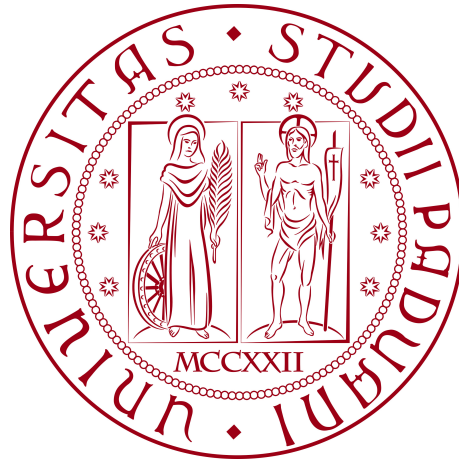


University of Padua

DEPARTMENT OF MATHEMATICS “TULLIO LEVI-CIVITA”

MASTER DEGREE IN COMPUTER SCIENCE



**Mobile content accessibility guidelines on the
Flutter framework**

Master's Thesis

Supervisor

Prof Ombretta Gaggi

Graduating

Matteo Budai

ID Number 2057217

ACADEMIC YEAR 2023-2024

Abstract

This thesis aims to describe accessibility problems encountered on mobile user interfaces using the framework Flutter. Accessibility is the ability of a user to fully perceive, understand, navigate, and interact with digital content regardless from their capabilities. A brief introduction explain problems that users can encountered today when they use their smartphones. First of all, we start from the guidelines defined in WCAG trying to the understand if they are sufficient or not, considering all categories of users, mainly focusing on users with disabilities. Then we integrate the WCAG guidelines with the new success criteria introduced with WCAG 2.1 and WCAG 2.2 and we define new guidelines for mobile that can help the end user. The thesis also showed how the framework Flutter addresses accessibility issues, proposing some new solutions. In conclusion, we define our results by trying to understand if the new mobile content accessibility guidelines are important and we analyze and understand how accessible the Flutter framework is for all categories of users.

Contents

1	Introduction	1
1.1	Related Works	5
2	Accessibility	9
2.1	WCAG	10
2.1.1	Principles	10
2.1.1.1	Perceivable	10
2.1.1.2	Operable	11
2.1.1.3	Understandable	11
2.1.1.4	Robust	12
2.1.2	Guidelines	13
2.1.3	Success criteria	13
2.1.4	WCAG 2.2	14
3	MCAG	28
3.1	Perceivable	28
3.1.1	New Ideas - Perceivable	32
3.2	Operable	33
3.2.1	New Ideas - Operable	36
3.3	Understandable	41
3.3.1	New Ideas - Understandable	44
3.4	Robust	48
3.4.1	New Ideas - Robust	49
3.5	Ideas for the future	49
4	Analysis on Flutter	55
4.1	Flutter	55

CONTENTS

4.2	Dart	56
4.3	Components	57
4.3.1	Embedder	58
4.3.2	Engine	58
4.3.3	Framework	58
4.4	Material and Cupertino libraries	58
4.5	Widget	59
4.5.1	Stateless widget	59
4.5.2	Stateful widget	59
4.6	Accessibility of the application	60
4.7	Analysis of the application	61
5	Conclusions	75
	Acronyms and abbreviations	77
	Glossary	78

List of Figures

4.1	Logo Flutter	55
4.2	Flutter layered architecture	57
4.3	First Page	62
4.4	Answer	62
4.5	Drawer	64
4.6	Second page	66
4.7	Third Page	68
4.8	Details	68
4.9	Fourth Page	70
4.10	DataPicker	71
4.11	Alert Dialog	71
4.12	Fifth Page	73

List of Tables

4.1	First Page Analysis	62
4.2	Drawer Analysis	64
4.3	Second Page Analysis	66
4.4	Third Page Analysis	68
4.5	Fourth Page Analysis	71
4.6	Fifth Page Analysis	73
4.7	Other Considerations to analyze	74

Listings

4.1	AppBar Widget Code	63
4.2	IconButton Widget Code	63
4.3	DrawerHeader Widget	65
4.4	ListTile Widget	65
4.5	Image Widget	69
4.6	Icon Widget	69
4.7	Text (header) Widget	69
4.8	TextFormField Widget	72
4.9	DropDownButtonFormField Widget	72
4.10	SwitchListTile Widget	72
4.11	Language Component	74

Chapter 1

Introduction

In recent years, the number of smartphone users [29] is growing very fast, with one billion users in 2013 to 7,41 billion in 2024. Furthermore the number of webpages loaded on mobile devices in October 2016 had overtaken desktops and laptops. To encourage this phenomenon, ensuring equal access to the Internet is essential to all people. The term accessibility, referring to the web, means the possibility of accessing information and services available online by diversified categories of users and from a range of different devices. It is the ability of a user to fully perceive, understand, navigate, and interact with digital content regardless from their capabilities. The disadvantaged classes [14] to access web content regard:

- *View*: Users who are blind or have low vision require alternative text descriptions, screen readers, or magnification tools;
- *Movement*: Individuals with limited dexterity may require adapted input methods such as voice commands or switch controls;
- *Hearing*: Users who are deaf or hard of hearing rely on captions, transcripts, or visual alternatives for audio content;
- *Convulsions*: Accessibility includes considerations to minimize flashing or rapidly changing content that could trigger seizures;
- *Intellect*: Users with cognitive disabilities benefit from simplified navigation, clear language and consistent design.

We have to consider also that there are different range of devices used by various categories of people:

- Phones and tablets;
- Digital TVs;
- Wearables;
- Devices in car dashboards and airplane seatbacks;
- Devices in household appliances;
- Other "Internet of Things".

Moreover, while we use our smartphones there are different issues [36] [32] to consider:

- *Touchscreen interfaces*: Accessibility solutions must accommodate touch gestures, pinch-to-zoom and multi-touch interactions;
- *Small screen sizes*: Design considerations include optimizing layout and content hierarchy for smaller screens;
- *Different input modalities*: Support for voice input, gesture recognition and adaptive controllers enhances accessibility;
- *Device use in different settings*: Mobile applications must function seamlessly in different lighting conditions, noise levels and physical contexts;
- etc.

In this thesis we focus on mobile devices where it is crucial that their applications are accessible to users with disabilities. According to the [World Health Organization \(WHO\)](#), it is estimated that over one billion members of the global population are affected by some form of disability. Hence, they can't be ignored by developers this issue.

We have to consider useful events which address accessibility issues:

- *Screen reader - Microsoft Narrator [26]*: It is a built-in screen reader software that was introduced by Microsoft in 2000. The idea is firstly for users of Microsoft Windows operating systems. The primary function of Narrator is to provide auditory feedback to visually impaired users, enabling them to interact with the operating system and applications without needing to rely on visual cues. Narrator reads the text on the screen, including menus, dialog boxes, and all the interface elements, making computers accessible to individuals with visual impairments;

- *Universal access in macOS devices [23]*: It is an initiative launched in 2002, aimed to enhance accessibility features across macOS operating system. The aim is to integrate a suite of accessibility tools directly into macOS, making them readily available to all users. These tools include features such as VoiceOver, Zoom, and other adjustments. VoiceOver, described below, is a powerful screen reader that provides spoken descriptions of on-screen elements and allows users to navigate their Mac using keyboard commands or gestures;
- *Apple's VoiceOver [43]*: It is Apple's screen reader technology introduced in 2005 and significantly improved in 2009. It is integrated across Apple's ecosystem, including macOS, iOS, iPadOS, and watchOS. VoiceOver uses speech synthesis to provide auditory descriptions of on-screen content, allowing users with visual impairments to navigate apps and interact with digital content. It supports multiple languages and offers customizable settings to provide to individual accessibility needs;
- *Google TalkBack screen reader [21]*: It is an accessibility feature developed by Google for Android devices. It was introduced in 2011 to help users with visual impairments in navigating Android interfaces and applications. Similar to other screen readers, TalkBack provides spoken feedback for on-screen content and allows users to interact with their devices using gestures and keyboard shortcuts. Over the years, Google has continued to improve TalkBack with new features and improvements for accessibility for Android users.

We used the screen reader in the next sections to test the accessibility on mobile applications.

An other important aspect to mention regards the legislation:

- *Americans with Disabilities (ADA) 1990 [4] [2]*: It is a comprehensive civil rights law that prohibits discrimination against individuals with disabilities in all areas of public life, including jobs, schools, transportation, and all public and private places open to the general public. Title III of the ADA specifically mandates that places of public accommodation, such as restaurants, hotels, and theaters, provide equal access to individuals with disabilities. This includes ensuring that websites and digital services offered by covered entities are accessible;
- *USA 2001 [42]*: Section 508 requires federal agencies to ensure that their **Electronic and Information Technology (EIT)** is accessible to people with disabilities, including federal employees and members of the public. The law sets forth technical standards

for EIT accessibility, ensuring that individuals with disabilities have comparable access to and use of information and data as individuals without disabilities;

- *ITA Legge Stanca [22] 2004 and updated in 2010*: The Legge Stanca, officially known as Law No. 4 of January 9, 2004, and subsequently updated in 2010, mandates that public administrations in Italy ensure that their websites and digital services are accessible to all citizens, including those with disabilities. The law aims to eliminate barriers to digital access and imposes penalties on non-compliant entities, thereby promoting equal access to public information and services;
- *EUROPE WCAG 1.0 e 2.0 (Last version WCAG 2.2 (October 5th 2023))*: The WCAG are a set of guidelines developed by the World Wide Web Consortium (W3C) to ensure that web content is accessible to people with disabilities. WCAG 1.0 was published in 1999, followed by WCAG 2.0 in 2008, and the latest version, WCAG 2.2, was released on October 5th, 2023. These guidelines provide a framework for making web content more perceivable, operable, understandable, and robust for users with diverse disabilities;
- *AGID (Agenzia per l'Italia Digitale) - 2012 [3]*: It oversees digital accessibility in Italy, monitoring compliance and enforcing accessibility standards across public and private sectors. The agency's initiatives include the appointment of a digital ombudsman in 2017 to address accessibility complaints and promote adherence to guidelines. AGID's efforts aim to ensure that digital services and information are accessible to all individuals, contributing to a more inclusive digital society.

1.1 Related Works

The first analyzed paper [25] is the one conducted by Aline Grazielle Silva Reis, Michael Cristian Nepomuceno Carvalho, Felipe Silva Dias, and André Pimenta Freire “Accessibility and usability problems encountered on websites and applications in mobile devices by blind and normal-vision users” that tests vision and blind users, which have the aim to collect empirical data of the problems encountered by visually disabled users on mobile apps and websites. During the experiment 10 participants (6 blind users and 4 normal-vision users) can use a Motorola MotoG with Android as operating system and the TalkBack Screen Reader or an iPhone 5c with iOS as operating system and the VoiceOver Screen Reader. They test the accessibility problems in 4 selected services (2 commercial and 2 governmental) considering that not all users performed tasks on all systems. The study found a total of 514 problems encountered by users. Of these, 409 problems (79%) are experienced by users with visual disabilities. No significant evidence was found in the average number of problems encountered in websites and apps when comparing governmental and commercial websites and apps. To remark the difficulty of a person with visual disability, the study conducted an experiment in which it analysed 91 tasks performed by all users, visual and blind users. The result is that blind users had lower success rate, with 19% completion on websites and 33.3% on applications, while normal-vision users have completely different percentages with 78.6% of successful tasks on websites and 91.67% on applications. Thanks to the experiment, the study also describes five principal categories of problems encountered by blind users and not encountered by normal-vision users:

- Inadequate feedback (controls, forms, and functionality): Insufficient feedback to inform that an action had an effect;
- No alternative to an image: Absence of alternative to an image, such as alternative text to an image;
- Sequence of interaction not clear: Users encountered problems with functionality in which they could not understand the sequence of interaction necessary to use the functionality;
- Lack of identification of buttons: Absence of identification to a button on apps or websites;
- Inference of functionality that does not exist: Users encountered a lot of problems when they tried to fill the form. Users identified the labels of the form elements and tried

to select the text thinking it was the functionality. The cause of the problem is the use of form elements such as input text field, radio button, and select element with no appropriate association.

In an other paper [35] conducted by Shahid Hussain Sher Badshah Arif Ali Khan and Bilal Khan “What users really think about the usability of smartphone applications: diversity based empirical investigation”, [International Organization for Standardization \(ISO\) 9241-11](#) defines usability as "the extent to which a system, product or service can be used by a specified users to achieve specified goal with effectiveness, efficiency, and satisfaction in a specific context of use". ISO 9241-20 states accessibility as the "usability of a product, service, environment, or facility by people within the widest range of capabilities". In the research *Thatcher* suggests that accessibility is a subset of usability and proposed that accessibility issues are distinct types of usability issues. Hence, usability is a subset of universal usability. Universal accessibility addresses three key challenges:

- Technology variety: wide range of hardware, software and network access;
- User diversity;
- Gaps in user knowledge.

Usability testing is a method used to assess system accessibility and usability by testing it with a representative number of end-users. *Nielsen* said that testing with five users lets you discover many usability issues as you would find by using additional test participants. The study is on two android pre-installed apps and uses a [Systematic Literature Review \(SLR\)](#). A SLR is a form of secondary study that uses explicit and rigorous methods to identify, analyse, and explore all available evidence associated to the research questions and area of interest. Results are composed in terms of usability which is composed by:

- Efficiency: It is measured in terms of time taken to complete a task, i.e. the time spends by a participant to perform a task successfully;
- Effectiveness: It means that the extent to which a user successfully performs a task;
- Satisfaction: It is the behaviour of the users toward using an app.

The results showed that 27 usability and accessibility problems were encountered by different users on both apps. As other articles the total number of problems encountered from users with disabilities is three times higher than other groups of participants and this is very important to consider. The study found the following problems as universal accessibility issues:

- Not clickable items;
- Home button not available;
- Images overlapped on text;
- Innapropriate content;
- Similar colors;
- No message for avoiding or correcting errors;
- No undo option (recovery);
- Poorly designed system feedback;
- No help facility available when an error occurred;
- Absence of instruction that provide useful user guidance.

Hence the usability and accessibility test results show that existing smartphone apps failed to meet the requirements of different types of users. SLR resulted in a huge number of usability guidelines, mapped in eleven categories. These guidelines can be used by developers and designers to prevent usability and accessibility problems in smartphone apps.

An article [8] conducted by Helen Petrie Christopher Power Andrè Freire and David Swallow “Guidelines are only half of the story: accessibility problems encountered by blind users on the web” describes accessibility found by blind users and it says that recent studies show that the web is becoming less accessible to people with disabilities over time. It classifies the following categories of problems:

- Content found in pages where not expected by users;
- Content not found in pages where expected by users;
- Pages too slow to load;
- No alternative to document format (e.g. PDF);
- Information architecture too complex (e.g. too many steps to find pages);
- Organization of content is inconsistent with web conventions/common sense;
- Broken links;
- Functionality does not work (as expected);

- Expected functionality not present;
- Irrelevant content before task content;
- Users cannot make sense of content;
- No/insufficient feedback to inform that actions has had an effect.

With other instruemnts it found also other problems:

- Language too complicated for perceive target audience;
- Link destination not clear;
- No enhancements to multimedia content;
- Meaning in content is lost or modified due to transformations;
- No alternative to information presented in tables;
- Heading structure violated.

Chapter 2

Accessibility

[World Wide Web Consortium \(W3C\)](#) [44] is an international consortium where Member organizations, a full-time staff, and developers work together to define web standards. W3C primarily pursues its mission through the creation of web standards and guidelines designed to ensure current and long-term growth for the web. It aims to spread the culture of Internet accessibility. [Web Content Accessibility Guidelines \(WCAG\)](#) [45] was developed through the W3C process in cooperation with individuals and organizations around the world, with a goal of providing a single shared standard for web content accessibility that meets the needs of individuals, organizations, and governments internationally. The WCAG documents explain how to make web content more accessible to people, focusing on people with disabilities. WCAG is intended for web content developers, web authoring tool developers, web accessibility evaluation tool developers and others who want or need a standard for web accessibility, including mobile accessibility. The evolution of WCAG is the following:

- WCAG 1.0 was defined in 1999 and organized into 14 principles. This was the first version of the Web Content Accessibility Guidelines for those developing websites and the guidelines were confusing and had problems encountered by users;
- A more precise and technologically flexible version was WCAG 2.0 [46]. WCAG 2.0 was defined in 2008 and organized with 12 guidelines. It delineates guidelines into three levels of compliance, 'A', 'AA', and 'AAA';
- WCAG for mobile introduced in 2015;
- WCAG 2.1 was defined in 2018 and updated in September 2023 adding 1 guideline and 17 success criteria;

- WCAG 2.2 [47] was defined on October 5th, 2023 and is the new version. It adds 9 success criteria.

