SOP 40

Mobile Apps

Projektleitung | Geschäftsführung | IT | Recht

So stellst du sicher, dass deine App nicht zur unbenutzbaren Barriere wird

Weil mobile Apps vom European Accessibility Act (EAA) genauso betroffen sind wie Websites. Die Interaktion auf einem Touchscreen stellt jedoch völlig neue Herausforderungen: Zu kleine Schaltflächen, nicht skalierbarer Text oder unverständliche Gesten machen eine App für Menschen mit Seh- oder motorischen Einschränkungen unbenutzbar.

Die eingebauten Screenreader der Betriebssysteme (VoiceOver für iOS, TalkBack für Android) sind für diese Nutzer:innen essenziell – aber sie funktionieren nur, wenn deine App ihnen die richtigen Informationen gibt. Eine nicht barrierefreie App ist nicht nur ein Gesetzesverstoß, sondern schließt Millionen potenzieller Nutzer:innen von vornherein aus.

Wann musst du das machen?

- Bei der Konzeption jeder neuen App.
- Während des gesamten Design- und Entwicklungsprozesses.
- Vor jeder Einreichung eines neuen App-Updates im Apple App Store oder Google Play Store.
- Bei der Durchführung eines Barrierefreiheits-Audits deiner bestehenden App.

Welches Gesetz verlangt das?

- European Accessibility Act (EAA) / Barrierefreiheitsstärkungsgesetz (BFSG): Mobile Apps von Anbietern bestimmter Produkte und Dienstleistungen fallen explizit unter das Gesetz.
- WCAG 2.1 (und höher): Die Richtlinien sind technologieunabhängig und gelten vollumfänglich auch für mobile Apps.
- Richtlinien der Plattformen:
- Apple Human Interface Guidelines (Accessibility)
- Google Material Design Guidelines (Accessibility)

Der Prozess im Detail

1 🟧 Phase 1: Das Design-Fundament – Die Hardware respektieren

Barrierefreiheit beginnt, bevor eine einzige Zeile Code geschrieben wird.

Anforderung

Umsetzung im Design

Warum ist das wichtig?

Ausreichend große Touch-Ziele

Bedienelemente (Buttons, Links, Icons) müssen eine Mindestgröße von 44×44 Pixeln (Apple) bzw. 48x48dp (Google) haben.

Menschen mit motorischen Einschränkungen müssen ein Ziel zuverlässig mit dem Finger treffen können.

Starke Kontraste

Text und wichtige grafische Elemente müssen ein Kontrastverhältnis von mindestens 4,5:1 (für normalen Text) bzw. 3:1 (für großen Text und Icons) erfüllen.

Gewährleistet Lesbarkeit bei Sehschwäche oder schlechten Lichtverhältnissen. (Siehe SOP 23)

Skalierbarer Text

Die App muss die vom Nutzer im Betriebssystem eingestellte Schriftgröße respektieren (Dynamic Type bei iOS, Font Size bei Android).

Ermöglicht es Nutzer:innen mit Sehschwäche, den Text auf eine für sie lesbare Größe einzustellen.

Keine reinen Farbinformationen

Verwende zusätzlich zu Farben immer auch Icons, Text-Labels oder Muster, um Zustände zu signalisieren (z. B. für „aktiv“ oder „Fehler“).

Stellt sicher, dass farbenblinde Nutzer:innen keine Informationen verpassen.

2 🟨 Phase 2: Die App zum Sprechen bringen – VoiceOver & TalkBack füttern

Ein Screenreader weiß nur, was deine App ihm sagt. Diese drei Informationen sind für jedes interaktive Element Pflicht:

Accessibility Property

Zweck

Beispiel (für einen Button)

Label (Name)

Was ist das? Der Name des Elements.

„Profilbild bearbeiten“

Hint (Hinweis)

Was passiert? Die Aktion, die bei Interaktion ausgelöst wird.

„Doppeltippen, um Ihr Foto zu ändern.“

Trait (Rolle)

Welche Art von Element ist das? Die Rolle des Elements.

„Button“

💡 Gute Praxis: Entwickler:innen müssen für jedes Element, das kein reiner Text ist, mindestens das Label setzen. Ohne Label liest VoiceOver oft nur „Button“ vor, was nutzlos ist.

3 🟩 Phase 3: Logische Reihenfolge & Gruppierung

Die Reihenfolge, in der ein Screenreader durch die Elemente navigiert, muss logisch sein und der visuellen Anordnung entsprechen.

