramienta** de análisis automático tendrá sus funciones, defectos y **fortalezas**, por lo que será necesario hacer uso de varias de ellas para alcanzar resultados óptimos. Con el tiempo, cuando el evaluador haya utilizado suficientes veces dichas herramientas, las conocerá mejor y será capaz de interpretar los distintos fallos de accesibilidad ofrecidos por cada una de ellas, de modo que pueda hacer una **selección inteligente** de aquellos criterios de conformidad para los que se tiene la certeza de que los resultados de evaluación ofrecidos son correctos.

A continuación, se brinda una comparación entre las características de las **validaciones automáticas** y las **validaciones manuales**, viendo cuáles son los puntos fuertes y débiles en cada caso.

Validación Automática	Validación Manual
Involucra herramientas que permiten realizar, con una periodicidad determinada, la recolección de **datos** y/o **análisis automáticos** sobre la accesibilidad de las vistas y componentes de una aplicación móvil.	Revisiones manuales realizadas por **personal experto**, que identifica las posibles **desviaciones** con respecto a los requisitos de accesibilidad y propone las **correcciones** pertinentes para mejorar la aplicación móvil. También puede implicar técnicas de recolección de **datos** de los **usuarios** (tests, encuestas, etc.) que deban realizarse o evaluarse manualmente.
Se pueden realizar tantas validaciones de este tipo como se estime necesario, pudiendo efectuar **seguimientos continuos** con la **regularidad** deseada (cada semana, cada quince días, cada mes, etc.).	Esta clase de revisiones requiere un **esfuerzo mayor**, ya que es más lenta en la en la detección y corrección de las desviaciones. Por este motivo suele tener una **periodicidad** más **amplia** que las revisiones automáticas (normalmente semestral o anualmente).
Permite revelar únicamente **problemas** detectables con métodos **automáticos**, aquellos en los que no es necesaria la intervención de un experto.	Evidencia **todo tipo** de **problemas**, con un gran nivel de detalle y fiabilidad.
Es una validación muy **exhaustiva** con respecto a los **datos** recolectados y los **problemas** detectados, ya que puede llevarse a cabo sobre un número elevado de vistas de la aplicación o, incluso, sobre la aplicación completa.	Salvo que la aplicación sea muy sencilla, sólo puede involucrar un **conjunto limitado** de sus **vistas** y **componentes**, lo que puede suponer pasar por alto algunos fallos.

Por supuesto, tal y como se ha comentado con anterioridad, para obtener los mejores resultados es recomendable emplear más de un método de validación. Por ejemplo, **combinando métodos automáticos** de varias fuentes que proporcionan abundante información **con otros métodos manuales** que proporcionan información más detallada, enriquecida y fiable.

6.2. - TABLA DE EQUIVALENCIAS RESPECTO A CONJUNTOS DE REQUISITOS DE ACCESIBILIDAD

A continuación se facilita una **tabla** en la que se establece la **relación** existente entre los **requisitos** concernientes a las aplicaciones móviles (capítulo 11) de la norma **EN 301 549** y las directrices de nivel AA del **WCAG 2.0**.

Como puede observarse, para mantener la secuencia original del conjunto de pautas ofrecido por el W3C, algunos **requisitos** de la norma EN 301 549 permanecen **vacíos**.

Descripción	EN 301 549	WCAG 2.0 (AA)
Contenido no textual (compatible con lector de pantalla)	11.2.1.1	1.1.1
Sólo audio y sólo vídeo (grabado)	11.2.1.2	1.2.1
Subtítulos (grabado)	11.2.1.3	1.2.2
Audiodescripción o contenido multimedia alternativo (grabado)	11.2.1.4	1.2.3
Subtítulos (en directo)	11.2.1.5	1.2.4
Audiodescripción (grabado)	11.2.1.6	1.2.5
Información y relaciones	11.2.1.7	1.3.1
Secuencia significativa	11.2.1.8	1.3.2
Características sensoriales	11.2.1.9	1.3.3
Uso del color	11.2.1.10	1.4.1
Control del audio	11.2.1.11	1.4.2
Contraste (mínimo)	11.2.1.12	1.4.3
Cambio de tamaño del texto	11.2.1.13	1.4.4
Imágenes de texto	11.2.1.14	1.4.5
Teclado	11.2.1.15	2.1.1
Sin trampas para el foco del teclado	11.2.1.16	2.1.2
Tiempo ajustable	11.2.1.17	2.2.1
Poner en pausa, detener o ajustar	11.2.1.18	2.2.2
Umbral de 3 destellos o menos	11.2.1.19	2.3.1
Vacío (Evitar bloques)	11.2.1.20	2.4.1
Vacío (Titulado de páginas)	11.2.1.21	2.4.2
Orden del foco	11.2.1.22	2.4.3
Propósito de los enlaces	11.2.1.23	2.4.4
Vacío (Múltiples vías)	11.2.1.24	2.4.5
Encabezados y etiquetas	11.2.1.25	2.4.6
Foco visible	11.2.1.26	2.4.7
Idioma del software	11.2.1.27	3.1.1
Vacío (Idioma de las partes)	11.2.1.28	3.1.2
Al recibir el foco	11.2.1.29	3.2.1
Al recibir entradas	11.2.1.30	3.2.2
Vacío (Navegación coherente)	11.2.1.31	3.2.3
Vacío (Identificación coherente)	11.2.1.32	3.2.4
Identificación de errores	11.2.1.33	3.3.1

Descripción	EN 301 549	WCAG 2.0 (AA)
Etiquetas o instrucciones	11.2.1.34	3.3.2
Sugerencias ante errores	11.2.1.35	3.3.3
Prevención de errores (legales, financieros, de datos)	11.2.1.36	3.3.4
Procesamiento	11.2.1.37	4.1.1
Nombre, función, valor	11.2.1.38	4.1.2

6.3. PROCESO DE VALIDACIÓN

Tal y como se ha comentado previamente, para obtener los **mejores resultados** es necesario que la evaluación de accesibilidad se realice ya desde el proceso de desarrollo de la aplicación móvil, contemplando los conceptos oportunos desde las **fases iniciales** en lugar de esperar a tener la aplicación terminada.

Sin embargo, una vez que el **producto** está **publicado** hay que realizar **evaluaciones** de accesibilidad **adicionales**, para verificar que se satisfaga el nivel de conformidad demandado. Dichas evaluaciones se pueden efectuar como parte de procesos de **validación**, consultoría/certificación o como análisis de accesibilidad **periódicos** dentro del proceso de **monitorización** del nivel de accesibilidad de la aplicación móvil (ver el apartado *"5.4 Monitorización"*).

En esta sección se describe una **metodología** general de trabajo para realizar **evaluaciones** de **accesibilidad** sobre aplicaciones móviles, exponiendo desde la especificación de los criterios de conformidad aplicables hasta un checklist de comprobación de las pautas de validación, pasando por la selección de escenarios de prueba y las estrategias de prueba de validación de accesibilidad.

6.3.1. Especificación de criterios de conformidad aplicables

Para conocer los **criterios de conformidad** de la norma **EN 301 549** que son **aplicables** a la hora de analizar la accesibilidad de la aplicación móvil objetivo, una alternativa válida consiste en la utilización del **árbol de decisión** planteado por Loïc Martínez Normand en 2017[5]. Para ello se tienen que contestar una serie de preguntas, a través de cuyas respuestas se revelan los criterios de conformidad o requisitos que deben ser considerados en la evaluación del nivel de accesibilidad de la aplicación a validar.

Esto **reduce** ostensiblemente el **esfuerzo** requerido para completar el análisis necesario, ya que en vez de contemplar todos los requisitos y recomendaciones de la norma EN 301 549, sólo se tienen en cuenta aquellos **relacionados** con las **características** de la aplicación móvil a evaluar.

[5] http://oa.upm.es/45580/

Por tanto, el evaluador sólo debe contestar a cada una de las **preguntas** del árbol de decisión y analizar el cumplimiento del conjunto de requisitos revelados como aplicables.

Para lograrlo se deben interpretar todos los **criterios** involucrados, donde cada criterio tiene **dos partes** claramente diferenciadas:

1. - En primer lugar se brinda una **precondición**, a partir de la cual se puede saber si el requisito es o no aplicable.

2. - A continuación se ofrece el texto correspondiente al **cuerpo del requisito**, con una descripción significativa acerca de cómo cumplir con el criterio de conformidad.

