Dobre praktyki WCAG 2.1 w zamówieniach publicznych

Wykonawco, przekazujemy nasze wymagania odnośnie dostępności cyfrowej.

Serwis/system powinien być całkowicie dostępny cyfrowo dla Użytkowników z wszelkimi niepełnosprawnościami, dla seniorów i wszystkich innych użytkowników Internetu. Ze względu na rolę, jaką pełni PFRON, Serwis/system powinien być wzorcowy w zakresie dostępności.

Wymóg dostępności serwisu/systemu PFRON wynika z:

- Ustawy z dnia 4 kwietnia 2019 r. o dostępności cyfrowej stron internetowych i aplikacji mobilnych podmiotów publicznych (Dz.U. 2019 poz. 848),
- Rozporządzenia Rady Ministrów z dnia 12 kwietnia 2012 r. w sprawie Krajowych Ram Interoperacyjności, minimalnych wymagań dla rejestrów publicznych i wymiany informacji w postaci elektronicznej oraz minimalnych wymagań dla systemów teleinformatycznych (Dz.U. 2017 poz. 2247).

Zgodnie z ustawą serwisy internetowe realizujących zadania publiczne muszą być zgodne z WCAG 2.1 na poziomie A i AA.

Zatem Wykonawco, jesteś zobowiązany do dostarczenia serwisu/systemu, który jest bezbłędny pod względem jakości kodu, zgodności z WCAG 2.1 i rzeczywistej dostępności dla wszelkich grup narażonych na wykluczenie cyfrowe.

1. Ogólne wymagania w zakresie dostępności cyfrowej

Twoim obowiązkiem jest zapewnić dostępność cyfrową serwisu/systemu na poziomie WCAG 2.1 A oraz AA, zgodnie z załącznikiem nr 1 do Ustawy z dnia 4 kwietnia 2019 r. o dostępności cyfrowej stron internetowych i aplikacji mobilnych podmiotów publicznych (Dz.U. 2019 poz. 848).

Materiałami referencyjnym odnośnie spełnienia wytycznych WCAG 2.1 są:

- WCAG 2.1 (oficjalne tłumaczenie na język polski)
- Techniques for WCAG 2.1 — jest to obszerny dokument on-line, który zawiera setki przydatnych fragmentów kodu i przykładów zastosowania kryteriów WCAG 2.1.

W trakcie projektowania elementów interfejsów (np. menu, nawigacja, okna modalne, formularze, nawigacja okruszkowa, tabele, karuzele itp.) powinieneś korzystać z wzorców projektowych i dobrych praktyk, opublikowanych na stronach:

- https://www.w3.org/TR/wai-aria-practices/
- https://www.w3.org/WAI/tutorials/

Wątpliwości dotyczące sposobów wdrażania dostępności cyfrowej będą rozstrzygane przez Zamawiającego na podstawie dokumentacji opracowanej przez www.w3.org.

2. Narzędzia wspierające budowę i testowanie dostępnych cyfrowo serwisów/systemów internetowych

Niżej wymienione narzędzia wspierają tworzenie dostępnych cyfrowo serwisów/systemów oraz umożliwiają wczesne wykrycie części problemów z obszaru dostępności cyfrowej. Pamiętaj jednak, że narzędzia automatyczne nie wykrywają wszystkich niezgodności z WCAG 2.1 – dlatego konieczna jest weryfikacja audytora WCAG.

Propozycja listy narzędzi:

- **NVDA** – czytnik ekranu https://nvda.pl/,
- **VoiceOver** - wbudowany w system Mac OS X mechanizm odczytywania komunikatów z ekranu - https://www.apple.com/pl/voiceover/info/guide/_1121.html,
- **WAVE** – narzędzie do wstępnej wizualnej ewaluacji zgodności strony z WCAG 2.1 https://wave.webaim.org/,
- **AXE Devtools** - narzędzie wspomagające badanie dostępności, generujące wstępną listę potencjalnych błędów https://chrome.google.com/webstore/detail/axe-devtools-web-accessib/lhdoppojpmngadmnindnejefpokejbdd,
- **ARC Toolkit** – rozszerzenie do przeglądarki Chrome, wspierające badanie kodu strony - https://chrome.google.com/webstore/detail/arc-toolkit/chdkkkccnlfncngelccgbgfmjebmkmce,
- **ANDI** – bookmarklet dla przeglądarki Chrome https://www.ssa.gov/accessibility/andi/help/install.html,
- **Colour Contrast Analyser** – narzędzie do weryfikowania kontrastu elementów strony https://www.tpgi.com/color-contrast-checker/,
- **HeadingsMap** – rozszerzenie pomagające określić strukturę oraz hierarchię nagłówków występujących na stronie https://chrome.google.com/webstore/detail/headingsmap/flbjommegcjonpdmenkdiocclhjacmbi,
- **Landmarks** – rozszerzenie pomagające określić punkty orientacyjne (tak zwane landmarki) występujące na stronie https://chrome.google.com/webstore/detail/landmark-navigation-via-k/ddpokpbjopmeeiiolheejjpkonlkklgp

