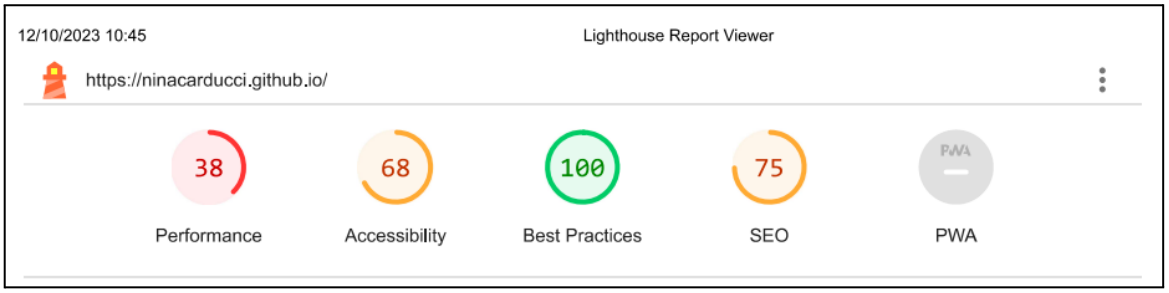
Rapport d’optimisation

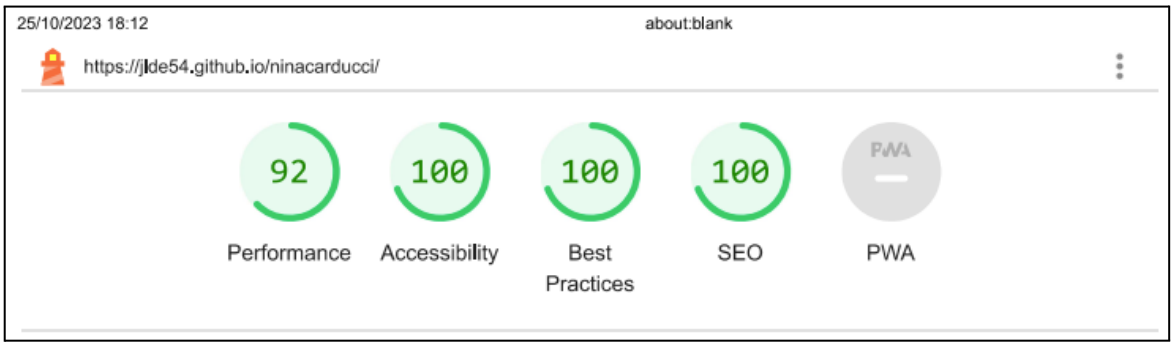
Site web de la photographe Nina Carducci

I. Comparatif avant et après optimisation

A. Score Lighthouse avant optimisation



B. Score Lighthouse après optimisation



II. Détails des optimisations effectuées

A. Optimisation des images

Le projet comporte originalement **14 images** pour un poids total de **29,4 Mo**.
Après modifications, le poids moyen total des images est de **253 Ko** soit un gain de **99.1%**.

Nous avons effectué les modifications suivantes aux images :

Images initiales	Taille initiale (Mo)	Dimension initiale (pixels)	Images finales	Taille finale (Ko)	Dimension finale (pixels)
camera.png	**1.58**	2024 x 2020	camera-100.webp	1.26	101 x 101
			camera-200.webp	3.47	202 x 202
			camera-300.webp	6.21	304 x 303
			camera-400.webp	**9.66**	405 x 404
			camera-500.webp	13.3	506 x 505
			camera-600.webp	16.9	607 x 606
nina.png	**2.05**	1460 x 1456	nina-100.webp	2.1	102 x 102
			nina-200.webp	5.59	197 x 197
			nina-300.webp	11.3	299 x 298
			nina-400.webp	**17.8**	394 x 393
			nina-500.webp	25.4	496 x 495
			nina-600.webp	33.2	584 x 582
edward-cisneros-3_h6-1NPDGw-unsplash.jpg	**5.43**	5653 x 2613	edward-cisneros-200.webp	4.83	198 x 91
			edward-cisneros-300.webp	8.33	297 x 137
			edward-cisneros-400.webp	11.9	396 x 183
			edward-cisneros-500.webp	16.4	509 x 235
			edward-cisneros-600.webp	**20.1**	594 x 274
			edward-cisneros-800.webp	29.0	791 x 366
			edward-cisneros-1200.webp	54.7	1187 x 549
nicholas-green-nPz8akkUmDI-unsplash.jpg	**1.81**	6000 x 2275	nicholas-green-200.webp	5.69	195 x 90
			nicholas-green-300.webp	10.3	300 x 139
			nicholas-green-400.webp	14.3	390 x 180

			nicholas-green-500.webp	20.2	525 x 243
			nicholas-green-600.webp	**24.2**	600 x 278
			nicholas-green-800.webp	32.7	780 x 361
			nicholas-green-1200.webp	54.0	1200 x 755
ryoji-iwata-wUZjnOv7t0g-unsplash.jpg	**1.54**	4540 x 2100	ryoji-iwata-200.webp	5.55	204 x 95
			ryoji-iwata-300.webp	9.68	306 x 142
			ryoji-iwata-400.webp	15.0	409 x 189
			ryoji-iwata-500.webp	21.1	499 x 231
			ryoji-iwata-600.webp	**30.2**	613 x 284
			ryoji-iwata-800.webp	51.1	817 x 378
			ryoji-iwata-1200.webp	101	1226 x 567
aaron-paul-wnX-fXzB6Cw-unsplash.jpg	**0.98**	2448 x 3264	aaron-paul-100.webp	3.61	110 x 147
			aaron-paul-200.webp	12.6	220 x 294
			aaron-paul-300.webp	25.6	330 x 441
			aaron-paul-400.webp	**40**	441 x 588
			aaron-paul-500.webp	56.1	551 x 734
			aaron-paul-600.webp	72	661 x 881
austin-neill-hgO1wFPXI3I-unsplash.jpg	**1.4**	5233 x 3489	austin-neill-100.webp	1.51	105 x 70
			austin-neill-200.webp	4.36	209 x 140
			austin-neill-300.webp	7.66	314 x 209
			austin-neill-400.webp	**11.0**	419 x 279
			austin-neill-500.webp	14.5	523 x 349
			austin-neill-600.webp	18.5	628 x 419
ali-morshedlou-WMD64tMfc4k-unsplash.jpg	**1.04**	4000 x 6000	ali-morshedlou-100.webp	1.33	100 x 150
			ali-morshedlou-200.webp	3.05	200 x 300
			ali-morshedlou-300.webp	5.05	300 x 450
			ali-morshedlou-400.webp	**7.31**	400 x 600
			ali-morshedlou-500.webp	9.62	500 x 750
			ali-morshedlou-600.webp	12.0	600 x 900