6.3.2. Selección de escenarios de prueba (muestra representativa)

Al completar las pruebas de validación lo ideal es analizar todas las **vistas** y **componentes** de la aplicación objetivo, pero esto no siempre es factible debido al tamaño y **complejidad** que pueden tener algunos productos hoy en día. Así, aunque es posible examinar una **aplicación completa** haciendo uso de ciertas herramientas de evaluación automática, llevar a cabo el examen manual sobre todos los elementos de la misma puede resultar imposible.

Por este motivo, en muchos casos es necesario **seleccionar una muestra representativa** sobre la que hacer la validación posterior, una que sea suficientemente genérica como para aportar una **visión global** de la aplicación móvil sin dejar de considerar todas las tipologías de componentes, vistas, contenidos y controles, servicios y tecnologías que la componen.

Además, se deben considerar en la muestra aquellos apartados de especial relevancia para las **personas con discapacidad**, como las secciones de ayuda, documentación, configuración de preferencias, mensajes de error, etc.

Considerando todo lo anterior, escoger una **muestra representativa** puede ser uno de los procedimientos más importantes de la evaluación y a la vez uno de los menos intuitivos. Esto se debe a que, salvo que la aplicación sea de un **tamaño reducido**, contará con un número tan grande de componentes que no será posible examinarla exhaustivamente y en su totalidad, lo que perfila la selección de la muestra a evaluar como un paso de suma **importancia** para obtener buenos resultados en la evaluación.

En esta sección vamos a detallar un **método** exhaustivo (parcialmente basado en WCAG-EM) con el que se puede obtener un **escenario de prueba significativo** con un alto porcentaje de confianza, garantizando unos resultados de evaluación fiables.

6.3.2.1. Identificación de vistas a evaluar

Identificar vistas comunes

Las vistas comunes son aquellas que son **más visibles** dentro de la aplicación. Se suele poder acceder a ellas directamente desde la vista principal o, en su caso, pestañas a lo largo de varias vistas.

En algunas aplicaciones se accede a éstas desde un menú de navegación. No todas las vistas que son accesibles desde éstos son comunes. Es trabajo del evaluador **identificar** aquellas que tienen **más relevancia** para el usuario.

Identificar la funcionalidad esencial

Es vital identificar correctamente la **funcionalidad** o funcionalidades esenciales de la aplicación para poder seleccionar las vistas de la muestra. Es posible que las vistas que dan acceso a estas funcionalidades **coincidan** con las **vistas comunes** identificadas anteriormente, pero no tiene por qué ser el caso.

Identificar tecnologías en las que se confíe

Las plataformas móviles suelen disponer de aplicaciones y **componentes** gráficos **nativos**, pero también pueden ofrecer **contenidos** basados en **HTML 5** y, si lo soportasen, **otras tecnologías** como flash. Es importante **evaluar la accesibilidad** de las vistas que utilicen tecnologías diferentes al grueso de la aplicación.

Además, si es posible hacerlo, identificar **librerías de terceros** que se hayan utilizado para el desarrollo de la aplicación puede ser también un buen punto para la evaluación, ya que los componentes gráficos reutilizados de éstas pueden no haber sido diseñados con la accesibilidad en mente, lo que suele provocar **problemas de accesibilidad** constantes a lo largo de la aplicación.

Identificar otras vistas relevantes

Por último debemos tener en cuenta también **cualquier otra vista** que no hayamos incluido en los pasos anteriores, pero que consideremos **importante** para el uso de la aplicación, como pueden ser apartados de ayuda o configuraciones imprescindibles.

6.3.2.2. Definición de muestras

Una vez hecho este proceso de identificación, ahora procedemos a definir las distintas **muestras** que vamos a **incluir** para la evaluación.

Definir la muestra estructurada

Ésta contendrá **todas las vistas** que hemos seleccionado anteriormente, lo que debería suponer una muestra representativa de la aplicación para la evaluación.

Definir la muestra aleatoria

Además de la muestra estructurada, definimos también una muestra aleatoria que deberá contener al menos un **10%** del número de **vistas** incluidas en la muestra estructurada. El objetivo de esta muestra es tener un **grupo de control** con el que podamos contrastar resultados.

Si los resultados de la muestra estructurada y la muestra aleatoria **difieren** considerablemente, entonces no habremos seleccionado correctamente las vistas para la muestra estructurada y deberemos **repetir** el **proceso**.

Incluir procesos completos

Si alguna de las **vistas** que hemos seleccionado en la muestra estructurada o aleatoria forma parte de un **proceso** más grande, entonces deberemos incluir todas las vistas pertenecientes a ese proceso para la evaluación.

6.3.3. Estrategias de prueba de validación de accesibilidad

Tras haber seleccionado la muestra, ya se puede llevar a cabo el **análisis** de accesibilidad de la aplicación móvil, evaluando las vistas y controles oportunos mediante los **requisitos** identificados como aplicables.

Para poder completar el proceso de **validación** con garantías es recomendable considerar unas **estrategias** de prueba concretas, especialmente útiles en caso de examinar una aplicación desarrollada por terceros:

- Las aplicaciones móviles suelen albergar toda clase de **contenidos** visuales, sonoros y **multimedia** en su interfaz de usuario, pudiendo además incluir tanto archivos de datos incorporados dentro de la propia aplicación como elementos que son descargados usando servicios web. Para tener acceso a dichos **recursos** (archivos de texto, imágenes, sonidos…) se recomienda ponerse en contacto con las personas encargadas de desarrollar la aplicación móvil, de modo que los pongan a disposición del **evaluador** para que pueda trabajar directamente sobre ellos en las labores de análisis del nivel de accesibilidad.

- Deben comprobarse todos los **componentes** de la muestra seleccionada, dado que con que uno de ellos no satisfaga los requisitos relacionados la aplicación no sería accesible. Esto se debe a que dicho elemento es un candidato firme a ser una **barrera de accesibilidad** que puede ser incluso bloqueante, evitando que el usuario pueda avanzar y/o navegar por la aplicación.

- Una **aplicación** nativa que cuente con versiones para distintas plataformas (**multiplataforma**) debe ser evaluada de forma separada en cada una de ellas. Esto se debe a que pueden existir grandes **diferencias** en cuanto a sus características y/o prestaciones (capa de accesibilidad + productos de apoyo), provocando que obtenga resultados de evaluación de accesibilidad muy distintos en sus diferentes versiones.

- Si la aplicación ofrece la posibilidad de hacer modificaciones de calado en su **configuración** (personalización), la validación debe realizarse haciendo comprobaciones con todas las posibles configuraciones existentes. De esta manera se verificará que el **comportamiento** es el **deseado** en todas ellas.

- En caso de que la aplicación tenga distintos **perfiles de usuario** y siempre que entre ellos existan cambios reseñables en las vistas, funcionalidades, permisos… se deben analizar todos los perfiles, evitando así que haya **fallos** de accesibilidad presentes en uno de ellos que puedan pasar **desapercibidos**.

- Siempre que sea posible, debe complementarse la **revisión manual** mediante análisis de requisitos con un examen en el que se utilicen **herramientas** de evaluación **automáticas** y/o de ayuda en el proceso de evaluación (descritas para cada plataforma dentro del apartado 7).

- También es conveniente emplear **productos de apoyo** (apartado 8) al completar la evaluación, intentando acercarse lo máximo posible a los **usos habituales** que personas de diferentes perfiles harán de la aplicación al utilizarla.

6.3.4. Checklist de comprobación de pautas de validación

Una vez generada la **aplicación móvil** y tras haberla puesto a disposición del público, hay que **seguir verificando** que responde a los criterios de accesibilidad pretendidos, chequeando los factores a cumplir.

Para lograrlo hay que crear una **lista de comprobación**, enumerando los requisitos de la norma EN 301 549 identificados como aplicables (dependientes de las características de la aplicación), y hacer **pruebas** en las que se valore su cumplimiento/incumplimiento.

Al verificar dichos requisitos se revelará el **nivel de accesibilidad** de la app, repasando los aspectos más importantes que pueden afectar a su utilización por parte de usuarios con discapacidad.

Dichas **pruebas**, que idealmente deberían ser realizadas por **personal** que **no haya participado** en el desarrollo de la aplicación, son fundamentales para conseguir que ésta sea realmente accesible, ya que permiten descubrir **problemas** con la **interacción** de los usuarios que no son evidentes durante las fases de diseño y desarrollo del producto SW.

