Platform for widespread uptake of certified, accessible and easy-to-use AAL mobile apps in Europe

ALL Agreement: AAL-2014-073

D4.2 Condition-dependent style guide

18.05.2017

V1.5

Responsible:	HSLU
Authors:	(C1, C2, … Cn)
Date of issue:	12/07/16
Status:	In progress(Issued/ Accepted)
Dissemination:	Public

Version	Date	Author(s)	Description
V0.1	12.07.16	Pascal Keel	Initial version
V1.1	05.03.17	Johanna Ludwig	User Interface for elderly people
V1.2	15.03.17	Gabriel Gaminde	Change structure and add content to the document
V1.3	21.03.17	Gabriel Gaminde	Change structure and add content to the document
V1.4	12.05.17	Gabriel Gaminde	Comments from COOSS. Added the Zocaalo styling FrameWork.
V1.5	18.05.2017	Iraitz Manterola	Minor changes, spelling mistakes corrected

Table of contents

1 Introduction

The aim of this document is to provide a guide to create multimodal accessible graphical user interfaces for the Android platform with the aid of the ZocAALo Framework design elements.

The user interface is one of the most critical parts of an application. Accessible and usable design enables the software product to reach more customers in business, government, and education. A software interface must be designed to be easy-to-learn. The user interface needs to facilitate the natural learning process. Final users must be able to start using the software application in a short time. This kind of interface must be as inclusive as possible following "Design for All" principles including alternative interaction devices, suited for the physical capabilities of each user.

It is important to remark the relation between accessibility and usability. Usability is related to how easy a product, hardware or software based, is to use. Generally, usability is measured against five criteria: memorability, efficiency, errors, learnability, and satisfaction. Accessibility on the other hand, means access to all, regardless of technological and physical means.

These two concepts are closely related as many accessibility requirements also improve usability for everyone. Every person could be in a temporary state imposed by their current environment that results in an accessibility issue. For example a screen will be difficult to see to everyone in certain light conditions. This is called situational disability.

Taking this in consideration accessibility requirements should include:

- Requirements which are more specific to people with disabilities, like ensuring that an application works well with assistive technologies, such as screen readers or screen magnifiers or automatic speech recognition. These requirements are technical and relate to the underlying code rather than to the visual appearance.
- Requirements that are also general usability principles which are included in accessibility requirements because they can act as barriers to people with disabilities. For example and application that could be used only with voice commands because some people with visual disabilities can’t see the contents of the screen. In defining accessibility requirements, care is usually taken not including aspects that impact all people similarly, and only include aspects that put persons with disabilities at a disadvantage relative to a person without a disability.

There are usability standards that have to be used in order to fully adapt the user interface to all kinds of people. To build the ZocAALo Styling Framework the guides we have followed are:

- Google Material design Accessibility Guidelines[i]
- Web Content Accessibility Guidelines 2.0[ii]
- The BBC Standards and Guidelines for Mobile Accessibility[iii]

2 Conditions and barriers that affect usability

There isn't a real estimation of how many people with some kind of disabilities use applications, either web based or on their mobile devices. A valuable source of these estimations could be Website analytics but unfortunately these do not reveal this information.

We could estimate how many people could benefit from designs with some kind of disability by reviewing the statistics:

- 15.3% of the world's population have some form of disability[iv], which includes conditions that affect seeing, hearing, motor abilities and cognitive abilities.
- About 4% of the population have low vision, whereas 0.6% are blind.
- 7 to 12% of men have some form of colour-vision deficiency (colour blindness), and less than 1% of women do.
- Low-vision conditions increase with age, and half of people over the age of 50 have some degree of low-vision condition.

There are evidences that disability is clearly related to age, 70% of all disabled are over 45 years[v]. Worldwide, the fastest-growing population is 60 years of age and older. The zocaalo styling framework is going to be mainly centred in providing tools for the software developers to address the barriers for the use of mobile applications for the elderly people.

2.1 Most common age related impairments

2.1.1 Cognitive

The field of cognitive ageing research has made a lot of progress in understanding the basic cognitive changes that accompany the normal ageing process[vi]. Cognitive ageing can be understood as a scientific field in which theorists and researchers try to conceptualize and answer the following fundamental question: what happens to the cognitive system as we get old?[vii.] Cognitive gerontology focuses on how biological(particularly in the brain and central nervous system), demographic, social and lifestyle changes alter mental abilities and behaviour in old age[viii].

It is important not to overstress the biological factors like prolonged education, lengthy marriage to an intelligent spouse, complexity of workplace environment, higher income and personality factors have all been shown to affect maintenance of cognitive functioning in old age in addition to, and independent of their effects on health or general biological wellbeing [ix]. Similarly, individuals' cognitive status is known to decline with increases in the levels of depression, mild unhappiness, and the levels of stress that they cannot control.

Some vital aspect to take into consideration related to general cognitive decline in older adults are the following:

- Biological, demographic, social and lifestyle changes alter mental abilities and behaviour in old age.
- Memory is often affected first, and most prominently.
- There is a considerable variation amongst individuals in the rate of decline and the amount of decline. Anyway there is a consensus about a „global“ model of age-related changes in mental abilities: all cognitive skills (efficiency of memory, perceptual processes and problem solving) decline at similar rates because they all strongly depend on the maximum information processing speed, and this markedly declines in old age.
- Cognition can be divided into several domains and the division varies considerably in the literature. One of the most commonly used division classifies cognition in five domains, attention, intelligence, memory, language and speech.

Regarding memory and learning functions we have to consider some important evidences:

- Older people are limited in the resources they have available for encoding information into
- memory and then for retrieving information from memory. Reduced processing speed is based on the concept of mental slowing, meaning that cognitive performance is degraded when processing is slow because relevant operations cannot be successfully executed.
- Even small declines in sensory memory may result in a large decline in long-term memory.

- Spatial memory (ability to recall where objects are in relationship to each other in space) also
- appear to decline with age.
- The perception of poor memory can seriously harm older people's self-concept as well as their performance on many tasks, it causes fear of dementia and can even result in depression.
- Older people are more likely to work for accuracy at the expense of speed.
- The main change which occurs in the memory with ageing is difficulty in recovering information learnt, not in the skill of learning.

Selective attention and memory for newly acquired information also show age-related decrements. These deficits combine to impact performance because older adults are more likely to forget command names and often must search through (often cluttered) displays for the relevant information. Even in comparatively simple displays, older adults require more time to visually search an array of information for a particular target piece of information. This difference is increased as the complexity of the display increases.

2.1.2 Visual

With advancing age, the normal function of eye tissues decreases and there is an increased incidence of ocular pathology. More than 82% of all people who are blind are 50 years of age and older. Demographic studies have shown that age is the best predictor of blindness and visual impairment.

There most common age-related low-vision conditions include the following:

- Presbyopia. A condition caused by the natural aging process that makes it more difficult to see very close. More than a billion people in the world were presbyopic as of 2005, according to the World Health Organization.
- Macular degeneration. Up to 50% of people are affected by age-related vision loss.
- Diabetic retinopathy. In people with diabetes, leaking blood vessels in the eyes can cloud vision and cause blind spots.
- Cataracts. They produce a clouding of the lens of the eye and decreases visual acuity.

- Retinitis pigmentosa. This inherited condition gradually causes a loss of vision.
- Pupil shrinkage. Resulting in the need for more light and a diminished capacity to adjust to changing light levels. For example, 60 year old retinas receive only 40% of the light that 20 year old retinas receivex while 80 year old retinas only receive around 15%.
- Reduction in visual field.

All of these conditions can generate problems for reading in short distance and they can also produce sensitivity to contrast, and in some cases reduce the ability to distinguish colours.

Colour-vision deficiencies, also called colour-blindness, are mostly inherited and can be caused by side effects of medication and age-related low-vision conditions. Nearly 8% of men and 0.5% of women will have some type of colour blindness, having difficulty with certain colour combinations so It is very important to take into consideration people with this visual deficiencies.

- Protanopia. Protans have difficulties distinguishing between blue and green colours and also between red and green colours. Pure reds cannot be seen, instead appearing black; purple colours cannot be distinguished from blues; more orange-tinted reds may appear as very dim yellows, and all orange–yellow–green shades of too long a wavelength to stimulate the blue receptors appear as a similar yellow hue. It is hereditary, sex-linked, and present in 1% of males.
- Deuteranopia is a type of colour vision deficiency where the green photoreceptors are absent. It affects hue discrimination in the same way as protanopia, but without the dimming effect. Like protanopia, it is hereditary, sex-linked, and found in about 1% of the male population.
- Tritanopia is a very rare colour vision disturbance in which there are only two cone pigments present and a total absence of blue retinal receptors. Blues appear greenish, yellows and oranges appear pinkish, and purple colours appear deep red. Unlike protanopia and deuteranopia, tritanopia and tritanomaly are not sex-linked traits and can be acquired rather than inherited and can be reversed in some cases.