jason-goodman-tHO1_OuKbg0-unsplash.jpg	**0.697**	4084 x 2764	jason-goodman-100.webp	1.67	109 x 69
			jason-goodman-200.webp	3.67	204 x 138
			jason-goodman-300.webp	5.98	306 x 207
			jason-goodman-400.webp	**8.51**	408 x 276
			jason-goodman-500.webp	11.3	511 x 346
			jason-goodman-600.webp	14.2	613 x 415
mateus-campos-felipe-Fsgzm8N0hIY-unsplash.jpg	**1.83**	6000 x 3674	mateus-campos-felipe-100.webp	1.23	90 x 55
			mateus-campos-felipe-200.webp	5.39	240 x 147
			mateus-campos-felipe-300.webp	6.46	270 x 165
			mateus-campos-felipe-400.webp	**11.3**	390 x 239
			mateus-campos-felipe-500.webp	14.2	450 x 276
			mateus-campos-felipe-600.webp	21.4	600 x 367
hannah-busing-RvF2R_qMpRk-unsplash.jpg	**1.69**	4463 x 3562	hannah-busing-100.webp	1.47	100 x 80
			hannah-busing-200.webp	3.15	200 x 160
			hannah-busing-300.webp	4.99	300 x 240
			hannah-busing-400.webp	**7.44**	400 x 320
			hannah-busing-500.webp	9.39	500 x 400
			hannah-busing-600.webp	11.6	600 x 480
jakob-owens-SiniLJkXhMc-unsplash.jpg	**5.98**	4480 x 6720	jakob-owens-100.webp	3.09	101 x 151
			jakob-owens-200.webp	9.56	202 x 302
			jakob-owens-300.webp	17.8	302 x 454
			jakob-owens-400.webp	**26.7**	400 x 600
			jakob-owens-500.webp	37.7	504 x 756
			jakob-owens-600.webp	47.0	604 x 900
ade-tunji-rVkhWWZFAtQ-unsplash.jpg	**0.978**	2211 x 3314	ade-tunji-100.webp	2.75	99 x 149
			ade-tunji-200.webp	6.93	199 x 298

			ade-tunji-300.webp	12.1	298 x 447
			ade-tunji-400.webp	**18.9**	398 x 597
			ade-tunji-500.webp	27.9	497 x 746
			ade-tunji-600.webp	38.6	597 x 895
nino-van-prattenburg--443cl1uR_8-unsplash.jpg	**2.41**	4024 x 6048	nino-van-prattenburg-100.webp	3.15	101 x 151
			nino-van-prattenburg-200.webp	7.87	201 x 302
			nino-van-prattenburg-300.webp	13.7	302 x 454
			nino-van-prattenburg-400.webp	**20.2**	402 x 605
			nino-van-prattenburg-500.webp	28.2	503 x 756
			nino-van-prattenburg-600.webp	37.1	604 x 907
TOTAL	**29.4 Mo**			**253 Ko**	