2.1 WCAG

Web Content Accessibility Guidelines (WCAG) explain the rules and the advices to make web content accessible. The last version is WCAG 2.2 which has 13 guidelines. These guidelines are organised under four principles [41]:

1. Perceivable;
2. Operable;
3. Understandable;
4. Robust.

2.1.1 Principles

In WCAG, success criteria and guidelines are organised under the four principles seen above that provide the foundation for web accessibility. If any of these are not provided, users, especially with disabilities, will not be able to use and navigate the web. This is important to allow anyone to use and access web content.

2.1.1.1 Perceivable

The first principle is *Perceivable* [33]. It requires that user interfaces components and information must be delivered to users in ways they can perceive. Information can't be invisible to all their senses. When information is communicated entirely through one sensory input channel such as sight or hearing perceptual barriers occur. One of the biggest barriers to perceivability is content that is only available in visual or audio format. Perceptual barriers also occur when information is updated too quickly or slowly for people to read. To ensure that all users can benefit from your website or application, your content must be easily convertible between different formats, or it must include alternatives that are functionally equivalent. Assistive technologies such as screen readers and braille displays are essential to perceivability. Many people with disabilities interact with the web using these technologies, and therefore rely on your website or application to be compatible with them. In most cases, the key to perceivability is converting non-textual content into text, which can then

be processed by the assistive technology of the user. Other perceivability concerns include the ability to resize text and the use of high contrast color schemes , so that people with low vision can successfully use and navigate your content. Hence it is necessary to pay particular attention to colour or size.

2.1.1.2 Operable

The second principle is *Operable* [31]. It requires that users must be able to operate the interface. The interface cannot require interaction that a user cannot perform. There are four "operable" guidelines to follow:

1. Make it possible to perform all tasks with a keyboard instead of a mouse or touches;
2. Give users enough time to perform all tasks;
3. Avoid information that flashes or flickers, as it may trigger seizures;
4. Make it possible for users to navigate, find content, and figure out where they are.

Users should also be given enough time to perceive and use your site's content. For example, keyboards are important for users with motor disabilities, who may not have the fine motor skills required to operate a mouse or touch. Another consideration is navigation, in fact, your website or application should be easy to navigate as possible. It should use clear titles, subtitles and headings to make clear the context. When users are pressing the tab key to redirect focus to elements, the order in which elements receive focus should make logical sense. To avoid seizures in users with particular pathologies, as epilepsy, your site or application should not have content that flashes more than three times per second.

2.1.1.3 Understandable

The third principle is *Understandable* [39]. It requires that users must be able to understand the information as well as the operation of the user interface. The content or operation cannot be beyond their understanding. This principle is organised around 3 considerations:

1. People must be able to read it. In fact, web content should identifying the languages that text is written in as well as any unusual words or abbreviations. Hence, the content must be "*readable*" for the user;
2. The site must behave in ways that people can predict. When content is in a predictable order, people with disabilities or with particular pathologies can successfully use and navigate the site or application. The content is "predictable";

3. The site must be designed to help people avoid mistakes. Assistive technologies can make mistakes when entering information. Hence websites should provide better error messages and help prevent errors whenever possible. The site provide "input assistance".

Starting from these considerations we found the same division in WCAG. Understandable content can be comprehended and read by users without too much effort. Main barriers of understandability refers and is important for three types of people with disabilities:

1. People with visual disabilities who rely on screen readers and braille displays to understand the text of a website;
2. People with learning disabilities who have difficulties with complex and long sentences and advanced vocabulary words;
3. People with cognitive disabilities who have difficulties focusing on long paragraphs of text.

For a website, usually, defining unusual expressions and abbreviations is better than redirecting users to a dedicated glossary.

2.1.1.4 Robust

The last principle is *Robust* [34]. It requires that users must be able to access the content as technologies advance. As technologies and user agents evolve, the content should remain accessible. Hence, content should be understandable both by the users and by assistive technologies such as screen readers. With only three success criteria, robustness is the shortest of the four WCAG principles. During the creation of a web page, errors tend to creep into the code. In fact, nowadays mistakes and errors are almost inevitable. These errors almost always affect the appearance and functionality of the page. The effect may be minor, such as, when the formatting is slightly off kilter, or major, for instance, when the page does not display at all. Robustness refers specifically to web content that is compatible with a variety of "user agents": browsers, assistive technologies, and other means of accessing web content. People with disabilities may view and interact with your website through any number of browsers: Google Chrome, Mozilla Firefox, Safari, Microsoft Edge and more. In addition, they may rely on different types of assistive technologies. We can consider a robust web page that:

- Displays content as the author intends;
- Functions as the author intends;

- Is compatible with current and future browsers, web-enabled devices, and assistive technologies.

To conform robustness principle, web authors are required to avoid specific kinds of coding errors. Developers can use automated validation tools to aid in testing.

2.1.2 Guidelines

Under the principles seen above there are the guidelines defined by WCAG. Guidelines provide the basic goals that authors should work toward in order to make content more accessible to users with different disabilities. Guidelines are not testable, but provide the framework and overall objectives to help authors understand the success criteria and better implement the techniques. The last version of WCAG (WCAG 2.2) has 13 guidelines to follow.

2.1.3 Success criteria

In WCAG, for each guideline there are testable success criteria [38]. They are used where requirements and conformance testing are necessary [10]. All success criteria must be important access issues for people with disabilities that address problems beyond the usability problems that might be faced by all users. We also have to consider that all success criteria must be testable, otherwise we can't determine whether a page meets the success criteria. For the success criteria we have three different levels of priority:

- **Priority 1:** The developer must comply with this control point, under penalty of precluding one or more categories of users from accessing the information. This is a basic requirement;
- **Priority 2:** The developer should comply with this checkpoint, otherwise one or more categories of users will find difficulties in accessing the information. This requirement removes significant barriers;
- **Priority 3:** The developer can take this checkpoint into consideration, otherwise one or more categories of users will be somehow hindered from accessing the information. This requirement improves access to web documents.

Considering these three priority levels we have three different levels of conformance [9]:

1. **Conformance level 'A' (lowest):** conformance with all Priority 1 checkpoints;
2. **Conformance level 'AA':** conformance with all Priority 1 and 2 checkpoints;

3. **Conformance level 'AAA' (highest):** conformance with all Priority 1, 2 and 3 checkpoints.

In addition, we can consider the following consideration to set the level of conformance:

- Whether the success criteria is essential;
- Whether it is possible to satisfy the success criteria for all web sites and types of content that the success criteria would apply to;
- Whether the success criteria requires skills that could reasonably be achieved by the content creators;
- Whether the success criteria would impose limits on the "look & feel" and/or function of the web page;
- Whether there are no workarounds if the success criteria is not met.

2.1.4 WCAG 2.2

WCAG 2.2 aims to provide a wide range of recommendations for making web content more accessible. It is a new document provided by W3C on 5th october 2023 which extends WCAG 2.1 published in june 2018. WCAG 2.2 was published with the goal to continue and pursue the work of WCAG 2.1. The goal is to improve accessibility guidance for three major groups: users with cognitive or learning disabilities, users with low vision, and users with disabilities who use mobile devices. It has 13 guidelines and 86 success criteria. In particular, WCAG 2.2 adds 9 success criteria and removes one (success criteria 4.1.1 - Parsing). Now we see the guidelines and success criteria provided by WCAG 2.2 and then we explain another document of W3C which explains how WCAG apply to mobile and we try to understand if this document is complete and when we found some absence we analyse if WCAG 2.2 is sufficient for mobile applications or not.

The structure and the guidelines of the document that we found also in the official [site](#) is the following:

1. **Percivable**

User interface and information components must be presentable to users in ways they can perceive. This includes:

- **Guideline 1.1 Text Alternatives**
Providing text alternatives for non-text content ensure that it can be changed into

forms such as braille, symbols, large print, speech, or simpler language to meet user needs.

- Success Criterion 1.1.1 Non-text Content (level A)

All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below.

- Guideline 1.2 Time-based Media

Providing alternatives for time-based media to ensure accessibility for users.

- Success Criterion 1.2.1 Audio-only and Video-only (Prerecorded) (level A)

For prerecorded audio-only and prerecorded video-only media an alternative for time-based media is provided and either an alternative for time-based media or an audio track is provided except when the audio or video is a media alternative for text and is clearly labeled as such.

- Success Criterion 1.2.2 Captions (Prerecorded) (level A)

Providing captions for all prerecorded audio content in synchronized media, except when the media is a media alternative for text and is clearly labeled as such.

- Success Criterion 1.2.3 Audio Description or Media Alternative (Prerecorded) (level A)

Providing an alternative version for time-based media or an audio description for prerecorded video content is offered in synchronized media, unless the media serves as an alternative format for textual content and is explicitly labeled as such.

- Success Criterion 1.2.4 Captions (Live) (level AA)

Providing captions for all live audio content within synchronized media.

- Success Criterion 1.2.5 Audio Description (Prerecorded) (level AA)

Providing audio description for all prerecorded video content within synchronized media.

- Success Criterion 1.2.6 Sign Language (Prerecorded) (level AAA)

Sign language interpretation is available for all prerecorded audio content presented in synchronized media.

- Success Criterion 1.2.7 Extended Audio Description (Prerecorded) (level AAA)

When pauses in foreground audio are insufficient for conveying the meaning

of the video through audio descriptions, extended audio descriptions are provided for all prerecorded video content in synchronized media.

– Success Criterion 1.2.8 Media Alternative (Prerecorded) (level AAA)

An alternative version of time-based media is provided for all prerecorded synchronized media and for all prerecorded video-only media.

– Success Criterion 1.2.9 Audio-only (Live) (level AAA)

An alternative version of time-based media that provides the same information for live audio-only content is available.

• Guideline 1.3 Adaptable

Develop content that can be presented in different ways, for example simpler layout, without compromising information or structure.

– Success Criterion 1.3.1 Info and Relationships (level A)

Information, structure, and relationships conveyed through presentation can be determined or are available in text.

– Success Criterion 1.3.2 Meaningful Sequence (level A)

When the sequence of content presentation affects its meaning, a correct reading order can be determinable.

– Success Criterion 1.3.3 Sensory Characteristics (level A)

Instructions for understanding and operating content do not rely solely on sensory characteristics of components such as shape, color, size, visual location, orientation, or sound.

– Success Criterion 1.3.4 Orientation (level AA)

Content must not restrict to a single display orientation for which concerns its view and operation, such as portrait or landscape, unless a specific display orientation is fundamental.

– Success Criterion 1.3.5 Identify Input Purpose (level AA)

The purpose of each input field that collects user information should be clearly identifiable. This is necessary when the input field corresponds to purposes specified in the input purposes for user interface components section, and the content must be implemented using technologies that support recognizing the intended meaning of the input data.

- Success Criterion 1.3.6 Identify Purpose (level AAA)

In content implemented using markup languages, the purpose of user interface components, icons, and regions can be identifiable.

- Guideline 1.4 Distinguishable

Make it easier for users to see and hear content including separating foreground from background.

- Success Criterion 1.4.1 Use of Color (level A)

Color is not the sole visual method used to convey information, indicate an action, prompt a response, or differentiate a visual element.

- Success Criterion 1.4.2 Audio Control (level A)

If any audio on a web page plays automatically for longer than 3 seconds, there must be a mechanism provided to either pause or stop the audio, or to independently control the audio volume separate from the overall system volume level

- Success Criterion 1.4.3 Contrast (Minimum) (level AA)

The visual presentation of text and images of text should have a contrast ratio of at least 4.5:1, except for instances of large text, incidental text, and logotypes.

- Success Criterion 1.4.4 Resize Text (level AA)

Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality.

- Success Criterion 1.4.5 Images of Text (level AA)

Where feasible with the technologies employed, textual content is utilized to convey information rather than images of text, except in cases where the presentation needs to be customizable or where the use of images of text is essential.

- Success Criterion 1.4.6 Contrast (Enhanced) (level AAA)

The visual presentation of text and images of text has a contrast ratio of at least 7:1, except for the following: large text, incidental and logotypes.

- Success Criterion 1.4.7 Low or No Background Audio (level AAA)

For prerecorded audio-only content that (1) contains primarily speech in the foreground, (2) is not an audio CAPTCHA or audio logo, and (3) is not

vocalization intended to be primarily musical expression such as singing or rapping, at least one of the following is true: no background, turn off and 20 dB.

– Success Criterion 1.4.8 Visual Presentation (level AAA)

For the visual presentation of blocks of text, a mechanism is available to achieve the following:

- * Foreground and background colors can be selected by the user;
- * Width is no more than 80 characters or glyphs (40 if CJK¹);
- * Text is not justified (aligned to both the left and the right margins);
- * Line spacing (leading) is at least space-and-a-half within paragraphs, and paragraph spacing is at least 1.5 times larger than the line spacing;
- * Text can be resized without assistive technology up to 200 percent in a way that does not require the user to scroll horizontally to read a line of text on a full-screen window.

– Success Criterion 1.4.9 Images of Text (No Exception) (level AAA)

Images of text are utilized solely for decorative purposes or when a specific presentation of text is crucial to conveying information.

– Success Criterion 1.4.10 Reflow (level AA)

Content can be presented in a way that preserves all information and functionality, without necessitating scrolling in two dimensions for:

- * Vertical scrolling content at a width equivalent to 320 CSS pixels;
- * Horizontal scrolling content at a height equivalent to 256 CSS pixels.

Except for parts of the content which require two-dimensional layout for usage or meaning.

– Success Criterion 1.4.11 Non-text Contrast (level AA)

The visual presentation of the following have a contrast ratio of at least 3:1 against adjacent color(s): user interface components and graphical objects.

– Success Criterion 1.4.12 Text Spacing (level AA)

In content implemented using markup languages that support the following text style properties, no loss of content or functionality should occur when all of the following properties are set, without changing any other style property:

¹CJK characters is a collective term for the Chinese, Japanese, and Korean languages

- * Line height (line spacing) to at least 1.5 times the font size;
 - * Spacing following paragraphs to at least 2 times the font size;
 - * Letter spacing (tracking) to at least 0.12 times the font size;
 - * Word spacing to at least 0.16 times the font size.
- Success Criterion 1.4.13 Content on Hover or Focus (level AA)
- When additional content becomes visible and then hidden upon receiving and removing pointer hover or keyboard focus, the following conditions are true: dismissible, hoverable and persistent.

2. Operable

User interface components and navigation must be operable.

- Guideline 2.1 Keyboard Accessible

Make all functionality available from a keyboard.

- Success Criterion 2.1.1 Keyboard (level A)

Every aspect of the content should be operable through a keyboard interface without requiring specific timings for individual keystrokes, except in cases where the underlying function requires input that depends on the path of the user's movement rather than just its endpoints.
- Success Criterion 2.1.2 No Keyboard Trap (level A)

Even if not used in the mobile context, if keyboard focus can be shifted to a component of the page using a keyboard interface, then it must be possible to move focus away from that component using only a keyboard interface. If it requires more than unmodified arrow or tab keys or other standard exit methods, users are informed about the method for shifting focus away.
- Success Criterion 2.1.3 Keyboard (No Exception) (level AAA)

All functionality of the content is operable through a keyboard interface without relying specific timings for individual keystrokes.
- Success Criterion 2.1.4 Character Key Shortcuts (level A)

If a keyboard shortcut is implemented in content using only letter (including upper- and lower-case letters), punctuation, number, or symbol characters, then at least one of the following is true: the shortcut can be turn off, remap and active only on focus.

- Guideline 2.2 Enough Time

Provide users enough time to read and use content.

- Success Criterion 2.2.1 Timing Adjustable (level A)

For each time limit imposed by the content, at least one of the following must be available: turning off the time limit, adjusting it, extending it, applying a real-time exception, granting an essential exception, or allowing a 20-hour exception.

- Success Criterion 2.2.2 Pause, Stop, Hide (level A)

For moving, blinking, scrolling, or auto-updating information, all of the following are true:

- * For any moving, blinking or scrolling information that (1) starts automatically, (2) lasts more than five seconds, and (3) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it unless the movement, blinking, or scrolling is part of an activity where it is essential;

- * For any auto-updating information that (1) starts automatically and (2) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it or to control the frequency of the update unless the auto-updating is part of an activity where it is essential.

- Success Criterion 2.2.3 No Timing (level AAA)

Timing is not an essential part of the event or activity presented by the content, except for non-interactive synchronized media and real-time events.

- Success Criterion 2.2.4 Interruptions (level AAA)

Interruptions can be deferred or suppressed by the user, except for interruptions related to emergencies.

- Success Criterion 2.2.5 Re-authenticating (level AAA)

When an authenticated session expires, users can resume their activity without losing data after re-authenticating.