- Achromatopsia. People with this condition cannot see colour at all, but it is not very common.

2.1.3 Auditive

Hearing is vital for communication. Decline in auditory function generally starts earlier than visual problems, and affects more people.

According to some studies xi10% of middle-aged adults suffer hearing losses, but in the case of people over 65 years old, the half of men (50%) and the 30% of women has hearing problems. Other studies also sayxii that the 75% of the people over 75-79 have audio impairments. So it is clear that the hearing problems appear with the age.

Older people usually notice a gradual age-related reduction and the increasing inability to hear highpitched sounds.The Royal National Institute for Deaf People (RNID) estimates for the UK that at around the age of 50 the proportion of deaf people begins to increase sharply and 55% of people over 60 are deaf or hard of hearing.

2.1.4 Motor

There is a general consensus that as people age, motor behaviours change such that, compared to younger adults, older adults take longer to make similar movements, their ability to maintain continuous movements declines, coordination is disrupted, and movements are more variablexiii.

The two most common age related motor conditions are Arthritis and Parkinson's disease.

Arthritis is a major cause of mobility issues for the elderly. The Us-based Arthritis Foundation reports that 50% of Americans over 65 experience arthritisxiv, while Arthritis Care in the UK report that 20% of all adults in the UK are affected[xv].

Another frequent age related condition is Parkinson's Disease, a progressive neurological condition affecting movements such as walking, talking, and writing. The National Institute of Neurological Disorders and Stroke in the US reports that the four primary symptoms of Parkinson's Disease are[xvi]:

- Tremor – trembling in the hands, arms, legs, jaw and face

- Rigidity – stiffness of the limbs and trunk
- Bradykinesia – slowness of movement
- Postural instability – impaired balance and coordination

Both arthritis and Parkinson's are likely to cause difficulties with the mouse use, and even other pointing devices, as well as keyboard and touchscreen use for some sufferers.

Regarding the use of tactile devices elderly commonly experience reduced sensitivity to tactile and vibrotactile stimuli. It is associated with a higher threshold for painxvii. Tactile sensitivity varies by body location. The detection threshold is lowest at the finger tips, lips and tip of the tongue and relatively high on the back of the hands and feet.

2.2 Generational and cultural barriers

A significant majority of older adults say they need assistance when it comes to using new digital devices. Just 18% would feel comfortable learning to use a new technology device such as a smartphone or tablet on their own, while 77% indicate they would need someone to help walk them through the process. And among seniors who go online but do not currently use social networking sites such as Facebook or Twitter, 56% would need assistance if they wanted to use these sites to connect with friends or family members[xviii]. Elder users have usually a skeptical attitude about the benefits of technology. Studies exploring computer use among the older adults, in general, indicate that the older the individual the less computer knowledge and interest they are likely to have[xix].

Some studies have been found that computer anxiety is positively correlated with agexx. However, some researchers argue that this relationship remains inconclusive and may not exhibit sufficient strength[xxi].

2.3 Impact on usage mobile aplications

The next table shows the type of impairment, its functional implications, impact of these implications when using an application on a mobile device and how to minimise their impact.

Impairment type	Functional implication	Impact on usage of mobile apps	Minimizing the impact
Visual	Focussing	Increasing difficulties to focus on fast moving objects	Only use animations when necessary and avoid complex paths and blinking elements (more than 3 times a second)
	Presbyopia	Inability to read small text, problems with close vision	Develop for scalable pixels so the user could use the zoom provided by the operating system.
	Bright-dark-adaption	Bright-dark-adaption decreases, increased sensitivity to dazzling	Follow the color rules regarding contrast and brightness.
	Colour perception	Difficulties distinguishing colours in blue-violett-scope	Follow the color rules regarding contrast and brightness.
	Text perception	Small font size is difficult to read	Develop for scalable pixels so the user could use the zoom provided by the operating system.
	Glaucoma	Tunnel vision, blindness.	Tag all the element in the GUI and always use clear caption so the user could use the app with the textt to speech engine provided by the OS.
Auditive	Presbycusis	High-frequency hearing loss, loss of ability to hear high-pitched sounds and sibilant consonants (ch, f, g, s, sh, t, th, z)..	Users could adjust the audio volume.
	Inability to hear low-pitched tones and a reduction in intensity of sound	This may affect the localization and perception of sounds, particularly those in the highest frequency ranges. Difficulty in hearing certain sounds.	Users could adjust the audio volume.
	Changes in auditory processing.	Listening in noisy conditions. This affects audio feedback on mobile applications.	Users could adjust the audio volume.
Motor	Reduced dexterity and fine motor control	Restricted dexterity and fine motor skills hinder operating and clicking of small symbols	The minimun size of touch targer should be 48 x 48 dp
Cognitive	Reaction time	More time to capture	Avoid the use of auto

		information is needed	closing infomation elements. For example a notification or pop up windows that auto closes after a number of seconds.
	Learning ability	Deceleration of uptaking new information and recalling previously learned knowledge	Provide reminders and alerts as cues for habitual actions.
	Reduced ability to ignore irrelevant information (selective attention)	Prerequisite for efficient behaviour, reading times of older adults were more negatively affected by the presence of distracting information than those of young adults, difficulty ignoring distractors embedded in text, comprehension of text and speech, attribution of memory. Some examples are (e.g. pop-ups, animations…).	Information should be presented in a clear way avoiding the use of technical words.
	Significant difficulties to divide attention (divided attention)	Reduced dual-task processing, general slowing in divided-attention situations.	Avoid splitting tasks across multiple screens if they require memory of previous actions.
	Age-related deficit in attention switching (Switching attention)	Difficult to maintain focus in the face of distraction.	Avoid using obstrusive elements. For example blocking notifications.
	Feeling of frustration when they cannot do what they need	They interact with virtual environment in a very similar way as they do with physical environment so they get easily frustrated.	Uunderstanding what the users' mental models are (with task analysis, observations, interviews, etc.), or designing a conceptual model to fit the users' mental model (interface design, iterations, validation testing, etc).
	Difficulties for	Difficulties to work with deep	Avoid using deep

	memorizing	hierarchies, lots of actions or lots of information. For example in menus or pages with too much information.	hierarchies. Don't use too much information or elements in one screen at once.
	Spatial cognition declines	Difficulties to work with maps or complex objects.	It is better to use waypoints or landmarks than maps.
Economic resources	Elderly people have usually lower income	Missing funds for acquiring a mobile device.	
Social resources	Lack of resources for training elderly people into using technology	Missing support out of the social environment for learning or using a mobile device.	
Technical biography	Elderly people are usually new to mobile technology	Elderly people were not socialized in their youth or professional life with smartphones and tablets.	
Personal attitude	Afraid of technology	Due to little technology experience, afraid to use new technologies or afraid to break something.	
	Skeptical attitude about the benefits of technology	Elderly people rate technology based on their direct benefit, hardware and software have to answer benefit expectations.	
	Lack of experience with technology	Problems to understand new technologies related words and images (icons).	

Table 1: Berries for elderly for using technologies

2.4 Units and measurements

Before describing in detail the zocaalo Styling Framework it is important to explain some basic concepts regarding screens in electronic devices.

Some units have different meanings in different contexts (OS) while others are not related to them. For example millimetres or inches are based on the physical size of the screen and are not related to any platform. When implanting a GUI it is important to know some concepts regarding metrics.

2.4.1 Pixel density

The number of pixels that fit into an inch is referred to as “pixel density.” High-density screens have more pixels per inch than low-density ones. As a result, UI elements (such as a button) appear physically larger on low-density screens and smaller on high-density screens.

Screen resolution refers to the total number pixels in a display.

screen density = screen width (or height) in pixels / screen width (or height) in inches

2.4.2 Density-independent pixels (dp)

"Density independence" refers to the uniform display of UI elements on screens with different densities.

Density-independent pixels (pronounced “dips”) are flexible units that scale to uniform dimensions on any screen. When developing an Android application, it is recommended to use dp to display elements uniformly on screens with different densities.

A dp is equal to one physical pixel on a screen with a density of 160. To calculate dp:

dp = (width in pixels * 160) / screen density

When writing CSS (Cascading Stylesheets), px must be used wherever dp or sp is stated. Dp only needs to be used in developing for Android.

In this table there is an example between screen density, screen width in pixels and screen width in density-independent pixels. If we have three screens, all 1.5 inches wide, with varying screen densities, the screen width will still be 240dp for all of them.

Screen density	Screen width in pixels	Screen width in density-independent pixels
120	180 px	240 dp
160	240 px	
240	360 px	

2.4.3 Scaleable pixels (sp)

When developing for Android, scalable pixels (sp) serve the same function as dp, but for fonts. The default value of an sp is the same as the default value for dp.

The primary difference between an sp and a dp is that sp preserves a user's font settings. Users who have larger text settings for accessibility will see the font size matched to their text size preferences.