Además, tal y como se ha comentado en las estrategias del apartado anterior, se recomienda realizar las pruebas de verificación con los **servicios de accesibilidad activados**, habilitando cada uno de ellos y comprobando que el funcionamiento de la aplicación es correcto en los requisitos que corresponda.

NOTA: a la hora de hacer las pruebas de validación es recomendable recurrir a los contenidos del **anexo C** de la norma EN 301 549, donde se concretan aspectos relativos al cumplimiento de los distintos **requisitos** de dicho documento.

NOTA 2: también puede resultar de utilidad el **anexo B** de la norma EN 301 549, donde se cotejan los distintos requisitos con los posibles **perfiles** atendiendo a las capacidades del usuario.

7. HERRAMIENTAS DE EVALUACIÓN DE ACCESIBILIDAD -

Para asegurar que las **aplicaciones** desarrolladas cumplen con las características necesarias para considerarse **accesibles**, hay que **validar** una serie de aspectos. El método más obvio para hacer esto es realizar un chequeo manual de todas las características expuestas en la guía.

No obstante, para agilizar el trabajo, disponemos de algunas **herramientas** que nos permiten **automatizar** hasta cierto punto esta tarea o que nos **facilitan** la labor de la **comprobación manual**. Éstas pueden ser de diversa índole, pero todas ellas tienen en común la capacidad de comprobar aspectos de la accesibilidad concretos de nuestra aplicación o, al menos, allanarnos el camino a la hora de analizarlos.

Desafortunadamente, la cantidad de herramientas existentes en el **ámbito móvil** en la actualidad y su nivel de madurez están **lejos** de lo observable en **materia web**, donde hay muchas más y más completas (valoran más requisitos).

Es necesario precisar que el hecho de utilizar estas herramientas no es suficiente para **garantizar** la **accesibilidad** del producto. En el mejor de los casos las utilidades podrán chequear todos los criterios, pero nada nos asegura que no existan **falsos positivos** o problemas **no detectados**. Un ejemplo típico sería el de un elemento gráfico etiquetado con un texto no significativo, lo cual no cumpliría con la directiva ni devolvería como resultado una detección de fallo.

Además, dependiendo de la **fase de desarrollo** en la que se encuentre el proyecto, se podrán usar unas herramientas u otras. Algunas permiten analizar aplicaciones en tiempo de **ejecución**, mientras que otras se limitan a la evaluación del **código fuente**. La frontera entre estas categorías es difusa, pues en muchos casos no existe diferencia entre utilizar la herramienta en fase de desarrollo o de explotación. Siempre hay que tener en cuenta que una herramienta que pueda aplicarse a un producto ya terminado puede usarse también si el desarrollo está en curso, pero no sucede lo mismo a la inversa.

Hay que aclarar que **ninguna** de las **herramientas** mencionadas en este apartado implica un **coste económico** para completar la evaluación del nivel de accesibilidad de una aplicación móvil, evitando así que se incrementen los gastos para mejorar las aplicaciones en esta materia. Por supuesto, pueden existir más herramientas gratuitas y herramientas comerciales a las que cada institución puede recurrir, en base a sus propios condicionantes y **preferencias**.

7.1. DEFINICIONES

Antes de presentar las herramientas generales y específicas de cada plataforma conviene fijar unas breves definiciones, en base a las cuales se **clasificará** posteriormente cada una de las **herramientas** descritas.

Según la necesidad de contar con la **intervención** de una persona durante el proceso de evaluación del nivel de accesibilidad, podemos distinguir dos tipos de herramienta:

- Una **herramienta de evaluación automatizada** permite generar un informe de forma automática, **sin intervención** del usuario en medio del proceso.

- Una **herramienta de evaluación manual** es aquella que **requiere** de la interacción del **usuario** para ofrecer la información. Ésta puede consistir en un reporte de errores sobre parte de la interfaz o la muestra de datos sin procesar, los cuales deberán ser juzgados por el usuario.

Además, tal y como se ha definido anteriormente, en función de si el análisis de accesibilidad se lleva a cabo mientras se desarrolla el producto o una vez que éste se está ejecutando, podemos diferenciar dos categorías:

- Una **herramienta en tiempo de desarrollo** trabaja sobre el **código fuente** de la aplicación, puesto que al ser ejecutada accede a información concreta (normalmente metadatos) presente en dicho código, recurso necesario para analizar el nivel de accesibilidad de la aplicación objetivo.

- Una **herramienta en tiempo de ejecución** puede ser utilizada sobre una aplicación que se esté ejecutando, ya sea en el sistema concreto de destino o en un emulador.

7.2. HERRAMIENTAS GENERALES

Existen herramientas de carácter general que no dependen de la plataforma para la que se esté desarrollando (no específicas del sistema), con las que se pueden medir **aspectos genéricos** como el **contraste** entre el texto y el fondo sobre el que se sitúa.

Dentro de este grupo de herramientas generales también se describen algunas que facilitan el proceso de **evaluación manual**, como **extensiones** para diferentes editores de código que permiten, por ejemplo, especificar la información de los componentes de la aplicación.

Por último, se detallan varias herramientas que permiten realizar comprobaciones de forma **automática** para analizar la **accesibilidad** de una aplicación móvil, reduciendo así significativamente el tiempo de revisión y detectando numerosos problemas que de otra forma serían difíciles de identificar.

Al respecto de la revisión automática de accesibilidad, es recomendable escoger varias herramientas diferentes, ya que no todas detectan los mismos problemas y además pueden producir falsos positivos, detectando como problema algo que en realidad no lo es. Por eso se recomienda utilizar varias herramientas automáticas, **al menos dos** de ellas, para poder cotejar los resultados.

Estas herramientas de evaluación automática se pueden usar, además de para analizar toda o gran parte de la aplicación móvil, como **ayuda en la detección** y selección de aquellas vistas que sean más problemáticas o que incluyan contenidos, funcionalidades o tipologías no detectadas en la selección manual de la muestra pero que merezcan especial atención.

Como comentamos antes, es importante recordar que a pesar de la utilidad de las herramientas de evaluación automática, existen determinados aspectos que no son comprobables automáticamente, siendo necesaria una **revisión manual complementaria**.

A continuación se describen brevemente las herramientas generales.

7.2.1. Colour Contrast Analyser (CCA)

Tipo: Manual, tiempo de desarrollo, tiempo de ejecución.

URL: https://www.paciellogroup.com/resources/contrastanalyser/

Se trata de una herramienta de escritorio que permite medir el **nivel de contraste** de los elementos visuales y controles gráficos (textos, botones, etc.). Está disponible para **Windows** y **Mac** en múltiples idiomas.

Para ello ofrece:

- Una funcionalidad que determina el **cumplimiento** o incumplimiento del nivel de contraste requerido por las directrices de **WCAG 2.0**, que coincide con lo solicitado en la norma EN 301 549 (ya que se basa en el nivel de conformidad AA de WCAG 2.0).

- Un **simulador** que permite recrear las condiciones visuales de **perfiles** muy diferentes, como el de aquellas personas con daltonismo (Protanopia, Deuteranopia, Tritanopia) o cataratas. También permite visualizar la interfaz gráfica en escala de grises o con los colores invertidos. Gracias a esta funcionalidad el desarrollador puede ponerse en la piel de **usuarios** con capacidades diversas.

Puede ser utilizada sobre capturas de pantalla de la aplicación móvil para valorar la legibilidad de los textos y componentes de la interfaz mostrados al usuario (según los ratios del WCAG 2.0), así como para evaluar cómo ven dicha aplicación las personas con las formas de **discapacidad visual** que permite simular.

7.2.2. Mobile Accessibility plugin [PhoneGap]

Tipo: Manual, tiempo de desarrollo.

URL: https://github.com/phonegap/phonegap-mobile-accessibility

Plugin de PhoneGap que brinda información de **estado** relacionada con varias de las **características** de **accesibilidad** de los sistemas operativos para móviles, permitiendo saber, por ejemplo, si está activado el lector de pantalla, si se ha habilitado la inversión de colores o cuál es el tamaño preferido para el texto. Además, permite a una aplicación mandar cadenas para ser leídas por el **lector de pantalla** o ejecutar comandos para detener el funcionamiento de dicho servicio de accesibilidad.

Se puede utilizar a la hora de generar **aplicaciones** móviles accesibles **híbridas**, basadas en componentes web, para las plataformas soportadas: Amazon Fire OS, Android o iOS.