- Lesereihenfolge prüfen: Aktiviere VoiceOver/TalkBack und wische von links nach rechts durch deine gesamte App. Springt der Fokus in einer logischen Reihenfolge?
- Elemente gruppieren: Fasse Elemente, die zusammengehören (z. B. ein Bild mit einer Überschrift und einem kurzen Text), in einem Container zusammen. So kann der Screenreader sie als eine Einheit vorlesen und Nutzer:innen müssen nicht unnötig oft wischen.

4 🟥 Phase 4: Gesten und Bedienbarkeit

- Standard-Gesten nicht überschreiben: Vermeide komplexe, benutzerdefinierte Gesten. Nutzer:innen verlassen sich auf die Standard-Gesten ihres Screenreaders (z. B. Wischen nach rechts für das nächste Element).
- Alternativen anbieten: Wenn deine App eine spezielle Geste erfordert (z. B. „Pinch-to-Zoom“ in einer Karte), muss es immer eine alternative Bedienmöglichkeit geben (z. B. „+“- und „-„-Buttons).
- Fokus-Management: Wenn sich ein neuer Bildschirm oder ein modales Fenster öffnet, muss der Fokus des Screenreaders automatisch auf dieses neue Element gesetzt werden.

5 🟦 Phase 5: Semantisches HTML schreiben, auch im Framework

Barrierefreiheit ist ein Qualitätsmerkmal. Zeige es.

- Erwähne Barrierefreiheit in der Beschreibung: Schreibe in die Beschreibung deiner App im App Store einen Satz wie: „Unsere App ist vollständig mit VoiceOver und TalkBack nutzbar, um allen Nutzer:innen ein barrierefreies Erlebnis zu bieten.“
- Auf Feedback reagieren: Wenn Nutzer:innen Barrieren über die App-Store-Bewertungen melden, nimm dieses Feedback ernst, bedanke dich und kündige eine Behebung im nächsten Update an.

Ergebnis

Am Ende dieser SOP hast du:

- Eine klare Grundlage, um die häufigsten Barrieren in mobilen Apps zu vermeiden.
- Sichergestellt, dass deine App von Menschen mit Seh- und motorischen Einschränkungen bedient werden kann.
- Die Anforderungen des EAA und der WCAG für mobile Apps verstanden.
- Ein positives Markenimage geschaffen, das Inklusion und Nutzerfreundlichkeit in den Vordergrund stellt.

Die Barrierefreiheit von Apps erfordert präzise Planung und detaillierte Tests. Dieser Baustein dient als direkt nutzbares Kit für deine Entwicklungs- und QA-Teams, um sicherzustellen, dass jede App-Komponente und jeder Screen den hohen Standards der Zugänglichkeit entspricht. Es ist eine handlungsleitende Spezifikation und Testanweisung in einem, die über reine Richtlinien hinausgeht und die Umsetzung signifikant beschleunigt.

Betreff: Anforderungen an den Einsatz von Pagebuildern / Frameworks für unser Projekt

App Accessibility Kit: Spezifikation & Prüfprotokoll

App-Name: [Name der App] Feature/Screen-Name: [Name des zu entwickelnden/prüfenden Features/ Screens, z.B. „Login-Screen“, „Produktliste“, „Einstellungsmenü“]

Verantwortliche(r) PM/PO: [Dein Name] Design-Lead: [Name] Entwicklungs-Lead: [Name] QA-Lead: [Name] Erstellungsdatum: [Datum] Geplantes Release: [Datum]

1. Design-Spezifikation (Verantwortlich: Design-Team)

- Touch-Zielgrößen geprüft und eingehalten (mind. 44x44px / 48x48dp)?
- Ja
- Nein (Aktion: [Was wird getan, um dies zu beheben?])
- Farbkontraste für Text und wichtige Grafiken gemäß WCAG (4.5:1 / 3:1) geprüft?
- Ja (Link zum Styleguide/Kontrastprüfung: [URL])
- Nein (Aktion: [Was wird getan, um dies zu beheben?])
- Skalierbarer Text (Dynamic Type/Font Size) im Design berücksichtigt?
- Ja
- Nein (Aktion: [Was wird getan, um dies zu beheben?])
- Informationen werden zusätzlich zu Farbe signalisiert (Icons, Text, Muster)?
- Ja (Beispiel: [Kurze Beschreibung])
- Nein (Aktion: [Was wird getan, um dies zu beheben?])