- Success Criterion 2.2.6 Timeouts (level AAA)

Users are alerted about the duration of any user inactivity that might lead to data loss, unless the data is preserved for more than 20 hours without user interaction.

- Guideline 2.3 Seizures and Physical Reactions

Do not design content in a way that is known to cause seizures or physical reactions.

- Success Criterion 2.3.1 Three Flashes or Below Threshold (level A)

Web pages do not include any content that flashes more than three times within any one-second period, unless the flash falls below the general flash and red flash thresholds.

- Success Criterion 2.3.2 Three Flashes (level AAA)

Web pages do not include any content that flashes more than three times within any one second period.

- Success Criterion 2.3.3 Animation from Interactions (level AAA)

Motion animation triggered by interaction can be disabled, unless the animation is essential to the functionality or the information being conveyed.

- Guideline 2.4 Navigable

Provide ways to help users navigate, find content, and determine where they are.

- Success Criterion 2.4.1 Bypass Blocks (level A)

A mechanism is available to bypass blocks of content that are repeated on multiple web pages.

- Success Criterion 2.4.2 Page Titled (level A)

Web pages must have titles that accurately describe their topic or purpose.

- Success Criterion 2.4.3 Focus Order (level A)

If a web page supports sequential navigation and the order of the navigation sequences affect meaning or operation, focusable components should receive focus in an order that preserves meaning and operability.

- Success Criterion 2.4.4 Link Purpose (In Context) (level A)

The purpose of each link should be ascertainable from the link text alone or from the link text along with its surrounding context, except in cases where the purpose of the link would be unclear to most users.

- Success Criterion 2.4.5 Multiple Ways (level AA)

More than one way is available to locate a web page within a set of web pages except where the web page is the result of, or a step in, a process.

- Success Criterion 2.4.6 Headings and Labels (level AA)

Headings and labels describe topic or purpose.

- Success Criterion 2.4.7 Focus Visible (level AA)
Any keyboard-operable user interface must have a mode of operation where the keyboard focus indicator is visibly present.
- Success Criterion 2.4.8 Location (level AAA)
Information regarding the user’s location within a set of web pages must be accessible.
- Success Criterion 2.4.9 Link Purpose (Link Only) (level AAA)
A mechanism is available to allow the purpose of each link to be identified from link text alone, except where the purpose of the link would be ambiguous to users in general.
- Success Criterion 2.4.10 Section Headings (level AAA)
Section headings are used to organize the content.
- Success Criterion 2.4.11 Focus Not Obscured (Minimum) (level AA) [NEW with WCAG 2.2]
When a user interface component receives keyboard focus, the component is not entirely hidden due to author-created content.
- Success Criterion 2.4.12 Focus Not Obscured (Enhanced) (level AAA) [NEW with WCAG 2.2]
When a user interface component receives keyboard focus, it should not be completely obscured by author-created content.
- Success Criterion 2.4.13 Focus Appearance (level AAA) [NEW with WCAG 2.2]
When the keyboard focus indicator is visible, it should adhere to the following criteria:
 - * The indicator’s area is at least as large as the area of a 2 CSS pixel thick perimeter of the unfocused component or sub-component;
 - * There is a contrast ratio of at least 3:1 between the same pixels in the focused and unfocused states of the indicator.
- Guideline 2.5 Input Modalities
Make it easier for users to operate functionality through various inputs beyond keyboard.
 - Success Criterion 2.5.1 Pointer Gestures (level A)
All functionality that relies on multipoint or path-based gestures for operation

must also be operable using a single pointer without requiring a path-based gesture, unless the use of multipoint or path-based gestures is essential.

– Success Criterion 2.5.2 Pointer Cancellation (level A)

For functionality that can be operated using a single pointer, at least one of the following conditions is true: no down-event, abort or undo, up reversal and essential.

– Success Criterion 2.5.3 Label in Name (level A)

For user interface components with labels that include text or images of text, the name contains the text that is presented visually.

– Success Criterion 2.5.4 Motion Actuation (level A)

Functionality that can be operated by device motion or user motion can also be operated by user interface components. There should be an option to disable responsiveness to motion to prevent accidental activation, except when motion support is essential.

– Success Criterion 2.5.5 Target Size (Enhanced) (level AAA)

The size of the target for pointer inputs is at least 44 by 44 CSS pixels except when: equivalent (The target is available through an equivalent link or control on the same page that is at least 44 by 44 CSS pixels), inline, user agent control and essential.

– Success Criterion 2.5.6 Concurrent Input Mechanisms (level AAA)

Web content does not restrict use of input modalities available on a platform except where the restriction is essential, required to ensure the security of the content, or required to respect user settings.

– Success Criterion 2.5.7 Dragging Movements (level AA) [NEW with WCAG 2.2]

All functionality that relies on dragging movements for operation should also be achievable using a single pointer without the need for dragging, unless dragging is essential or the functionality is determined solely by the user agent and remains unmodified by the author.

– Success Criterion 2.5.8 Target Size (Minimum) (level AA) [NEW with WCAG 2.2]

The size of the target for pointer inputs is at least 24 by 24 CSS pixels, except where: spacing (undersized targets (those less than 24 by 24 CSS pixels) are

positioned so that if a 24 CSS pixel diameter circle is centered on the bounding box of each, the circles do not intersect another target or the circle for another undersized target), equivalent, inline, user agent control and essential.

3. Understandable

Information and the operation of the user interface must be understandable.

- Guideline 3.1 Readable

Make text content readable and understandable.

- Success Criterion 3.1.1 Language of Page (level A)

The default human language of each web page should be determinable.

- Success Criterion 3.1.2 Language of Parts (level AA)

The human language of each passage or phrase in the content should be determinable, with exceptions for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the surrounding text.

- Success Criterion 3.1.3 Unusual Words (level AAA)

A mechanism must be available to identify specific definitions of words or phrases used in an unusual or restricted manner, including idioms and jargon.

- Success Criterion 3.1.4 Abbreviations (level AAA)

A mechanism for identifying the expanded form or meaning of abbreviations is available.

- Success Criterion 3.1.5 Reading Level (level AAA)

When text necessitates reading skills beyond lower secondary education level after excluding proper names and titles, supplemental content or an alternative version that does not require skills beyond lower secondary education level should be provided.

- Success Criterion 3.1.6 Pronunciation (level AAA)

A mechanism must be available to identify the specific pronunciation of words in cases where the meaning of the words in context is ambiguous without knowing their pronunciation.

- Guideline 3.2 Predictable

Make Web pages appear and operate in predictable ways.

- Success Criterion 3.2.1 On Focus (level A)
When any user interface component gains focus, it should not initiate a change of context.
- Success Criterion 3.2.2 On Input (level A)
Modifying the setting of any user interface component should not automatically result in a change of context unless the user has been informed of this behavior prior to using the component.
- Success Criterion 3.2.3 Consistent Navigation (level AA)
Navigation mechanisms that are repeated across multiple web pages within a set maintain the same relative order each time they appear, unless the user initiates a change.
- Success Criterion 3.2.4 Consistent Identification (level AA)
Components that have the same functionality within a set of web pages are identified consistently.
- Success Criterion 3.2.5 Change on Request (level AAA)
Changes of context are initiated only by user request or a mechanism is available to turn off such changes.
- Success Criterion 3.2.6 Consistent Help (level A) [NEW with WCAG 2.2]
If a web page includes any of the following help mechanisms, and these mechanisms are repeated across multiple web pages within a set, they maintain the same order relative to other page content, unless a user-initiated change occurs: human contact details, human contact mechanism, self-help options, and fully automated contact mechanisms.
- Guideline 3.3 Input Assistance
Help users avoid and correct mistakes.
 - Success Criterion 3.3.1 Error Identification (level A)
If an input error is automatically detected, the system should identify the erroneous item and describe the error to the user in text.
 - Success Criterion 3.3.2 Labels or Instructions (level A)
Labels or instructions are provided when content requires user input.
 - Success Criterion 3.3.3 Error Suggestion (level AA)
If an input error is automatically detected and suggestions for correction are

available, they should be provided to the user, except when doing so would compromise the security or purpose of the content.

- Success Criterion 3.3.4 Error Prevention (Legal, Financial, Data) (level AA)
For web pages that cause legal commitments or financial transactions for the user to occur, that modify or delete user-controllable data in data storage systems, or that submit user test responses, at least one of the following is true: reversible, checked and confirmed.
- Success Criterion 3.3.5 Help (level AAA)
Context-sensitive help is available.
- Success Criterion 3.3.6 Error Prevention (All) (level AAA)
For Web pages that require the user to submit information, at least one of the following conditions is true: reversible, checked and confirmed.
- Success Criterion 3.3.7 Redundant Entry (level A) [NEW with WCAG 2.2]
Information previously entered by or provided to the user that is required to be entered again in the same process should either be auto-populated or made available for the user to select.
- Success Criterion 3.3.8 Accessible Authentication (Minimum) (level AA) [NEW with WCAG 2.2]
A cognitive function test (such as remembering a password or solving a puzzle) is not required for any step in an authentication process unless that step provides at least one of the following conditions: alternative, mechanism, object recognition and personal content.
- Success Criterion 3.3.9 Accessible Authentication (Enhanced) (level AAA) [NEW with WCAG 2.2]
A cognitive function test (such as remembering a password or solving a puzzle) is not required for any step in an authentication process unless that step provides at least one of the following conditions: alternative and mechanism.

4. Robust

Content must be robust enough that it can be interpreted by a wide variety of user agents, including assistive technologies.

- Guideline 4.1 Compatible

Maximize compatibility with current and future user agents, including assistive technologies.

- Success Criterion 4.1.1 Parsing (level A) (**obsolete and removed** with WCAG 2.2)

- Success Criterion 4.1.2 Name, Role, Value (level A)

For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be determined; states, properties, and values that can be set by the user can be set; and notification of changes to these items is available to user agents, including assistive technologies.

- Success Criterion 4.1.3 Status Messages (level AA)

In content implemented using markup languages, status messages should be identifiable through roles or properties so that they can be presented to the user by assistive technologies without requiring the messages to receive focus.

Now that we have well-defined the official WCAG 2.2 document, we will use it in the next sections and chapters to understand what is already there and what is missing for the mobile guidelines.

Chapter 3

MCAG

WCAG for mobile [48] denominated as "Mobile Accessibility: How WCAG 2.0 and Other W3C/WAI Guidelines Apply to Mobile" was an official document provided by W3C. It aims to pursue the accessibility of the content in the mobile world. Its principles, guidelines and success criteria can be applied to mobile web content, mobile web apps, native apps and hybrid apps using web components inside native apps.

The document was published on 26th february 2015 providing recommendations, notes and guidelines of mobile technologies including mobile web best practices and mobile web application best practices.

With the term "mobile" we can consider small handheld devices such as smartphones, wearables, for instance smart-watches and also large tablets. We have to note that the last version of WCAG, WCAG 2.2, can be applied and was relevant to both web and non-web mobile content and applications. But we know that mobile devices present a mix of accessibility issues that are different from desktop. WCAG 2.2 did not provide testable success criteria for some of the mobile-related issues that we will see. The document we analyzed refers to WCAG 2.0 which was closely related to WCAG 2.2. Now we analyse the document provide by W3C trying to understand if it is complete starting from its structure and then we integrate the principles with the success criteria in WCAG 2.1 and 2.2 and some other important best practices. The document is structured as described in the following sections.

3.1 Perceivable

1. Perceivable

For this principle WCAG 2.0 identifies the following mobile issues:

1.1. Small Screen Size

The official document precises:

Small screen size is one of the most common characteristics of mobile devices. While these screens boast exceptional resolution, their compact size imposes practical limits on how much information users can view simultaneously, especially for individuals using magnification due to low vision. Effective practices to optimize user experience on small screens include:

- Minimizing the amount of information that is displayed on each page compared to desktop/laptop versions by implementing either a dedicated mobile version or a responsive design:
 - a dedicated mobile version contains content tailored for mobile use. For example, the content may contain fewer content modules, fewer images, or focus on important mobile usage scenarios.
 - a responsive design maintains the same content, but adjusts its presentation using CSS stylesheets based on viewport width. For instance, on narrow screens the navigation menus may be hidden until the user taps a menu button.
- Providing default sizes for content and touch controls to minimize the need for users, particularly those with low vision, to zoom in and out.
- Adapting the length of link text based on the viewport width to ensure readability and usability.
- Positioning form fields below, rather than beside, their labels in portrait layout.

1.2. Zoom/Magnification

The official document precises:

Various methods exist to allow users to control content size on mobile devices with small screens. At the browser level these methods are generally available to assist a wide audience of users. At the platform level these methods are available as accessibility features to help people with cognitive disabilities and/or visual impairments.

The methods include the following:

- OS-level features
 - Adjust default text size through Display Settings. Note that system text size is often not supported by mobile browsers.

- Enable screen magnification through Accessibility Settings. Note that using this setting requires the user to pan vertically and horizontally.
- Use a magnifying lens view that follows the user’s finger, typically controlled via Accessibility Settings.
- Browser-level features (Not necessary to consider in mobile applications. It is only mentioned for a complete description)
 - Adjust the default text size of text rendered within the browser’s viewport
 - * Activate a reading mode that displays main content at a user-defined text size.
 - Utilize pinch-zoom functionality to magnify the browser’s viewport. Using this setting requires the user to pan vertically and horizontally.
 - * Some browsers may implement features that alter this type of magnification, such as re-flowing the content at the new magnification level, overriding author attempts to prevent pinch-zoom.

The WCAG 2.0 success criterion that is most related to *Zoom/Magnification* is 1.4.4 Resize Text (level AA). Success criteria 1.4.4 requires text to be resizable without assistive technology up to 200 percent. To meet this requirement content must not prevent text magnification by the user.

The following methods can be implemented to enhance accessibility on mobile devices:

- Ensure that the page’s viewport meta element does not block browser pinch zoom functionality, enabling users to zoom the page to 200%. Avoid restrictive values for user-scalable and maximum-scale attributes in this meta element. Note that full viewport zooming (without blocking pinch zoom) may require users to pan horizontally and vertically. While effective in meeting success criteria, this approach is less user-friendly compared to supporting text resizing features that reflow content to fit the user’s chosen viewport size. Best practices emphasize techniques that facilitate text resizing without necessitating horizontal panning.
- Support for system fonts that follow platform level user preferences for text size.
- Provide on-page controls to change the text size.

It is essential to note that while these methods align with WCAG success criteria,

they should not exclusively depend on accessibility features designed for specific populations with disabilities, as these are classified as assistive technologies under WCAG guidelines. For instance, a platform-level zoom feature that magnifies all platform content and support people with low vision is likely considered an assistive technology.

1.3. Contrast

The official document precises:

Mobile devices are more frequently used in diverse settings, including outdoors where glare from sunlight or other strong light sources is common. This situation emphasizes the critical importance of ensuring good contrast for all users and underscores the challenges faced by individuals with low vision when accessing content with insufficient contrast on mobile devices. The WCAG 2.0 guidelines address this issue through two success criteria:

- 1.4.3 Contrast (Minimum) (level AA) which requires a contrast of at least 4.5:1 (or 3:1 for large-scale text);
- 1.4.6 Contrast (Enhanced) (level AAA) which requires a contrast of at least 7:1 (or 4.5:1 for large-scale text).

Success criteria 1.4.3 allows for different contrast ratios for large text. Allowing different contrast ratios for larger text is useful because larger text with wider character strokes is easier to read at a lower contrast. This allows designers more leeway for contrast of larger text, which is helpful for content such as titles. The specified ratios for 18-point text or 14-point bold text under 1.4.3 were determined based on readability considerations for web pages viewed on a 15-inch monitor at 1024x768 resolution, from a distance of 24 inches. However, mobile device content is viewed on smaller screens and in different conditions, necessitating careful consideration of these allowances for reduced contrast on larger text in mobile applications.

default text sizes on mobile platforms tend to be larger than those on non-mobile devices. When determining appropriate contrast ratios, developers should apply lower ratios only when text is approximately 1.2 times bold or 1.5 times (120% bold or 150%) the default size on the platform. It is important to note that the use of text that is 1.5 times the default size on mobile platforms does not imply that the text will be readable by a person with low vision. People with low vision will likely need and use additional platform level accessibility features and assistive

technology such as increased text size and zoom features to access mobile content.