7.2.3. Accessibility Checker [NetBeans IDE]

Tipo: Automatizada, tiempo de desarrollo.

URL: http://plugins.netbeans.org/plugin/7577/

Es un **plugin** que puede ser incorporado y utilizado dentro del entorno de desarrollo integrado **NetBeans**. A través de dicho módulo se puede **testear** fácilmente una aplicación en tiempo de diseño, verificando la corrección a nivel de **accesibilidad** de sus componentes. Como resultado ofrece una lista con los **errores** de accesibilidad detectados, así como advertencias e información adicional para poder mejorar el producto final.

Accessibility Checker permite inspeccionar fácilmente: que las **descripciones** sean accesibles, que los **nombres** sean accesibles, que exista correspondencia en los controles con respecto a sus **etiquetas** (usando labelFor), los **mnemónicos** y que los componentes sean alcanzables mediante **tabulación**.

7.2.4. Accessibility Tools Framework (ACTF) [Eclipse IDE]

Tipo: Automatizada, manual, tiempo de desarrollo.

URL: http://www.eclipse.org/actf/

ACTF es un **framework** con el que los desarrolladores pueden construir herramientas para **evaluar** y **mejorar** la **accesibilidad** de las aplicaciones y contenidos, de modo que puedan ser utilizados sin dificultad por las personas con discapacidad. Ofrece un conjunto de **herramientas** de ejemplo muy diversas, generadas a partir del framework, como herramientas de validación, aplicaciones de simulación de tecnologías de apoyo, herramientas de visualización de la usabilidad, herramientas de pruebas unitarias e interfaces alternativas accesibles para las aplicaciones. Todo ello se encuentra definido sobre el propio entorno provisto por **Eclipse**, lo que facilita su integración en el flujo de trabajo y favorece la creación de aplicaciones y contenidos accesibles de forma asequible.

7.3. HERRAMIENTAS POR PLATAFORMA

7.3.1. Android

7.3.1.1. Accessibility Scanner

Tipo: Automatizado, tiempo de ejecución.

URL:
https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.auditor

Esta herramienta de **análisis automático** se instala como una aplicación y se integra en las opciones de accesibilidad del dispositivo. Una vez hecho esto, podemos ejecutarla sobre una **vista** de cualquier **aplicación**. El resultado será una lista de sugerencias que podemos poner en práctica para mejorar la accesibilidad de la misma.

7.3.1.2. Accessibility Test Framework

Tipo: Automatizada, tiempo de desarrollo.

URL: https://github.com/google/Accessibility-Test-Framework-for-Android

Se trata de una **API** que da acceso a las propiedades de accesibilidad de las vistas de la aplicación. Con ella podemos incluir criterios de accesibilidad en nuestros **test automatizados**.

7.3.1.3. Node Tree Debugging

Tipo: Manual, tiempo de desarrollo.

URL: https://developer.android.com/guide/topics/ui/accessibility/node-tree-debugging.html

Esta herramienta está incluida dentro del lector de pantalla TalkBack. Permite visualizar la **jerarquía** de elementos de la interfaz tal y como se expone de cara a la **capa de accesibilidad**, por lo que puede ser útil para fases de desarrollo y debugging.

7.3.1.4. UI Automator Viewer

Tipo: Manual, tiempo de desarrollo.

URL:

https://developer.android.com/topic/libraries/testing-support-library/index.html#UIAutomator

https://developer.android.com/topic/libraries/testing-support-library/index.html#uia-viewer

Herramienta que permite **inspeccionar** los elementos de la interfaz gráfica, obteniendo las **propiedades** de la capa de accesibilidad. Es útil para encontrar dónde se hayan los **errores** que hayamos podido detectar mediante otras formas de testing. La aplicación se encuentra en el directorio %android-sdk%/tools/.

7.3.1.5. Lint

Tipo: Automatizado, tiempo de desarrollo.

URL: http://tools.android.com/tips/lint

Lint es una herramienta integrada en **Android Studio**. Muestra advertencias sobre posibles **problemas** de accesibilidad localizados en el **código fuente**. Con ella, se puede acceder

rápidamente al fragmento que presenta el inconveniente y resolverlo, ofreciéndonos un mensaje descriptivo del fallo.

7.3.1.6. Espresso

Tipo: Automatizado, tiempo de desarrollo.

URL:

https://developer.android.com/topic/libraries/testing-support-library/index.html#Espresso

https://developer.android.com/reference/android/support/test/espresso/Espresso.html

Expresso es una **librería** de test de accesibilidad orientada a proporcionar una manera rápida y sencilla de realizar **test automatizados**. Además, permite comprobar si ciertos comportamientos o condiciones se cumplen.

7.3.1.7. Robolectric

Tipo: Automatizado, tiempo de desarrollo. -

URL: http://robolectric.org/ -

Se trata de una **librería** que permite ejecutar **test** de aplicaciones Android directamente sobre una **JVM** (Java Virtual Machine), en lugar de requerir de un emulador. -

7.3.2. iOS

7.3.2.1. Accessibility Inspector

Tipo: Manual, tiempo de desarrollo.

URL:
https://developer.apple.com/library/content/technotes/TestingAccessibilityOfiOSApps/TestAccessibilityiniOSSimulatorwithAccessibilityInspector/TestAccessibilityiniOSSimulatorwithAccessibilityInspector.html

Accessibility Inspector es una herramienta integrada en el entorno de desarrollo **XCode**. Permite observar las **propiedades** de accesibilidad de los componentes de la interfaz, sus **acciones** asociadas y su posición en la **jerarquía** de accesibilidad.

7.3.2.2. Accessibility Verifier

Tipo: Automatizado, tiempo de ejecución.

URL:
https://developer.apple.com/library/content/documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXTestingApps.html#//apple_ref/doc/uid/TP40001078-CH210-SW4

Esta herramienta ofrece la generación de un **informe** con los problemas de **accesibilidad** detectados de manera automática sobre la aplicación.

7.3.3. Windows

Las tres herramientas que se detallan a continuación vienen incorporadas con el **SDK** de **Windows** a partir de la versión **8.1** del sistema operativo. Se pueden encontrar en la ruta de instalación del SDK: %RUTA_SDK%\VERSION\bin\PLATFORM.

Donde RUTA_SDK es la ruta de instalación (por defecto, "C:\Program Files (x86)\Windows Kits", VERSION es la versión de Windows y PLATFORM es la arquitectura del procesador.

7.3.3.1. UI Accessibility Checker (AccChecker)

Tipo: Automatizada, manual, tiempo de desarrollo, tiempo de ejecución. -

URL: https://msdn.microsoft.com/en-us/library/windows/desktop/hh920985(v=vs.85).aspx -

Permite **probar** diferentes **escenarios**, verificar la exactitud de la información accesible y descubrir **problemas** de accesibilidad en tiempo de ejecución. -

También es conocida como AccChecker. Se trata de una herramienta **multipropósito** que engloba tres funcionalidades diferentes: -

- Una aplicación gráfica que permite el **testing manual**, servicio de logging y generación de supresión.
- Una API para usar en frameworks de **testing automatizados**.
- Una aplicación de consola que soporta **escenarios** de test no gestionados en los que la API no puede utilizarse.

Es compatible con las dos capas de accesibilidad de Windows, MSAA y UIA.

7.3.3.2. AccScope

Tipo: Manual, tiempo de desarrollo.

URL: https://msdn.microsoft.com/en-us/library/windows/desktop/dn433239(v=vs.85).aspx

AccScope es una herramienta pensada para **chequear** la **accesibilidad** de nuestra aplicación en las etapas de diseño y desarrollo de la misma. Gracias a ella, podemos ver las

propiedades que se expondrán a un lector de pantalla y añadir información si lo consideramos necesario.

7.3.3.3. Inspect

Tipo: Manual, tiempo de ejecución.

URL: https://msdn.microsoft.com/en-us/library/windows/desktop/dd318521(v=vs.85).aspx

Inspect expone los datos de **accesibilidad** de los **componentes** de la aplicación, así como los **patrones** de control (concretamente propiedades y patrones de control de Microsoft UI Automation y propiedades de Microsoft Active Accessibility). También nos permite examinar la **jerarquía** de elementos y saber aquellos que estarán incluidos en la **capa de accesibilidad**, con lo que se puede verificar la estructura de navegación de elementos de automatización en árbol de UI Automation y objetos accesibles en jerarquía de Microsoft Active Accessibility.