2. VoiceOver/TalkBack-Fütterung (Verantwortlich: Entwicklung-Team)

Für jedes interaktive oder informative UI-Element in diesem Feature/Screen:

- Label (Name) gesetzt? (Pflicht für alle Elemente außer rein dekorativen)
- Ja
- Nein (Aktion: [Was wird getan, um dies zu beheben?])
- Beispiel für kritisches Element und Label: [Element, z.B. „Profilbild-Icon“; Label: „Profilbild bearbeiten“]
- Trait (Rolle) korrekt definiert? (z.B. Button, Text, Image, Link, Header)
- Ja
- Nein (Aktion: [Was wird getan, um dies zu beheben?])
- Hint (Hinweis für Aktion) gesetzt? (Für komplexe oder nicht-standardisierte Elemente empfehlenswert)
- Ja
- Nein (Aktion: [Was wird getan, um dies zu beheben?])

3. Logische Reihenfolge & Gruppierung (Verantwortlich: Entwicklung-Team)

- Visuelle Lesereihenfolge = Screenreader-Lesereihenfolge? (Prüfung durch Development / QA)
- Ja
- Nein (Beschreibung des Problems & Aktion: [Details])
- Zusammengehörige Elemente gruppiert (als eine Einheit vorlesbar)? (z.B. Bild+Text, Label+Input)
- Ja
- Nein (Beschreibung des Problems & Aktion: [Details])

4. Gesten & Fokus-Management (Verantwortlich: Entwicklung-Team)

- Standard-Gesten nicht überschrieben oder alternative Bedienung vorhanden?
- Ja
- Nein (Beschreibung des Problems & Aktion: [Details])
- Fokus-Management bei Screen-/Modal-Öffnung implementiert? (Fokus springt automatisch auf neues Element)
- Ja
- Nein (Beschreibung des Problems & Aktion: [Details])
- Fokus-Rückgabe bei Schließen von Modal/Overlay implementiert?
- Ja
- Nein (Beschreibung des Problems & Aktion: [Details])

5. Test & Verifizierung (Verantwortlich: QA-Team)

- VoiceOver/TalkBack-Test durchgeführt?
- Ja (Datum: [Datum], Tester: [Name])
- Nein (Geplanter Termin: [Datum])
- Gesten-Test mit Screenreader durchgeführt?
- Ja (Datum: [Datum], Tester: [Name])
- Nein (Geplanter Termin: [Datum])
- Barrierefreiheit gemäß SOP für dieses Feature/Screen bestätigt?
- Ja
- Nein (Begründung und erforderliche Maßnahmen: [Beschreibung])

Zusätzliche Kommentare/Aktionen: [Hier können spezifische Notizen, Herausforderungen oder besondere Umstände festgehalten werden.]

Unterschriften zur Abnahme:

- Projektleitung / EAA-Verantwortliche:r: _____ Datum: _____
- Design-Lead: _____ Datum: _____
- Entwicklungs-Lead: _____ Datum: _____
- QA-Lead: _____ Datum: _____

Fragen & Antworten

Warum ist es so wichtig, dass meine App die im Betriebssystem eingestellte Schriftgröße des Nutzers respektiert (Dynamic Type/Font Size), anstatt eine feste Schriftgröße zu verwenden?

Das ist entscheidend für Nutzer:innen mit Sehschwäche oder ältere Menschen , die eine größere Schrift benötigen, um Texte lesen zu können. Wenn deine App die systemweite Einstellung ignoriert, zwingst du diese Nutzer:innen, die App zu deinstallieren oder ein Vergrößerungsglas zu benutzen. Die Betriebssysteme (iOS, Android) bieten hierfür Mechanismen (wie Apples Dynamic Type oder Androids skalierbare Pixel sp), die Entwickler:innen nutzen müssen. Das Ignorieren dieser Einstellungen führt dazu, dass Text abgeschnitten wird oder überlappt, wenn der Nutzer die Schriftgröße erhöht, was die App unbenutzbar macht.

Die SOP erwähnt %22reine Farbinformationen%22 als Problem. Was ist, wenn mein Design ausschließlich farbige Indikatoren verwendet, um den Status anzuzeigen (z.B. ein roter Punkt für Fehler, ein grüner für Erfolg)?

Das ist ein häufiger Designfehler Für farbenblinde Nutzer:innen sind diese Informationen nicht erkennbar. Du musst immer einen zusätzlichen, nicht-farblichen Indikator verwenden. Beispiele:

- Fehler/Erfolg: Neben einem roten/grünen Punkt einen Text („Fehler“, „Erfolgreich“), ein Icon (Häkchen, Ausrufezeichen) oder eine Formänderung hinzufügen.
- Aktiv/Inaktiv: Bei einem aktiven Tab nicht nur die Farbe ändern, sondern auch eine Unterstreichung , eine fette Schrift oder ein Icon hinzufügen.