2.4.4 Designing layouts for dp

When designing layouts for the screen, an element‘s measurements must be calculated in dp:

dp = (width in pixels * 160) / density

For example, a 32 x 32 px icon with a screen density of 320 equals 16 x 16 dp.

2.4.5 Image scaling

Images can be scaled to look the same across different screen resolutions by using these ratios:

Screen resolution	dpi	Pixel ratio	Image size (pixels)
xxxhdpi	640	4.0	400 x 400
xxhdpi	480	3.0	300 x 300
xhdpi	320	2.0	200 x 200
hdpi	240	1.5	150 x 150
mdpi	160	1.0	100 x 100

-

Table 2: Image scaling

3 Accessibility guidelines

3.1 Principles

Zocaalo Styling Framework is based on three basic principles. These principles are general to the design and development of inclusive and usable application and website applications for all.

The three basic principles are:

- Platform and web standards will be used as intended
- Standard user interface controls will be used where possible
- Support platform accessibility

3.2 Platform and web standards will be used

The zocaalo Styling Framework will always use web and platform specific standards as intended. These must be used because when standards and guidelines are implemented using non-standard platform techniques (for example not labeling certain elements for text readers using the platform specific feature) there is a risk that users who are dependent on platform specific accessibility features, such as accessibility settings or screen readers, could not access the content.

There are various degrees of visual impairment so people accessing the software application may view it in a number of different ways. Some may need to have the screen zoomed in, some may need to have the colour settings changed to high contrast and others will also require the help of a screen reader (Android provides this functionality as an in-built accessibility feature).

3.3 Standard user interface controls will be used where possible

Standard UI controls, objects, and elements will be used to ensure a greater level of accessibility. Custom controls tend to not implement accessibility as fully as standard platform controls.

3.4 Support platform accessibility

All content and functionality will work alongside, and not suppress, native accessibility and features and settings.

3.4.1.1 Navigation methods

The users will have to be able to navigate and access all the content using the platforms navigation paradigm for assistive technology.

For example, the directional controller will be supported on Android to allow users of the TalkBack screen reader to review and navigate page content. Android requires that all elements be keyboard accessible so they can be accessed with a d-pad or track ball. Android 4.0 has lessened this requirement a bit by including an “Explore by Touch" method.

3.4.1.2 Support platform assistive technologies or features

The applications built with components from The Zocaalo Styling Framework will not block, disable, or interfere with platform specific accessibility features or technology as users with disabilities may not be able to use the app.

Some users with disabilities may require multiple accessibility features because they have multiple disabilities. For example, a user may be deaf and blind or may have low vision and unable to use a pointing device or touch screen. Multiple modes of operation will be supported allowing users to access content according to their preferences.

3.5 Graphical User Interface

3.5.1 Styles and themes

A style is a collection of attributes that specify the look and format for a View (Android) or window. A style can specify attributes such as height, padding, font colour, font size, background colour, and much more. Depending on the OS (Operative System) environment a style could be defined in different formats. For example in a Web app it is defined with CSS files that could be separated or included in the html file that specifies the layout. In Android a style is defined in an XML resource that is separate from the XML that specifies the layout.

Core content will be accessible when styling is unsupported or removed. For example older mobile devices may have poor support for fonts, colours and styles. Additionally, assistive technology cannot draw meaning from styling as some users will change settings (fonts, styles, colours, etc.) to suit their needs.

On the other hand we have the concept of themes. Themes and styles are both declared in exactly the same way, the difference comes in how they're used.

Themes are meant to be the global source of styling for the app. So in short themes are global (all the app) and styles are local (views).

As we said styles are meant to be applied at a view level so internally, a style is set on a View, the LayoutInflater will read the style and apply it to the AttributeSet before any explicit attributes (this allows you to override style values on a view).

Values in an attribute set can reference values from the View's theme.

3.5.2 Colour

Colour is arguably the second most important aspect of a software application, after functionality. The human to computer interaction is heavily based on interacting with graphical UI elements, and colour plays a critical role in this interaction. It helps users see and interpret the software application's content, interact with the correct elements, and understand actions. Every software application has a colour scheme, and it uses the primary colours for its main areas. A good colour contrast also accommodates lower specification devices with poor colour support and assists all users in a brightly lit environment.

3.5.2.1 Colour contrast

Good colour contrast is also essential when using colour as a differentiator, for example, when colour is used to indicate the presence of a link or a selected tab with text. The colour difference between the link text and non-link text must have sufficient contrast.

There is not a formal rule regarding colour contrast in mobile devices applications so the accepted recommendation is that the (Web Content Accessibility Guidelines) WCAG 2.0 Level AA contrast ratio must be met or ideally exceeded. WCAG 2.0 is divided into three conformance levels (A-AA-AAA). The higher the level, the more restraining it becomes on design. Success criteria from level A should be invisible or barely noticeable to the interface. On the other hand, level AAA will have such a high impact on design, that even the W3C claims that most organizations will not be able to achieve that level (as the compromises on design will be too important) "Note 2: It is not recommended that Level AAA conformance be required as a general policy for entire sites because it is not possible to satisfy all Level AAA Success Criteria for

some content"[xxii]. For that reason the zocaalo Styling Framework will follow at least the AA conformance level which rules are:

- Small text will have a contrast ratio of at least 4.5:1 against its background.
- Large text (at 14 pt bold/18 pt regular and up) will have a contrast ratio of at least 3:1 against its background.
- Icons or other critical elements will also use the above recommended contrast ratios.
- Decorative elements (such as logos or illustrations) won't have to meet contrast ratios but they will be distinguishable if they possess important functionality.
- Colourblindness takes different forms (including red-green, blue-yellow, and monochromatic). The GUI must use multiple visual cues to communicate important states. For example elements such as strokes, indicators, patterns, texture, or text to describe actions and content.

The zocaalo Styling Framework components will use a palette of colours to ensure the foreground and background colours provide sufficient contrast following the WCAG 2.0 Level AA colour contrast guidelines.

If we plan to use different colours schemes for the application and not the recommended ones it is vital to check the contrast ratios during the GUI design. To check the contrast ratios there are several free checkers. Some checkers can also calculate what it would look like for different levels of colour blindness, and if it has enough contrast for that as well.

Some free tools to test the colour contrast ratio for accessibility are:

- Colour Contrast Analyzer (CCA). This standalone application is available for Windows and OSX. Apart of calculating the colour contrast ratio for accessibility is also capable of checking different levels of colour blindness (Deuteranopia, Protanopia, Tritanopia). (https://www.paciellogroup.com/resources/contrastanalyser/)
- Accessibility Colour Wheel. This online tool is also capable of checking the colour contrast ratio for accessibility and also different levels of colour blindness (Deuteranopia, Protanopia, Tritanopia). (http://gmazzocato.altervista.org/colourwheel/wheel.php)

- ACE. The Accessibility Colour evaluator. This online tool brings the same functionality as the other two tools (Deuteranopia, Protanopia, Tritanopia). (http://daprlab.com/ace/)
- Colour Contrast Check (snook.ca). This online tool capable of checking the colour contrast ratio for accessibility. (https://snook.ca/technical/colour_contrast/colour.html#fg=33FF33,bg=333333)

3.5.2.2 Colour and meaning

As it was told before, nearly 8% of men and 0.5% of women will have some type of colour blindness, having difficulty with certain colour combinations, so it is very important not using only colours as the only way information is conveyed.

Some of the reasons for this, apart from colour blindness, are that colours can be difficult to distinguish in bright sunlight and cannot be perceived by users who are colour blind, or vision impaired. Also screen readers do not detect colour and some users will change colour settings for their whole computer. For example, setting their computer to greyscale or applying a tint to help with reading. Lower specification mobile devices also offer poor colour support.

Additional visual and non-visual methods of identifying information or meaning will be applied to support the use of colour like the ones listed below:

- Visual cues as, text attributes with suitable mark up (such as underline, bold or italic), patterns, or icons with suitable alternative text will be used.
- Non-visual cues will be used, which are programmatically available to assistive technologies, could be element tags, hidden text or suitable labels, for example ARIA or UILabel.

3.5.3 Fonts

It is recommended that the font size is at least 12-14 point. So at least the font size must be 12 points. Anyway users must be able to increase font size to improve readability and to adapt the text size to their needs. Mobile devices and browsers include features to allow users to adjust font size system-wide. To enable system font size in an Android app, mark text and their associated containers to be measured in scaleable pixels (sp).

All users benefit when they can adapt the size of content to see and read it. This may be a constant or temporary adaption due, for example, to screen size, screen glare, or vision impairment.

In order to increase legibility, text lines will be separated so that there is at least a 1 stroke width separation between the top of diacritical marks (accents) on upper case letters (e.g. the top of the umlaut on "Ü") and the bottom of descenders (e.g. the downward stroke of "g"). Increased separation improves reading speed.