Su funcionamiento se asemeja a AccScope, aunque en este caso nos encontramos una estructura de **navegación** en forma de **árbol**, lo que lo convierte también en una herramienta accesible de por sí.

7.4. COMPARATIVA DE HERRAMIENTAS

A continuación se ofrece una tabla en la que se clasifican las diferentes **herramientas**, con el objeto de que los desarrolladores sepan cuáles deben usar cuando se muevan entre las diferentes **plataformas**.

Característica	Android	iOS	Windows
Generación de informes	Accessibility Scanner	Accessibility Verifier	UI Accessibility Checker
Test automatizado	Accessibility Test Framework Espresso		UI Accessibility Checker
Exploración de los componentes gráficos	UI Automator Viewer Robolectric	Accessibility Inspector	AccScope
Exploración de la jerarquía	Node Tree Debugging	Accessibility Inspector	Inspect
Acciones de usuario	UI Automator Viewer	Accessibility Inspector	AccScope
Detección de problemas en desarrollo	Lint		

8. PRODUCTOS O HERRAMIENTAS DE APOYO -

Además de utilizar herramientas para evaluar la accesibilidad de una aplicación móvil, es recomendable hacer **pruebas** de forma similar a como acceden a ella las personas con discapacidad, empleando productos o **herramientas de apoyo**. Así, con unas condiciones de prueba significativas, es más fácil evidenciar **problemas** de acceso al contenido que de otro modo podrían pasar desapercibidos.

8.1. ¿QUÉ ES UNA HERRAMIENTA DE APOYO?

Una herramienta de apoyo es un producto **software** o **hardware** que permite el **acceso** de una persona con discapacidad a una **característica** a la que no tendría la oportunidad de acceder de otra forma. Junto al diseño universal, constituye el pilar fundamental de la accesibilidad.

Al contrario de lo que se pueda pensar, las herramientas de apoyo no sólo ofrecen ventajas a personas con un alto grado de **discapacidad**. El reconocimiento de voz, el software de texto a voz o las configuraciones de texto grande y alto contraste, son ejemplos de tecnologías de **asistencia** que han pasado a formar parte del día a día de la sociedad, aportando **ventajas** a todos los **usuarios**. Es por esto que no se debe pensar en estas herramientas como un mercado de nicho, sino como utilidades que potencialmente pueden servir a un gran número de personas.

Estas herramientas son muy importantes, ya que el cumplimiento de los principios de desarrollo, así como la evaluación del nivel de accesibilidad de una aplicación móvil, depende enormemente del aprovechamiento y la **compatibilidad** de las aplicaciones con los **servicios de accesibilidad** de los sistemas operativos.

8.2. HERRAMIENTAS NATIVAS Y HERRAMIENTAS DE TERCEROS

Actualmente, la mayoría de los **sistemas operativos** maduros ofrecen herramientas de apoyo **por defecto**, dando la oportunidad a sus usuarios de utilizar sus dispositivos de forma accesible desde el encendido, tal y como recoge la norma EN 301 549, salvo que el aparato esté averiado.

Gracias a estas herramientas, los sistemas operativos **facilitan** su **acceso** a toda clase de **usuarios**, incluyendo a las personas con discapacidad. Normalmente, todos los SSOO ofrecen los mismos **tipos de servicio**, con alguna que otra variación en cuanto a las características entre plataformas. Ejemplos de estas herramientas son los lectores de pantalla, los magnificadores, las opciones de texto grande y alto contraste, el dictado y el reconocimiento de voz, los asistentes virtuales…

Sin embargo, esto no quiere decir que no haya **herramientas** desarrolladas por **terceros** que traten de competir con los productos nativos. Esto no es así en todas las plataformas, pero podemos encontrar varias **alternativas** en aquellas más populares y abiertas. Las hay gratuitas, de código abierto e incluso privativas, que utilizan licencias comerciales para financiar su desarrollo.

Esto es posible gracias a que los sistemas operativos ofrecen ciertas **características** en sus **capas de accesibilidad** que nos permiten desarrollar aplicaciones no sólo accesibles, sino que funcionen como soporte para los usuarios con discapacidad.

Esta posibilidad de desarrollo es un detalle importante, ya que en caso de que las opciones de accesibilidad de los SSOO no cubran las **necesidades** de algún perfil de usuario particular, será necesario contar con **alternativas externas** que den respuesta a dichas necesidades.

Considerando todo lo anterior, hay que recalcar que no hay una **opción** mejor que otra. Dependiendo del sistema y de la herramienta concreta, es posible que las nativas sean mejor opción que las desarrolladas por terceros o viceversa. También puede darse el caso de que una herramienta esté mejor optimizada para un método de entrada/salida, como pueden ser el teclado y la pantalla táctil en el caso de los lectores de pantalla. Por tanto, debemos tener en cuenta estas **variaciones** y comprobar el funcionamiento de nuestras aplicaciones con la mayor cantidad de **herramientas** de apoyo que podamos.

8.3. TIPOS DE HERRAMIENTAS DE APOYO

A continuación vamos a dar una breve introducción a cada una de las **clases** de **herramientas de apoyo** más extendidas:

8.3.1. Lector de pantalla

Un lector de pantalla es un producto **software** que interactúa con la **capa de accesibilidad** de la plataforma para obtener los datos sobre la presentación de la aplicación y trasladárselos al usuario a través de un **canal alternativo** al visual. Este canal puede ser el **auditivo**, con una narración en voz alta (TTS), o **táctil**, mediante un dispositivo conectado al terminal conocido como línea braille.

Hay que tener en cuenta que el uso de un lector de pantalla es generalmente acompañado por un **teclado** en dispositivos de sobremesa y portátiles, y de una **pantalla táctil** en terminales móviles y tabletas, aunque existen métodos de entrada alternativos. En cualquier caso, una característica común a todos estos métodos de entrada es la **navegación secuencial**. En muchas ocasiones, el usuario irá recorriendo los distintos elementos de la interfaz gráfica utilizando el tabulador, las teclas de cursor o gestos táctiles sobre la pantalla. Por eso es muy importante el **orden** de navegación de los elementos y que todos aquellos que sean importantes para la funcionalidad de la aplicación puedan ser alcanzados por el **foco** del sistema.

Existen otros métodos de navegación, como puede ser la **exploración táctil**, en la que el usuario recibe información sobre el elemento gráfico que tiene debajo del dedo para decidir si activarlo o no, o simplemente para leer su contenido; también se puede encontrar la **navegación** en forma de **árbol**, en la que se dispone la interfaz como si se tratara de un árbol lógico con contenedores (raíces) y elementos contenidos (hojas), de modo que se puede ir alcanzando mayor o menor profundidad; modos de **revisión** todavía más complejos incluyen la revisión de **texto plano** o la verbalización del texto bajo el **cursor** del ratón.

En cualquier caso, aparte de la navegación secuencial, todos los lectores de pantalla suelen ofrecer diversos **comandos** que facilitan la navegación, saltando directamente a ciertos contenidos u obviando otros. Esto agiliza bastante el proceso de **búsqueda** de la información interesante. Es muy típico especialmente en aquellas aplicaciones de carácter web, como los navegadores, donde se puede navegar directamente entre **encabezados** o **puntos de interés** semánticos.

8.3.2. Magnificadores

El magnificador es un **software** pensado para **aumentar** drásticamente el **tamaño** del **contenido** visualizado, a modo de lupa virtual. Puede encontrarse en las vertientes de pantalla completa o pantalla dividida, en la que una parte de la interfaz continúa mostrándose al tamaño habitual para ofrecer una mejor orientación espacial al usuario, que puede perderse si únicamente dispone de la imagen magnificada.

Algunas aplicaciones comunes de esta tecnología han llegado a los dispositivos móviles en forma de **gestos** para poder agrandar imágenes y disfrutar de una visualización más cómoda de ciertos detalles.

8.3.3. Texto grande y alto contraste

Estas dos **características** quizás sean las que más tiempo llevan presentes en los **dispositivos móviles**, ya que desde que las pantallas empezaron a hacerse en color, solían disponer de temas que favorecían la visualización. No sólo están pensadas para personas con **baja visión**. También favorecen a aquellos usuarios que deben leer durante periodos prolongados de tiempo, ya que un **tamaño** adecuado del **texto** y un **contraste** suficiente disminuyen la fatiga visual.

La introducción de aplicaciones extensivamente en estas plataformas ha supuesto un problema para estas características, ya que los desarrolladores pueden obviarlas en sus trabajos. Es por eso que cada **fabricante** ha elaborado una serie de **pautas** de **estilo** con las que estas herramientas puedan operar correctamente sobre cualquier tipo de aplicación.