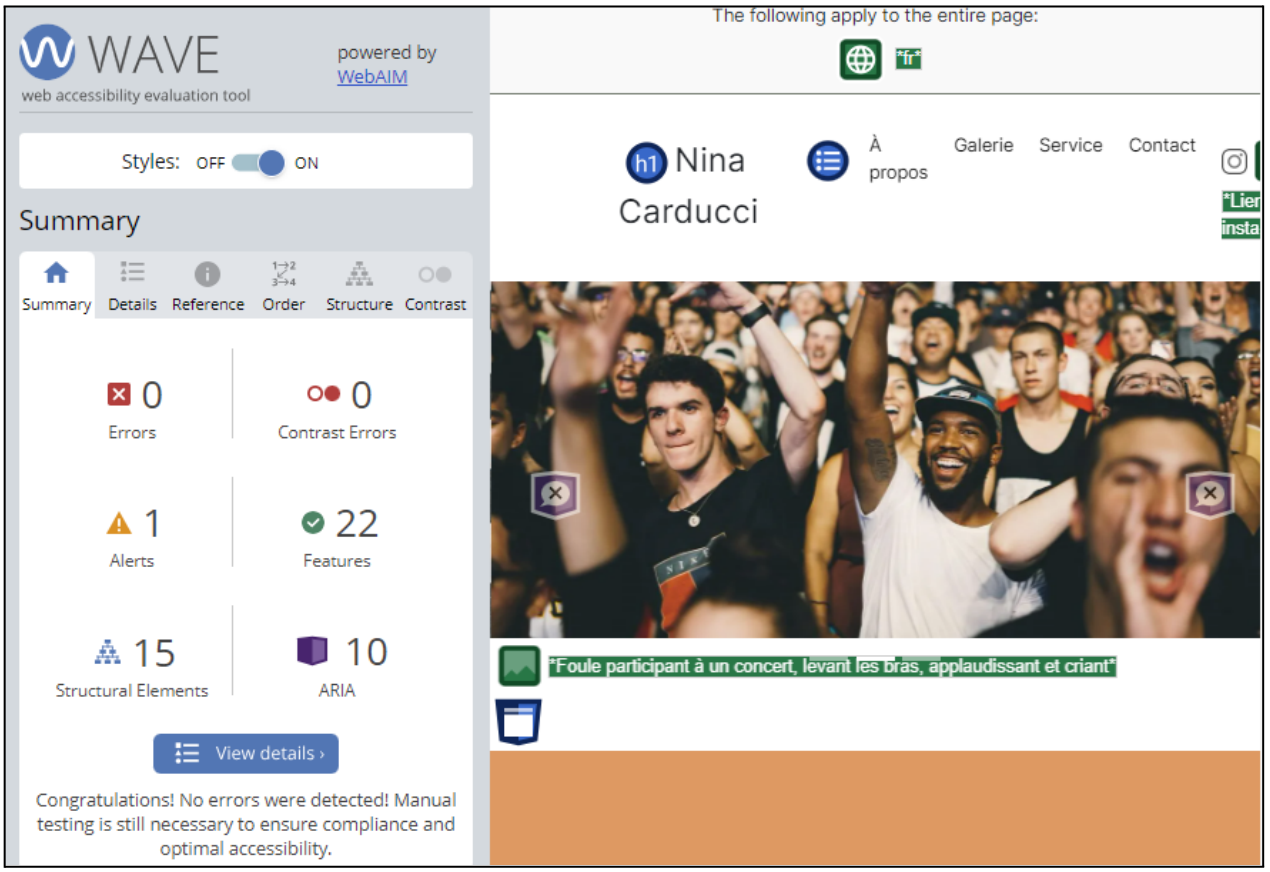
B. Optimisation des performances

#	Modification effectuée	Objectif recherché	Commentaires
1	Réduire la taille des images .jpg et .png	Gagner en vitesse d'affichage des images	Effectué pour les 14 images du site Outil utilisé : ImageMagick
2	Transformer les images en format .webp	Les images WebP sont plus petites que leurs homologues JPEG et PNG et permet de gagner encore plus en vitesse d'affichage des images	Effectué pour les 14 images du site Outil utilisé : libewebp-1.2.3-windows-x64
3	Utiliser les images "responsive" avec l'attribut "srcset"	Utiliser des images de dimensions différentes au lieu d'une seule. Le browser choisira laquelle utiliser en fonction de l'écran	Effectué pour les 14 images du site
4	Renseigner les attributs LOADING="lazy" des images dans index.html	Les images non visibles avant que l'utilisateur ne les affiche sont chargées après les autres ressources plus critiques	2 images du carrousel, 9 images de la galerie et l'image du pied de page sont concernées
5	Ajouter les propriétés "width" et "height" manquantes	S'assurer que le navigateur peut allouer la quantité d'espace appropriée dans le document pendant le chargement de l'image.	Images instagram (.social-link img) + nina (.picture img) dans style.css et ryoji-iwata (.carousel-item) dans bootstrap.css
6	Éliminer le CSS inutilisé dans bootstrap.css	Diminuer le temps du navigateur pour traiter et télécharger les fichiers CSS.	Nous sommes passé de 11.266 à 950 lignes
7	Éliminer le JS inutilisé dans bootstrap.bundle.js	Diminuer le temps du navigateur pour traiter et télécharger les fichiers JS.	Nous sommes passé de 6.812 à 2.070 lignes
8	Minifier les fichiers CSS : bootstrap.css et style.css	Éliminer tous les caractères et espaces inutiles du balisage CSS. La minification accélère le chargement des pages web.	
9	Minifier les fichiers JS : bootstrap.bundle.js, script.js et maugallery.js	Suppression des espaces, commentaires et points-virgules, etc. La minimisation du code JavaScript entraîne une taille de fichier compacte et accélère le chargement des pages web.	
10	Éliminer l'effet de blocage de rendu des Google Fonts	Optimiser la vitesse de la page en chargeant la police CSS de manière asynchrone	https://pagespeedchecklist.com/asynchronous-google-fonts
11	Différer l'exécution des scripts avec l'attribut "defer"	Éviter de bloquer le chargement de la page pendant l'exécution des scripts. L'attribut defer indique au navigateur de ne pas attendre le script.	Appliqué sur les scripts Javascript

C. Optimisation de l’accessibilité

#	Modification effectuée	Objectif recherché	Commentaires / Référence
1	Associer l’étiquette (label) du champ au champ du formulaire	Permettre aux technologies d’assistance de restituer correctement l’information. L’utilisateur comprend la saisie qui est attendue dans le champ.	
2	Renseigner un nom reconnaissable dans les liens.	Les liens manquants ou avec un texte vague peuvent rendre la navigation confuse et être une source de frustration pour les personnes handicapées.	index.html (lignes 51 à 55) : liens vers “A propos”, “Galerie”, “Services” et “Contact”
3	Appliquer un meilleur contraste entre la couleur d'avant-plan et d'arrière-plan	Le contraste des couleurs est important pour les utilisateurs malvoyants ou daltoniens. Un bon contraste des couleurs signifie que tous les utilisateurs peuvent voir votre contenu, quel que soit l’appareil qu’ils utilisent ou l’éclairage de leur environnement.	Appliqué : color: #000 au lieu de color : #FFF sur un background-color: #BEB45A; style.css (lignes 163 à 167)
4	Ajouter l'attribut "lang" dans la balise <html>	Préciser la langue d'origine d'une page Web afin de garantir la bonne restitution des contenus textuels.	<html lang="fr">
5	Ordonner de façon séquentielle et ascendante les éléments <hn>	Permettre aux autres moteurs de recherche d’analyser de quoi parle une page. Ils ne voient pas une page comme nous la voyons, ces balises permettent de savoir comment une page est construite et de comprendre le contexte de chaque partie.	<h1 class="name">Nina Carducci</h1> <h2 class="about-me__title">A propos de moi</h2> <h2 class="title">Portfolio</h2> <h2 class="title">Mes services</h2> <h3>Shooting photo</h3> <h3>Retouches</h3> <h3>Album photos</h3> <h3>Une question ? Une demande de devis ?</h3> <h4>350€/demi journée</h4> <h4>50€/photo</h4> <h4>400€ album A4</h4>
6	Renseigner les attributs ALT des images	Pour les personnes malvoyantes : le lecteur d’écran indiquera oralement le contenu de l’image. Le texte du alt sera visible même si l’image ne se charge pas.	
7	Renseigner les attributs TITLE des images	Afficher la description de l’image lors du passage du curseur du visiteur sur l'image.	