Characters must be clear and legible. So text will use lower case lettering, with capitals used for the beginning of sentences and in accordance with National language conventions

The choice of typeface is generally less important than size and contrast. It is preferable to use typefaces that people are familiar with and will recognize easily. The zocaalo Styling Framework will avoiding italic, simulated handwriting and ornate typefaces as these can be difficult to read particularly by users with impaired vision.

People with sight problems often prefer bold or semi-bold weights to normal ones. We will avoid light type weights.

A good choice of typeface could be Arial or Helvetica but other fonts could be used if they follow the previous rules.

The use of a different language than mother-language (original language that user learnt when was a child) is discouraged, for this reason, the interface will be internationalised (translated) and will be dependent on the user profile. This is particularly important in countries like Spain where some languages (Spanish, Galician, Catalan, and Basque) coexist or like Switzerland where French, German, Italian and some other languages are spoken officially.

The content will respond to platform native text resizing, ans scaling (or "zoom") will be supported.

3.5.4 Actionable elements

All users must be able to determine if an element is actionable or if it is static content. Actionable elements are links, buttons, navigation and other control features, including swipe areas on touch devices that might be less obvious.

Actionable elements will be identified visually, by convention, and by information provided to assistive technologies. This will be achieved by using platform native elements such as links, buttons, inputs, etc. All elements will have clear labels and, when applicable, a suitable role or trait. See 'Roles, traits and properties' for further information.

Users will be able to control interfaces naturally and intuitively. This is the case where realistic controls are used, such as toggle-buttons and sliders, users expect to interact with them in a literal and familiar way.

Regarding links that solely use text sometimes text that is big enough to read is too small to touch. For example, a linked letter in an A-Z listing would be too fine to touch accurately and should be placed in a linked container to increase the target area. So the text size should be big enough for the users to be able to touch it without problems. Hover states will only act as confirmation that an element is actionable.

Some styling of the actionable elements will use:

- Underlines for links inline with text.
- Colour highlights and weight variants to make inline actionable elements clear. The colour difference between the link text and non-link text must have sufficient contrast.
- Visible state changes to indicate focus (can be same as hover).
- Arrows to indicate direction of swipe areas.
- Try avoiding scrolls as much as possible.

3.5.5 Touch targets size

Touch targets are the parts of the screen that respond to user input (buttons, interactive controls, etc). They extend beyond the visual bounds of an element. For example, an icon may appear to be 24 x 24 dp (Density-independent Pixels), but the padding surrounding it comprises the full 48 x 48 dp touch target. Density-independent Pixels is an abstract unit that is based on the physical density of the screen. These units are relative to a 160 dpi screen, so one dp is one pixel on a 160 dpi screen.

Keeping this is in mind touch targets will be at least 48 x 48 dp. A touch target of this size results in a physical size of about 9mm, regardless of screen size. The recommended target size for touchscreen elements is 7-10mm. This size is

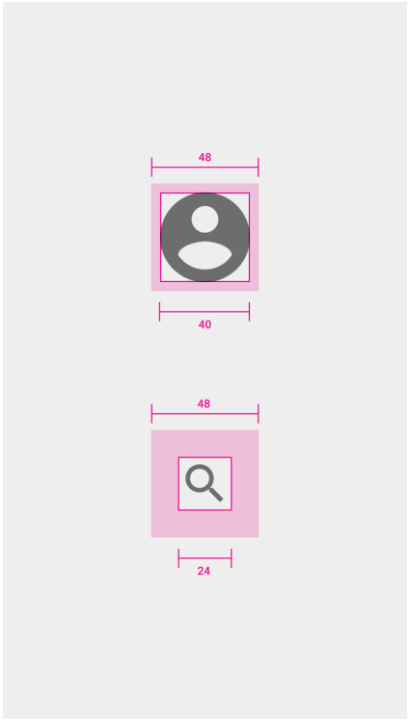
equivalent to the smallest average finger. It may be appropriate to use larger touch targets to accommodate a larger spectrum of users when possible.

3.5.6 Touch target spacing

Actionable elements will not touch or overlap, and there will be an inactive space between actionable elements in order to reduce the risk of activating the wrong control.

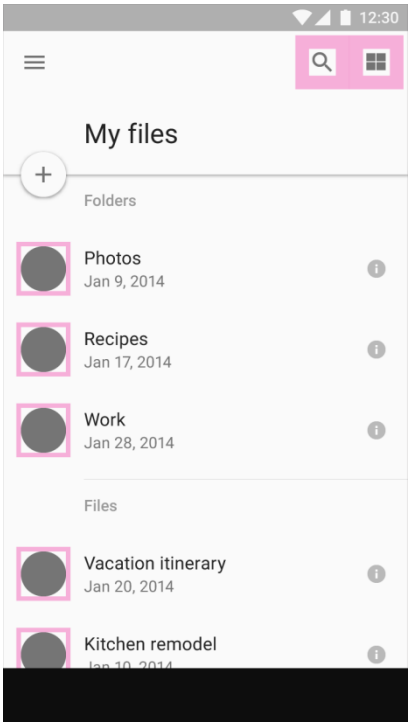
In most cases, touch targets should be separated by 8dp of space or more to ensure balanced information density and usability.

Below we have some examples of touch targets and buttons on a smartphone screen.

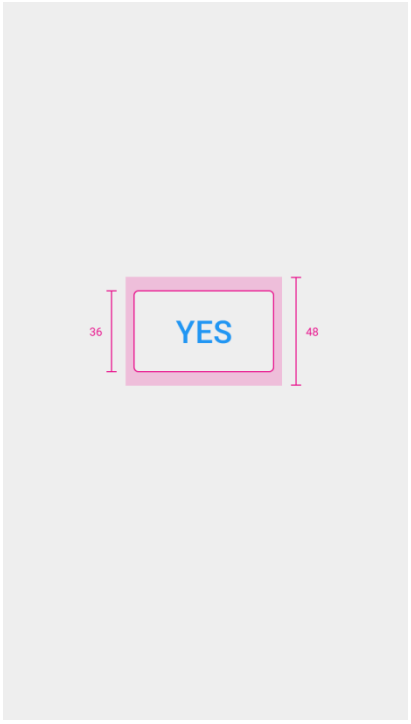


Avatar: 40dp

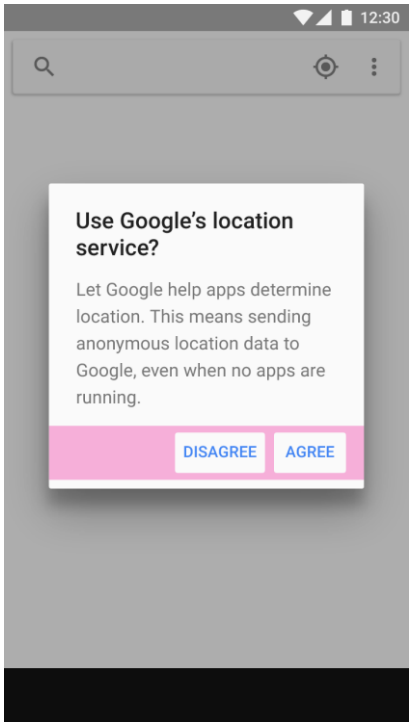
Icon: 24dp
Touch target on both: 48dp



Examples of touch targets



Touch target height: 48dp
Button height: 36dp



Buttons on a popup window

Figure 1: Style examples

3.6 Layout

The zocaalo Styling Framework will follow the specific Android usability section in their Material design guidelines.

Material design's touch target guidelines will enable users who aren't able to see the screen, or who have motor-dexterity problems, to tap elements in the app designed with the zocaalo Styling Framework.

3.7 Information architecture

It is very important to take into consideration the hierachy of the elements on the screen.

Items must be placed on the screen according to their relative level of importance.

Important actions must be placed at the top or bottom of the screen (reachable with shortcuts).

Related items of a similar hierarchy must be placed next to each other.

Regarding the text content is crucial to avoid using humour, jargon and too informal language and the use of foreign language terminology (including English).

The app should always provide a clear exit or back button. Small problems can escalate to abandonment when user cannot find an exit or back button.

3.8 Consistency

Consistency allows all users to predict where to find information and how to use it. This is particularly helpful for users with cognitive impairments.

Consistent and logical structure and language help all users understand where they are and how to navigate or perform a task. Consistent and logical layout helps both sighted and non-sighted users predict where they should touch or interact.

For example, navigational aids such as back buttons should consistently move the user back to the previous step and act as a breadcrumb trail.

The look and sound of a control, object or element should inform the user how to interact with it.

3.9 Grouped elements

It is easier and quicker for people using a keyboard or screen reader to interact with content when not overwhelmed and confused by extraneous elements. Grouping elements into a single overall control makes things clearer, simplifies interactions, and can provide larger touch targets.

Related content should be arranged into groups so that accessibility services announce the content in a way that reflects its natural groupings. Users of assistive technology then don't need to swipe, scan, or wait as much to discover all information on the screen.

The two most effective methods of grouping related content are the following:

- For smaller or simpler groups of content is best to organize all content into a single announcement.
- For larger or more complex content structures, is recommended to create natural groupings for the content.