8.3.4. Escala de grises

Esta utilidad está especialmente pensada para aquellas personas que sufren algún tipo de **disfunción visual** que les impide distinguir ciertos **colores** dentro de la paleta. Al convertir la interfaz a una escala de grises, pueden **distinguir** claramente los contornos y las formas de los distintos elementos, así como leer correctamente cualquier texto.

8.3.5. Sonido monoaural

Estamos acostumbrados hoy en día a encontrar dispositivos con **sonido** estéreo e incluso de una calidad superior, como pueden ser aquellos certificados con tecnologías como Dolby. Sin embargo, para personas que no dispongan de una audición aceptable en ambos oídos, esta

división del sonido a través de **distintos canales** puede suponer una **pérdida de información**.

Es por este motivo que los sistemas ofrecen la posibilidad de utilizar una **salida monoaural**. Así estos usuarios podrán acceder a toda la información sonora sin tener que perder ni un ápice de ella.

8.3.6. Control por voz

En general, puede ser interesante poder **controlar** nuestro dispositivo mediante **comandos de voz** o, incluso, mediante lenguaje natural, gracias a los asistentes virtuales inteligentes que están cada día más integrados en las distintas plataformas. Sin embargo, esto puede ser especialmente útil para aquellas personas que, por cualquier causa, encuentran dificultades o imposibilidades a la hora de **interactuar** con el método estándar de **entrada** del dispositivo, como la pantalla táctil.

Podemos encontrar este tipo de usuarios en personas que padecen algún tipo de **discapacidad motora** o incluso otros perfiles no tan obvios. Siempre es más fácil para alguien poco habituado a tratar con la tecnología dar órdenes a su terminal en lugar de estar navegando a través de distintos menús para obtener el mismo resultado.

8.3.7. Dictado por voz

Como en el caso anterior, la posibilidad de **introducir texto** a través de la **voz** es una característica ampliamente utilizada por la sociedad. Es útil cuando no se pueden utilizar las manos para **escribir** debido a las circunstancias, como estar ocupado haciendo otras tareas, tales como cocinar o conducir.

Especialmente destacable es su utilidad para aquellos usuarios que no son capaces de introducir texto mediante las **interfaces** habituales o que disponen de una **velocidad** de escritura **lenta**, en comparación con aquellas personas con una diversidad funcional común. Esto no sólo se restringe a personas con dificultades motoras, ya que los usuarios con baja visión o ceguera también presentan esta dificultad, aunque el motivo sea completamente diferente.

8.3.8. Navegación mediante interruptor y por barrido

Este estilo de navegación, generalmente, utiliza un dispositivo externo de tipo **pulsador**. Con él se desplaza el **foco** del sistema a lo largo de la interfaz de manera **secuencial**, de modo que se activa el elemento deseado cuando el foco lo alcanza.

La navegación por **barrido** actúa de forma similar, aunque la diferencia reside en el método de **desplazamiento** del foco, que en este caso es **automático**. El usuario sólo realiza la pulsación o acción de activación cuando el foco está situado en el elemento que desea.

8.3.9. Texto predictivo

Esta característica está pensada para **acelerar** la **escritura** de texto para cualquier persona. En un primer momento, se diseñó como una alternativa para aquellos usuarios que tenían una **movilidad reducida** y su método de entrada era objetivamente mucho más lento que el usual, como puede ser el caso de la navegación por interruptor o por barrido. En cualquier caso, se ha convertido en una herramienta tremendamente útil y ampliamente **extendida**.

8.4. HERRAMIENTAS DE APOYO MÁS UTILIZADAS

La siguiente tabla ofrece una referencia rápida para las distintas **herramientas de apoyo** disponibles en cada **sistema operativo** móvil:

Producto de apoyo/SO	Android	iOS	Windows
Lector de pantalla	Talkback Voice Assistant ** ShinePlus	Voice Over	Narrador NVDA * Jaws *
Magnificador	Lupa ShinePlus	Zoom	Lupa Magic * Zoomtext *
Alto contraste	Sí	Sí	Sí
Texto grande	Sí	Sí	Sí
Escala de grises	Sí***	Sí	No
Sonido monoaural	Sí	Sí	Sí
Control por voz	Google Now Google Assistant ¡Y muchas otras de terceros!	Siri	Cortana Dragon NaturallySpeaking *
Dictado por voz	Sí	Sí	Sí
Control por interruptor	Sí	No	No
Control por barrido	Sí	No	No
Texto predictivo	Sí	Sí	Sí

* Sólo disponible en versiones **x86/x64** de Windows, aplicable tanto a equipos de escritorio como a tablets con este tipo de **procesadores**. Estas herramientas han sido incluidas por su gran popularidad y por la posibilidad de que sean **compatibles** con los smartphones en un futuro cercano, gracias a la convergencia del sistema Windows 10.

** Sólo disponible para dispositivos **Samsung Galaxy**.

*** Disponibilidad dependiente del **fabricante** del dispositivo móvil.

NOTA: las herramientas de apoyo presentes en el Sistema Operativo dependen del **dispositivo** y de la **distribución**, por lo que en algunos casos puede que no se correspondan estrictamente con las descritas en este apartado.

9. BIBLIOGRAFÍA -

1. - Guía de Validación de Accesibilidad Web, J. González (Ministerio de Hacienda y Administraciones Públicas), Diciembre 2014 [en sección Guías Prácticas de http://administracionelectronica.gob.es/PAe/accesibilidad/documentacion].

2. - Cómo hacer "Apps" accesibles (nº1 de "Infórmate sobre..."), Santiago Gil González (CEAPAT-IMSERSO), Febrero 2013 [http://www.ceapat.es/InterPresent1/groups/imserso/documents/binario/appsaccesibles.pdf].

3. - Metodología para Evaluar la Accesibilidad de Aplicaciones Nativas, ILUNION, Septiembre 2015 [http://www.amovil.es/es/blogs/metodologia-evaluar-accesibilidad-aplicaciones-nativas].

4. - EN 301 549 (Accessibility requirements suitable for public procurement of ICT products and services in Europe), ETSI+CEN+CENELEC, Abril 2015 [http://www.etsi.org/deliver/etsi_en/301500_301599/301549/01.01.02_60/en_301549v010102p.pdf].

5. - Website Accessibility Conformance Evaluation Methodology (WCAG-EM) 1.0, W3C, Julio 2014 [https://www.w3.org/TR/WCAG-EM/].

6. - Árbol de decisión para EN 301 549, Loïc Martínez Normand, Abril 2017 [http://oa.upm.es/45580/].

10. ANEXO I: EQUIPO RESPONSABLE DEL PROYECTO -

Coordinación del proyecto

Elena Muñoz Salinero (Ministerio de Hacienda y Función Pública).

Financiación del proyecto

Guía realizada con el apoyo de la Red de Cooperación ESVI-AL, de la Comunidad Autónoma de Madrid y de la Universidad de Alcalá (UAH).

Autores

Juan Aguado Delgado. -

Francisco Javier Estrada Martínez. -

Web de proyectos de accesibilidad software de la Universidad de Alcalá (UAH)

http://tifyc-pmi.cc.uah.es/appacces/

Editores y revisores

Universidad de Alcalá (UAH)

- José Ramón Hilera González.
- José María Gutiérrez Martínez.
- Salvador Otón Tortosa.

Ministerio de Hacienda y Función Pública

- Elena Muñoz Salinero.
- César Cuadros Moreno.

Ilunion

- Verónica Martorell Martínez.
- Daniel Montalvo Charameli.
- Fabiola Isabel Ávila López.

CEAPAT

- Miguel Ángel Valero Duboy.
- José Pascual Gallego.

Calidad y Publicación

Paloma Juez Alonso (Ministerio de Hacienda y Función Pública).

11. ANEXO II: EJEMPLOS CONCRETOS DE CÓDIGO EN DISTINTOS SISTEMAS OPERATIVOS

Esta sección contiene un **ejemplo** de desarrollo concreto de una **aplicación accesible**, en la que se plantean algunas de las cuestiones de accesibilidad más comunes. Se trata de un **gestor de tareas** que utiliza las vistas más típicas que pueden encontrarse en cualquier aplicación.