3.1.1 New Ideas - Perceivable

For the first perceivable mobile issue, *Small Screen Size*, we can add some of the WCAG 2.1 and WCAG 2.2 guidelines, to fully complete the principle. Initially we can consider guideline 1.3 "*adaptable*" which said that we have to create content that can be presented in different ways, such as a simpler layout, without losing information or structure. We have to mention also guideline 1.4 "*distinguishable*" which specifies to make easier for users to see and hear content including separating foreground from background. Success criteria and best practices that we can consider are:

- For a dedicated mobile version and a responsive design, WCAG 2.1 introduced the success criteria 1.3.4 Orientation (level AA) who emphasizes the importance of an adaptable interface.
- An other important success criteria introduced with WCAG 2.1 is 1.4.12 Text Spacing (level A) contributing to well-distributed and easily understandable text, emphasizing the need to consider text scalability.

Reading WCAG 2.1 and WCAG 2.2 guidelines, we can observe that for the second perceivable mobile issue, *Zoom/Magnification*, there are not new success criteria to consider except 1.4.4 mentioned before and the success criteria 1.4.10 from WCAG 2.1 that we will define below. However, to guarantee guidelines 1.3 "*adaptable*" and 1.4 "*distinguishable*" there are some best practices to mention:

- Information, structure and relationships conveyed through presentation must remain consistent and easily understandable even when enlarged. For doing so we have to respect the success criteria 1.3.1 Info and Relationships (level A).
- An other point to consider is the success criteria 1.4.10 Reflow (level AA) introduced with WCAG 2.1, and as before, we have to ensure that the application supports browser zoom technologies, such as pinch-zoom, without compromising usability or content layout.

For the third and the last perceivable mobile issue, *Contrast*, we can consider all the guidelines and success criteria regarding colors mentioned in WCAG 2.2 but also present in WCAG 2.0.

Now we can add some of the WCAG 2.1 and WCAG 2.2 guidelines, to fully complete *Contrast* principle. The main success criteria to consider is 1.4.11 Non-text Contrast (level AA) introduced with WCAG 2.1 which specifies that the visual presentation have a contrast ratio of at least 3:1 against adjacent colors.

An other important success criteria to consider is 2.4.13 Focus Appearance (level AAA) introduced with WCAG 2.2 which specifies that when the keyboard focus indicator is visible, an area of the focus indicator has a contrast ratio of at least 3:1 between the same pixels in the focused and unfocused states.

We have to note that contrast calculations can be based on colors defined within the technology and for this reason considering colors with the right contrast is very important. Furthermore, pixels modified by user agent resolution enhancements and anti-aliasing can be ignored. Other best practices that we have to mention are the following:

- If it is possible to add an automatic contrast adaptation based on the device's brightness settings;
- To include options for light and dark themes selected by the users.

3.2 Operable

2. Operable

For this principle WCAG 2.0 identifies the following mobile issues:

2.1. Keyboard Control for Touchscreen Devices

The official document precises:

Mobile device design has evolved away from built-in physical keyboards (e.g. fixed, slide-out) towards devices that maximize touchscreen area and display an on-screen keyboard only when the user has selected a user interface control that accepts text input (e.g. a textbox).

Despite this shift, ensuring keyboard accessibility remains crucial. Most major mobile operating systems now support keyboard interfaces, enabling users to connect external physical keyboards (e.g., Bluetooth, USB On-The-Go) or utilize alternative on-screen keyboards (e.g., scanning keyboards).

Supporting these keyboard interfaces benefits several groups with disabilities:

- Individuals with visual impairments who can benefit from some characteristics of physical keyboards over touchscreen keyboards (e.g. clearly separated

keys, key nibs and more predictable key layouts).

- People with dexterity or mobility impairments benefit from keyboards designed to minimize accidental keystrokes through features such as differently shaped and spaced keys, as well as guarded layouts. They also benefit from specialized input methods that simulate keyboard interactions.
- Users who can be confused by the dynamic nature of onscreen keyboards and who can benefit from the consistency of a physical keyboard.

Several WCAG 2.0 success criteria are relevant to effective keyboard control:

- 2.1.1 Keyboard (level A)
- 2.1.2 No Keyboard Trap (level A)
- 2.4.3 Focus Order (level A)
- 2.4.7 Focus Visible (level AA)

These success criteria allow that all functionality of the content is operable through a keyboard interface and focusable components receive focus in an order that preserves meaning and operability where the keyboard focus indicator is visible.

2.2. Touch Target Size and Spacing

The official document precises:

The increasing resolution of mobile device allows for different interactive elements to be displayed simultaneously on small screens. However, these elements must be adequately sized and spaced to ensure accurate user interaction via touch input.

According to best practices for touch target size include the following:

- Ensuring that touch targets are at least 9 mm high by 9 mm wide.
- Ensuring that touch targets close to the minimum size are surrounded by a small amount of inactive space.

It is crucial to note that this size is not dependent on the screen size, device or resolution. Screen magnification should not need to be used to obtain this size, because magnifying the screen often introduces the need to pan horizontally as well as vertically, which can decrease usability.

2.3. Touchscreen Gestures

The official document precises:

Many mobile devices are designed primarily for interaction through touchscreen gestures. These gestures can range from simple, single-finger taps to complex actions involving multiple fingers and drawn shapes.

While some but not all mobile operating systems provide work-around features that allow users simulate complex gestures with simpler ones using an onscreen menu.

Best practices for implementing touchscreen gestures include:

- Gestures within applications should be as easy as possible to execute. This consideration is crucial for users relying on screen readers, as they substitute direct touch manipulation with a two-step process of focusing and activating elements. Furthermore, it is also a challenge for users with dexterity and/or motor impairments or people who rely on head pointers or a stylus where multi-touch gestures may be difficult or impossible to perform. Often, interface designers have different options for how to implement an action. Widgets that necessitate complex gestures can pose difficulties or be impractical for users relying on screen readers. Typically, design alternatives are available to allow adjustments to settings through simple tap or swipe gestures.
- Activating elements via the mouseup or touchend event. Using the mouseup or touchend event to trigger actions helps prevent unintentional actions during touch and mouse interaction. Mouse users clicking on actionable elements like links, buttons, or submit inputs should have the ability to move the cursor outside the element to prevent triggering the event accidentally. This flexibility enables users to change their minds without being forced to commit to an action. In the same way, elements accessed via touch interaction should generally trigger an event (e.g. navigation, submits) only when the touchend event is fired (i.e. when all of the following are true: the user has lifted the finger off the screen, the last position of the finger is inside the actionable element, and the last position of the finger equals the position at touchstart).

Another common issue with touchscreen gestures is that they might lack onscreen indicators that remind people how and when to use them. For instance, gestures like swiping from the left side of the screen to open a menu may not be discoverable without visible indicators or guidance.

2.4. Device Manipulation Gestures

The official document precises:

In addition to touchscreen gestures, many mobile operating systems often include options for developers to integrate device manipulation gestures, such as shaking or tilting, into applications. While these gestures can enhance user interaction by

enabling innovative interfaces, they present challenges for users who have difficulty physically handling a mobile device.

Some (but not all) mobile operating systems provide work-around features that let the user simulate device shakes, tilts, etc. from an onscreen menu.

Therefore, even when device manipulation gestures are provided, developers should still provide touch and keyboard operable alternative control options. The most related WCAG 2.0 success criteria is 2.1.1 Keyboard (level A). Another concern with device manipulation gestures is the potential lack of onscreen indicators to guide users on how and when to use them.

2.5. Placing buttons where they are easy to access

The official document precises:

Mobile sites and applications should position interactive elements where they can be easily reached when the device is held in different positions.

When designing mobile content, developers often aim to optimize usability for one-handed operation, which can benefit users with disabilities who may have limited hand mobility.

However, it's important to consider that what is ergonomic for some users (e.g., assuming thumb reach) may pose challenges for others based on handedness or range of motion. Therefore, flexibility in button placement is essential to accommodate diverse user needs.

Some (but not all) mobile operating systems provide work-around features that let the user temporarily shift the display downwards or sideways to facilitate one-handed operation.

3.2.1 New Ideas - Operable

For the first operable mobile issue, *Keyboard Control for Touchscreen Devices*, we can add some of the WCAG 2.1 and WCAG 2.2 guidelines, to fully complete the principle. Initially we can consider guideline 1.3 "*adaptable*" which said that we have to create content that can be presented in different ways, such as a simpler layout, without losing information or structure. Then we have to consider also guideline 2.4 "*navigable*" which provides ways to help users navigate, find content, and determine where they are. Finally, we use guideline 2.5 "*input modalities*" which makes it easier for users to operate functionality through various inputs beyond keyboard. Starting from WCAG 2.1, new useful success criteria introduced to help in understanding this principle are the following:

- WCAG 2.1 introduced the success criteria 1.3.5 Identify Input Purpose (level AA) which underlines the importance of clearly identifying the purpose of an input area, ensuring that input areas are clearly labeled and described. This improves the experience for users who navigate using the keyboard.
- The other success criteria introduced with WCAG 2.1 is 2.5.5 Target Size (Enhanced) (level AAA) which requires interactive elements to be sufficiently large for easy selection via touch input or pointer. This point focuses on touch accessibility but an adequate target size can also facilitate navigation via keyboard.

Also WCAG 2.2 introduced important success criteria for this principle:

- Firstly, success criterias to consider introduced with WCAG 2.2 are 2.4.11 Focus Not Obscured (Minimum) (level AA) and 2.4.12 Focus Not Obscured (Enhanced) (level AAA). This two success criterias state that when a user interface component receives keyboard focus, no part of the component is hidden by author-created content. Hence, it is important to ensure that focused elements are visible to users navigating via keyboard.
- An other important success criteria introduced with WCAG 2.2 is 2.4.13 Focus Appearance (level AAA) which requires that when the keyboard focus indicator is visible, an area of the indicator meets specific size and contrast requirements. This is important because if the focus indicator is visible and has sufficient contrast it can improve the experience of users using keyboards.
- Finally, the last success criteria introduced with WCAG 2.2 to note is 2.5.8 Target Size (Minimum) (level AA) which said that the size of the target for pointer inputs be at least 24 by 24 CSS pixels. As before, an adequate target size can facilitate user selection via keyboard or other alternative input devices.

For the second operable mobile issue, *Touch Target Size and Spacing*, as before, we can add some new success criteria from WCAG 2.1 and WCAG 2.2 to complete the principle. Again we consider guidelines 1.3 "*adatable*", 2.4 "*navigable*" and 2.5 "*input modalities*" and the new success criterias to mention are the following:

- WCAG 2.1 introduced the success criteria 1.3.4 Orientation (level AA) which said us that content does not restrict its view and operation to a single display orientation. Hence, if touch targets are well-sized and spaced correctly, they can mantain their usability even when the screen is rotated. This is mandatory when we talk about

websites, instead in applications when rotation is enabled, then the interface must fit well, otherwise it is necessary to lock it to maintain the usability.

- WCAG 2.2 introduced success criteria 2.4.13 Focus Appearance (level AAA) which states that when the keyboard focus indicator is visible, an area of the focus indicator meets certain size and contrast requirements. Even if this success criteria aimed at improving keyboard accessibility, it can also improve the usability of touch targets.
- WCAG 2.1 introduced success criteria 2.5.5 Target Size (Enhanced) (level AAA) which said that the size of the target for pointer inputs is at least 44 by 44 CSS pixels instead WCAG 2.2 introduced success criteria 2.5.8 Target Size (Minimum) (level AA) which said that the size of the target for pointer inputs is at least 24 by 24 CSS pixels. As before, its primary purpose is another but ensuring that focused elements are clearly visible can also enhance the usability of touch targets. We noted that WCAG considers targets points reachable with the mouse instead MCAG considers primarily touches with fingers and in this case, that we study, the practical minimum size for any tap target is 44x30 (or 30x44) pixels, where 44 pixels are equal to 7mm.

For this principle we have to remember to use appropriate colors and contrasts to make touch targets distinct and easily distinguishable by users, especially those with visual disabilities. For the third operable mobile issue, *Touchscreen Gestures*, we consider the guideline 2.5 "input modalities" which makes it easier for users to operate functionality through various inputs beyond keyboard. We have to mention that WCAG 2.1 and WCAG 2.2 introduced important success criteria to understand this principle. We can see the following:

- WCAG 2.1 introduced success criteria 2.5.1 Pointer Gestures (level A) which states that actions requiring pointer gestures are also accessible via keyboard input to ensure users can interact with the content without solely relying on touchscreen gestures.
- WCAG 2.1 introduced success criteria 2.5.2 Pointer Cancellation (level A) which underlines that actions activated through a pointer gesture can be cancelled or terminated without causing significant changes to the application state. This is important to allow users to correct errors and navigate content flexibly.
- WCAG 2.2 introduced success criteria 2.5.7 Dragging Movements (level AA) which said that all functionality using a dragging movement for operation can be achieved by a single pointer without dragging, unless dragging is essential or the functionality is determined by the user agent and not modified by the author. This is important for

users with limited motor abilities. In particular they can access functionality without needing to perform complex gestures.

For this principle is useful to provide clear and intuitive visual indicators to guide users on using on-screen gestures.

For the fourth operable mobile issue, *Device Manipulation Gestures*, as before, we focus on guideline 2.5 "*input modalities*" which makes it easier for users to operate functionality through various inputs beyond keyboard. Even if keyboards are not usually used in the mobile context we mention the new success criteria for completeness and to help some particular users who need keyboards, for example with some motor disabilities. In particular we can describe the following success criterias studied with WCAG 2.1 and WCAG 2.2:

- WCAG 2.1 introduced success criteria 2.5.1 Pointer Gestures (level A) which states that all functionality that uses multipoint or path-based gestures for operation can be operated with a single pointer without a path-based gesture, unless a multipoint or path-based gesture is essential. This is important to consider because it requires that actions requiring pointer gestures should also be accessible through keyboards. Indeed users can interact with the content without relying solely on touchscreen gestures.
- WCAG 2.1 introduced success criteria 2.5.2 Pointer Cancellation (level A) which underlines that actions activated through a pointer gesture can be cancelled or terminated without causing significant changes to the application state. This allows to have actions triggered by a pointer gesture that can be canceled or undone without significant changes to the application's state. This is important to allow users to correct errors and navigate the content flexibly.
- WCAG 2.1 introduced success criteria 2.5.4 Motion Actuation (level A) which said us that functionality that can be operated by device motion or user motion can also be operated by user interface components and responding to the motion can be disabled to prevent accidental actuation. We saw that the principle focuses on gestures that involve physical movements of the device, such as shaking or tilting, hence, by requiring alternatives for motion-based functionality, this criterion ensures that users who may have difficulty with such gestures can still interact with the content effectively, promoting accessibility and inclusivity.
- WCAG 2.2 introduced success criteria 2.5.7 Dragging Movements (level AA) which states that all functionality that uses a dragging movement for operation can be

achieved by a single pointer without dragging, unless dragging is essential or the functionality is determined by the user agent and not modified by the author. This ensures that users with limited motor abilities can access functionalities without having to perform complex gestures.

Finally, for the fifth and the last operable mobile issue, *Placing buttons where they are easy to access*, we consider guideline 1.4 "*distinguishable*" which specifies to make easier for users to see and hear content including separating foreground from background, guideline 2.4 "*navigable*" which provides ways to help users navigate, find content, and determine where they are and guideline 2.5 "*input modalities*" which makes it easier for users to operate functionality through various inputs beyond keyboard.

New success criteria to consider are the following:

- WCAG 2.1 introduced success criteria 1.4.11 Non-Text Contrast (level AA) which is not directly related but it can help this principle because an adequate visual contrast for non-text elements such as buttons and interactive elements can ensure that they are easily distinguishable and accessible, especially when the device is used in varying lighting conditions.
- WCAG 2.2 introduced success criteria 2.4.11 Focus Not Obscured (Minimum) (level AA) and success criteria 2.4.12 Focus Not Obscured (Enhanced) (level AAA). These two success criterias state that when a user interface component receives keyboard focus, the component is not entirely hidden due to author-created content. This is important because interactive elements must be always visible when activated via keyboard or pointer can improve accessibility for users who use only one hand.
- WCAG 2.1 introduced success criteria 2.5.5 Target Size (Enhanced) level (AAA) and to complete it, WCAG 2.2 introduced success criteria 2.5.8 Target Size (Minimum) (level AA). This two success criteria ensure that interactive elements are of an adequate size, facilitates one-handed use and improves accessibility for users with motor or dexterity disabilities.
- WCAG 2.2 introduced success criteria 2.5.7 Dragging Movements (level AA) which states that all functionality that uses a dragging movement for operation can be achieved by a single pointer without dragging, unless dragging is essential or the functionality is determined by the user agent and not modified by the author. This supports accessibility for users who use only one hand, allowing them to interact with content without the need for complex gestures.

In conclusion, for this principle is important to ensure that interactive elements are ergonomically positioned to allow easy one-handed access and use.