3.10 Unique page/screen titles

An important rule is that all pages or screens in the app will be uniquely and clearly identifiable.

The page/screen title is often the first thing people will see or hear and acts as a confirmation of where they have arrived at, helping people orientate themselves within websites and apps. It is particularly helpful for vision impaired users who cannot perceive the whole page/screen at once.

Page titles are standard for HTML. Apps have the facility to add screen titles. However, when visible space is in short supply other means may be used to identify location, such as a logo on a homepage, the first item of content presented as a heading, or a selected tab on top navigation.

3.11 Navigation

The GUI must enable Focus-based navigation so users could navigate the screen layouts using hardware-based or software directional controls (D-pads, trackballs, keyboards and navigation gestures). In some cases the user interface components will have to be made focusable or the focus order changed to be more logical for user actions.

All interactive elements must be focusable and accesible and non-interactive elements must not be focusable.

Depending on the user needs some people may only use a keyboard, switch device or voice control for navigation and input.

Any menu or drawer component that opens from a user action may follow the modal pattern, or may automatically close or dismiss the component and return focus to the trigger element after the user moves focus onward from the last element. For example, a drop-down menu, side-drawer menu, or accordion panel control.

3.12 Notifications

Messages and notifications must be descriptive enough to be understood by people with different academic grades, people that are not used to read, etc.

It is recommended to provide multimodal feedback: Audio feedback must always have a secondary feedback mechanism to support users who are deaf or hard of

hearing. For example, a sound alert for the arrival of a message must be accompanied by a system Notification, haptic feedback (if available) or other visual alert.

3.13 Sound

Android and other OS support Text to speech natively. Android includes TalkBack which provides spoken, audible, and vibration feedback to current Android devices.

In order to provide sound and text to speech functionalities to applications built with the zocaalo Styling Framework some rules will be applied:

- Give visual alternatives to sound, and vice versa. Provide closed captions, a transcript, or other visual alternatives to critical audio elements and sound alerts.
- Allow users to navigate the app using sound by adding descriptive labels to UI elements. When using a screen reader such as TalkBack and navigating by touch exploration, labels are spoken aloud when users touch UI elements with their fingertips.
- There will not be any unnecessary sounds that play over a screen reader, such as background music that autoplays when entering a web page.
- Extra sounds will not be added to native elements (screen readers will be able to interpret native elements correctly).

Text to speech rate should be 140 words per minute or less as some experts recommend.

The volume adjustment must be configurable from the App from the according to each users needs.

3.14 Animations

Animations could be used in the GUI but some rules must be followed.

- Animating an element should be functional. Every use of animation should be purposeful. Animation without purpose only has the potential to distract.
- Animations should be quick and unobtrusive. Great care should be taken in not distracting users or cause discomfort with obtrusive or visually disconcerting animations. Animations grab attention easily. The pleasant

surprise of first noticing an animation quickly wears off if it makes routine actions take longer to complete.

- Animations should take a short path. When animating a panel for a portrait-oriented mobile device, sliding in from the side is less distracting than sliding in from the bottom. There's less distance to cover, so the animation can be completed more quickly without needing to move any faster.
- Animation paths should be consistent. Modals and panels flying in from multiple directions are shocking and distracting. The assumption that multiple paths will lead to a spatial understanding of an application's patterns is not true. It's easier for a user to understand and expect that all overlays appear from the left than to process that modals slide from the top, tooltips from the right, and navigation menus from the left. When it comes to animation, surprises are obtrusive and disorienting.
- Avoid the use of several transformations at once, excluding opacity. Slide or grow, but don't do both. Animations with multiple transformations are a red flag for potential friction with users.
- Avoid rapidly flashing content. If used it shouldn't flash more than three times a second as it may trigger epileptic seizures in individuals with photosensitive epilepsy.

Some element could benefit from animation. For example:

- Modals opening and closing. Typical modals are centrally placed and disconnected from the viewport's edges. It is better to fade modals in and out rather than sliding them as this element won't benefit from motion.
- Scrolling. Clicking an anchor link and immediately jumping halfway down the page can cause a user to lose their spatial context. A smoothly-eased scroll, neither too fast nor too slow, can be useful as a hint to your user of where you've taken them.
- Changing content. Adding or removing a small amount of content from view can be easy to miss. An animation can both draw attention to the change and maintain a user's spatial context if there's jumpiness to the layout resulting from the content change. For this reason, components such as accordions and error messages often benefit from subtle animation.
- Feedback for user interaction. Any dragging or touch input should be smooth and follow the path of a user's gesture.

4 Zocaalo Styling Framework

The zocaalo styling Framework provides two libraries to the app developer which make the process of building accessible applications that fulfil the requirements of the zocaalo store target users much easier.

The custom views library is a collection of views and helper classes, which allow the adjustment of various attributes, e.g. colours and text formatting at application runtime.

The Styling Wizard library allows a developer to embed a wizard in his application in order to create themes which dictate the custom views looks.

In this chapter we will have a look at the main concept behind these libraries, the core functions of each of the libraries and how they work together.

4.1 Custom views

The custom view library is a collection of custom views, helper classes and interfaces. The custom views implement methods to adjust colour, font size, font type and the general theme. They will be discussed thoroughly in the next chapter. The helper classes implement methods to calculate, get and set the adjustable parameters. There are currently three helper classes for different kind of parameter types: ColorHelper, TextHelper and ShowCaseViewHelper. To allow multiple pre-sets, the adjustable parameters can be saved in a ZocaaloTheme and applied to all the ZocaaloViews at a later time. The ZocaaloThemes are saved as a List in the Shared Preferences. The following sub sections discuss the currently adjustable parameters (also shown in the following table). All of these can be set via Java setter methods or XML attributes of the specific view. These parameters are derived from the accessibility guidelines described in this document combined with experiences acquired by developing other AAL applications.

The developer could use the Zocaalo custom views for every view element that wants to make it adjustable (or just use them for everything and disable the listeners on the ones you don't want to be adjustable). Currently the following views exist:

- ZocaaloButton
- ZocaaloImageButton
- ZocaaloImageView

- ZocaaloProgressBar
- ZocaaloRadioButton
- ZocaaloCheckBox
- ZocaaloSpinner
- ZocaaloSeekBar
- ZocaaloTextView
- ZocaaloToolBar

These custom views can be customized using different xml attributes. The following snippet uses an implementation of the ZocaaloButton as an example. These attributes don't have to be set via xml, there are setter methods in Java as well.

```
<ZocaaloButton
    android:id="@+id/buttonTextSizeB"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="left"
    android:layout_marginLeft="8dp"
```

Color **Text** **ShowCaseVie w**

```
   android:layout_marginRight="8dp"
    android:layout_marginTop="8dp"
    android:layout_weight="1"
    android:text="@string/smaller"
    app:colorType="AccentDark"
    app:textType="Subtitle"
    app:fontListenerDisable="true"
    app:helperText="@string/helpertext_textsmaller" />
```

Each custom view can be customized in the exact same way as the ZocaaloButton in the example. Some implement different attributes than others though.

	colorListenerDisable	colorType	textColorListener	statusBarColorType	statusBarColorListenerDisable	textType	textListenerDisable	fontListenerDisable	helpeListenerDisable	helperText
ZocaaloActivity				✓	✓					
ZocaaloButton	✓	✓				✓	✓	✓	✓	
ZocaaloImageButton	✓	✓	✓						✓	
ZoccaloImageView	✓	✓	✓						✓	
ZocaaloProgressBar	✓	✓							✓	
ZocaaloRadioButton	✓	✓	✓			✓	✓	✓	✓	
ZocaaloCheckBox	✓	✓	✓			✓	✓	✓	✓	
ZocalloSpinner	✓	✓	✓			✓	✓	✓	✓	
ZocaaloSeekBar	✓	✓							✓	
ZocaaloTextView	✓	✓				✓	✓	✓	✓	
ZoocaloToolBar	✓	✓							✓	

Table 3: Attributes

Since not all attributes are quite self-explanatory, the following list provides a short description for each one.

4.1.1 colorListenerDisable

This parameter is used to disable the colorListener for Views that shouldn't adjust their colour. The colorListener is enabled (colorListenerDisable = false) by default.

4.1.2 colorType

With the colorType, the type of the color can be defined. For most elements there are six possibilities (Primary, PrimaryDark, PrimaryLight, Accent, AccentDark, AccentLight), although some elements only allow to choose between two (e.g. the progressbar, for the simple reason that it already uses all 3 variations of a color).

4.1.3 textColorListener

This parameter is used to enable the textColorListener and make a view listen to the text color corresponding to its defined colorType (meaning black or white, depending on the luminance of the color) instead of the colorType directly. It is mainly used for icons and disabled by default.

4.1.4 onlyPrimaryColor

This parameter disables the accent color for the view and forces it to adjust its color to the primary colour at all times. It is disabled by default.