- **Text Spacing** – narzędzie wspomagające symulację strony ze zwiększonymi odstępami w zakresie podanym w WCAG 2.1.
  https://dylanb.github.io/bookmarklets.html

3. Najważniejsze wymagania techniczne w zakresie dostępności (programistyczne)

Poniższa część dokumentacji ma za zadanie zwrócić Twoją uwagę na kluczowe aspekty zapewnienia dostępności cyfrowej serwisu/systemu internetowego. Musisz wiedzieć, że są to tylko wytyczne wspierające Ciebie w realizacji kluczowych wytycznych WCAG 2.1, nie zaś pełna lista sposobów zapewnienia zgodności strony ze standardem WCAG 2.1.

3.1. Zgodność składni z walidatorem HTML

Wszystkie strony serwisu/systemu muszą być bezbłędne pod względem jakości kodu HTML z walidatorem https://validator.w3.org/nu/.

W trakcie wdrożenia mogą się pojawić sytuacje, w których możemy zaakceptować błędy HTML. Muszą to być jednak uzasadnione i udokumentowane przypadki, związane z niestabilnością specyfikacji HTML5, które będą działać np. na rzecz dostępności.

Uwaga: Poza zgodnością z walidatorem samych szablonów serwisu/systemu, także treści zapisane przy użyciu edytora wizualnego WYSIWYG nie mogą powodować problemów. Dlatego edytor wizualny powinien generować prawidłowy kod HTML.

3.2. Jakość semantyczna kodu HTML

Podstawowym warunkiem dostępności jest prawidłowe — adekwatne stosowanie znaczników HTML. Najprościej rzecz ujmując, serwis/system powinieneś realizować w pełnej zgodności ze specyfikacją HTML5.

Przykłady poprawności semantycznej:

W ramach prac nad serwisem/systemem pamiętaj, że poszczególne elementy należy wykonać w określony sposób:

1. Linki za pomocą znacznika **<a>**, czyli natywnego semantycznego znacznika HTML. Jeśli jest to niemożliwe dopuszczalne są również niesemantyczne elementy ** ** wraz z odpowiednią rolą **role=”link”**;
2. Nagłówki za pomocą znaczników **<h1>**...**<h6>** (przy czym nagłówek **<h1>** winien występować tylko raz), czyli natywnego semantycznego znacznika HTML. Jeśli jest to niemożliwe dopuszczalne są również niesemantyczne elementy ** ** wraz z odpowiednią rolą, na przykład dla nagłówka poziomu 1 **role=”heading” ARIA-level="1"**;

3. Przyciski za pomocą znaczników **<button>** lub **<input type="button">**, czyli natywnego semantycznego znacznika HTML. Jeśli jest to niemożliwe dopuszczalne są również niesemantyczne elementy ** ** wraz z odpowiednią rolą **role=”button”**;
4. Listy za pomocą znaczników **<ul>**/**<ol>** i **<li>** dla poszczególnych elementów;
5. Rozwijane listy formularzy za pomocą znaczników **<select>**/**<option>**.

Przykłady błędów **semantycznych**:

Unikaj poniższych rozwiązań.

1. Link wykonany za pomocą **** (oskryptowany JavaScript);
2. Nagłówek w formie **<p class="heading">**;
3. Lista rozwijana w formularzu, wykonana za pomocą znaczników listy **<ul>**/**<li>**.

3.3. Uzupełnienia semantyczne za pomocą ARIA

Atrybuty ARIA muszą być uzupełnieniem semantyki HTML. To technologia przeznaczona przede wszystkim dla użytkowników czytników ekranu. Szczególnie ważne jest jej stosowanie w komponentach stron internetowych, które opierają się na rozbudowanej interakcji JavaScript.