Para aumentar su utilidad, se ha desarrollado dicho ejemplo para las **plataformas mayoritarias** atendiendo a la cuota de mercado móvil en la actualidad: Android e iOS. Los ejemplos se han puesto a disposición del público en **GitHub** de forma independiente, distinguiendo el caso de cada plataforma para que sea más fácil de manipular y revisar.

Por supuesto, hay que decir que no se trata de una aplicación plenamente funcional. Sólo se ha desarrollado la **interfaz** y la **funcionalidad básica**, dejando el resto como mero adorno estético para demostrar varios de los puntos importantes a la hora de tener en cuenta la accesibilidad.

Los ejemplos se han creado siguiendo un ciclo de **dos fases**: en la primera etapa, se ha desarrollado la aplicación **sin** tener en cuenta la **accesibilidad**, agregando únicamente la funcionalidad principal y todas las vistas necesarias; después, en una segunda iteración, se han **corregido** los **fallos** de accesibilidad que no satisfacían lo estipulado en la norma EN 301 549. Esta última revisión se llevó a cabo en una **rama diferente** denominada “accesible”. De esta manera, es más fácil detectar los cambios que fueron acometidos para cumplir los criterios de accesibilidad.

Este planteamiento se ha adoptado tras una reflexión profunda del tema. Se prefirió un **ejemplo** más o menos **grande** con **múltiples fallos** de accesibilidad simultáneos en lugar de muchos ejemplos pequeños con un solo fallo, ya que este primer supuesto se acerca más a lo que un desarrollador puede encontrarse en el **mundo real**. Además, la alternativa es más común y fácil de encontrar, ya que tanto en la documentación técnica de los diferentes sistemas como en distintos sitios web de consulta abundan ejemplos de pequeños fragmentos de código.

A continuación se encuentran los diferentes **enlaces** de los repositorios de GitHub:

- Android: https://github.com/ctt-gob-es/Ejemplo-App-Accesible-Android
- iOS: https://github.com/ctt-gob-es/Ejemplo-App-Accesible-iOS

NOTA: No se incluye un ejemplo para la plataforma **Universal Windows Platform**, dado que en el transcurso del proceso de creación de la guía Microsoft anunció su intención de **descontinuar** Windows 10 Mobile. Sin embargo, los autores han decidido mantener en el documento el resto de materiales relacionados con Windows, dado su valor a la hora de trabajar en aplicaciones ya existentes y al llevar a cabo el proceso de monitorización.

12. ANEXO III: REQUISITOS A SATISFACER EN EVALUACIÓN DEL NIVEL DE ACCESIBILIDAD DE APLICACIÓN MÓVIL

Los requisitos a satisfacer a la hora de evaluar la **accesibilidad** de una aplicación móvil dependen de las características de la propia aplicación, de modo que examinar las **propiedades** del software será suficiente para considerar o descartar **criterios** de conformidad de **EN 301 549**.

- En **cualquier caso** => requisitos genéricos del capítulo 5 (la directiva no obliga).
- En **aplicaciones** móviles **híbridas** (con componentes web) => capítulo 9.
- En **aplicaciones** móviles **nativas** => capítulo 11. Concretamente:
  - Si es software **no web** sin funcionalidad cerrada => capítulo 11.2.1.
  - Si es software **no web** con **funcionalidad cerrada** => capítulo 11.2.2.
  - Para respetar la **interoperabilidad** con productos de apoyo => capítulo 11.3.
  - Requisitos de **uso** de **accesibilidad** documentado => capítulo 11.4.
  - Considerar los criterios de conformidad relativos a las preferencias de usuario si posee **interfaz de usuario** => capítulo 11.5.
  - Si la aplicación es **herramienta de autor** => capítulo 11.6.

Opcionalmente, según las **funcionalidades** que ofrezca la aplicación, puede ser conveniente aplicar criterios de conformidad de otras secciones:

- Del capítulo 6 si implica **comunicación bidireccional**.
- Del capítulo 7 si tiene capacidades de **vídeo**.
- Del capítulo 8 si involucra **Hardware**.
- Del capítulo 10 si alberga en su interior **documentos no web**.
- Del capítulo 12 si ofrece **documentación** o **servicios de apoyo** al usuario.
- Del capítulo 13 si proporciona acceso a servicios de **intermediación** o emergencia.

13. ANEXO IV: TABLA RESUMEN DE HERRAMIENTAS -

A continuación se facilita una tabla que sintetiza la información más relevante y las principales **características** de las diversas **herramientas** comentadas en la guía, organizadas en orden alfabético.

Nombre	Sitio web	Descripción	Plataforma	Automática	Manual	Tiempo de desarrollo	Tiempo de ejecución	Librería
Accessibility Checker [NetBeans IDE]	http://plugins.netbeans.org/plugin/7577/	Plugin de NetBeans a través del cual se puede testear fácilmente una aplicación en tiempo de diseño, verificando su corrección a nivel de accesibilidad.	Múltiple	X		X		
Accessibility Inspector	https://developer.apple.com/library/content/technotes/TestingAccessibilityOfiOSApps/TestAccessibilityiniOSSimulatorwithAccessibilityInspector/TestAccessibilityiniOSSimulatorwithAccessibilityInspector.html	Herramienta integrada en el entorno de desarrollo XCode que permite observar las propiedades de accesibilidad de los componentes de la interfaz, sus acciones asociadas y su posición en la jerarquía de accesibilidad.	iOS		X	X		
Accessibility Scanner	https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.auditor	Herramienta de análisis automático que puede ser ejecutada sobre una vista de cualquier aplicación, ofreciendo una lista de sugerencias para mejorar la accesibilidad de la misma.	Android	X			X	
Accessibility Test Framework	https://github.com/google/Accessibility-Test-Framework-for-Android	API que da acceso a las propiedades de accesibilidad de las vistas de la aplicación. Permite incluir criterios de accesibilidad en test automatizados.	Android	X		X		
Accessibility Tools Framework (ACTF) [Eclipse IDE]	http://www.eclipse.org/actf/	Framework con el que se pueden construir herramientas para evaluar y mejorar la accesibilidad de las aplicaciones y contenidos.	Múltiple	X	X	X		
Accessibility Verifier	https://developer.apple.com/library/content/documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXTestingApps.html#//apple_ref/doc/uid/TP40001078-CH210-SW4	Esta herramienta ofrece la generación de un informe con los problemas de accesibilidad detectados de manera automática sobre la aplicación.	iOS	X			X	
AccScope	https://msdn.microsoft.com/en-us/library/windows/desktop/dn433239(v=vs.85).aspx	Herramienta pensada para chequear la accesibilidad de nuestra aplicación en las etapas de diseño y desarrollo de la misma.	Windows		X	X		

Nombre	Sitio web	Descripción	Plataforma	Automática	Manual	Tiempo de desarrollo	Tiempo de ejecución	Librería
Colour Contrast Analyser (CCA)	https://www.paciellogroup.com/resources/contrastanalyser/	Permite medir el contraste de los elementos visuales y controles gráficos (textos, botones, etc.).	Escritorio (Windows o Mac)		X	X	X	
Espresso	https://developer.android.com/topic/libraries/testing-support-library/index.html#Espresso https://developer.android.com/reference/android/support/test/espresso/Espresso.html	Librería de test de accesibilidad orientada a proporcionar una manera rápida y sencilla de realizar test automatizados.	Android	X		X		X
Inspect	https://msdn.microsoft.com/en-us/library/windows/desktop/dd318521(v=vs.85).aspx	Expone datos de accesibilidad de componentes de la aplicación y patrones de control, permitiendo examinar la jerarquía de elementos y saber cuáles estarán incluidos en la capa de accesibilidad.	Windows		X		X	
Lint	http://tools.android.com/tips/lint	Revela posibles problemas de accesibilidad localizados en el código fuente.	Android	X		X		
Mobile Accessibility plugin [PhoneGap]	https://github.com/phonegap/phonegap-mobile-accessibility	Plugin que brinda información de estado relacionada con características de accesibilidad de los SSOO para móviles.	Amazon Fire OS, Android o iOS		X	X		
Node Tree Debugging	https://developer.android.com/guide/topics/ui/accessibility/node-tree-debugging.html	Permite visualizar la jerarquía de elementos de la interfaz tal y como se expone de cara a la capa de accesibilidad.	Android		X	X		
Robolectric	http://robolectric.org/	Permite ejecutar test de aplicaciones sobre una JVM (Java Virtual Machine).	Android	X		X		X
UI Accessibility Checker (AccChecker)	https://msdn.microsoft.com/en-us/library/windows/desktop/hh920985(v=vs.85).aspx	Permite probar diferentes escenarios, verificar la exactitud de la información accesible y descubrir problemas de accesibilidad en tiempo de ejecución.	Windows	X	X	X	X	
UI Automator Viewer	https://developer.android.com/topic/libraries/testing-support-library/index.html#UIAutomator https://developer.android.com/topic/libraries/testing-support-library/index.html#uia-viewer	Herramienta que permite inspeccionar los elementos de la interfaz gráfica, obteniendo las propiedades de la capa de accesibilidad.	Android		X	X		

14. ANEXO V: GUÍA RÁPIDA POR PLATAFORMAS -

En este apartado se ofrece una breve recopilación de las recomendaciones para la accesibilidad recogidas en la **documentación** para **desarrolladores** de las plataformas móviles Android, iOS y Windows.