3.3 Understandable

3. Understandable

For this principle WCAG 2.0 identifies the following mobile issues:

3.1. Changing Screen Orientation (Portrait/Landscape)

The official document precises:

Some mobile applications automatically adjust the screen to a particular display orientation (landscape or portrait) and expect users to rotate their devices accordingly. However, some users have their mobile devices mounted in a fixed orientation (e.g. on the arm of a power wheelchair).

Therefore, to accommodate diverse user needs, mobile application developers should try to support both orientations. If supporting both orientations is not feasible for certain functionalities or layouts, developers should ensure that users can easily adjust the orientation to return to a supported state.

Changes in orientation must be programmatically exposed to ensure detection by assistive technology such as screen readers. For instance, if a screen reader user is unaware of a change in orientation, they might inadvertently perform incorrect navigation commands.

3.2. Consistent Layout

The official document precises:

It is crucial for components repeated across multiple pages in web applications to maintain a consistent layout. In responsive web design, where components may rearrange based on device size and screen orientation, pages within the same view (defined size and orientation) should consistently place repeated components and navigational elements. Consistency between the different screen sizes and screen orientations is not a requirement under WCAG 2.0.

For example:

- A website features a logo, title, search form, and navigation bar at the top of each page, consistently ordered. On smaller screens in portrait mode, the navigation bar condenses into a single icon, but the entries in the drop-down menu maintain their relative order.

- A Web site, when viewed on the different screen sizes and in different orientations, has some components that are hidden or appear in a different order. The components that show, however, remain consistent for any screen size and orientation.

The WCAG 2.0 success criteria that are most related to the issue of consistency are:

- 3.2.3 Consistent Navigation (level AA)
- 3.2.4 Consistent Identification (level AA)

3.3. Positioning important page elements before the page scroll

The official document precises:

The small screen size on many mobile devices limits the amount of content that can be displayed without scrolling.

Positioning important page information so it is visible without requiring scrolling can assist users with low vision and users with cognitive impairments.

For users with low vision who rely on screen magnification, only a small portion of the page may be visible at any time. Placing important elements before the need to scroll enables these users to easily locate vital information without having to adjust the magnified view or perform additional interactions. This approach also aids individuals with cognitive impairments, such as short-term memory disabilities, by reducing the cognitive load associated with navigating through content.

Furthermore, positioning essential elements before scrolling promotes consistency in their location across pages. Consistent placement of elements enhances predictability and usability for users with cognitive impairments and low vision.

3.4. Grouping operable elements that perform the same action

The official document precises:

When multiple elements perform the same action or lead to the same destination (e.g., an icon and its associated text acting as links), they should be enclosed within a single actionable element. This increases the touch target size for all users and benefits people with dexterity impairments. It also reduces the number of redundant focus targets, which benefits people using screen readers and keyboard/switch control.

The WCAG 2.0 success criterion that is most related to grouping of actionable elements is:

- 2.4.4 Link Purpose (In Context) (level A)
- 2.4.9 Link Purpose (Link Only) (level AA)

3.5. Provide clear indication that elements are actionable

The official document precises:

It is crucial that elements triggering changes in applications are distinctly recognizable from non-actionable elements such as content or status information. This distinction is especially relevant for web and native mobile applications where users primarily interact through touch or mouse interfaces, relying on visual cues to identify actionable elements. Ensuring these elements are perceptible to users who depend on programmatically determined accessible names, such as screen reader users, is essential.

Visual users interacting with content using touch or visual cursors (e.g. mice, touchpads, joysticks) should be able to clearly distinguish actionable elements such as links or buttons. Existing interface design conventions are aimed at indicating that these visual elements are actionable. The principle of redundant coding ensures that elements are indicated as actionable by more than one distinguishing visual feature. Following these conventions benefits all users, but especially users with vision impairments.

Visual features that can set an actionable element apart include shape, color, style, positioning, text label for an action, and conventional iconography.

Examples of distinguishing features:

- i. Conventional shape: Button shape (rounded corners, drop shadows), checkbox, select rectangle with arrow pointing downwards;
- ii. Iconography: conventional visual icons (question mark, home icon, burger icon for menu, floppy disk for save, back arrow, etc);
- iii. Color offset: shape with different background color to distinguish the element from the page background, different text color;
- iv. Conventional style: Underlined text for links, color for links;
- v. Conventional positioning: Commonly used position such as a top left position for back button (iOS), position of menu items within left-aligned lists in drop-down menus for navigation.

The WCAG 2.0 success criteria do not directly address issue of clear visual indication that elements are actionable but are related to the following success criteria:

- 3.2.3 Consistent Navigation (level AA)
- 3.2.4 Consistent Identification (level AA)

3.6. Provide instructions for custom touchscreen and device manipulation gestures

The official document precises:

The ability to provide control via custom touchscreen and device manipulation gestures can help developers create efficient new interfaces. However, for many people, custom gestures can be a challenge to discover, perform and remember.

Therefore, instructions (e.g. overlays, tooltips, tutorials, etc.) should be provided to explain what gestures can be used to control a given interface and whether there are alternatives. To be effective, the instructions should, themselves, be easily discoverable and accessible. The instructions should also be available anytime the user needs them, not just on first use, though on first use they may be made more apparent through highlighting or some other mechanism.

These WCAG 2.0 success criteria are relevant to providing instructions for gestures:

- 3.3.2 Labels or Instructions (level A)
- 3.3.5 Help (level AAA)

3.3.1 New Ideas - Understandable

For the first understandable mobile issue, *Changing Screen Orientation (Portrait/Landscape)*, the most related success criteria introduced with WCAG 2.1 under the guideline "adatable" is 1.3.4 Orientation (level AA) which states that content does not restrict its view and operation to a single display orientation, such as portrait or landscape, unless a specific display orientation is essential. By ensuring that the application supports both screen orientations, accessibility is promoted for users who use mobile devices mounted in fixed positions. An other important related success criteria introduced with WCAG 2.2 under the guideline 1.4 "distinguishable" is 1.4.10 Reflow (level AA) which states that web content be presented in a layout that does not require horizontal scrolling to view the entire content when the user zooms in to 400%. In this case if the application is able to support both screen orientations, the success criteria ensures that the layout adapts properly to both orientations without compromising accessibility.

For the second understandable mobile issue, *Consistent Layout*, we refer, as before, to the guideline 1.3 "adatable", to the guideline 2.4 "navigable" which provides ways to help users

navigate, find content, and determine where they are and finally to the guideline 3.2 "*predictable*" which makes web pages appear and operate in predictable ways. We can add the following new success criteria introduced with WCAG 2.1 and WCAG 2.2:

- WCAG 2.1 introduced the success criteria 1.3.4 Orientation (level AA) which states that content does not restrict its view and operation to a single display orientation, such as portrait or landscape, unless a specific display orientation is essential. Maintaining consistency in layouts and organization of components can contribute to better adaptability to various display modes.
- WCAG 2.2 introduced the success criterias 2.4.11 Focus Not Obscured (Minimum) (level AA) and 2.4.12 Focus Not Obscured (Enhanced) (level AAA) which require that when a user interface component receives keyboard focus, the component is not entirely hidden due to author-created content. This is important because maintaining visibility of focused components contributes to consistency and user interface comprehension.
- Finally, WCAG 2.2 introduced the success criteria 3.2.6 Consistent Help (level A) which said that help mechanisms must be presented consistently across all pages to assist users in navigating uniformly throughout the site. This contributes in helping consistency.

For the third understandable mobile issue, *Positioning important page elements before the page scroll*, we can see that there are not success criteria directly related to this principle. However, we can mention some related and useful success criterias:

- 1.3.1 Info and Relationships (level A);
- 2.4.10 Section Headings (level AAA);
- 2.4.13 Focus Appearance (level AAA);
- 3.2.4 Consistent Identification (level AA).

These success criteria do not focus directly on the placement of elements before page scrolling, but are still relevant to ensuring a consistent and accessible user experience.

Some useful best practices for this principle are the following:

- To identify crucial content that users need to see immediately, such as the logo, navigation menu, main headings, and calls to action.
- To position important elements at the top of the page so they can be perceived without the need for scrolling.

- To position headings above their corresponding content to facilitate scanning and quick access to important sections.
- To adapt the layout of the page so that important elements are prominently positioned on screens of different sizes and orientations.
- To use responsive design to ensure that content automatically adjusts to the available space on the device screen.

For the fourth understandable mobile issue, *Grouping operable elements that perform the same action*, we have to note the guideline 2.5 *"input modalities"* which makes it easier for users to operate functionality through various inputs beyond keyboard. No new success criteria were introduced with WCAG 2.2, instead for WCAG 2.1 we can mention the following success criteria:

- Success criteria 2.5.3 Label in Name (level A) states that visible text for all user interface components, including links, accurately describes the purpose or function of the component. This ensures that users understand the action associated with the element, benefiting users who rely on screen readers and keyboard navigation.
- Success criteria 2.5.6 Concurrent Input Mechanisms (level AAA) which encourages providing multiple input methods for interactive elements, such as allowing both touch and keyboard input. By supporting various input methods, users with different abilities and preferences can interact with the content effectively.

For this principle we have to ensure that interactive elements have clear and descriptive labels to communicate their purpose or function, helping users who rely on assistive technologies. It also reduces the number of redundant focus targets, which benefits people using screen readers and keyboard/switch control.

For the fifth understandable mobile issue, *Provide clear indication that elements are actionable*, there are no new strictly related success criteria introduced with WCAG 2.1 and WCAG 2.2 but under the guideline 2.4 *"navigable"*, the guideline 2.5 *"input modalities"* and the guideline 3.2 *"predictable"* we can find some useful success criteria. In particular we mention the following:

- WCAG 2.2 introduced the success criteria 2.4.11 Focus Not Obscured (Minimum) (level AA) which helps in maintaining visible focus. Hence, users, especially those with visual impairments, can clearly perceive which elements are currently focused and therefore actionable.

- WCAG 2.2 introduced the success criteria 2.4.12 Focus Not Obscured (Enhanced) (level AAA) which, as before, ensures that users can confidently interact with actionable elements without losing sight of their focus state.
- WCAG 2.2 introduced the success criteria 2.4.13 Focus Appearance (level AA) which helps users, especially those who rely on keyboard navigation, in identifying actionable elements.
- WCAG 2.1 introduced the success criteria 2.5.1 Pointer Gestures (level A) which ensures keyboard accessibility for all users, including those with motor disabilities. Hence, they can access actionable elements.
- WCAG 2.1 introduced the success criteria 2.5.5 Target Size (Enhanced) (level AAA) which enlarging target sizes benefits users with motor impairments, ensuring they can interact with elements more easily and accurately.
- WCAG 2.2 introduced the success criteria 2.5.8 Target Size (Minimum) (level AA) which, similar to the previous point, an adequate target size prevents accidental activations and benefits users across various abilities, enhancing the clarity and usability of actionable elements.
- WCAG 2.2 introduced the success criteria 3.2.6 Consistent Help (level A) which even if not directly related to making elements actionable, consistent help ensures that users have access to assistance when interacting with elements, which can indirectly contribute to their clarity and usability.

For the sixth understandable mobile issue, *Provide instructions for custom touchscreen and device manipulation gestures*, we refer to the guideline 2.5 "*input modalities*" and to the guideline 3.2 "*predictable*". In particular the most related success criterias are the following:

- WCAG 2.1 introduced the success criteria 2.5.1 Pointer Gestures (level A) which even if not directly related to the principle, it ensures that alternative methods of interaction are available for users who may have difficulty with touchscreen gestures.
- WCAG 2.1 introduced the success criteria 2.5.4 Motion Actuation (level A) which ensures that such functionality is also accessible to users who cannot perform motion-based actions.
- WCAG 2.2 introduced the success criteria 2.5.7 Dragging Movements (level AA) which ensures that all operations using dragging movements can also be performed with a

single pointer without dragging. It supports users who may have difficulty executing complex gestures. Integrating this criterion ensures that users who rely on alternative input methods, such as keyboard or switch control, can access and operate functionalities that involve dragging movements.

- WCAG 2.2 introduced the success criteria 3.2.6 Consistent Help (level A) which provides instructions for custom touch screen and device manipulation gestures. Hence, consistent help ensures that users can access guidance whenever needed, regardless of their familiarity with the interface. This is important because maintaining consistency in the presentation and availability of help resources, users can rely on clear instructions to understand and execute gestures, enhancing their overall user experience.

3.4 Robust

4. Robust

For this principle WCAG 2.0 identifies the following mobile issues:

4.1. Set the virtual keyboard to the type of data entry required

The official document precises:

Mobile devices offer users the ability to customize standard keyboards through device settings and to install additional custom keyboards. Depending on the type of data entry, some devices automatically switch between different virtual keyboards. Users can configure these settings themselves or opt for a specific keyboard type. For instance, HTML5 form field controls trigger different keyboards when users input information on a website. While this customization aids in error prevention and ensures correct formatting, it can present challenges for individuals using screen readers due to subtle variations in keyboard layouts.

4.2. Provide easy methods for data entry

The official document precises:

Users can enter information on mobile devices in multiple ways such as on-screen keyboard, Bluetooth keyboard, touch, and speech. Text entry can be time-consuming and difficult in certain circumstances. Reduce the amount of text entry needed by providing select menus, radio buttons, check boxes or by automatically entering known information (e.g. date, time, location).

4.3. Support the characteristic properties of the platform

The official document precises:

Mobile devices provide many features to help users with disabilities interact with content. These include platform characteristics such as zoom, larger fonts, and captions. The features and functions available differ depending on the device and operating system version. For instance, while most platforms support the option to enlarge fonts, not all applications consistently apply this setting across all text. Additionally, some applications may increase font size without properly wrapping text, leading to issues like horizontal scrolling.

3.4.1 New Ideas - Robust

Before starting the analysis we have to remember that with WCAG 2.2 the success criteria 4.1.1 Parsing (level A) under the guideline 4.1 *"compatible"* was removed.

For the first robust mobile issue, *Set the virtual keyboard to the type of data entry required*, no new success criteria can be added from WCAG 2.1 or WCAG 2.2.

For the second robust mobile issue, *Provide easy methods for data entry*, we can add the following success criteria:

- Under the guideline 4.1 *"compatible"*, WCAG 2.1 introduced the success criteria 4.1.3 Status Messages (level AA) which requires clear feedback to users during data entry. This is applied by providing status notifications after form submission, indicating errors in data entry, and updating the status in real-time during interaction with dynamic user interface elements.

For the third robust mobile issue, *Support the characteristic properties of the platform*, no new success criteria can be added from WCAG 2.1 or WCAG 2.2.

3.5 Ideas for the future

In the previous sections we have seen how the existing guidelines can be integrated with the new success criterias introduced with WCAG 2.1 and WCAG 2.2 and some best practices to follow. The aim of this section is to find new guidelines to complete a document of indications for a mobile application.

As before, we can start from the first principle *Perceivable*:

- *Typography*: The default system font in Android is Roboto and San Francisco in iOS. Using these fonts or closely related fonts is recommended for a consistent experience across the Android and iOS platforms. A useful alternative is to use highly readable

fonts to favor the various categories of user. In practice, a character that does not confuse the end user is recommended;

- *Themes:* Support both light and dark modes in your application and using Material Theming to provide a unique and consistent look in all the application.

For the second principle *Operable*:

- *Voice control:* Offer the ability to fully control the application using advanced voice commands. This includes fully navigating the application through voice commands and activating specific features with precision and reliability. It is useful to implement advanced voice search that allows users to find and access application content quickly and accurately through intuitive voice commands and natural language understanding. An other aspect regards to provide contextual suggestions or real-time help for users with disabilities during use. For example, to describe what buttons or controls do when they are highlighted or selected to help users with screen readers. Moreover, introduce audio feedback to indicate user interactions, such as confirmation of a successful action, so that even blind users or users with low vision can better understand the actions taken in the application.

Possible success criteria: The application offers complete navigation functionality through voice commands with precise voice recognition on specific words and phrases. From WCAG we can use the following success criteria:

- WCAG 2.1 Success Criterion 1.2.1 Audio-only and Video-only (Prerecorded) (A)
For prerecorded audio-only and prerecorded video-only media, the following are true, except when the audio or video is a media alternative for text and is clearly labeled;
- WCAG 2.1 Success Criterion 1.2.5 Audio Description (Prerecorded) (AA)
Audio description is provided for all prerecorded video content in synchronized media;
- WCAG 2.1 Success Criterion 1.2.9 Audio-only (Live) (AAA)
An alternative for time-based media that presents equivalent information for live audio-only content is provided;
- WCAG 2.1 Success Criterion 1.4.2 Audio Control (A)
If any audio on a Web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level;

- WCAG 2.1 Success Criterion 2.1.1 Keyboard (A)

All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes;

- *Customization and simplification of the interface:* Allow the user to customize the interface based on the surrounding environment (bright sunlight, low light conditions, etc...). Furthermore, we must try to offer a simplified mode that reduces the complexity of the interface for less experienced users or those with motor, hearing or visual difficulties and it is important to optimize the interface for one-handed navigation. During the experience, users can rearrange UI elements to make screen interactions more accessible. Introducing text typing assistance tools, such as smart suggestions, autocorrect, or text prediction, to help users with learning or motor disabilities when entering data. Allow users to customize touchscreen gestures to perform specific actions, providing flexibility in interacting with the application.