4.1.5 statusBarColorType

This parameter is used to set the color of the status bar. The possible colors are: Primary, PrimaryDark, PrimaryLight, Accent, AccentDark, AccentLight.

4.1.6 statusBarColorListenerDisable

This parameter is used to disable the status bar colorListener, if the developer doesn't want the status bar to change its color. It is enabled (statusBarColorListenerDisable = false) by default.

4.1.7 textType

The textType parameter is used to define the textType (Title, Subtitle or Text) of all the text inside the view.

4.1.8 textListenerDisable

This parameter is used to disable the textListener for Views that shouldn't adjust their textSize. It is enabled (textListenerDisable = false) by default.

4.1.9 fontListenerDisable

This parameter is used to disable the font adjustment. The font type is, much like the text size, a global parameter, which can't be defined separately for every view element. It is enabled (fontListenerDisable = false) by default.

4.1.10 helperListenerDisable

This parameter is used to disable the helper framework (the ShowcaseViews). It is enabled by default.

4.1.11 helperText

With this attribute, the text to be shown in the ShowcaseView on a long click of the element can be defined.

4.1.12 Flow inside the custom view library

The following diagram illustrates what happens within the custom views library from the point when an attribute is changed, until the changes become visible to the user.

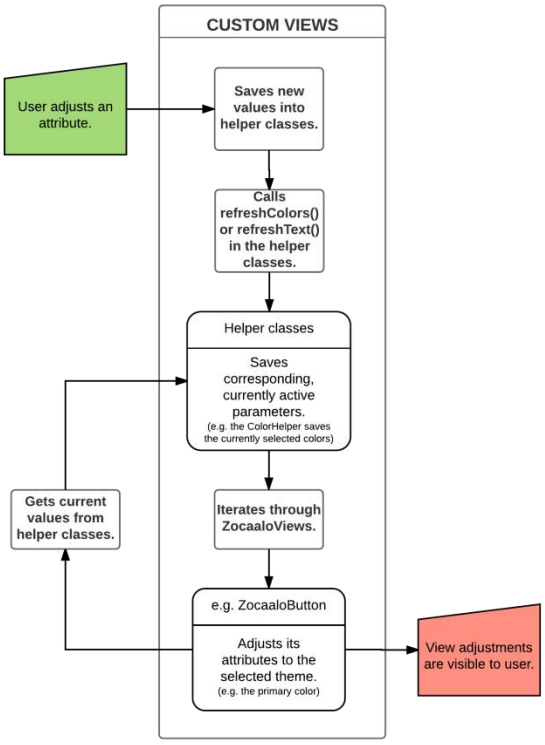


Figure 2: Flow inside a custom library

4.1.13 Styling Wizard

The Styling Wizard library consists of various fragments in order to create a wizard to adjust all the different parameters which make up a ZocaaloTheme.

There are 3 main types of parameters (similar to the helper classes in the custom view libraries): Color, text and the helper.

Additionally there is a ZocaaloWizardFragment to embed in an application which displays a list of all available ZocaaloThemes and allows starting the wizard and creating a new theme. Similarly there is a ZocaaloWizardActivity which can be used instead of the fragment or as a demo application.

The Wizard itself facilitates the adjustment of the following attributes.

4.1.14 Color

Currently the user can choose between 19 primary and 16 accent colours. Which view element adjusts to which colour can be defined by the developer. Once a colour has been chosen, its luminance can be adjusted to improve the visibility. If a view adjusts its background and has text on said background, it automatically adjusts the text color in a way that provides the best contrast. The standard to calculate whether to change the text colour to black or white, has been taken from the “W3C Techniques For Accessibility Evaluation And Repair Tools” (https://www.w3.org/TR/AERT, visited on March 14, 2016).

To save and distribute the chosen colour to the custom views, the ColorHelper class is used. This helper class saves the colour in the HSL format to allow an easy adjustment of the luminance. The class also implements the refreshColor() method, which can be used to update the colours of all ZocaaloViews.

4.1.15 Text

The way text gets displayed on ZocaaloViews is also adjustable in multiple ways. The text size can be set by the user (in density-independent pixels). The developer can set a minimum and a maximum to make sure the layout keeps its form more or less. Additionally one of the following text types can be defined: text, subtitle, title. While the subtitle type multiplies the text size by 1.2, the title type multiplies it by 1.4. There are also 4 font types to be chosen from: Normal, normal bold, serif and serif bold. All these parameters are managed by the TextHelper class, which is a lot like the

ColorHelper class: It saves, loads and distributes the text related attributes discussed above. The method refreshText() can be called to update the textsize and formatting in all ZocaaloViews.

4.1.16 Additional Information

Should you wish to update the views at any point in your code, the ColorHelper.refreshColor() and TextHelper.refreshText() methods can be called. These methods are also called at the start of each ZocaaloActivity. Additionally those helper classes implement various getter methods in case you need the different attributes for your own purposes. Check the columns of the tables above for an overview where to find which getter method.

4.1.17 Creating an adjustable view

To implement your own custom view, create a new class that extends the view you wish to use. Analyse what kind of adjustable parameters your view should have. To do so, you can simply go through the table in the last chapter one by one and choose the correct parameters for your view. Pay attention to the helper classes listed in the table and implement the corresponding interfaces in your view:

- Colorhelper -> ColorListener
- TextHelper -> TextListener
- ShowCaseViewHelper -> ShowCaseViewListener

To get access to this functionality the developer have to implement an updateView() method. In this method the attributes of the view can be adjusted. Different values can be obtained from the corresponding helper class, e.g. to get the primary color. To do the ColorHelper.getColorPrimary() must be called.

This pattern can be applied to most simple views. If the developer wish to implement a more complex view, some adjustment could be necessary. The custom views in the library could be used as a reference.

4.1.18 Helper framework

All of the ZocaaloViews implement a default onLongClickListener, which starts a ShowcaseView to highlight the clicked element. A helpertext can be defined for every element, which will be shown in the corresponding ShowcaseView. The

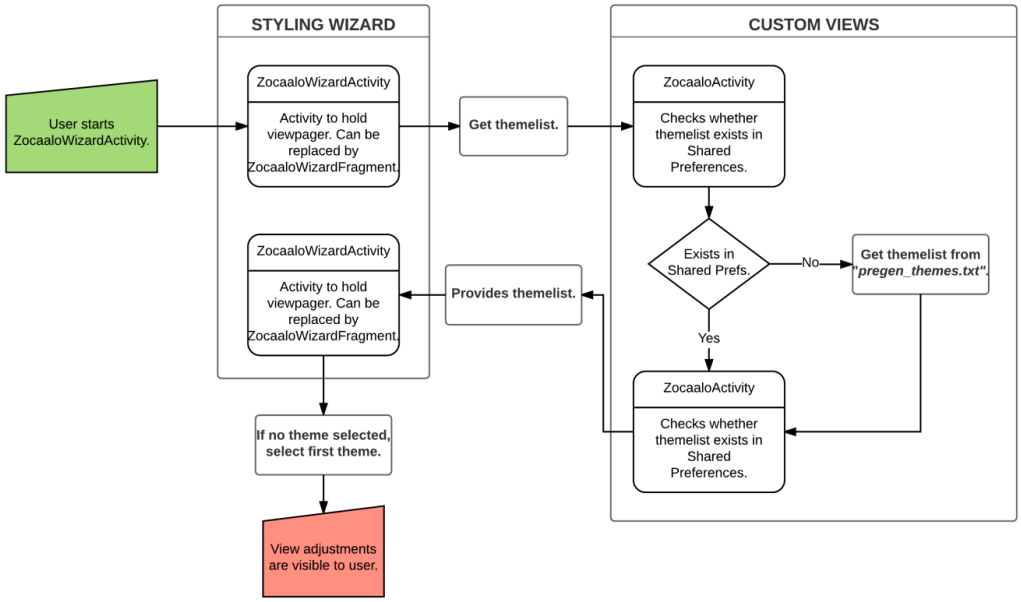
ShowcaseViews are created with the help of the ShowCaseViewHelper class. It also contains methods to set the dismiss text as well as the colortype of the ShowcaseViews.

4.1.19 Combining the libraries

The custom view library can be used on its own if you want to be able to adjust your views at runtime, but you'll have to provide your own ZocaaloThemes. The Styling Wizard library on the other hand can only be used in conjunction with the custom view library. If the developer chooses to embed a Styling Wizard in their app, the developer will simply add a ZocaaloWizardFragment or a ZocaaloWizardActivity (which adds the ZocaaloWizardFragment for you).