Rapport Wave Evaluation Tool

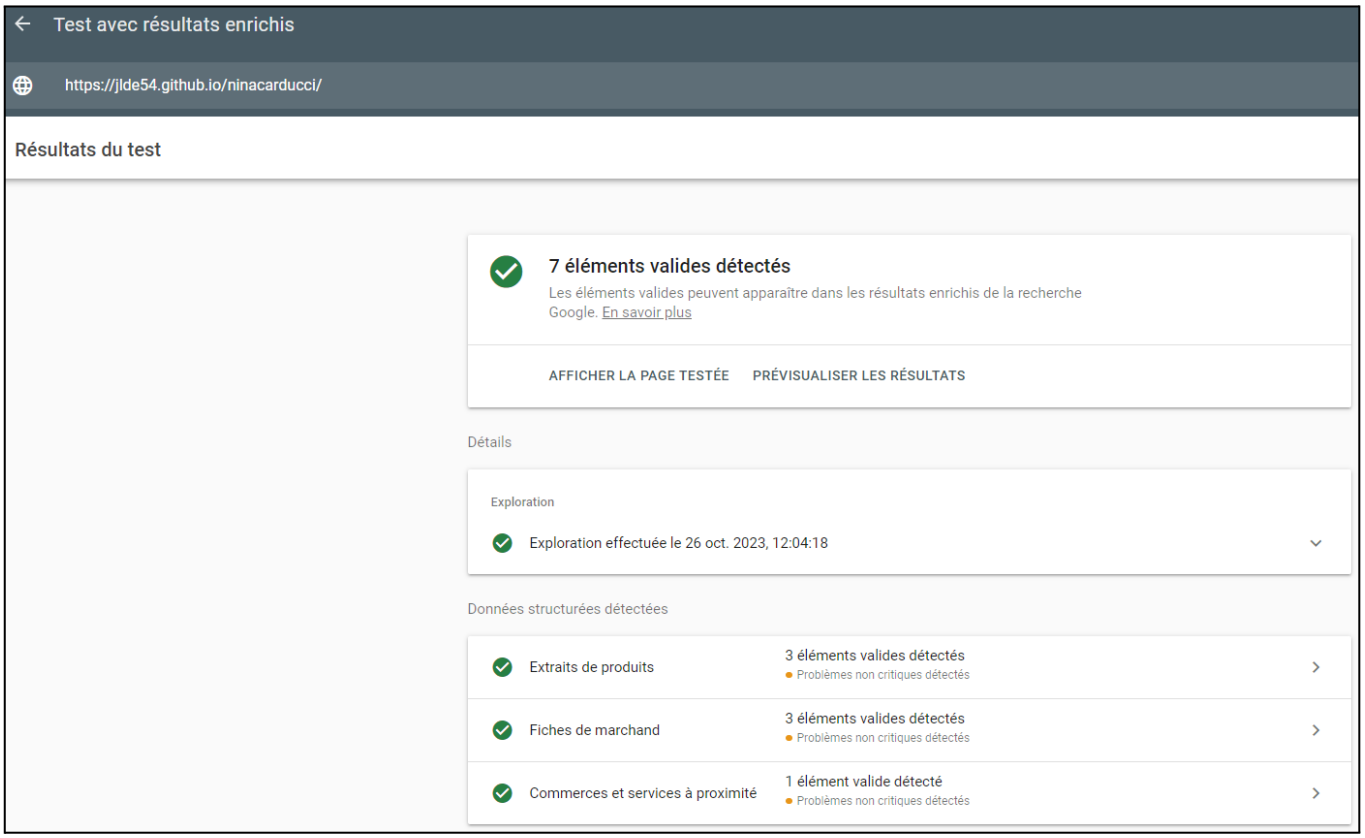


D. Optimisation du SEO (référencement naturel)

#	Modification effectuée	Objectif recherché	Commentaires / Référence
1	Renseigner les attributs ALT des images	Les mots clés contenus dans le ALT sont pris en compte pour le positionnement dans Google. La recherche universelle (c'est-à-dire la possibilité de chercher sur Google via divers types de médias : images – vidéos-textes) est de plus en plus grande. Référencer ses images dans Google devient donc primordial si l'on souhaite être visible sur ces différents canaux. Il permet aux robots de Google d'interpréter avec plus de précision le contenu de la page. Il génère du trafic vers votre site : il peut en effet vous apporter des visiteurs depuis depuis Google Images.	
2	Renseigner les attributs TITLE des images	Fournir aux moteurs de recherche des informations importantes concernant le sujet de votre image.	
3	Optimiser le référencement local du site web	Obtenir un meilleur positionnement du site internet en haut des résultats de recherche sur les moteurs de recherche pour attirer des internautes et générer du trafic	Utiliser les microdonnées "schema.org" (<script type="application/ld+json">)
4	Mettre en place le référencement sur les réseaux sociaux	Contrôler comment le site web s'affiche sur les réseaux sociaux	Utiliser Open Graph (balises <meta> property="og:") et Twitter Cards (balises <meta> name="twitter:")
5	Renommer les images	Eviter les mots non significatifs pour les moteurs de recherche	
6	Mention de la balise <title>	Indiquer aux internautes et aux moteurs de recherche le sujet d'une page web.	
7	Mention de la balise <meta name= "description">	Montrer la pertinence d'un site web et inciter les internautes à visiter le site.	

Rapport du test Google Rich Snippets (référencement local).

https://search.google.com/test/rich-results



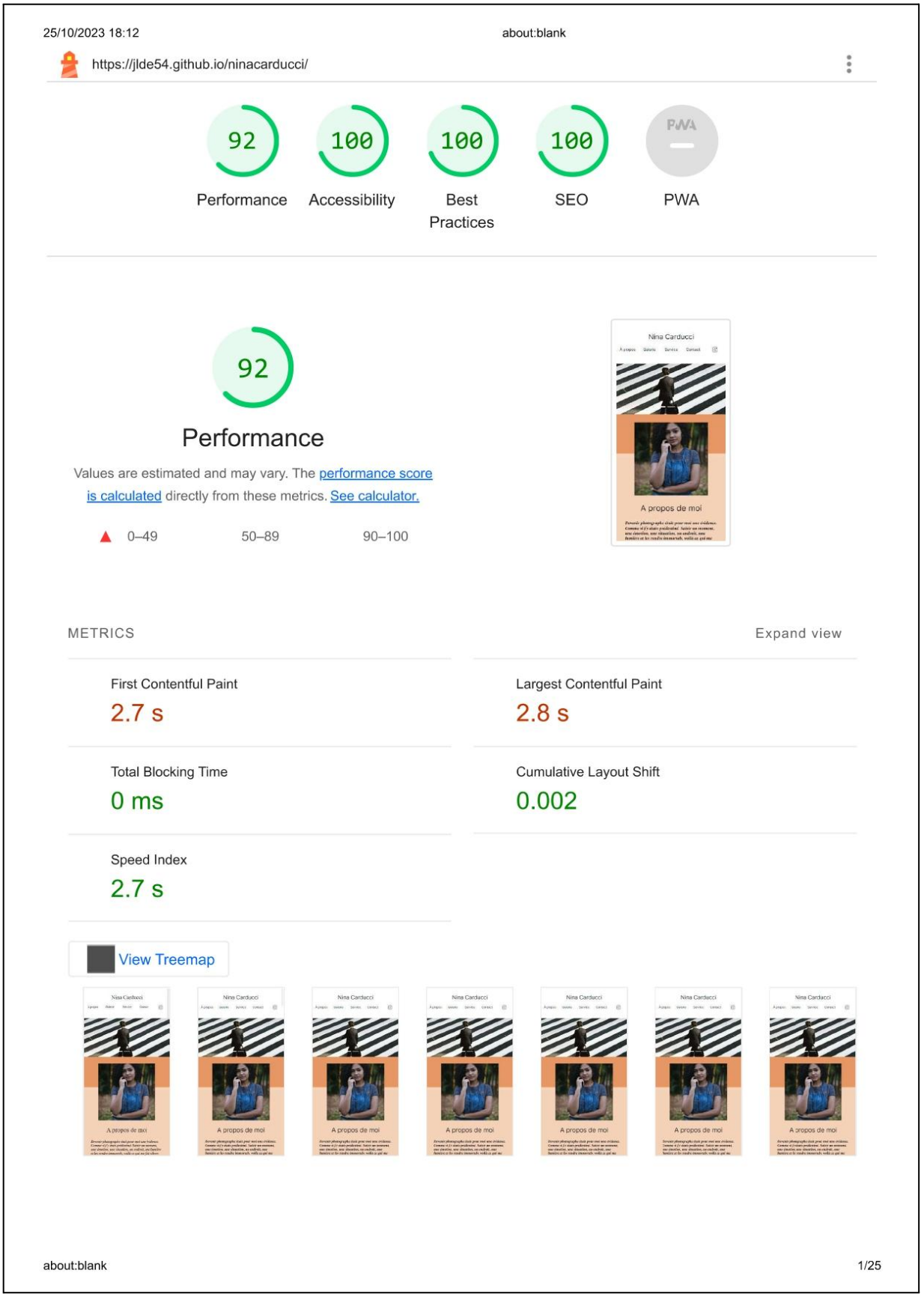
III. Réalisations additionnelles à la demande du client

A. La cliente a demandé l'ajout des informations suivantes :

1. Son adresse
2. Son n° de téléphone
3. La mention : "Je réponds au téléphone du lundi au vendredi de 10h à 19h

Ces informations ont été placées en dessous du formulaire de contact.