Stosowanie atrybutów ARIA można podzielić na dwie części:

1. Uzupełnienie głównych bloków serwisu/systemu o punkty orientacyjne;
2. Dodatki do formularzy lub takich komponentów stron, jak karuzele, zakładki (**tabs**), menu rozwijane, bloki rozwijane, okna modalne, alerty, slidery.

Głównym źródłem informacji jak stosować ARIA powinna być dla Ciebie dokumentacja Aria Techniques for WCAG 2.1.

Niestety, nie jest to wystarczające źródło wiedzy. Nie ma jednego miejsca w Internecie zawierającego aktualną, pewną i gotową do stosowania wiedzę w zakresie ARIA.

Zastrzegamy sobie prawo do weryfikacji serwisu/systemu, w każdy dostępny sposób, pod względem zgodności ze specyfikacją ARIA w całym okresie obowiązywania Umowy. W przypadku stwierdzenia niezgodności ze specyfikacją ARIA będziesz zobowiązany do ich usunięcia na własny koszt w terminie wskazanym przez Zamawiającego.

3.4. Tytuły stron serwisu/systemu internetowego

Wszystkie tytuły stron serwisu/systemu muszą być automatycznie generowane na podstawie informacji, które pozwolą użytkownikowi dowiedzieć się, co jest treścią danej strony.

Przykłady:

1. Strona główna serwisu/systemu powinna mieć tytuł — „Serwis informacyjny iPFRON+”.
2. Strona „Program Wsparcie Inicjatyw” powinna mieć tytuł — „Program Wsparcie Inicjatyw – Serwis informacyjny iPFRON+”.

Wszystkie strony mają mieć tytuł wg zasady - od szczegółu do ogółu.

Do uzgodnienia z Tobą pozostanie kwestia, ile elementów ścieżki ma być widocznych w tytule:

- tytuł strony + nazwa serwisu lub
- tytuł stron + nazwa działu + np. nazwa nadrzędnego działu + nazwa serwisu.

Zgodnie z Wymaganiami dla Systemu Zarządzania treścią (CMS), opis dodatkowych modułów i funkcjonalności CMS oraz CMS serwisu - Redaktor musi mieć możliwość indywidualnego definiowania zawartości atrybutu metatagu **title**, niezależnie od tytułu redakcyjnego.

3.5. Oznaczenie języka strony i treści

Język naturalny treści na stronie powinieneś zawsze oznaczać odpowiednim atrybutem lang. W założeniu wszystkie strony serwisu/systemu będą miały atrybut **lang** o treści **"pl"** lub **“pl-PL”**.

Dodatkowo powinieneś zapewnić redaktorom serwisu w edytorze **WYSIWYG** możliwość oznaczenia takim atrybutem dowolnego ciągu znaków, tak by użytkownik korzystający z technologii asystujących mógł zorientować się, że treść jest w innym języku, niż domyślny język strony.

3.6. Nagłówki stałe

W serwisie będą stałe bloki treści i bloki funkcjonalne. Powinieneś je oznaczyć nagłówkami na odpowiednim poziomie.

3.7. Nagłówki dla redaktorów

Redaktorom powinieneś zapewnić możliwość ustawiania odpowiedniej struktury nagłówkowej stron. Nagłówki dostępne dla redaktora powinny się zawierać od **h2** do **h6**. Nagłówek **h1** powinien najtrafniej opisywać główną treść strony.

3.8. Linki

W serwisie wszystkie linki powinny być zrozumiałe poza kontekstem tekstowym bądź wizualnym. W stałych częściach serwisu/systemu może oznaczać to potrzebę uzupełniania krótkich linków o treści uzupełniające. Linki powinny być uzupełniane przez treści niewidoczne dla użytkowników niekorzystających z czytników ekranu, na przykład za pomocą klasy **sr-only** czy **visually-hidden**.

Przykłady linków, które będzie można uzupełnić o dodatkową treść, to: zamknij, przewiń, następny, poprzedni, więcej, pobierz, pokaż wszystkie, itp.

3.9. Opisy alternatywne

Wszystkie grafiki, które zamieścisz w szablonach za pomocą znacznika **<img>** powinny mieć atrybut **alt**.