4.1.20 Loading themelist

Figure 3: Loading theme list



4.1.21 Choosing a theme

The following diagram illustrates the flow when the user choses another theme:

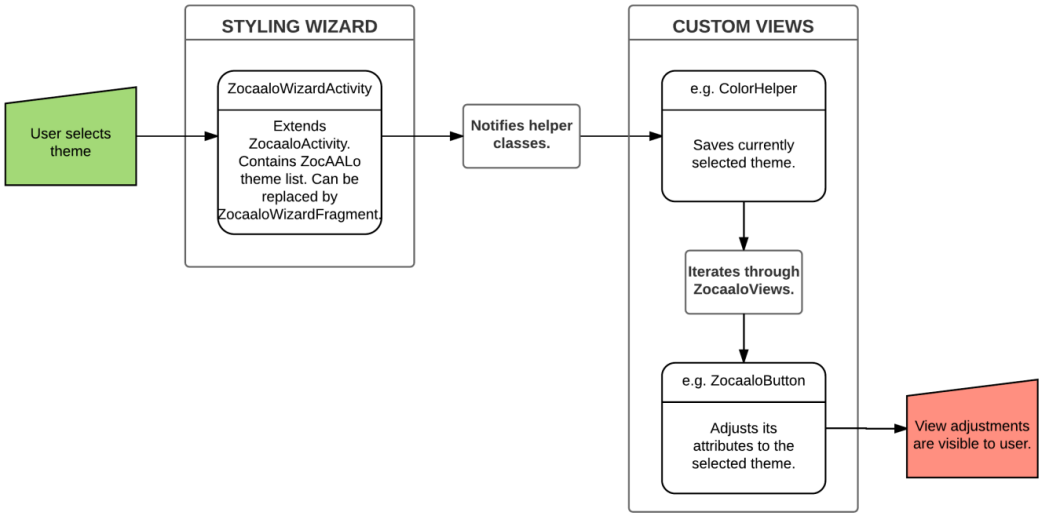


Figure 4: Choosing a theme

4.1.22 Creating/editing a theme

The next diagram illustrates the flow when the user creates a new theme or edits an existing one:

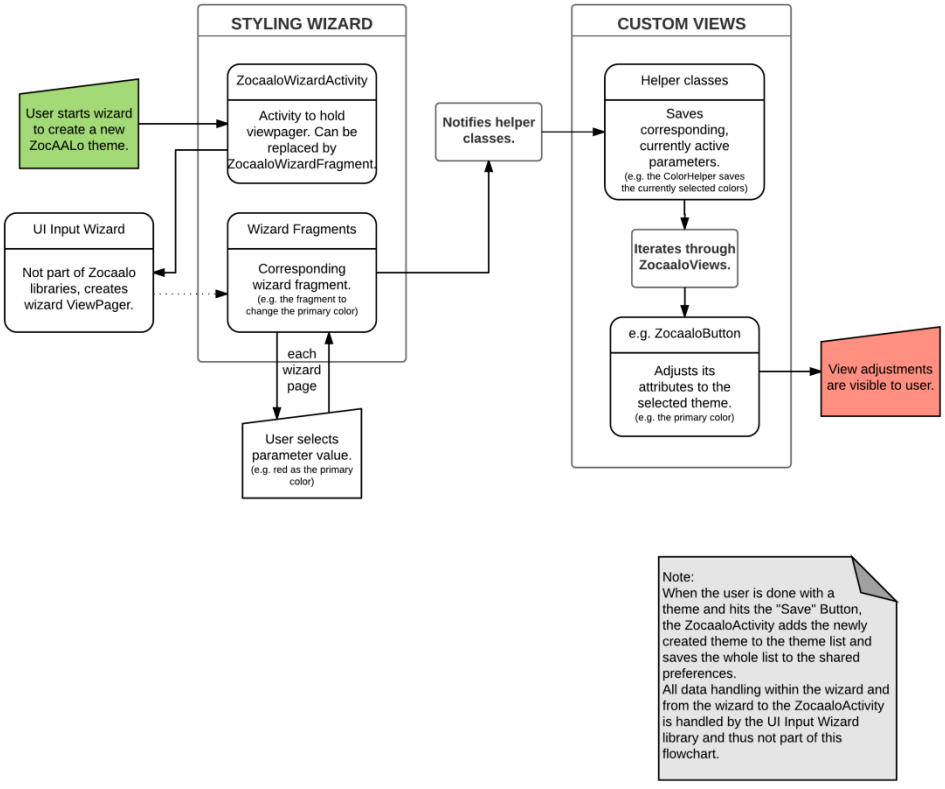


Figure 5: Creating a theme

5 Analisys and Testing

The zocaalo Styling Framework will help to improve the accessibility of the applications that implement it, but do not guarantee a fully accessible experience. The developer must:

- Test the app for full task completion, beginning to end, with various assistive technologies turned on. For example, turn on Explore by Touch in TalkBack and change the speed at which text is spoken out loud.
- Have users with impairments test your app.
- Make sure the major tasks that users must complete are possible for everyone.
- Talk to users, particularly those who use assistive technology, to learn about their needs, what they want out of your app, which tools they use, and how they use them. Become familiar with these tools so you can give them the best experience.

In the next section there is a list of third party tools.

5.1 Manual testing

5.1.1 Talkback

TalkBack is an accessibility service that helps blind and vision-impaired users interact with their devices.

TalkBack adds spoken, audible, and vibration feedback to your device. TalkBack comes pre-installed on most Android devices.

https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback&hl=en_GB

5.1.2 Switch Access

Switch Access lets users interact with Android devices using a switch instead of the touch screen. There are several kinds of switches: assistive technology devices such as those sold by AbleNet, Enabling Devices, RJ Cooper, or Tecla*; external keyboard keys; or buttons. This service can be helpful for users with motor impairments.

https://support.google.com/accessibility/android/?hl=en#topic=6151780

5.2 Other services

5.2.1 Google BrailleBack

BrailleBack is an Accessibility Service that helps blind users make use of braille devices. It works together with the TalkBack app to give a combined braille and speech experience. This app lets you connect a supported refreshable braille display to your device via Bluetooth.

Screen content will be presented on the braille display and you can navigate and interact with your device using the keys on the display. It is possible to input text using the braille keyboard.

There is a large list of devices supported including: APH Refreshabraille, Baum VarioConnect , Esys EuroBraille , Freedom Scientific Focus Blue (14 and 40 cell models), HandyTech (Basic Braille, Active Braille, Braille Star, Braille Wave, Braillino, Easy Braille), Harpo (Braillepen 12, Braillepen 12 Touch), HIMS (BrailleSense, Braille EDGE), Humanware Brailliant (1st generation and BI models), Optelec Alva (BC640, BC680), Easylink 12 touch, Papenmeier Braillex Trio, Seika (notetaker and 40 cell display) .

This app is not active by default.

https://play.google.com/store/apps/details?id=com.googlecode.eyesfree.brailleback&hl=en

5.2.2 Voice Access (beta)

Voice Access is an accessibility service that helps users who have difficulty manipulating a touch screen (e.g. due to paralysis, tremor, or temporary injury) use their Android devices by voice.

For help using Voice Access, see Voice Access support at https://support.google.com/accessibility/android/#topic=6151842.

Voice Access provides voice commands (currently English only) in three categories:

- Basics and navigation from any screen (e.g. "go back", "go home")
- Gestures to interact with things on the current screen (e.g. "click next", "scroll down")
- Text editing and dictation (e.g. "type hello", "replace coffee with tea")

On-screen help can be accessed by saying "What can I say?" within Voice Access. It is also possible to see the full list of voice commands by going to Voice Access Settings and selecting "Show all commands."

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

5.3 Analysis tools

5.3.1 Accessibility Scanner

Accessibility Scanner is a tool that suggests accessibility improvements for Android apps without requiring technical skills. Just open the app you want to scan, then tap the Accessibility Scanner button to find items in the app that might benefit from accessibility improvements. You can use this app to suggest changes to developers or to make changes yourself.

Accessibility Scanner suggests improvements such as enlarging small touch targets, increasing contrast, and providing content descriptions so that your app can be more easily used by individuals with accessibility needs. Designing for accessibility can allow you to reach a larger audience and provide a more inclusive experience.

To begin using Accessibility Scanner:

- Navigate to Settings > Accessibility
- Locate and turn on Accessibility Scanner

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

5.3.2 Node tree debugging

Accessibility services use a separate representation of an app's UI to operate. When debugging, might be useful to view the hierarchy and attributes of UI elements in the same way accessibility services view them. To accomplish this task, node tree debugging can be used. This tool, available in TalkBack, provides information about how an AccessibilityService, such as TalkBack, views UI elements within the app.

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

5.3.3 UI Automator Viewer

The uiautomatorviewer tool provides a convenient GUI to scan and analyze the UI components currently displayed on an Android device. UI Automator can be used to inspect the layout hierarchy and view the properties of UI components that are visible on the foreground of the device. This information lets create more fine-grained tests, for example by creating a UI selector that matches a specific visible property. The tool is located in the <android-sdk>/tools/ directory.

In accessibility testing, this tool is useful for debugging issues found using other testing methods. For example, if manual testing results in a view that does not have speakable text and should or a view that receives focus and should not, the tool can be used to help locate the source of the bug.