IV. Rapport complet de l’audit Lighthouse



Show audits relevant to: All FCP LCP TBT CLS

DIAGNOSTICS

▲ Serve static assets with an efficient cache policy — 10 resources found

A long cache lifetime can speed up repeat visits to your page. Learn more about efficient cache policies.

URL	Cache TTL	Transfer Size
GitHub Utility 1st Party		**187 KiB**
…concerts/aaron-paul-600.webp (jlde54.github.io)	10m	72 KiB
…slider/ryoji-iwata-800.webp (jlde54.github.io)	10m	51 KiB
…images/nina-600.webp (jlde54.github.io)	10m	33 KiB
…entreprise/ali-morshedlou-600.webp (jlde54.github.io)	10m	12 KiB
…bootstrap/bootstrap.bundle.min.js (jlde54.github.io)	10m	9 KiB
…bootstrap/bootstrap.min.css (jlde54.github.io)	10m	4 KiB
…assets/maugallery.min.js (jlde54.github.io)	10m	2 KiB
…assets/style.min.css (jlde54.github.io)	10m	1 KiB
…images/instagram.png (jlde54.github.io)	10m	1 KiB
…assets/scripts.min.js (jlde54.github.io)	10m	0 KiB

○ Avoid chaining critical requests — 3 chains found

The Critical Request Chains below show you what resources are loaded with a high priority. Consider reducing the length of chains, reducing the download size of resources, or deferring the download of unnecessary resources to improve page load. Learn how to avoid chaining critical requests. FCP LCP

Maximum critical path latency: **178.295 ms**

Initial Navigation

/ninacarducci/ (jlde54.github.io)

…bootstrap/bootstrap.min.css (jlde54.github.io) **- 40.623 ms, 3.93 KiB**

…assets/style.min.css (jlde54.github.io) **- 41.533 ms, 1.47 KiB**

/jquery-3.4.1.min.js (code.jquery.com) **- 123.624 ms, 30.03 KiB**

Largest Contentful Paint element — 2,760 ms

This is the largest contentful element painted within the viewport. Learn more about the Largest Contentful Paint element LCP

Element	
	img.d-block.w-100

Phase	% of LCP	Timing
TTFB	23%	640 ms
Load Delay	0%	0 ms
Load Time	29%	810 ms
Render Delay	48%	1,310 ms

Avoid large layout shifts — 5 elements found

These DOM elements contribute most to the CLS of the page. Learn how to improve CLS CLS

Element	CLS Contribution
 h2.about-me__title	0.001
p.about-me__introduction	0.000

Element	CLS Contribution
h1.name	0.000
h1.name	0.000
a	0.000

Avoid long main-thread tasks — 2 long tasks found

Lists the longest tasks on the main thread, useful for identifying worst contributors to input delay. Learn how to avoid long main-thread tasks TBT

URL	Start Time	Duration
GitHub Utility 1st Party		154 ms
/ninacarducci/ (jlde54.github.io)	841 ms	103 ms
/ninacarducci/ (jlde54.github.io)	786 ms	51 ms

More information about the performance of your application. These numbers don't directly affect the Performance score.

PASSED AUDITS (34) Hide

Eliminate render-blocking resources — Potential savings of 0 ms

Resources are blocking the first paint of your page. Consider delivering critical JS/CSS inline and deferring all non-critical JS/styles. Learn how to eliminate render-blocking resources. FCP LCP

URL	Transfer Size	Potential Savings
jQuery CDN Cdn	**30.0 KiB**	**1,090 ms**
/jquery-3.4.1.min.js (code.jquery.com)	30.0 KiB	1,090 ms

Properly size images

Serve images that are appropriately-sized to save cellular data and improve load time. Learn how to size images.

Defer offscreen images

Consider lazy-loading offscreen and hidden images after all critical resources have finished loading to lower time to interactive. Learn how to defer offscreen images.

Minify CSS

Minifying CSS files can reduce network payload sizes. Learn how to minify CSS. FCP LCP

Minify JavaScript

Minifying JavaScript files can reduce payload sizes and script parse time. Learn how to minify JavaScript. FCP LCP

Reduce unused CSS

Reduce unused rules from stylesheets and defer CSS not used for above-the-fold content to decrease bytes consumed by network activity. Learn how to reduce unused CSS. FCP LCP

Reduce unused JavaScript

Reduce unused JavaScript and defer loading scripts until they are required to decrease bytes consumed by network activity. Learn how to reduce unused JavaScript. LCP

Efficiently encode images

Optimized images load faster and consume less cellular data. Learn how to efficiently encode images.

Serve images in next-gen formats

Image formats like WebP and AVIF often provide better compression than PNG or JPEG, which means faster downloads and less data consumption. Learn more about modern image formats.

Enable text compression

Text-based resources should be served with compression (gzip, deflate or brotli) to minimize total network bytes. Learn more about text compression. FCP LCP

Preconnect to required origins

Consider adding `preconnect` or `dns-prefetch` resource hints to establish early connections to important third-party origins. Learn how to preconnect to required origins. FCP LCP

Initial server response time was short — Root document took 30 ms

Keep the server response time for the main document short because all other requests depend on it. Learn more about the Time to First Byte metric. FCP LCP

URL	Time Spent
GitHub Utility 1st Party	30 ms
/ninacarducci/ (jlde54.github.io)	30 ms

Avoid multiple page redirects

Redirects introduce additional delays before the page can be loaded. Learn how to avoid page redirects. FCP LCP

Preload key requests

Consider using `<link rel=preload>` to prioritize fetching resources that are currently requested later in page load. Learn how to preload key requests. FCP LCP

Use HTTP/2

HTTP/2 offers many benefits over HTTP/1.1, including binary headers and multiplexing. Learn more about HTTP/2.