- W przypadku, gdy grafika nie będzie przekazywać żadnej treści (grafiki dekoracyjne), powinieneś je umieszczać za pomocą CSS, czyli stosując właściwość **background-image**. Inną metodą jest dodanie do **<img>** - pustego **alt**— zapis **alt** lub **alt=""**.
- Jeśli grafika będzie przekazywać treść, atrybut **alt** powinieneś uzupełnić o adekwatny opis.
- Jeśli grafika będzie linkiem, to w opisie alternatywnym powinieneś przekazywać funkcję linku, tak jakby to był link tekstowy lub zastosować opis **ARIA-label** lub ukrytą klasę np. **<sr-only>** do opisu celu linku. Jeśli zastosujesz drugie rozwiązanie atrybut **alt** powinien być pusty.
- Elementy, które zaimplementujesz za pomocą SVG powinny posiadać znacznik **<title></title>**, w którym należy umieścić tekst alternatywny lub też dodać atrybut **ARIA-hidden=”true”**, jeśli ma to być grafika dekoracyjna.

3.10. Formularze — semantyka.

Budowa formularzy pod względem dostępności musi opierać się o dobre praktyki HTML5. Należy uwzględnić, że formularze mogą być używane przez osoby z niepełnosprawnością wzroku, niepełnosprawne ruchowo czy głucho-niewidome.

Powinieneś wiedzieć, jakie są popularne sposoby użycia formularzy, np. bez użycia myszki czy bez patrzenia na ekran.

W większości przypadków jako podstawy semantyki HTML dla formularzy rozumiemy:

- użycie etykiet do wszystkich pól, etykiety mogą być ukryte lub widoczne,
- zrozumiałość etykiet,
- dostęp do wszelkich wskazówek bez konieczności patrzenia na ekran, np. za pośrednictwem czytnika ekranu (wskazówki, sugestie poprawy błędów, komunikaty błędów do obiektów formularzy powinny być powiązane semantycznie z tym obiektem, np. poprzez **ARIA-describedby**),
- kolejność treści i pól formularzy wspierająca użyteczność i zrozumiałość,
- zdefiniowanie atrybutu **“autocomplete”**,
- zdefiniowanie wymagalności pól (**ARIA-required=”true/false” or required**)

3.11. Formularze — wsparcie użytkownika i informacja o błędach.

Większym wyzwaniem w przypadku formularzy jest właściwa dostępność informacji o tym, w jaki sposób wypełnić pola oraz informacje o błędach.

W tym przypadku kieruj się następującym podejściem:

1. wszystko, co możliwe, wykonaj za pomocą podstawowych elementów HTML + JavaScript — im dalej będzie sięgać wsteczna kompatybilność, tym lepiej,
2. jeśli formularz będzie tego wymagał, zastosuj atrybuty ARIA.

Kolejność w powyższym wypunktowaniu jest ważna - Użytkownicy mogą korzystać z przestarzałego oprogramowania. Dlatego zagwarantuj wsteczną kompatybilność w jak największym stopniu.

Nie rekomendujemy stosowania walidacji HTML. Prezentacja informacji w tym rozwiązaniu jest ograniczona czasem.

Przykłady poprawnych rozwiązań:

- Przykład z użyciem role="alert" https://www.upyoura11y.com/handling-form-errors/
- Przykład wykorzystania aria-describedby: https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA1

3.12. Tabele.

W przypadku tabel, kluczowe jest stosowanie odpowiedniej składni i semantyki HTML. Czytniki ekranu wspierają obsługę tabel bardzo dobrze.

Wskazówki, które pomogą Ci w tworzeniu dostępnych tabel znajdziesz na stronie https://www.w3.org/WAI/tutorials/tables/.

3.13. Działanie serwisu/systemu za pomocą klawiatury.

Prawidłowe zastosowanie semantyki HTML powinno gwarantować dostępność za pomocą klawiatury każdego aktywnego elementu na stronie – zadbaj o to na etapie wdrożenia, aby zapewnić bezbłędne działanie tej funkcjonalności.

Programiści muszą stosować zarządzanie fokusem przez JavaScript w taki sposób, aby nie stworzyć tzw. pułapki klawiaturowej. Taki błąd powoduje barierę dla użytkowników z niepełnosprawnością ruchu oraz korzystających z czytników ekranu.

3.14. Kolejność fokusu.

Fokus klawiatury powinien mieć kolejność wedle reguły od lewej do prawej i od góry do dołu. Na przykład, po przejściu fokusem menu głównego, powinien on trafić do głównego bloku treści lub lewej kolumny.

3.15. Ukrywanie treści.

W niektórych przypadkach, np. w linkach, może być konieczne stosowanie ukrytej treści. Takie rozwiązanie wspiera korzystanie z serwisu/systemu przez użytkowników z niepełnosprawnością wzroku.