14.1. ANDROID

Página de referencia

https://developer.android.com/guide/topics/ui/accessibility/index.html

Herramientas de apoyo

- TalkBack.
- Switch Access.
- BrailleBack.
- Voice Access.
- Texto grande.
- Alto contraste.
- Lupa.

Pautas de accesibilidad

- Diseño: Elementos claramente visibles.
    - Contraste:
        - Texto pequeño: 4.5:1.
        - Texto grande (14pt en negrita/18pt+): 3:1.
    - Tamaño de áreas táctiles: elemento + padding.
        - 48 x 48 dp (9 mm aproximadamente). Recomendado entre 7 y 10 mm.
        - Separación de 8 dp entre áreas táctiles.
        - Seleccionar el tamaño de la fuente en sp (píxeles escalables) para habilitar las funciones de tamaño del texto del sistema.
        - Reservar suficiente espacio para poder incluir el texto en cualquier idioma.

- Jerarquía clara de elementos según su importancia.
- Elementos relacionados agrupados.
- Información clave discernible de un vistazo.
- Ofrecer alternativas a sonidos y eventos visuales.
- Permitir para, pausar u ocultar elementos que se muevan, hagan scroll o parpadeen durante más de 5 segundos.
- Limitar el parpadeo/destello de elementos a un máximo de 3 veces por segundo.
- Evitar destellos en regiones grandes y centrales de la pantalla.

- Notificar visualmente no sólo con colores, sino también con diseño.
- Robustez:
  - Navegación – Da a los usuarios confianza para saber dónde están y qué es importante.
  - Entender las tareas importantes – Refuerza los elementos importantes mediante texto, elementos visuales, movimiento…
  - Acceso a tu app: Proporciona un etiquetado adecuado para ofrecer un acceso textual a tu app.
    - Proporcionar atributo contentDescription para todo elemento gráfico o que no tenga una representación textual.
    - Proporcionar descripción de la acción con el atributo hint cuando pueda haber confusión.
  - Soporta las herramientas de asistencia específicas de tu plataforma.
  - Evitar usar temporizador en controles importantes para su ocultación.
  - Proveer una alternativa para acceder a controles ocultados por temporizador.
  - La funcionalidad debería poder usarse con un número de pasos mínimos. El control del foco debería estar implementado para las funciones más importantes.
  - Cualquier característica con consideraciones especiales de accesibilidad debería estar incluida en la documentación de ayuda.

- Confirmar acciones que puedan ser potencialmente destructivas mediante diálogos y otro tipo de mensajes.
- Asegurar que las vistas personalizadas cumplen con la accesibilidad.

Antipatrones de accesibilidad

- Sonidos innecesarios de fondo, como música, que puedan entorpecer la experiencia.
- Sonidos añadidos a elementos de la vista.
- Etiquetas largas y poco concisas.
- Incluir el tipo de elemento en el etiquetado.
- El etiquetado es una descripción de la apariencia y no de la funcionalidad.

Herramientas de testing de accesibilidad

- Accessibility Scanner.
- Node Tree Debugging.
- UI Automator Viewer.
- Lint.
- Espresso.
- Robolectric.

14.2. IOS

Página de referencia

https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/iPhoneAccessibility/Accessibility_on_iPhone/Accessibility_on_iPhone.html

Herramientas de apoyo

- Zoom.
- White and black.
- Mono Audio.
- Speak auto-text.
- Voice Control.
- VoiceOver.

Librería de accesibilidad

UIaccessibility Programming Interface.

Componentes

- UIAccessibility Informal Protocol: Reporta estado de accesibilidad e información descriptiva del elemento.
- UIAccessibilityContainer Informal Protocol: Este protocolo permite que los elementos de un contenedor sean accesibles como elementos individuales.
- UIAccessibilityElement class: Define un objeto que puede ser devuelto mediante el UIAccessibilityContainer Protocol. Se utiliza para representar elementos que no son automáticamente accesibles.
- UIAccessibilityConstants.h header file: Define los rasgos de accesibilidad que una aplicación puede exhibir y las notificaciones que puede enviar.

Atributos de accesibilidad

- Label: Describe el elemento, pero no su tipo.
- Traits: Una combinación de uno o más traits individuales, cada uno de los cuales describe un aspecto del elemento.
- Hint: Frase corta que describe el resultado de la activación de una acción.
- Frame: Describe la localización y el tamaño del elemento en la pantalla.
- Value: El valor actual del elemento, si puede tenerlo.

Pautas de accesibilidad

- Todos los elementos con los que el usuario puede interactuar son accesibles.
    - Las vistas predefinidas en la API estándar son accesibles.
    - Ofrecer un atributo label descriptivo para aquellos elementos que contengan gráficos.
    - Si se crean vistas personalizadas, debe asegurarse que son accesibles.
- Todos los elementos accesibles ofrecen información precisa y que sirve de ayuda.
    - Se ha de asegurar que los elementos utilizados en la vista son los correctos y no dan lugar a confusión.

- Ofrecer una descripción del resultado de la acción cuando éste no esté claro sólo con el atributo label.

- El contenido dinámico debe ofrecer información precisa y actualizada. Se deben notificar los cambios en la vista al usuario.

Antipatrones de accesibilidad

- Especificar el tipo de elemento dentro del atributo label.
- Ofrecer información en el atributo label que no empiece por mayúscula o termine en punto.
- No ofrecer la información de hint en tercera persona.

Herramientas de testing

- Accessibility Inspector.

14.3. UNIVERSAL WINDOWS PLATFORM

Página de referencia

https://developer.microsoft.com/en-us/windows/accessible-apps

Librería de accesibilidad

Windows Automation API.

Herramientas de apoyo

- Narrador.
- Lupa.
- Alto contraste.
- Texto grande.

Pautas de accesibilidad

- Exponer los elementos de interfaz de usuario al acceso mediante programación.
  - Establecer nombre accesible.
  - Establecer descripción accesible (si fuera necesario).
  - Establecer rol (si fuera necesario).

  - Establecer valor accesible (si fuera necesario).
  - Determina si la vista aparece en el árbol de accesibilidad.
  - Asociar campos de formulario con etiquetas de texto.
  - Los nombres accesibles deben estar localizados.
  - Las regiones dinámicas deben notificar los cambios.
- La aplicación admite navegación por teclado.
  - Establecer una jerarquía lógica de controles.
  - Agrupar controles relacionados.
  - Determinar el método de desplazamiento entre controles.
  - Definir atajos de teclado.
  - Los controles deben poder recibir el foco.
  - El orden de navegación debe ser lo más adecuado posible, sin tener por qué coincidir con el visual o el del documento estructurado.
  - Los eventos deben poder activarse mediante el teclado.
- Configuración de color y contraste accesibles.
  - Coherencia con los colores del sistema.
  - En caso de estilo personalizado, el contraste debe ser:
    - Texto pequeño: 5:1.
    - Texto grande (14pt negrita/18 pt+): 3:1.
  - No usar sólo el color para transmitir información.
  - No utilizar elementos que parpadeen más de 3 veces por segundo.

Antipatrones de accesibilidad

- Utilizar componentes personalizados si se pueden utilizar los estándar de la interfaz o algunos que ya hayan implementado la accesibilidad.
- Establecer elementos de texto estático o no interactivo en el orden de navegación. Puede confundir a los usuarios.

- Utilizar posicionamiento absoluto.
- Actualizar automáticamente el lienzo de la aplicación.

Herramienta de testing de accesibilidad

- AccScope.
- Inspect.
- UI Accessibility Checker.
- UI Automation Verify.
- Accessible Event Watcher.