Possible success criteria: The application offers interface customization options that automatically adjust the contrast, brightness or size of elements based on the surrounding environment, or allow users to do so manually. Provide a clear and easily accessible option to activate simplified mode, reducing advanced features and simplifying the app layout. From WCAG we can use the following success criteria:

- WCAG 2.1 Success Criterion 1.4.8 Visual Presentation (AAA)

For the visual presentation of blocks of text, a mechanism is available;

- WCAG 2.1 Success Criterion 2.1.4 Character key Shortcuts (A)

If a keyboard shortcut is implemented in content using only letter (including upper- and lower-case letters), punctuation, number, or symbol characters, then at least one of the following is true: turn off, remap and active only on focus;

- WCAG 2.2 Success Criterion 2.5.8 Target Size (Minimum) (AA) (2.5.5)

The size (minimum) of the target for pointer inputs is at least 24 by 24 CSS pixels, or the size (enhanced) of the target for pointer inputs is at least 44 by 44 CSS pixels except in particular cases;

- *All text elements are visible to screen reader:* All elements usable and non-usable by a user must be clearly visible and readable by the screen reader, to facilitate all categories of users, including those with visual disabilities. It is useful to verify that screen readers can correctly detect and read all text and content in the application, including that

within graphs, images, and other visual elements, ensuring that the screen reader is served the elements in the correct order of reading;

- *Action confirmations:* When users take important actions, such as submitting a form or deleting data, ensure they receive clear and understandable confirmation before the action is finally performed;

For the third principle *Understandable*:

- *Action progression display:* Given the increase in actions within an application it becomes necessary to provide clear and progressive indicators to help users understand the status and progress of an action within the application so as to better understand what happens inside the application.

Possible success criteria: The application offers options to view the status and progress of an action in the application. From WCAG we can use the following success criteria:

- WCAG 2.1 Success Criterion 3.2.3 Consistent Navigation (AA)
Navigational mechanisms that are repeated on multiple web pages within a set of web pages occur in the same relative order each time they are repeated;
 - WCAG 2.1 Success Criterion 3.2.4 Consistent Identification (AA)
Components that have the same functionality within a set of web pages are identified consistently;
 - WCAG 2.1 Success Criterion 3.2.5 Change on Request (AAA)
Changes of context are initiated only by user request or a mechanism is available to turn off such changes;
 - WCAG 2.2 Success Criterion 3.2.6 Consistent Help (A)
If a web page contains any of the following help mechanisms, and those mechanisms are repeated on multiple web pages;
 - WCAG 2.1 Success Criterion 4.1.3 Status Messages (AA)
In content implemented using markup languages, status messages can be determined through role or properties;
- *Avoid non necessary elements:* It is important not to introduce elements into the application that create confusion for the end user, in particular elements that are not qualitative. Furthermore given the size of mobile screens must also be avoided. It is essential to reduce the number of touches to achieve your goals. This helps all categories

of users but especially those with visual or motor difficulties or difficulties in processing information;

- *Destination of the navigation buttons:* Make sure that all navigation buttons within the application are clearly distinguishable, avoiding ambiguous or unclear navigation buttons, especially by users with motor or visual difficulties and that their destination is also clear so as to avoid confusion and that the user gets lost in navigation;
- *Guided start-up tour:* Provide a guided tour when the application launches to guide users through the main features and options available. This can help users quickly familiarize themselves with the user interface and understand how to use the application effectively;

For the fourth principle *Robust:*

- *Advanced feedback:* Provide more sophisticated and customizable haptic feedback to indicate application interactions, notifications or state changes. Users should be able to adjust the duration and intensity of this feedback to suit their preferences and needs, so as to facilitate all categories of users. Identify critical or urgent notifications with customizable audible or vibration alerts to ensure users with hearing or visual impairments can be aware of those notifications. Introduce audio feedback to indicate user interactions, such as confirmation of a successful action, so that even blind users or users with low vision can better understand the actions taken in the application.

Possible success criteria: The app provides a range of customizable haptic feedback (vibrations, sounds, etc...) for different actions or notifications. Users can adjust the duration and intensity of feedback. From WCAG we can use the following success criteria:

- WCAG 2.1 Success Criterion 1.4.8 Visual Presentation (AAA)
For the visual presentation of blocks of text, a mechanism is available;
- WCAG 2.1 Success Criterion 3.2.5 Change on Request (AAA)
Changes of context are initiated only by user request or a mechanism is available to turn off such changes;
- WCAG 2.1 Success Criterion 4.1.3 Status Messages (AA)
In content implemented using markup languages, status messages can be determined through role or properties;

- *Minimize data entry and maintain user data:* Minimize the data entry required of users and securely maintain user data, using automatic storage of previously entered data or providing intelligent suggestions during data entry to facilitate the use of the application and prevent data loss during use;
- *Help users avoid errors and correct mistakes when entering input:* Provide clear feedback and contextual instructions to guide users during data entry and help them avoid errors, thus facilitating all categories of users. Implement data validation checks to detect common errors and alert users in a timely manner. Offer suggestions and automatic corrections for input errors to simplify the correction process;
- *Multilingual support:* Make sure the application is correctly translated for all languages it makes available. Offer users the possibility of selecting the language of their preference, ensuring adequate translation of the contents, including texts, labels and feedback messages for optimal use of the application by users of various languages;
- *Provide users enough time to read and use the content:* This is not purely defined in WCAG or WCAG for mobile but it would be useful to define it as a criterion. Initially we need to define timescales to determine enough time, so users can read and use the content of the application without feeling under pressure or stressed, also considering the different reading and understanding speeds of the different types of users.

Chapter 4

Analysis on Flutter

In this chapter we start with a description of the Flutter framework and then we continue with an analysis of the accessibility of the framework testing various ad hoc mini-applications.

4.1 Flutter

Flutter [15], whose logo is shown in the Figure 5.1, is a framework [19] created quite recently for the development of applications for different platforms. It was created by Google as an open source project and published officially for the first time in December 2018 in version 1.0 at the Flutter event Live. On March 3, 2021, version 2.0 was released which allows developers to stably generate multiplatform applications. On May 30, 2023, Flutter 3.0 was released, introducing further improvements and advanced features for cross-platform application development.



Figure 4.1: Logo Flutter

Flutter offers a large set of UI element libraries and combines the ease of development with performance similar to native performance, maintaining a correct visual distinction between different platforms without the programmer having to pay particular attention. It is mainly used for Android applications and iOS and works like a true native application. Using the same codebase, i.e. the entire collection of source code used for build the application,

you can create the application for different platforms.

Flutter's programming language is Dart, also developed by Google, and was designed to replace JavaScript. Flutter is completely free and its popularity and usage has grown significantly.

Although very recent, in the Google Play Store we can count over 50,000 Flutter's applications. The advantages of using Flutter are:

- One codebase for all platforms;
- Use of Dart which is an easy language to learn;
- It is easier to develop applications with Flutter which then enter the market faster;
- It is based on the *Everything is a Widget* principle which will be explained in the Widget section;
- Low consumption of resources;
- Good performance execution of native apps on smartphones;
- User interface which can also be customized;
- Hot reload allows you to see changes in real time, accelerating development.

Instead, among the disadvantages we can mention that:

- Dart is a new language and therefore is not very widespread;
- Being new, there is a lack of third-party libraries that facilitate development;
- Large apps are created compared to other frameworks or even a Java itself.

4.2 Dart

Dart [11] is a programming language developed by Google and presented for the first time on 10 October 2011 at the GOTO Aarhus 2011 conference. The main purpose is to replace JavaScript for application development. Dart forms the foundation of Flutter and also supports many core development tasks such as formatting, parsing, testing the code, ...

Dart is type-safe. Also values in Dart cannot be *null* except in cases where it is indicated that these values may be, so as to avoid possible errors in the code.

Dart [12] has a large number of core libraries and additional [Application Program Interface\(API\)](#) packages. It allows you to write programs through two distinct platforms:

- Dart Native: it is for applications developed for mobile and desktop devices. Includes both a Dart virtual machine with just-in-time (JIT) compilation, that is, the code is compiled during program execution and facilitates hot reloading, or an AOT (Ahead-of-Time) compiler for the production of machine code, i.e. it is compiled before execution, typically during installing the program, so as to improve performance by decreasing startup times and avoiding the compilation phase during the execution of the plan;
- Dart Web: it is for applications targeting the Web. Includes both a development time (dartdevc) which allows you to debug your application in the Chrome browser and see the changes almost immediately, which a production time compiler (dart2js) that provides suggestions for improving the Dart code and removing unused code. Both compilers translate Dart in JavaScript.

4.3 Components

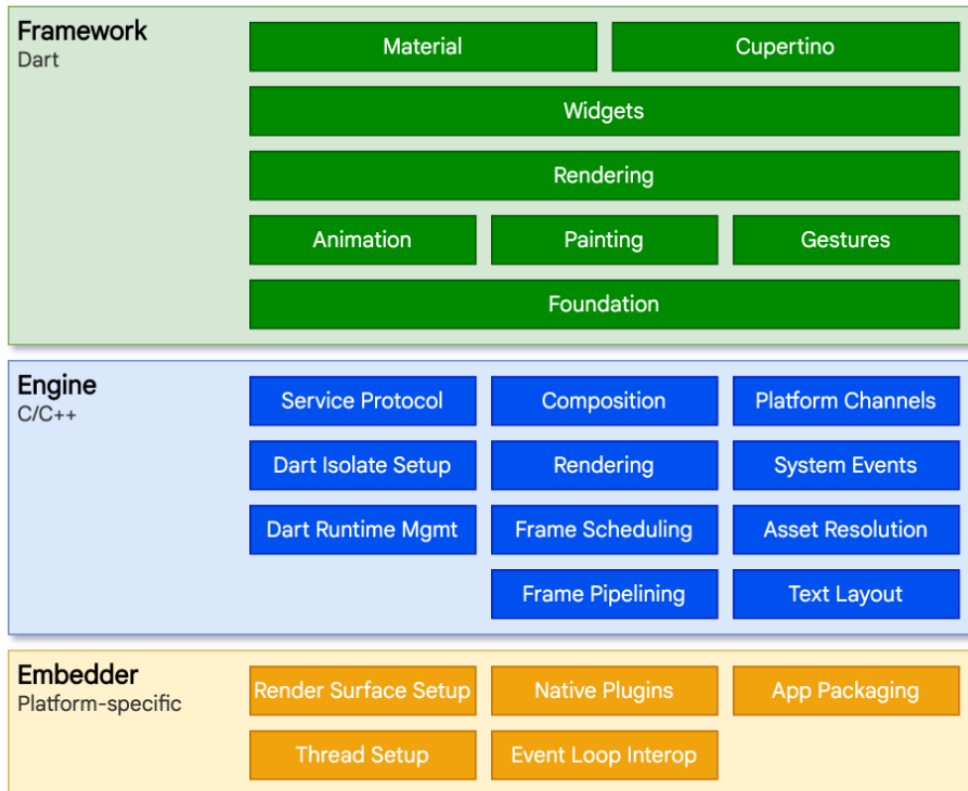


Figure 4.2: Flutter layered architecture

Flutter [6], as seen in Figure 5.2, is made up of a layered architecture and its three basic

layers are:

- Framework;
- Engine;
- Embedder.

4.3.1 Embedder

Embedder is the lowest level of the architecture.

The language used is defined according to the platform and currently can be: Java and C++ for Android, Objective-C/Objective-C++ for iOS and macOS and C++ for Windows and Linux. It aims to tie native screen rendering, handling of events and other elements. To do this the Embedder layer interacts with the layer Engine via C/C++ APIs. Furthermore the Embedder layer is composed of a Shell which also hosts the Dart VM. Each Shell is specific to each platform and offers access to the native APIs of the platform in question.

4.3.2 Engine

Engine is the middle layer of the architecture.

The language used is mainly C++ and C to make it faster and more efficient applications created in Flutter. Contains essential low-level components for the operation of the framework. Inside we find the Skia graphics engine, a Open source 2D graphics library written in C++ created by Google, and the shells it is to can be accessed via the API exposed by the *dart:ui* library.

4.3.3 Framework

It is the most important layer of architecture.

The language used is Dart. There are classes inside basic fundamentals and basic services like animations. Rendering is also present which allows you to manage the layout through a tree of objects that is displayed on the screen and updates automatically. Always within this layer they are present the Widgets and the two libraries: Material and Cupertino.

4.4 Material and Cupertino libraries

Within Flutter [24] it is possible to understand which platform the application is running on through *Platform.isIOS*. If it returns a Boolean value equal to *true* it means that we will

be on iOS otherwise we will be on Android. The Cupertino library allows you to implement a specific design for applications iOS. Unlike Cupertino, the Material library allows you to implement a specific design for Android applications.

4.5 Widget

Flutter is based on the principle *Everything is a widget* as a program's interface is made up of several nested widgets. Each widget [18] can be text or a button or any other graphic element that contains various characteristics. All these widgets can influence other widgets in the construction of the application. The main advantage of this widget structure is flexibility, however disadvantage is that all widgets are located in the program's source code and therefore they will be strongly nested. The framework contains two main classes of widgets [49]:

- Stateless widget;
- Stateful widget.

4.5.1 Stateless widget

Stateless widgets do not have a mutable state and therefore will not change over time, not even following behavior carried out by the user. Some examples of these widgets are: Text, Row, Column and Container. To create a stateless widget, you need to extend the *StatelessWidget* class it requires the override of the *build()* method. This method is invoked the first time to build the widget tree and then whenever their dependencies change. You can use a stateless widget only when the widget fields do not change over time not even after user actions. In other cases a stateful widget will be used.

4.5.2 Stateful widget

Stateful widgets are dynamic and change over time based on user interaction or based on other factors. Some examples of these widgets are: Image, Form and Checkbox. To create a stateful widget you need to extend the *StatefulWidget* class. As the name suggests, these widgets depend on the state of the object. Indeed every time you want to change the state of any object you have to call the *setState()* method thus signaling the framework to update the user interface calling the *build()* method and taking into account the new states of the

objects assigned in the method just described. We therefore need to create a stateful widget when the widget might change during its lifecycle.

4.6 Accessibility of the application

Flutter [16] uses textual widgets and semantics to enhance accessibility for users with visual impairments. It achieves this by providing appropriate labels and attributes to widgets, ensuring that text colors and sizes meet accessibility standards. Flutter also supports screen reading capabilities, enabling users to interact with UI elements effectively. Furthermore, Flutter promotes sufficient contrast between foreground and background colors, enhancing readability and complying with accessibility guidelines. In particular the framework Flutter supports:

- Large fonts: Render text widgets with user-specified font sizes;
- Screen readers: Communicate spoken feedback about UI contents;
- Sufficient contrast: Ensure that widgets are rendered with colors that provide adequate contrast. Flutter provides various tools and guidelines to ensure that widgets are rendered with colors that offer adequate contrast, which is crucial for readability and accessibility in user interfaces. Insufficient contrast between foreground and background colors can make text or UI elements difficult to distinguish, especially for users with visual impairments or when viewed in different lighting conditions. In particular Flutter:
 1. Offers a set of accessibility guidelines, including recommendations on color contrast. These guidelines align with the WCAG standards, which specify minimum contrast ratios for text and UI elements;
 2. Provides tools like the *AccessibilityChecker*, which help developers evaluate whether their applications adhere to accessibility guidelines;
 3. Offers specific widgets and properties that facilitate the creation of accessible interfaces. For example, the Text widget has properties like *style*, which allow developers to define text colors that meet contrast requirements;
 4. Supports the use of themes *ThemeData* that can be configured to ensure all interface components adhere to accessibility guidelines, including color contrast;

To study the accessibility we start with the creation of a mini-application in Android Studio [5] as [Integrated Development Environment \(IDE\)](#). Android studio displays all the properties that a widget makes available, promoting accessibility. To study the accessibility it is sufficient to follow the official documentation provided by Flutter where there are all the presented widgets. To test the application for the Android operating system, initially we used the emulator present on the [IDE](#) and then for a deeper analysis we use an Android *Google Pixel 7* [20] to see manually if the screen reader (Talkback) works correctly and respect the accessibility. The aim of the thesis is to analyse accessibility aspects of the different components and widgets provided by Flutter. The goal is to answer the following questions:

- *Question 1:* Are the components and widgets provided by the framework accessible by default?
- *Question 2:* If a component or a widget is not accessible by default, is it possible to make it accessible?
- *Question 3:* If a component or a widget is not accessible by default, and can be made accessible by developers, how much does it cost - in terms of additional required code?