Polecamy artykuły opisujące techniki ukrywania treści:

- http://webaim.org/techniques/css/invisiblecontent.
- https://getbootstrap.com/docs/5.0/helpers/visually-hidden/

Poza tymi obszarami, w których Wykonawca zaproponuje użycie techniki ukrywania, w ramach monitoringu wdrożenia, ekspert ds. dostępności pracujący w ramach zespołu projektowego będzie rekomendować miejsca, w których warto dodatkowo zastosować tę technikę.

3.16. Zabezpieczenie formularzy.

Zabezpiecz formularz w taki sposób, aby nie stwarzał barier dla użytkownika serwisu/systemu.

W takim przypadku musisz uważać z rozwiązaniami typu **CAPTCHA**. Tego typu zabezpieczenia najczęściej nie są w stanie zapewnić dostępności dla wszystkich odbiorców.

W miarę możliwości filtrowanie spamu i działań niepożądanych pozostaw po stronie serwera lub wykonaj zabezpieczenia tak aby nie wymagały dodatkowego działania po stronie użytkownika. Jeśli zaproponujesz rozwiązanie typu **CAPTCHA**, to będzie ono dokładnie testowane pod kątem dostępności dla wszystkich użytkowników.

Więcej informacji na temat rozwiązania **CAPTCHA** znajdziesz pod adresem:

https://developers.google.com/recaptcha/docs/invisible

3.17. Działanie filtrów / przeładowanie.

Wszelkie działania związane z przeładowaniem widoku takie jak:

- filtrowanie,
- sortowanie,
- wyszukiwanie,

przetestuj z czytnikami ekranu. W takich sytuacjach kluczowy będzie komfort obsługi bezwzrokowej. Użytkownik powinien mieć pełną wiedzę na temat działania interfejsu i świadomość tego, że treść strony została zaktualizowana.

Przyjmij ogólną zasadę, że zmiany treści strony bez przeładowania stosujemy tylko w uzasadnionych sytuacjach.

W niektórych przypadkach, po zmianie przefiltrowania może być konieczna automatyczna zmiana tytułu strony **<title>**.

3.18. Działanie w trybie wysokiego kontrastu (WINDOWS).

Serwis/system powinien bezproblemowo działać w trybie wysokiego kontrastu Windows. Wykonawca powinien prowadzić takie testy na bieżąco w trakcie wdrożenia.

Typowe problemy w takim trybie mogą być związane z użyciem CSS-owego zastępowania tekstu grafiką. Dlatego w niektórych przypadkach zamiast użycia takiej techniki, będzie konieczne zastosowanie typowych linków graficznych **<a><img>**.

3.19. Skip linki.

Na każdej stronie serwisu powinien działać link „**Przejdź do wyszukiwania**”, „**Przejdź do głównej treści**” (jeżeli takie elementy występują), które pomagają przeskoczyć fokusem bezpośrednio do głównej funkcjonalności danej strony. Najczęściej będzie to oznaczać przeskoczenie nawigacji lub też innych powtarzających się elementów na stronie. Takie elementy zaprojektuj i skonsultuj z zespołem zleceniodawcy.

4. Inne wymagania techniczne

4.1. Szybkość działania serwisu/systemu

Serwis/system powinien być maksymalnie zoptymalizowany do szybkiego działania. Lekkość serwisu wpływa pozytywnie na działanie z oprogramowaniem wspomagającym, takim jak np. czytnik ekranu. Takie działanie powoduje również komfortową obsługę w urządzeniach mobilnych. W ramach optymalizacji pod kątem szybkości działania trzeba będzie zwrócić uwagę na następujące kwestie:

- brak nadmiarowego kodu HTML / CSS / JS,
- nieobciążanie serwisu/systemu zbędnymi dodatkami JS,
- dobrą optymalizację grafiki,
- minimalizację liczby plików pobieranych wraz z unikalną stroną,
- cache serwisu, który zminimalizuje zapytania do bazy danych.

4.2. Responsywność (RWD).

Przy budowaniu serwisu/systemu pamiętaj o urządzeniach mobilnych, które pełnią ważną rolę w odbiorze treści internetowych.

Serwis/system buduj w oparciu o najlepsze i aktualne praktyki tworzenia serwisów responsywnych.

Przygotuj wszystkie projekty graficzne z zastosowaniem skoków responsywnych szerokości w odniesieniu do typów urządzeń (standardów):

- smartfon – z rozdzielczością 360x640 (wartość średnia) w wersji pionowej oraz poziomej (z uwzględnieniem wartości minimalnej 320 px),
- tablet - z rozdzielczością 768x1024 w wersji pionowej oraz poziomej,
- monitor komputerowy - z rozdzielczością 1366x768 (wartość średnia), z uwzględnieniem wartości minimalnej – 900 px oraz wartości maksymalnej - 1920 px.