Use video formats for animated content

Large GIFs are inefficient for delivering animated content. Consider using MPEG4/WebM videos for animations and PNG/WebP for static images instead of GIF to save network bytes. Learn more about efficient video formats LCP

Remove duplicate modules in JavaScript bundles

Remove large, duplicate JavaScript modules from bundles to reduce unnecessary bytes consumed by network activity. TBT

Avoid serving legacy JavaScript to modern browsers

Polyfills and transforms enable legacy browsers to use new JavaScript features. However, many aren't necessary for modern browsers. For your bundled JavaScript, adopt a modern script deployment strategy using module/nomodule feature detection to reduce the amount of code shipped to modern browsers, while retaining support for legacy browsers. Learn how to use modern JavaScript TBT

Preload Largest Contentful Paint image

If the LCP element is dynamically added to the page, you should preload the image in order to improve LCP. Learn more about preloading LCP elements. LCP

Avoids enormous network payloads — Total size was 287 KiB

Large network payloads cost users real money and are highly correlated with long load times. Learn how to reduce payload sizes. LCP

Show 3rd-party resources (5)

URL	Transfer Size
GitHub Utility 1st Party	**178.3 KiB**
…concerts/aaron-paul-600.webp (jlde54.github.io)	72.3 KiB
…slider/ryoji-iwata-800.webp (jlde54.github.io)	51.4 KiB
…images/nina-600.webp (jlde54.github.io)	33.4 KiB
…entreprise/ali-morshedlou-600.webp (jlde54.github.io)	12.2 KiB
…bootstrap/bootstrap.bundle.min.js (jlde54.github.io)	9.0 KiB
Google Fonts Cdn	**64.5 KiB**
…v13/UcCO3FwrK….woff2 (fonts.gstatic.com)	21.1 KiB
…v13/rnCu-xNNw….woff2 (fonts.gstatic.com)	15.3 KiB
…v13/rnCu-xNNw….woff2 (fonts.gstatic.com)	14.5 KiB
…v13/rnCr-xNNw….woff2 (fonts.gstatic.com)	13.5 KiB
jQuery CDN Cdn	**30.0 KiB**
/jquery-3.4.1.min.js (code.jquery.com)	30.0 KiB

Avoids an excessive DOM size — 144 elements

A large DOM will increase memory usage, cause longer style calculations, and produce costly layout reflows. Learn how to avoid an excessive DOM size. TBT

Statistic	Element	Value
Total DOM Elements		144
Maximum DOM Depth	div.mg-prev	10
Maximum Child Elements	body	9

User Timing marks and measures

Consider instrumenting your app with the User Timing API to measure your app's real-world performance during key user experiences. Learn more about User Timing marks.

JavaScript execution time — 0.1 s

Consider reducing the time spent parsing, compiling, and executing JS. You may find delivering smaller JS payloads helps with this. Learn how to reduce Javascript execution time. TBT

Show 3rd-party resources (1)

URL	Total CPU Time	Script Evaluation	Script Parse
GitHub Utility 1st Party	**381 ms**	**10 ms**	**3 ms**
/ninacarducci/ (jlde54.github.io)	381 ms	10 ms	3 ms
jQuery CDN Cdn	**265 ms**	**116 ms**	**5 ms**
/jquery-3.4.1.min.js (code.jquery.com)	265 ms	116 ms	5 ms
Unattributable	**147 ms**	**5 ms**	**0 ms**
Unattributable	147 ms	5 ms	0 ms

Minimizes main-thread work — 0.8 s

Consider reducing the time spent parsing, compiling and executing JS. You may find delivering smaller JS payloads helps with this. Learn how to minimize main-thread work TBT

Category	Time Spent
Other	331 ms
Style & Layout	290 ms
Script Evaluation	153 ms
Rendering	32 ms
Parse HTML & CSS	19 ms
Script Parsing & Compilation	13 ms

All text remains visible during webfont loads

Leverage the `font-display` CSS feature to ensure text is user-visible while webfonts are loading. Learn more about `font-display`. FCP LCP

Minimize third-party usage — Third-party code blocked the main thread for 0 ms

Third-party code can significantly impact load performance. Limit the number of redundant third-party providers and try to load third-party code after your page has primarily finished loading. Learn how to minimize third-party impact. TBT

Third-Party	Transfer Size	Main-Thread Blocking Time
Google Fonts Cdn	**65 KiB**	**0 ms**
…v13/UcCO3FwrK….woff2 (fonts.gstatic.com)	21 KiB	0 ms
…v13/rnCu-xNNw….woff2 (fonts.gstatic.com)	15 KiB	0 ms
…v13/rnCu-xNNw….woff2 (fonts.gstatic.com)	15 KiB	0 ms
…v13/rnCr-xNNw….woff2 (fonts.gstatic.com)	14 KiB	0 ms
jQuery CDN Cdn	**30 KiB**	**0 ms**
/jquery-3.4.1.min.js (code.jquery.com)	30 KiB	0 ms

Lazy load third-party resources with facades

Some third-party embeds can be lazy loaded. Consider replacing them with a facade until they are required. Learn how to defer third-parties with a facade. TBT

Largest Contentful Paint image was not lazily loaded

Above-the-fold images that are lazily loaded render later in the page lifecycle, which can delay the largest contentful paint. Learn more about optimal lazy loading. LCP

Element	
	img.d-block.w-100

Uses passive listeners to improve scrolling performance

Consider marking your touch and wheel event listeners as `passive` to improve your page's scroll performance. Learn more about adopting passive event listeners.

Avoids `document.write()`

For users on slow connections, external scripts dynamically injected via `document.write()` can delay page load by tens of seconds. Learn how to avoid document.write().

Avoid non-composited animations

Animations which are not composited can be janky and increase CLS. Learn how to avoid non-composited animations CLS

Image elements have explicit `width` and `height`

Set an explicit width and height on image elements to reduce layout shifts and improve CLS. Learn how to set image dimensions CLS

Has a `<meta name="viewport">` tag with `width` or `initial-scale`

A `<meta name="viewport">` not only optimizes your app for mobile screen sizes, but also prevents a 300 millisecond delay to user input. Learn more about using the viewport meta tag. TBT

Page didn't prevent back/forward cache restoration

Many navigations are performed by going back to a previous page, or forwards again. The back/forward cache (bfcache) can speed up these return navigations. Learn more about the bfcache

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

ADDITIONAL ITEMS TO MANUALLY CHECK (10)

Hide

- Interactive controls are keyboard focusable

  Custom interactive controls are keyboard focusable and display a focus indicator. Learn how to make custom controls focusable.

- Interactive elements indicate their purpose and state

  Interactive elements, such as links and buttons, should indicate their state and be distinguishable from non-interactive elements. Learn how to decorate interactive elements with affordance hints.

- The page has a logical tab order

  Tabbing through the page follows the visual layout. Users cannot focus elements that are offscreen. Learn more about logical tab ordering.

- Visual order on the page follows DOM order

  DOM order matches the visual order, improving navigation for assistive technology. Learn more about DOM and visual ordering.

- User focus is not accidentally trapped in a region

  A user can tab into and out of any control or region without accidentally trapping their focus. Learn how to avoid focus traps.

- The user's focus is directed to new content added to the page

  If new content, such as a dialog, is added to the page, the user's focus is directed to it. Learn how to direct focus to new content.

- HTML5 landmark elements are used to improve navigation

  Landmark elements (`<main>`, `<nav>`, etc.) are used to improve the keyboard navigation of the page for assistive technology. Learn more about landmark elements.