In the Question 1 we are interested in evaluating how much the framework is accessible by default hence in the following tables we use a tick or a cross to say if they are accessible or not. For the Question 2 we use P(Property) and W(Widget) with a number to indicate the number of properties and/or widgets that the analyzed widget or component needs to become accessible. Finally for the Question 3 we write the number of additional lines of code needed to make the component or widget accessible.

4.7 Analysis of the application

We called the analyzed application *Flaccess*, consisting of five pages and created ad hoc for the purposes of the thesis without the purpose of publishing it on the Play Store or App Store.

Let's start from the first page that has a simple quiz showed in Figure 4.3 and 4.4:

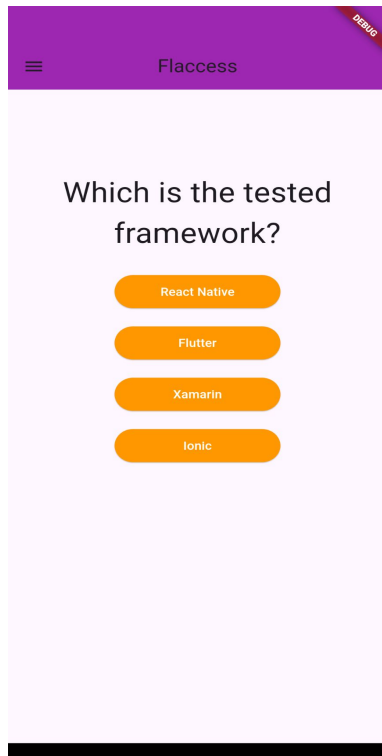


Figure 4.3: First Page

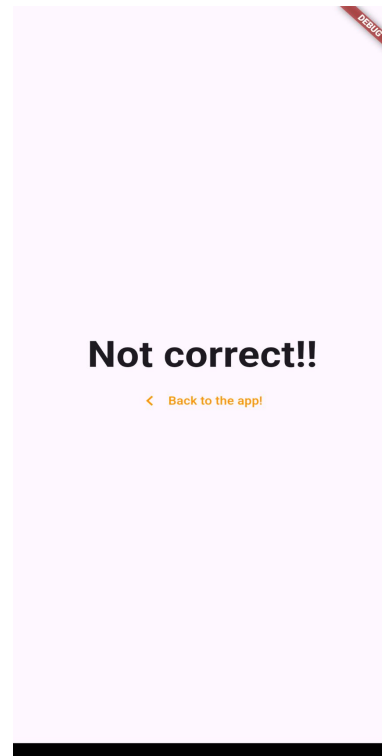


Figure 4.4: Answer

The main important widgets of the first page are listed in Table 4.1:

Widget/Component	Question 1	Question 2	Question 3
AppBar	×	+1W +1P	2
Text	✓		
Button (all)	✓		
IconButton	×	+1P	1

Legend: ✓ = Accessible, × = Not Accessible, +1W = Number of Widgets, +1P = Number of Properties, Question 3 = Additional lines of code

Table 4.1: First Page Analysis

1. **AppBar:** Flutter does not provide a proper widget to define headers hence the solution is to use a *Semantic* widget with the property *'header: true'*. Thanks to this property the screen reader can read that this is a header. Code is in Listing 4.1:

```
1   appBar: AppBar(  
    title: Semantics(  
      header: true,  
      child: Text(  
        'Flaccess',  
6      ),  
    ),  
  ),  
),
```

Listing 4.1: AppBar Widget Code

2. **Text:** This widget in flutter is accessible and the screen reader is always able to read the text in the application. We can note that the Text widget have the property *'semanticsLabel'* to hide the original text content and read other one. Exercise caution when opting to utilize it;
3. **Button:** All the button widgets are accessible in Flutter and it is not necessary to use a Semantic widget with the property *'button:true'* because the screen reader is able to recognize that there is a button and it is able to guide the user correctly;
4. **IconButton:** The iconButton is not accessible by default in Flutter and indeed you need the property *'semanticLabel'* to describe the behaviour of the icon. Code is in Listing 4.2:

```
2   IconButton(  
    icon: Icon(Icons.arrow_back, semanticLabel: 'Back to the HomePage')  
    ),
```

Listing 4.2: IconButton Widget Code

In the first page showed in Figure 4.5 there is also the drawer with its components:

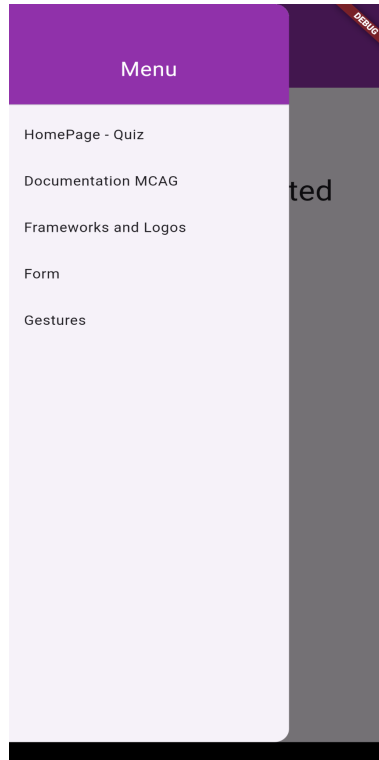


Figure 4.5: Drawer

The main important widgets of the drawer are listed in Table 4.2:

Widget/Component	Question 1	Question 2	Question 3
Drawer	✓		
DrawerHeader	×	+1W +1P	2
ListTile	×	+1W +1P	2

Table 4.2: Drawer Analysis

1. **Drawer:** This widget is accessible by default in Flutter and the screen reader explains very well how to use it;
2. **DrawerHeader:** As before the header of the Drawer is not accessible, hence we have to use a Semantic widget with the property *'header'* set to true. Code is in Listing 4.3:

```
child: DrawerHeader(  
2   child: Semantics(  
    header: true,  
    child: Text(  
      'Menu',  
    ),  
7  ),  
),
```

Listing 4.3: DrawerHeader Widget

3. **ListTile:** This widget is not accessible in Flutter. To make it accessible we use a Semantic widget with the property *'button'* set to true to signal the screen reader that this is a button and what to do. Code is in Listing 4.4:

```
ListTile(  
2   title: Semantics(  
    button: true,  
    child: Text(  
      'HomePage - Quiz',  
    ),  
7  ),  
  onTap: () {  
    _resetQuiz();  
    _closeDrawer();  
  },  
12 ),
```

Listing 4.4: ListTile Widget

The second page where there is the documentation showed in Figure 4.6:

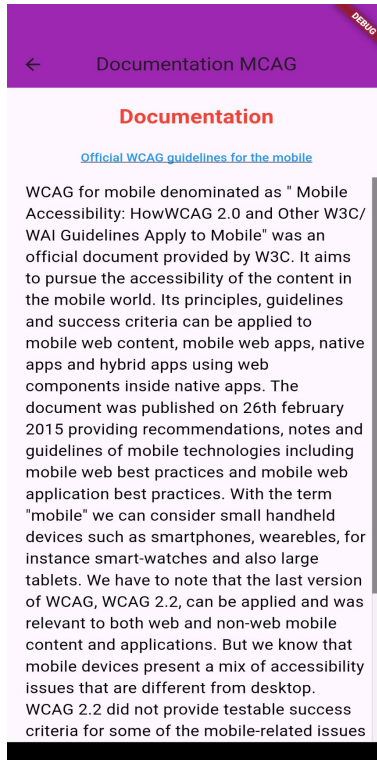


Figure 4.6: Second page

The main important widgets of this page are listed in Table 4.3:

Widget/Component	Question 1	Question 2	Question 3
Text (link)	✓		
ScrollBar	✓		
SinglechildScrollView	✓		

Table 4.3: Second Page Analysis

1. **Text with link:** In Flutter the text is always accessible to the screen reader but in this case with the embedded link we can think that the screen could have some problems to read it correctly. We use a `TextButton` and the screen reader said to the user that it is a button and to touch two times to active it. Note that as before you can use the property `'semanticLabel'` but pay particular attention when you use it;
2. **ScrollBar and SingleChildScrollView:** A lot of people can suppose that these widgets are not accessible but instead they are. If we use a Semantic widget with the property `'label'` to explain that you use a scrollable content for the user is only a waste of time. Maybe it can help users who are not expert in using the screen readers but in most of cases is additional time in the use of the application, which is not a good thing when you develop an application.

The third page where there are the cards and the images showed in Figure 4.7 and 4.8:

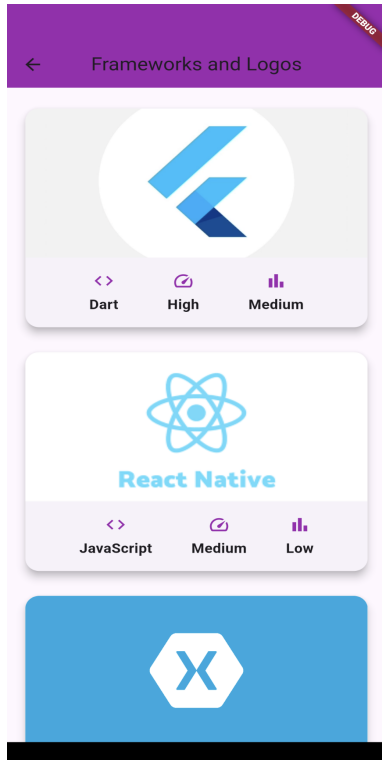


Figure 4.7: Third Page

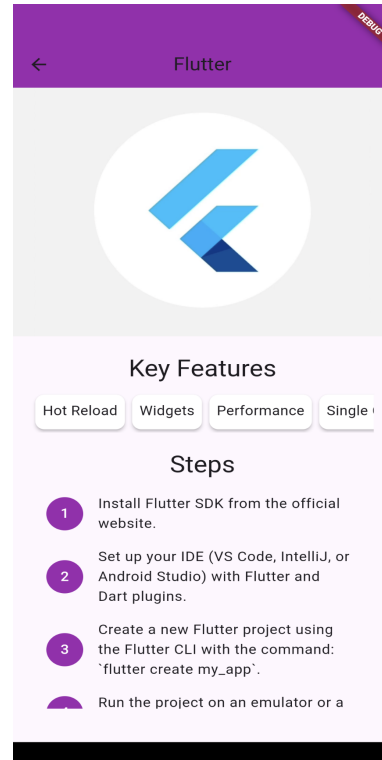


Figure 4.8: Details

The main important widgets of this page are listed in Table 4.4:

Widget/Component	Question 1	Question 2	Question 3
Image	×	+1P	1
Card	✓		
Icon	×	+1P	1
Text (header)	×	+1W +1P	2
ClipRect	✓		
CircleAvatar	✓		

Table 4.4: Third Page Analysis

1. **Image:** In Flutter, images are not accessible by default but adding the property `'semanticLabel'` the screen reader reads the text that you put, in this case it is useful to describe the image, and then it also says that the current element is an image. An alternative but less efficient solution is to use a Semantic widget with the property `'label'`. Code is in Listing 4.5:

```
Image.asset(  
  imagePath,  
  semanticLabel: 'Flutter',  
),
```

Listing 4.5: Image Widget

2. **Card:** This widget is accessible especially because it is composed by different widgets that make the Card widget accessible because it works as a container of other widgets;
3. **Icon:** As the Image widget the Icon widget is not accessible in Flutter and to make it accessible you have to use the property `'semanticLabel'`. Code is in Listing 4.6:

```
Icon(  
  Icons.schedule,  
  semanticLabel: 'The time needed is',  
),
```

Listing 4.6: Icon Widget

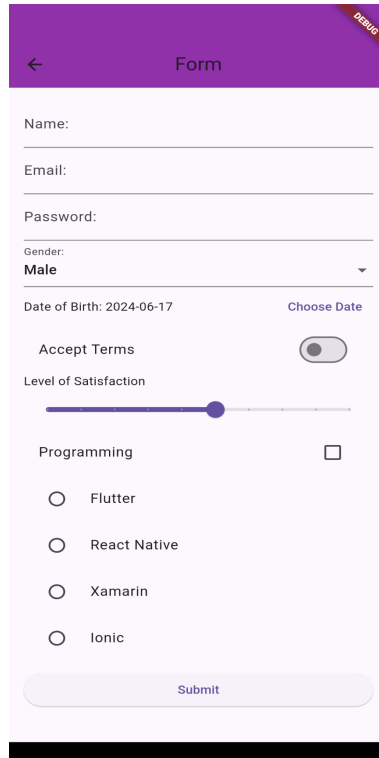
4. **Text (header):** As we saw in the previous analysis, all elements which require to be read as a header need a Semantic widget with the property `'header'` set to true. Code is in Listing 4.7:

```
child: Semantics(  
  header: true,  
  child: Text(  
    'Steps',  
  ),  
),
```

Listing 4.7: Text (header) Widget

5. **ClipRect:** An accessible element in Flutter used with the Card widget;
6. **CircleAvatar:** An accessible element in Flutter used in the details page to give a better aspect to the numbers which are an accessible Text widget;

The fourth page where there is the form showed in Figure 4.9, 4.10 and 4.11:



The screenshot shows a mobile application interface for a form titled "Form". The form is displayed on a light purple background. At the top, there is a purple header bar with a back arrow on the left and the title "Form" in the center. Below the header, the form consists of several fields: "Name:" with a text input field; "Email:" with a text input field; "Password:" with a text input field; "Gender:" with a dropdown menu showing "Male"; "Date of Birth:" with the text "2024-06-17" and a "Choose Date" button; "Accept Terms" with a toggle switch that is currently turned on; "Level of Satisfaction" with a horizontal slider; "Programming" with a checkbox that is currently unchecked; and a list of radio button options: "Flutter", "React Native", "Xamarin", and "Ionic". At the bottom of the form, there is a "Submit" button.

Figure 4.9: Fourth Page

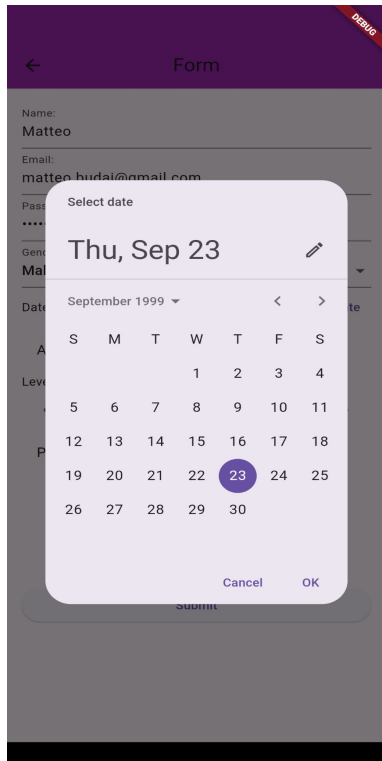


Figure 4.10: DatePicker

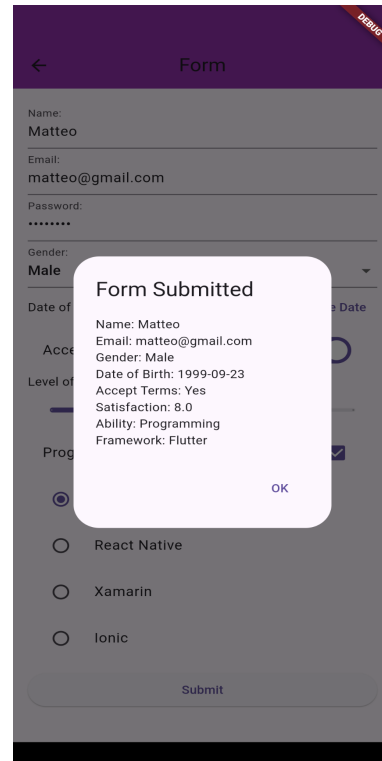


Figure 4.11: Alert Dialog

The main important widgets of this page are listed in Table 4.5:

Widget/Component	Question 1	Question 2	Question 3
Form	✓		
TextFormField	✓		
DropDownButtonFormField	✓		
showDatePicker	✓		
SwitchListTile	✓		
RadioListTile	✓		
CheckBoxTile	✓		
Slider	✓		
AlertDialog	✓		

Table 4.5: Fourth Page Analysis

1. **Form:** In Flutter the Form widget is accessible by default. It is a widget composed by other widgets used to create a form. It is not necessary to use a Semantic widget with the property `'form'` set to true;
2. **TextFormField:** The widget is accessible in Flutter. The screen reader know that there is a field to complete with a text but it does not read what to enter in the field. To complete it we have to use the property `'decoration'` with a label. Noted that this property is important also without the screen reader. Code is in Listing 4.8:

```
TextFormField(  
  decoration: InputDecoration(labelText: 'Name:'),  
),  
4
```

Listing 4.8: TextFormField Widget

3. **DropDownButtonFormField:** As the previous widget is accessible in Flutter but as before it needs a property `'decoration'` to signal what to enter in the field. Code is in Listing 4.9:

```
DropDownButtonFormField<String>(  
2  value: _gender,  
  decoration: InputDecoration(labelText: 'Gender:'),  
  ...  
),
```

Listing 4.9: DropDownButtonFormField Widget

4. **showDatePicker:** It is a widget accessible in Flutter. It shows an accessible calendar where you can pick a data;
5. **SwitchListTile, RadioListTile and CheckBoxTile:** These three widgets are accessible by default in Flutter. Noted that they need a property `'title'` to specify what you will check. This property is important for all the users. Code is in Listing 4.10:

```
SwitchListTile(  
  title: Text('Accept Terms'),  
  value: _acceptTerms,  
  onChanged: (value) {  
5    setState(() {  
      _acceptTerms = value;  
    });  
  },  
10 ),
```

Listing 4.10: SwitchListTile Widget

6. **Slider:** In Flutter this widget is accessible by default. With a particular gesture you can use it with the screen reader which says immediately when you change the current value;
7. **AlertDialog:** As the previous widgets, it is accessible. It is composed by different widgets and note that it is useful with and without the screen reader to add the property `'title'`.