Zwróć uwagę, aby obiekty nie zachodziły na siebie i nie przykrywały treści bądź funkcjonalności.

Przy projektowaniu widoków mobilnych uwzględnij minimalną wielkość fontów – 16 px. Jest to wartość ważna podczas analizy czytelności strony.

4.3. Możliwości edytora WYSIWYG

W edytorze wizualnym poza standardowymi funkcjami, udostępnisz redaktorom kilka dodatkowych narzędzi. Dokonując wyboru **WYSIWYG**a trzeba będzie sprawdzić, czy rozwiązanie obsługuje dane funkcje natywnie, np. na podstawie pluginów.

Wstępna rekomendacja na rozwiązanie WYSIWYG to TinyMCE. W trakcie produkcji systemu trzeba będzie dokonać wyboru, która wersja edytora powinna być dostępna dla redaktorów strony internetowej lub aplikacji internetowej.

5. Szczegółowe wytyczne w zakresie dostępności (graficzne)

5.1. Kontrast treści.

Kontrast między kolorem tekstu a kolorem jego tła, musi wynosić minimum 4,5:1 lub 3:1 dla większego tekstu (krój pisma powyżej 18 punktów).

Prostym narzędziem do analizy poziomu kontrastu jest Colour Contrast Analyzer.

W związku z wymogami dotyczącymi kontrastu, nie powinieneś stosować elementów prezentujących tekst na tle niejednorodnym, np. bezpośrednio na tle zdjęcia. Istnieje możliwość dodania takiego tekstu wraz z zastosowaniem atrybutu **CSS opacity** o wartości mniejszej niż 1.

Możesz stosować kolorystykę o mniejszym kontraście, ale tylko w zakresie elementów dekoracyjnych w serwisie. Kryterium kontrastu nie obejmuje logo serwisu.

5.2. Identyfikacja linków.

Linki tekstowe muszą być łatwe do odnalezienia przez wszystkich użytkowników serwisu.

Muszą odróżniać się od tekstu zarówno kolorem jak i podkreśleniem. Niedopuszczalne jest zastosowanie tylko koloru do wyróżnienia linku.

Podkreślenia użyj w projekcie graficznym wyłącznie do oznaczenia linków. To samo dotyczy koloru linków. Nie może być on powtórzony na żadnym elemencie nieklikalnym i musi spełniać wymogi wskazane w punkcie “Kontrast treści”.

Po oznaczeniu linku kursorem myszy (**hover**) podkreślenie linku powinno znikać, a kolor linku zmieniać się na kolor o wyższym wskaźniku kontrastu do tła, niż przy kolorze bazowym linku.

5.3. Formularze.

Wymóg widoczności dotyczy również formularzy stosowanych w serwisie/systemie. W szczególności odnosi się to do widoczności ramek pól, etykiet pól oraz przycisków.

Wszystkie elementy formularzy muszą spełniać wymóg kontrastu w stosunku do tła na poziomie przynajmniej 3:1.

Tak jak w przypadku linków, przyciski formularzy po oznaczeniu kursorem myszy bądź fokusem klawiatury muszą stawać się widoczne dla użytkowników (zwiększenie kontrastu między kolorem przycisku a kolorem tekstu przycisku).

Etykiety pól powinny być widoczne (w niektórych przypadkach mogą być ukryte, jednakże muszą być możliwe do przetworzenia przez narzędzia asystujące - na przykład **<label>** do elementu **<input>** wyszukiwarki) i prezentowane bezpośrednio obok pola. Etykiety powinny być programistycznie powiązane z polami formularzy za pomocą atrybutów **“for”** i **“id”**.

Dodatkowe informacje, które ułatwią użytkownikowi wypełnić formularz powinny być powiązane z elementem **<input>** za pomocą atrybutu **ARIA-labelledby**.

Informacje o błędach powinny być prezentowane tekstowo, bezpośrednio obok pól których dotyczą (dodatkowo powiązane z polem poprzez **ARIA-describedby**) oraz pod nagłówkiem rozpoczynającym blok z formularzem. Powinien istnieć jeden, ogólny komunikat informujący użytkownika o błędnym wypełnieniu formularza wraz z rolą alert https://www.w3.org/TR/WCAG20-TECHS/ARIA19.htm

5.4. Fokus klawiatury.

Cały serwis będzie umożliwiał nawigację za pomocą samej klawiatury.