- Offscreen content is hidden from assistive technology

  Offscreen content is hidden with display: none or aria-hidden=true. Learn how to properly hide offscreen content.

- Custom controls have associated labels

  Custom interactive controls have associated labels, provided by aria-label or aria-labelledby. Learn more about custom controls and labels.

- Custom controls have ARIA roles

  Custom interactive controls have appropriate ARIA roles. Learn how to add roles to custom controls.

These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

PASSED AUDITS (21) Hide

- `[aria-*]` attributes match their roles

  Each ARIA `role` supports a specific subset of `aria-*` attributes. Mismatching these invalidates the `aria-*` attributes. Learn how to match ARIA attributes to their roles.

- `[aria-hidden="true"]` is not present on the document `<body>`

  Assistive technologies, like screen readers, work inconsistently when `aria-hidden="true"` is set on the document `<body>`. Learn how `aria-hidden` affects the document body.

- `[aria-*]` attributes have valid values

  Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid values. Learn more about valid values for ARIA attributes.

- `[aria-*]` attributes are valid and not misspelled

  Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid names. Learn more about valid ARIA attributes.

- Buttons have an accessible name

  When a button doesn't have an accessible name, screen readers announce it as "button", making it unusable for users who rely on screen readers. Learn how to make buttons more accessible.

- ARIA IDs are unique

  The value of an ARIA ID must be unique to prevent other instances from being overlooked by assistive technologies. Learn how to fix duplicate ARIA IDs.

- Image elements have `[alt]` attributes

Informative elements should aim for short, descriptive alternate text. Decorative elements can be ignored with an empty alt attribute. Learn more about the `alt` attribute.

Input buttons have discernible text.

Adding discernable and accessible text to input buttons may help screen reader users understand the purpose of the input button. Learn more about input buttons.

`[user-scalable="no"]` is not used in the `<meta name="viewport">` element and the `[maximum-scale]` attribute is not less than 5.

Disabling zooming is problematic for users with low vision who rely on screen magnification to properly see the contents of a web page. Learn more about the viewport meta tag.

`[aria-hidden="true"]` elements do not contain focusable descendents

Focusable descendents within an `[aria-hidden="true"]` element prevent those interactive elements from being available to users of assistive technologies like screen readers. Learn how `aria-hidden` affects focusable elements.

Background and foreground colors have a sufficient contrast ratio

Low-contrast text is difficult or impossible for many users to read. Learn how to provide sufficient color contrast.

Document has a `<title>` element

The title gives screen reader users an overview of the page, and search engine users rely on it heavily to determine if a page is relevant to their search. Learn more about document titles.

`<html>` element has a `[lang]` attribute

If a page doesn't specify a `lang` attribute, a screen reader assumes that the page is in the default language that the user chose when setting up the screen reader. If the page isn't actually in the default language, then the screen reader might not announce the page's text correctly. Learn more about the `lang` attribute.

`<html>` element has a valid value for its `[lang]` attribute

Specifying a valid BCP 47 language helps screen readers announce text properly. Learn how to use the `lang` attribute.

Form elements have associated labels

Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Learn more about form element labels.

Links have a discernible name

Link text (and alternate text for images, when used as links) that is discernible, unique, and focusable improves the navigation experience for screen reader users. Learn how to make links accessible.

Lists contain only `<li>` elements and script supporting elements (`<script>` and `<template>`).

Screen readers have a specific way of announcing lists. Ensuring proper list structure aids screen reader output. Learn more about proper list structure.

List items (`<li>`) are contained within `<ul>`, `<ol>` or `<menu>` parent elements

Screen readers require list items (`<li>`) to be contained within a parent `<ul>`, `<ol>` or `<menu>` to be announced properly. Learn more about proper list structure.

Heading elements appear in a sequentially-descending order

Properly ordered headings that do not skip levels convey the semantic structure of the page, making it easier to navigate and understand when using assistive technologies. Learn more about heading order.

Values assigned to `role=""` are valid ARIA roles.

ARIA `roles` enable assistive technologies to know the role of each element on the web page. If the `role` values are misspelled, not existing ARIA `role` values, or abstract roles, then the purpose of the element will not be communicated to users of assistive technologies. Learn more about ARIA roles.

Image elements do not have `[alt]` attributes that are redundant text.

Informative elements should aim for short, descriptive alternative text. Alternative text that is exactly the same as the text adjacent to the link or image is potentially confusing for screen reader users, because the text will be read twice. Learn more about the `alt` attribute.

NOT APPLICABLE (40) Hide

`[accesskey]` values are unique

Access keys let users quickly focus a part of the page. For proper navigation, each access key must be unique. Learn more about access keys.

`button`, `link`, and `menuitem` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to make command elements more accessible.

Elements with `role="dialog"` or `role="alertdialog"` have accessible names.

ARIA dialog elements without accessible names may prevent screen readers users from discerning the purpose of these elements. Learn how to make ARIA dialog elements more accessible.

- ARIA input fields have accessible names

  When an input field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more about input field labels.

- ARIA `meter` elements have accessible names

  When a meter element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to name `meter` elements.

- ARIA `progressbar` elements have accessible names

  When a `progressbar` element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to label `progressbar` elements.

- `[role]`s have all required `[aria-*]` attributes

  Some ARIA roles have required attributes that describe the state of the element to screen readers. Learn more about roles and required attributes.

- Elements with an ARIA `[role]` that require children to contain a specific `[role]` have all required children.

  Some ARIA parent roles must contain specific child roles to perform their intended accessibility functions. Learn more about roles and required children elements.

- `[role]`s are contained by their required parent element

  Some ARIA child roles must be contained by specific parent roles to properly perform their intended accessibility functions. Learn more about ARIA roles and required parent element.

- `[role]` values are valid

  ARIA roles must have valid values in order to perform their intended accessibility functions. Learn more about valid ARIA roles.

- Elements with the `role=text` attribute do not have focusable descendents.

  Adding `role=text` around a text node split by markup enables VoiceOver to treat it as one phrase, but the element's focusable descendents will not be announced. Learn more about the `role=text` attribute.

- ARIA toggle fields have accessible names

When a toggle field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more about toggle fields.

- ARIA `tooltip` elements have accessible names

When a tooltip element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to name `tooltip` elements.

- ARIA `treeitem` elements have accessible names

When a `treeitem` element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more about labeling `treeitem` elements.

- The page contains a heading, skip link, or landmark region

Adding ways to bypass repetitive content lets keyboard users navigate the page more efficiently. Learn more about bypass blocks.

- `<dl>`'s contain only properly-ordered `<dt>` and `<dd>` groups, `<script>`, `<template>` or `<div>` elements.

When definition lists are not properly marked up, screen readers may produce confusing or inaccurate output. Learn how to structure definition lists correctly.

- Definition list items are wrapped in `<dl>` elements

Definition list items (`<dt>` and `<dd>`) must be wrapped in a parent `<dl>` element to ensure that screen readers can properly announce them. Learn how to structure definition lists correctly.