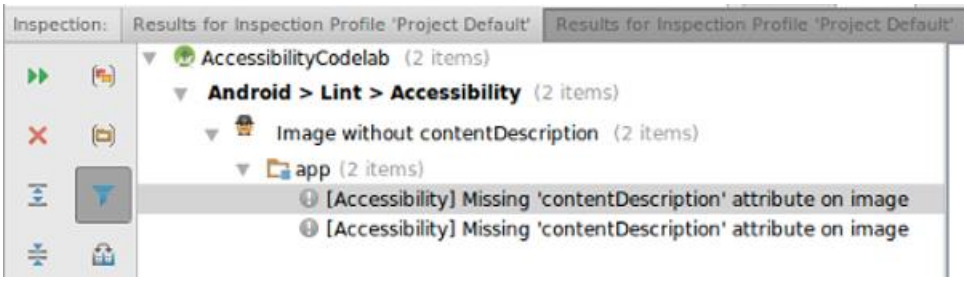
https://developer.android.com/training/testing/ui-testing/uiautomator-testing.html

5.3.4 Lint

Android Studio shows lint warnings for various accessibility issues and provides links to the places in the source code containing these issues. In the following example, an image is missing a contentDescription attribute. The missing content description results in the following message:

[Accessibility] Missing 'contentDescription' attribute on image

The next figure shows an example of how this message appears in Android Studio:



Message in Android Studio showing missing contentDescription attribute

Figure 6: Screen

If users of accessibility services, such as screen readers, encountered this image within the app itself, they wouldn't be able to understand the image's meaning.

https://developer.android.com/studio/write/lint.html

It is possible to customize the check Lint performs. Moreover more rules could be added to the core lint rules so aditional elements and code aspects could be checked writing custom rules.

5.3.5 Keyline Pushing

In Material Design, all components align to an 8dp square baseline grid, except type and toolbar iconography, which align to a 4dp square baseline grid. Keyline Pushing draws an 8dp square baseline grid and keylines based on the device's form factor, simplifying the design testing phase of the app and helping to analyze any other app.

https://play.google.com/store/apps/details?id=com.faizmalkani.keylines&hl=en

5.4 Automated testing

5.4.1 Espresso

Espresso is an Android testing library designed to make UI testing fast and easy. It allows you to interact with UI components under test in your app and assert that certain behaviors occur or specific conditions are met.

5.4.2 Enable checks

You can enable and configure accessibility testing through the AccessibilityChecks class:

AccessibilityChecks.enable();

By default, the checks run when you perform any view action defined in ViewActions. The check includes the view on which the action is performed as well as all descendent views. You can check the entire view hierarchy of a screen using the following code:

AccessibilityChecks.enable().setRunChecksFromRootView(true);

Suppress known issues

When first enabling checks, you may encounter a number of issues you may not be able to deal with immediately. You can suppress test failures resulting from these issues by setting a matcher for the results that you would like to suppress. To do so, obtain an AccessibilityValidator object by calling the enable() method of the

AccessibilityChecks class, then use the returned AccessibilityValidator's setSuppressingResultMatcher() method to configure a suppressing matcher.

In the following example, all issues related to touch target size on View objects with a resource ID of "overflow" are suppressed:

```
AccessibilityValidator validator = AccessibilityChecks.enable();
Matcher<AccessibilityViewCheckResult> myMatcher =
    allOf(
        matchesCheckNames(is("TouchTargetSizeViewCheck")),
        matchesViews(withId(R.id.my_overflow)));
validator.setSuppressingResultMatcher(myMatcher);
```

https://google.github.io/android-testing-support-library/docs/espresso/index.html

Code examples: https://github.com/googlesamples/android-testing

5.4.3 Robolectric

Robolectric is an open-source Android testing library that lets you test real Android code on a JVM, without needing to start an emulator.

Note: UI testing with Robolectric has some shortcomings, so you should use other forms of testing in addition to this tool. For example, Robolectric cannot give reliable results for touch target size and duplicate clickable items. To detect these issues, consider using Accessibility Scanner.

When first enabling checks for Robolectric, you may encounter a number of issues you may not be able to deal with immediately. You can suppress these errors by setting a matcher for the results that you would like to suppress. For more information, see the documentation for the setSuppressingResultMatcher() method of the AccessibilityUtil class that is available in Robolectric.

5.5 User testing

Along with the other testing methods in this guide, user testing can provide specific and valuable insights about the usability of your app.

5.6 Framework Tests

5.6.1 Accessibility-Test-Framework-for-Android

To help people with disabilities access Android apps, developers of those apps need to consider how their apps will be presented to accessibility services.

Some good practices can be checked by automated tools, such as if a View has a contentDescription. Other rules require human judgment, such as whether or not a contentDescription makes sense to all users.

For more information about Mobile Accessibility, see:

http://www.w3.org/WAI/mobile/

This library collects various accessibility-related checks on View objects as well as AccessibilityNodeInfo objects (which the Android framework derives from Views and sends to AccessibilityServices).

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

5.7 Custom accessibility services

An accessibility service can be bundled with a normal application, or created as a standalone Android project. The steps to creating the service are the same in either situation.

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

6 Appendix

6.1 Definitions

AAL

Ambient assisted living (AAL) can be defined as “the use of information and communication technologies (ICT) in a person’s daily living and working environment to enable them to stay active longer, remain socially connected and live independently into old age” (www.aal-europe.eu).

GUI

The graphical user interface, is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.

OS

An operating system (OS) is system software that manages computer and mobile devices hardware and software resources and provides common services for computer programs. All computer programs, excluding firmware, require an operating system to function. Currently Android is the leading mobile OS with a market share of 86.8% followed by iOS with the 12.5%.

Mental Model

In the field of user interface design, a mental model refers to the representation of something—the real world, a device, software, etc.—that the user has in mind. It is a representation of an external reality. Users’ mental models come from their prior experience with similar software or devices, assumptions they have, things they’ve heard others say, and also from their direct experience with the product or device.

Conceptual Model

A conceptual model is the actual model that is given to the user through the interface of the product. The actual interface (layout, navigation, colours, etc.) is representing the conceptual model.

6.2 References

[i] https://material.io/guidelines/usability/accessibility.html
[ii] https://www.w3.org/TR/WCAG20/
[iii] http://www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile
[iv] Word Healt Organisation: World report on disability, (2011)
[v] Barth, Erling: People with Disabilites in Norway, (1987)
[vi] Park, D., & Schwarz, N.: Cognitive Ageing. A Primer. Philadelphia, PA: Psychology press. (1999)
[vii] Park, D., & Schwartz, N. (2000): Cognitive Aging: A Primer. Philadelphia, PA: Psychology Press. (2000)
viii Rabbitt, P.: Cognitive Changes Across the Lifespan. In: Johnson, M. L., Bengtson, V. L. (2005)
[ix] Arbuckle, T. Y., Gold, D. & Andres, D. Cognitive functioning of older people in relation to social and personality variables. Psychology and Aging, 1, 55-62 (1986):.
[x] University of Illinois Eye and Ear Infirmary. Eye changes with aging. In The Eye Digest (17 June 2007)
[xi] Fisk, A.D., Rogers, W.A., Charness, N., Czaja, S.J., Sharit, J.: Designing for Older Adults: Principles and Creative Human Factors Approaches, 2nd edn. CRC Press J Taylor&. Francis Croup (2009)
[xii] Hawthorn, D.: Possible implications of aging for interface designers. Interacting with Computers 12, 507–528 (2000)
[xiii] Mynatt, E. D., & Rogers, W. A.;Developing technology to support the functional independence of older adults. Ageing International, 27 (1), 24-41 (2001).
[xiv] Arthritis Foundation: Arthritris Prevalence: A Nation in Pain. (2008)
[xv] Arthritis Care. General information about arthritis. (August 2007)
[xvi] National Institute of Neurological Disorders and Stroke: NINDS Parkinson's Disease Information Page. (10 April 2008)
[xvii] Meisami, E.: Aging of the sensory system. In: Timiras, P.S. (Ed): Physiological basis of aging and geriatrics. Boca raton, Fla.: CRC Press Inc., 115-31 (1994).
[xviii] Jack Loechner.: American Seniors and Digital Technology (http://www.pewinternet.org/2014/04/03/older-adults-and-technology-use/) (2014)
[xix] Maria Karavidas , Nicholas K. Lim, Steve L. Katsikas, Carlos Albizu University :The effects of computers on older adult users. (Available online 8 May 2004)
[xx] Ellis, R. D., & Allaire, J.: Modeling computer interest in older adults: The role of age, education, computer knowledge, and computer anxiety. Human Factors, 41(3), 345–355. (1999)
[xxi] Chua, S. L., Chen, D., & Wong, A. F. L.: Computer anxiety and its correlates: A meta-analysis. Computers in Human Behavior, 15, 609–623. (1999)
[xxii] https://www.w3.org/TR/WCAG20/#conformance-reqs