Fokus klawiatury powinien mieć formę wzmocnioną w stosunku do fokusu domyślnego przeglądarki i być widoczny przy nawigacji za pomocą klawiatury w formie ramki, wokół wybranego elementu.

Kolor ramki fokusu dobierz do schematu kolorystycznego serwisu tak aby był dobrze widoczny na oznaczonym elemencie (minimalny kontrast – 3:1).

Przykład dobrze widocznego fokusu możesz zobaczyć w serwisie www.pfron.org.pl - wystarczy zacząć nawigację w serwisie/systemie za pomocą przycisku TAB.

Do rozróżnienia fokusa klawiatury i myszki możesz wykorzystać bibliotekę dostępną na stronie: https://github.com/ten1seven/what-input

5.5. Typografia.

Czcionki, których użyjesz w serwisie/systemie powinny być bezszeryfowe, o wysokim poziomie czytelności - także przy dużym powiększeniu. Przykładami tego typu czcionek są: Lato, Open Sans czy PT Sans.

Liczbę czcionek (kroju i wielkości) powinieneś ograniczyć w projekcie graficznym serwisu do niezbędnego minimum.

5.6. Spójna identyfikacja.

W ramach serwisu/systemu zaplanuj i zaprezentuj widoki tekstowych elementów semantycznych, takich jak:

- nagłówek poziomu 1, (każda strona powinna posiadać jeden nagłówek poziomu 1, pozostałe w odpowiedniej hierarchii, jeżeli treść jest wymagana.
- lista numerowana (uporządkowana),
- lista wypunktowana (nieuporządkowana),
- listy obu typów wielokrotnie zagnieżdżone,
- link,
- tekst podstawowy,
- tekst podstawowy wyróżniony,
- przycisk (3 schematy dla różnych funkcjonalności),
- listy rozwijane (**select**),
- przyciski typu radio,
- pola wyboru,
- pole edycyjne.

Wielkość krojów pisma , których użyjesz w poszczególnych stylach powinna odpowiadać hierarchii tych stylów względem siebie. Dobrym rozwiązaniem jest, abyś przyjął zasadę, iż nagłówek poziomu 6 powinien być co najmniej wielkości kroju pisma podstawowego, tylko pogrubionego.

Minimalna wielkość kroju pisma, którą dopuszczamy w projekcie graficznym to 12 px., przy czym treść podstawowa powinna mieć wielkość minimum 16 px.

Dla treści nie powinieneś stosować formatowania wersalikami.

Odstępy między wierszami w akapitach powinieneś ustawić na co najmniej 1,3-1,5 wysokości linii, a odległość między akapitami powinna być przynajmniej 1,5 razy większa niż ta pomiędzy wierszami. W innym przypadku powinieneś zapewnić możliwość zmiany wielkości, bez utraty treści (np. za pomocą **1.4.12 Text Spacing** – narzędzie wspomagające symulację strony ze zwiększonymi odstępami w zakresie podanym w WCAG 2.1.)

W jednym wersie powinieneś zaprezentować do 85 znaków.

Nie justuj (równoczesne wyrównanie do lewej i prawej) żadnej treści w projekcie graficznym. Dopuszczamy tylko wyrównanie do lewej, a w uzasadnionych sytuacjach wyśrodkowanie tekstu.

Tam, gdzie to możliwe, treść prezentuj w formie tekstu, a nie grafiki tekstu. Do osiągnięcia pożądanego wyglądu użyj odpowiednich stylów CSS.

Spójna identyfikacja to nie tylko spójność użycia krojów pisma lub stylów. Rozumiemy to jako jednolitą implementację tych samych elementów na różnych podstronach. Przykładem jest ten sam opis logo serwisu/systemu we wszystkich miejscach, w których występuje bądź też elementu ukazującego podpowiedź przy wypełnianiu formularza (nie może raz być to “otwórz podpowiedź”, a za innym razem “pomoc”).

5.7. Tabele.

Pamiętaj, że tabele z danymi prezentowane w projekcie graficznym powinny posiadać wyraźnie odróżniające się od reszty komórek wersy/kolumny nagłówkowe. Prawidłowa implementacja jest kluczowa dla zrozumienia tabeli przez narzędzia asystujące.