Die Farbe kann weiterhin als visuelle Verstärkung dienen, aber niemals als einziger Informationsträger.

Wie teste ich am besten die %22logische Reihenfolge%22 von Elementen mit VoiceOver oder TalkBack, besonders wenn die visuelle Anordnung komplex ist?

Der Test der logischen Reihenfolge erfordert systematisches Vorgehen , du:

- Starte oben links: Beginne mit dem ersten Element auf dem Bildschirm.
- Wische immer nach rechts: Gehe Element für Element durch (mit dem Standard-Wischgeste nach rechts bei VoiceOver/TalkBack).
- Vergleiche mit visueller Reihenfolge: Notiere dir die Reihenfolge, in der die Elemente vorgelesen werden. Entspricht diese der erwarteten Reihenfolge, in der ein sehender Nutzer die Elemente von oben nach unten, links nach rechts erfassen würde?
- Achte auf Sprünge: Wenn der Fokus unerwartet springt oder Elemente übersprungen werden, ist das ein Problem. Typische Fehler sind unsichtbare Elemente im Fokusbaum oder eine falsche Gruppierung.
- Element-Rotor (VoiceOver): Bei iOS kannst du auch den Rotor nutzen, um nach Überschriften, Links oder Bedienelementen zu springen und zu prüfen, ob alle vorhanden sind und die Hierarchie stimmt.

Ein langsames, bewusstes Durchgehen des Bildschirms ist hier der Schlüssel.

Meine App verwendet viele benutzerdefinierte Gesten (z.B. Wischen in bestimmten Mustern, langes Drücken mit mehreren Fingern). Wie kann ich hier Barrierefreiheit gewährleisten?

as ist eine große Herausforderung, da Screenreader selbst komplexe Gesten nutzen und deine App-Gesten damit kollidieren könnten.

- Vermeide komplexe Gesten: Wann immer möglich, setze auf Standard-Gesten oder Klicks/Taps.
- Biete Alternativen: Wenn eine spezielle Geste unverzichtbar ist, muss es IMMER eine gleichwertige, alternative Bedienung über Standard-UI-Elemente geben. Beispiele:
- Statt „Pinch-to-Zoom“ in einer Karte: Biete „Zoom-In“ und „Zoom-Out“ Buttons (+ und -).
- Statt Wischen für Navigation: Biete sichtbare Vor-/Zurück-Pfeile oder eine Liste.
- Statt Langdrücken für Kontextmenü: Biete ein Kontextmenü an, das über einen normalen Button (... -Icon) geöffnet wird.
- Dokumentation: Wenn du doch benutzerdefinierte Gesten nutzen musst, dokumentiere sie klar in der Hilfe der App und weise auf die Alternativen hin.

Das Ziel ist, dass keine Funktion der App ausschließlich über eine Geste bedienbar ist, die nicht allen Nutzern zugänglich ist.

Die SOP erwähnt, Barrierefreiheit in der App-Beschreibung zu erwähnen. Gibt es weitere Marketing- oder Kommunikationsmaßnahmen, die ich ergreifen sollte, um unsere Bemühungen zu zeigen?

Ja, du kannst deine Bemühungen auf verschiedenen Kanälen hervorheben:

- Eigene App-Website/Support-Bereich: Erstelle eine spezielle Seite oder einen FAQ-Bereich, der detaillierter auf die Barrierefreiheitsfunktionen deiner App eingeht. Hier kannst du auch Tutorials zur Nutzung mit Screenreadern anbieten.
- Blogposts/Pressemitteilungen: Berichte über wichtige Barrierefreiheits-Updates oder die Implementierung neuer Funktionen.
- Soziale Medien: Teile kurze Demos oder Tipps zur Nutzung der App mit assistiven Technologien.
- Nutzer-Feedback: Ermutige Nutzer:innen aktiv, Barrieren zu melden, und reagiere transparent auf Feedback und behebe gemeldete Probleme. Nichts schafft mehr Vertrauen als eine sichtbare Reaktion auf Nutzerbedürfnisse.

Das Zeigen deines Engagements für Barrierefreiheit kann die Wahrnehmung deiner Marke positiv beeinflussen und eine breitere Nutzerbasis ansprechen