The fifth page where there are different gestures is in Figure 4.12. In this page there are four gestures to complete:

1. Blue Box: It requires to touch twice;
2. Orange Box: It consists in touch three times rapidly;
3. Green Box: It requires to touch two times with a long press on the second tap;
4. Purple Box: It consists in touch two times to swipe the box.

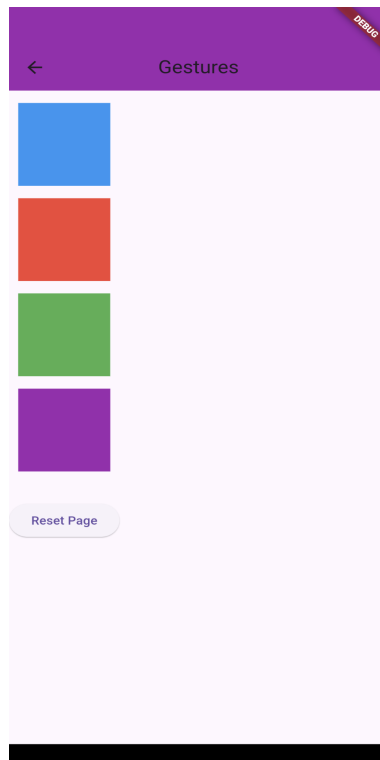


Figure 4.12: Fifth Page

The main important widgets of this page are listed in Table 4.6:

Widget/Component	Question 1	Question 2	Question 3
GestureDetector	✓		

Table 4.6: Fifth Page Analysis

1. **GestureDetector:** In Flutter the widget GestureDetector is accessible by default. The screen reader is able to recognize simple gestures and explains them to the user, for example for a double touch the screen reader says to tap twice to complete the gesture. Pay attention that for difficult gestures, for example pinch and zoom or others, the widget is not accessible and you have to use a Semantic widget with the property 'label' because the screen reader is not able to recognize the gesture and it says nothing to the user. Remember that for gesture is useful to instruct the user with a mini tutorial.

Other important widgets and components to consider are listed in Table 4.7:

Widget/Component	Question 1	Question 2	Question 3
Language	×	+1W +1P	10
Layout	✓		

Table 4.7: Other Considerations to analyze

1. **Language:** In Flutter the language is not accessible by default. It requires to wrap the Text widget with a Semantic widget with the 'attributedString' property. Code is in Listing 4.11

```

1   Semantics(
      attributedLabel: AttributedString(widget.title, attributes: [
        LocaleStringAttribute(
          range: TextRange(start: 0, end: widget.title.length),
          locale: const Locale("en"))
        ]),
6   header: true,
      child: Text(
        'Flaccess',
      ),
11  )
    \label{1st:Language}

```

Listing 4.11: Language Component

2. **Layout:** In Flutter all the layouts widget are accessible. We can use Container, Row, Column, Expanded, Center and other Layout widgets to group together different widgets and build an accessible application. It is not necessary to add properties or other stuffs because we use them only as containers of other widgets.

Chapter 5

Conclusions

In this thesis, we conducted a comprehensive analysis of the accessibility guidelines defined by the [Web Content Accessibility Guidelines \(WCAG\)](#), with a particular focus on their applicability to the mobile context. The purpose was to evaluate if the current guidelines are sufficient to address the unique challenges posed by mobile applications and to identify areas where the guidelines might be not sufficient to cover all the problems. When we noticed deficiencies, we proposed potential enhancements and additional guidelines to better serve the needs of mobile users. Our study started with an in-depth examination of the existing WCAG documentation. We aimed to determine if the guidelines as currently documented adequately cover the accessibility issues specific to mobile applications. Through this analysis, we identified several areas where the guidelines were either insufficient or not fully applicable in the mobile context. In response, we proposed new guidelines and modifications to the existing ones, studying new ideas for the future. Following the theoretical analysis, we turned our focus to the practical implementation of these guidelines using the Flutter framework. Flutter provided a relevant platform for our study. We structured our investigation around three primary research questions:

1. *Research Question (RQ1)*: Are the default widgets and components in Flutter accessible? This question aimed to assess the inherent accessibility features provided by Flutter and identify any default components that may pose accessibility challenges;
2. *Research Question (RQ2)*: What is necessary to make non-accessible widgets or components accessible? We determine the number of additional properties and widgets required to enhance accessibility;
3. *Research Question (RQ3)*: Specifies how many lines of code the not accessible widget

or component needs to become accessible;

For doing so we created an application in Flutter to study with the screen reader the available widgets and components. Our analysis showed that most widgets in the framework are accessible by default and the ones not accessible by default needs only a maximum of one additional widget and/or one additional property except in particular cases as the *language* component for example.

In conclusion, accessibility of mobile applications is not only a compliance requirement, but also an ethical decision that enhances users experience. By ensuring that mobile applications are accessible to everyone, developers can reach a more extensive user base. In this context, Flutter provides good solutions to fit accessibility but there are some not covered points which are important for a total understanding of the final user.

Acronyms and abbreviations

ADA Americans with Disabilities. [3](#)

AGID Agenzia per l'Italia Digitale. [4](#)

API Application Program Interface. [56](#)

EIT Electronic and Information Technology. [3](#)

IDE Integrated Development Environment. [61](#)

ISO International Organization for Standardization. [6](#)

SLR Systematic Literature Review. [6](#)

UI User Interface. [60](#)

W3C World Wide Web Consortium. [9](#)

WCAG Web Content Accessibility Guidelines. [9](#), [60](#), [75](#)

WHO World Health Organization. [2](#)

Glossary

ADA Prohibits discrimination based on disability and establishes accessibility requirements for public and private entities. [3](#)

AGID Monitoring and intervention in Italy towards noncompliers, has within it the ombudsman for digital, has guidelines on the accessibility of IT tools from WCAG 2.1 and has accessibility declaration and self-assessment model. [4](#)

API In informatics, an API is a set of procedures available to programmers, typically grouped to form a toolkit for a specific task within a program. Its purpose is to provide an abstraction, usually between hardware and the programmer or between low-level and high-level software, simplifying the programming process. [56](#)

EIT It is information technology (IT), as defined at FAR 2.101, and any equipment or interconnected system or subsystem of equipment that is used in the creation, conversion, or duplication of data or information. [3](#)

IDE It is an application that provides various tools for software development , in particular a source code editor, development automation modes , and a debugger . Some environments also include a compiler. [61](#)

ISO It brings global experts together to agree on the best way of doing things – for anything from making a product to managing a process. As one of the oldest non-governmental international organizations, ISO has enabled trade and cooperation between people and companies the world over since 1946. The International Standards published by ISO serve to make lives easier, safer and better. [6](#)

SLR It is an independent academic method that aims to identify and evaluate all relevant literature on a topic in order to derive conclusions about the question under consideration. [6](#)

W3C It is an international consortium where Member organizations, a full-time staff, and developers work together to define web standards. W3C primarily pursues its mission through the creation of web standards and guidelines designed to ensure current and long-term growth for the web. It aims to spread the culture of Internet accessibility. [9](#)

WCAG It was developed through the W3C process in cooperation with individuals and organizations around the world, with a goal of providing a single shared standard for web content accessibility that meets the needs of individuals, organizations, and governments internationally. The WCAG documents explain how to make web content more accessible to people, focusing on people with disabilities. [9](#), [75](#)

WHO Founded in 1948, WHO is the United Nations agency that connects nations, partners and people to promote health, keep the world safe and serve the vulnerable – so everyone, everywhere can attain the highest level of health. WHO leads global efforts to expand universal health coverage. We direct and coordinate the world’s response to health emergencies. And we promote healthier lives – from pregnancy care through old age. Our Triple Billion targets outline an ambitious plan for the world to achieve good health for all using science-based policies and programmes. [2](#)

Bibliography

- [1] *Accessibility Principles*. URL: <https://www.w3.org/WAI/fundamentals/accessibility-principles/>.
- [2] *ADA*. URL: <https://adata.org/learn-about-ada>.
- [3] *Agenzia per l'Italia Digitale*. URL: <https://www.agid.gov.it/en/intervention-areas/accessibility-usability>.
- [4] *Americans with Disabilities*. URL: [https://adata.org/factsheet/ADA-overview#:~:text=The%20Americans%20with%20Disabilities%20Act%20\(ADA\)%20became%20law%20in%201990,open%20to%20the%20general%20public..](https://adata.org/factsheet/ADA-overview#:~:text=The%20Americans%20with%20Disabilities%20Act%20(ADA)%20became%20law%20in%201990,open%20to%20the%20general%20public..)
- [5] *Android Studio*. URL: <https://developer.android.com/studio?hl=it>.
- [6] *Architecture and Widget Flutter*. URL: <https://flutter.dev/docs/resources/architectural-overview>.
- [7] *Characteristics Flutter*. URL: <https://www.ionos.it/digitalguide/siti-web/programmazione-del-sito-web/cose-flutter/>.
- [8] Helen Petrie Christopher Power Andrè Freire and David Swallow. “Guidelines are only half of the story: accessibility problems encountered by blind users on the web”. In: (2012), pp. 433–442. DOI: [10.1145/2207676.2207736](https://doi.org/10.1145/2207676.2207736).
- [9] *Conformance Levels*. URL: <https://www.w3.org/WAI/WCAG21/Understanding/conformance#levels>.
- [10] *Conformance WCAG*. URL: <https://www.w3.org/TR/UNDERSTANDING-WCAG20/conformance..>
- [11] *Dart*. URL: <https://dart.dev/>.
- [12] *Dart Overview*. URL: <https://dart.dev/overview>.
- [13] *Dart Wikipedia*. URL: [https://it.wikipedia.org/wiki/Dart_\(linguaggio\)](https://it.wikipedia.org/wiki/Dart_(linguaggio)).

BIBLIOGRAPHY

- [14] *Disadvantaged classes of users*. URL: <https://web.dev/articles/what-is-accessibility?hl=it>.
- [15] *Flutter*. URL: <https://flutter.dev/>.
- [16] *Flutter Accessibility*. URL: <https://docs.flutter.dev/ui/accessibility-and-internationalization/accessibility>.
- [17] *Flutter for the mobile*. URL: <https://nix-united.com/blog/thepros-and-cons-of-flutter-in-mobile-application-developmen/#benefits>.
- [18] *Flutter Widgets*. URL: <https://docs.flutter.dev/ui/widgets>.
- [19] *Framework Flutter*. URL: <https://www.html.it/pag/377313/il-frameworkdettagli-tecnici/>.
- [20] *Google Pixel 7*. URL: https://www.hdblog.it/schede-tecniche/google-pixel-7_i5670/.
- [21] *Google TalkBack*. URL: https://en.wikipedia.org/wiki/Google_TalkBack.
- [22] *Legge Stanca*. URL: <https://www.tpgi.com/understanding-the-stanca-act-italys-digital-accessibility-law/>.
- [23] *Mac's accessibility*. URL: <https://support.apple.com/it-it/guide/mac-help/mh35884/mac>.
- [24] *Material and Cupertino libraries*. URL: <https://aurimas-%20deimantas.medium.com/p1-%20flutter-%20making-%20platform-%20specific-%20ui-%20mobile-%20applicationandroid-%20with-material-ios-with-b50bc958a31>.
- [25] Aline Grazielle Silva Reis Michael Crystian Nepomuceno Carvalho Felipe Silva Dias and André Pimenta Freire. "Accessibility and usability problems encountered on websites and applications in mobile devices by blind and normal-vision users". In: (2018), pp. 2022–2029. DOI: [10.1145/3167132.3167349](https://doi.org/10.1145/3167132.3167349).
- [26] *Microsoft Narrator*. URL: [https://en.wikipedia.org/wiki/Narrator_\(Windows\)](https://en.wikipedia.org/wiki/Narrator_(Windows)).
- [27] *Mobile Accessibility*. URL: <https://www.w3.org/WAI/standards-guidelines/mobile/>.
- [28] *Mobile device accessibility for the visually impaired*. URL: <https://dl.acm.org/doi/10.1007/s10209-017-0540-1>.
- [29] *Mobile Users*. URL: <https://www.statista.com/statistics/218984/number-of-global-mobile-users-since-2010/>.

BIBLIOGRAPHY

- [30] *Observation based analysis on the use of mobile applications for visually impaired users*. URL: <https://dl.acm.org/doi/10.1145/2957265.2961848>.
- [31] *Operability Principle*. URL: <https://www.boia.org/blog/wcag-2.1-principles-explained-operability>.
- [32] Andreas Orphanides and Chang Nam. “Touchscreen interfaces in context: A systematic review of research into touchscreens across settings, populations, and implementations”. In: *Applied Ergonomics* 61 (May 2017), pp. 116–143. DOI: [10.1016/j.apergo.2017.01.013](https://doi.org/10.1016/j.apergo.2017.01.013).
- [33] *Perceivability Principle*. URL: <https://www.boia.org/blog/wcag-2.1-principles-explained-perceivability>.
- [34] *Robustness Principle*. URL: <https://www.boia.org/blog/wcag-2.1-principles-explained-robustness>.
- [35] Shahid Hussain Sher Badshah Arif Ali Khan and Bilal Khan. “What users really think about the usability of smartphone applications: diversity based empirical investigation”. In: (2020), pp. 9177–9207. DOI: [10.1007/s11042-020-10099-x](https://doi.org/10.1007/s11042-020-10099-x).
- [36] *Smartphone issues*. URL: https://www.researchgate.net/publication/261551651_Design_Guidelines_and_Design_Recommendations_of_Multi-Touch_Interfaces_for_Elders.
- [37] *Smartphone Popularity*. URL: https://www.researchgate.net/figure/The-global-popularity-of-smartphones_fig1_347480307.
- [38] *Success Criteria*. URL: <https://www.davidmacd.com/blog/what-are-WCAG-success-criteria.html>.
- [39] *Understandability Principle*. URL: <https://www.boia.org/blog/wcag-2.1-principles-explained-understandability>.
- [40] *Understanding perceivable principle*. URL: https://courses.idrc.ocadu.ca/understanding1/1_perceivable.html.
- [41] *Understanding the four principles*. URL: <https://www.w3.org/WAI/WCAG22/Understanding/intro#understanding-the-four-principles-of-accessibility>.
- [42] *USA 2001: Section 508*. URL: <https://www.section508.gov/manage/laws-and-policies/>.
- [43] *VoiceOver*. URL: https://www.apple.com/voiceover/info/guide/_1121.html.
- [44] *W3C*. URL: <https://www.w3.org/>.

BIBLIOGRAPHY

- [45] *WCAG*. URL: <https://wcag.it/>.
- [46] *WCAG 2*. URL: <https://www.w3.org/WAI/standards-guidelines/wcag/>.
- [47] *WCAG 2.2 guidelines*. URL: <https://www.w3.org/TR/WCAG22/>.
- [48] *WCAG Mobile*. URL: <https://www.w3.org/TR/mobile-accessibility-mapping/>.
- [49] *Widget States*. URL: <https://www.html.it/pag/378347/stateful-estateless-widget-le-fondamenta-di-unapp-flutter/>.