Musisz zwrócić szczególną uwagę na informowanie technologii asystującej na temat stanu sortowania/filtrowania oraz ilości danych w tabeli

5.8. Możliwość swobodnej zmiany wielkości widoku.

Pamiętaj, że koncepcja serwisu zakłada możliwość swobodnej zmiany wielkości strony (Ctrl + oraz Ctrl -). Przy każdej szerokości ekranu/poziomie powiększenia (nie tylko przeznaczonej dla tabletów i smartfonów) wszystkie treści i funkcje serwisu powinny być czytelne. Projekt graficzny musi umożliwiać zaprogramowanie w ten sposób serwisu.

5.9. Elementy ruchome.

Dopuszczamy elementy ruchome w serwisie, ale tylko w połączeniu z przyciskiem, który umożliwi użytkownikowi ich zatrzymanie i ponowne uruchomienie.

Żaden element serwisu/systemu nie może migać, jeśli czynność ta powtarza się więcej niż 3 razy na sekundę.

5.10. Elementy rozwijane.

Wszystkim elementom, które są rozwijane powinieneś przypisać atrybut **ARIA-expanded**. Jego wartość należy ustawić z poziomu JS (**true** albo **false**) - w zależności czy element jest zwinięty czy rozwinięty: **ARIA-expanded="true"** jeśli jest rozwinięty, **ARIA-expanded="false”** jeśli jest zwinięty. Dzięki temu użytkownicy korzystający z aplikacji asystujących będą wiedzieli jaka jest aktualna struktura zamieszczonych informacji.

5.11. Multimedia.

Naszą rekomendacją odnośnie materiałów wideo jest ich prezentacja za pomocą standardowego odtwarzacza YouTube. Treści wideo powinny posiadać napisy i audiodeskrypcję. Projekt graficzny powinien uwzględniać zamieszczanie bezpośrednio pod materiałem wideo linku do transkrypcji tekstowej materiału, jeśli nie jest umieszczona bezpośrednio w filmie.

5.12. Elementy zmienne.

Wszelkie elementy, które zmieniają swoją wartość, dzięki działaniu jakiegoś mechanizmu (na przykład kalkulatora czy formularza), powinny mieć atrybut **ARIA-live**. Dzięki niemu użytkownik jest informowany o zmianie treści na stronie. Przykłady działania atrybutu znajdziesz na stronie https://dequeuniversity.com/library/aria/liveregion-playground

5.13. Kolorystyka serwisu/systemu

Jedyne ograniczenia kolorystyczne w serwisie/systemie dotyczą logotypu Państwowego Funduszu Rehabilitacji Osób Niepełnosprawnych (patrz załącznik do wytycznych) oraz minimalnego kontrastu treści do tła oraz wymagania w 2.7. Spójna identyfikacja.

6. Zalecenia na poziomie AAA

Interfejs graficzny serwisu/systemu będzie zgodny z wytycznymi WCAG 2.1 poziomu A oraz AA. Dla wskazanych poniżej elementów interfejsu spełnione zostaną zalecenia na poziomie AAA:

1. 1.4.8 Prezentacja wizualna:
    a. szerokość nie przekracza 80 znaków, tekst nie jest wyjustowany,
    b. interlinia to przynajmniej 150%, a odstęp pomiędzy paragrafami 1.5 razy wartości interlinii,
    c. tekst powiększony do 200% nie wymaga przesuwania horyzontalnego.
2. 2.4.9 Cel łącza (z samego łącza): wymaganie opisaliśmy w punkcie 3.8 Linki;
3. 2.5.5 Rozmiar celu dotykowego: wielkość kontrolki (poziom AAA). Wielkość obiektu, który trzeba dotknąć lub kliknąć myszą, musi być na tyle duża, by Użytkownik mógł łatwo trafić palcem lub kursorem myszy;

7. Dokumenty

Wszystkie dokumenty, które będziesz publikował w Systemie, muszą spełniać wymagania WCAG w odniesieniu do dokumentów cyfrowych (zalecenia w tym zakresie dostępne są na stronie W3C opisujące techniki WCAG dla PDF - https://www.w3.org/TR/WCAG20-TECHS/pdf).

Jesteś zobowiązany do każdorazowej adaptacji dokumentów dostarczanych przez Zamawiającego oraz prawidłowego (zgodnego z wytycznymi WCAG) przygotowania Dokumentacji Użytkownika.

Dokumentację Użytkownika przygotujesz zgodnie z zasadami prostego języka umieszczonymi w serwisie gov.pl (https://www.gov.pl/web/sluzbacywilna/prosty-jezyk).