- `[id]` attributes on active, focusable elements are unique

All focusable elements must have a unique `id` to ensure that they're visible to assistive technologies. Learn how to fix duplicate `id`s.

- No form fields have multiple labels

Form fields with multiple labels can be confusingly announced by assistive technologies like screen readers which use either the first, the last, or all of the labels. Learn how to use form labels.

- `<frame>` or `<iframe>` elements have a title

Screen reader users rely on frame titles to describe the contents of frames. Learn more about frame titles.

- `<html>` element has an `[xml:lang]` attribute with the same base language as the `[lang]` attribute.

If the webpage does not specify a consistent language, then the screen reader might not announce the page's text correctly.

Learn more about the `lang` attribute.

- `<input type="image">` elements have `[alt]` text

  When an image is being used as an `<input>` button, providing alternative text can help screen reader users understand the purpose of the button. Learn about input image alt text.

- Elements with visible text labels have matching accessible names.

  Visible text labels that do not match the accessible name can result in a confusing experience for screen reader users. Learn more about accessible names.

- Links are distinguishable without relying on color.

  Low-contrast text is difficult or impossible for many users to read. Link text that is discernible improves the experience for users with low vision. Learn how to make links distinguishable.

- The document does not use `<meta http-equiv="refresh">`

  Users do not expect a page to refresh automatically, and doing so will move focus back to the top of the page. This may create a frustrating or confusing experience. Learn more about the refresh meta tag.

- `<object>` elements have alternate text

  Screen readers cannot translate non-text content. Adding alternate text to `<object>` elements helps screen readers convey meaning to users. Learn more about alt text for `object` elements.

- Select elements have associated label elements.

  Form elements without effective labels can create frustrating experiences for screen reader users. Learn more about the `select` element.

- Skip links are focusable.

  Including a skip link can help users skip to the main content to save time. Learn more about skip links.

- No element has a `[tabindex]` value greater than 0

  A value greater than 0 implies an explicit navigation ordering. Although technically valid, this often creates frustrating experiences for users who rely on assistive technologies. Learn more about the `tabindex` attribute.

- Tables have different content in the summary attribute and `<caption>`.

  The summary attribute should describe the table structure, while `<caption>` should have the onscreen title. Accurate table mark-up helps users of screen readers. Learn more about summary and caption.

- Tables use `<caption>` instead of cells with the `[colspan]` attribute to indicate a caption.

  Screen readers have features to make navigating tables easier. Ensuring that tables use the actual caption element instead of cells with the `[colspan]` attribute may improve the experience for screen reader users. Learn more about captions.

- `<td>` elements in a large `<table>` have one or more table headers.

  Screen readers have features to make navigating tables easier. Ensuring that `<td>` elements in a large table (3 or more cells in width and height) have an associated table header may improve the experience for screen reader users. Learn more about table headers.

- Cells in a `<table>` element that use the `[headers]` attribute refer to table cells within the same table.

  Screen readers have features to make navigating tables easier. Ensuring `<td>` cells using the `[headers]` attribute only refer to other cells in the same table may improve the experience for screen reader users. Learn more about the `headers` attribute.

- `<th>` elements and elements with `[role="columnheader"/"rowheader"]` have data cells they describe.

  Screen readers have features to make navigating tables easier. Ensuring table headers always refer to some set of cells may improve the experience for screen reader users. Learn more about table headers.

- `[lang]` attributes have a valid value

  Specifying a valid BCP 47 language on elements helps ensure that text is pronounced correctly by a screen reader. Learn how to use the `lang` attribute.

- `<video>` elements contain a `<track>` element with `[kind="captions"]`

  When a video provides a caption it is easier for deaf and hearing impaired users to access its information. Learn more about video captions.

- All heading elements contain content.

  A heading with no content or inaccessible text prevent screen reader users from accessing information on the page's structure. Learn more about headings.

- Identical links have the same purpose.

  Links with the same destination should have the same description, to help users understand the link's purpose and decide whether to follow it. Learn more about identical links.

- Document has a main landmark.

  One main landmark helps screen reader users navigate a web page. Learn more about landmarks.

Touch targets have sufficient size and spacing.

Touch targets with sufficient size and spacing help users who may have difficulty targeting small controls to activate the targets. Learn more about touch targets.

Best Practices

TRUST AND SAFETY

Ensure CSP is effective against XSS attacks

A strong Content Security Policy (CSP) significantly reduces the risk of cross-site scripting (XSS) attacks. Learn how to use a CSP to prevent XSS

Description	Directive	Severity
No CSP found in enforcement mode		High

GENERAL

Detected JavaScript libraries

All front-end JavaScript libraries detected on the page. Learn more about this JavaScript library detection diagnostic audit.

Name	Version
jQuery	3.4.1

PASSED AUDITS (13) Hide

Uses HTTPS

All sites should be protected with HTTPS, even ones that don't handle sensitive data. This includes avoiding mixed content, where some resources are loaded over HTTP despite the initial request being served over HTTPS. HTTPS prevents intruders from tampering with or passively listening in on the communications between your app and your users, and is a prerequisite for HTTP/2 and many new web platform APIs. Learn more about HTTPS.

Avoids deprecated APIs

Deprecated APIs will eventually be removed from the browser. Learn more about deprecated APIs.

Allows users to paste into input fields

Preventing input pasting is a bad practice for the UX, and weakens security by blocking password managers.Learn more about user-friendly input fields.

Avoids requesting the geolocation permission on page load

Users are mistrustful of or confused by sites that request their location without context. Consider tying the request to a user action instead. Learn more about the geolocation permission.

Avoids requesting the notification permission on page load

Users are mistrustful of or confused by sites that request to send notifications without context. Consider tying the request to user gestures instead. Learn more about responsibly getting permission for notifications.

Displays images with correct aspect ratio

Image display dimensions should match natural aspect ratio. Learn more about image aspect ratio.

Serves images with appropriate resolution

Image natural dimensions should be proportional to the display size and the pixel ratio to maximize image clarity. Learn how to provide responsive images.

Page has the HTML doctype

Specifying a doctype prevents the browser from switching to quirks-mode. Learn more about the doctype declaration.

Properly defines charset

A character encoding declaration is required. It can be done with a `<meta>` tag in the first 1024 bytes of the HTML or in the Content-Type HTTP response header. Learn more about declaring the character encoding.

Avoids `unload` event listeners

The `unload` event does not fire reliably and listening for it can prevent browser optimizations like the Back-Forward Cache. Use `pagehide` or `visibilitychange` events instead. Learn more about unload event listeners

No browser errors logged to the console

Errors logged to the console indicate unresolved problems. They can come from network request failures and other browser concerns. Learn more about this errors in console diagnostic audit

No issues in the `Issues` panel in Chrome Devtools

Issues logged to the `Issues` panel in Chrome Devtools indicate unresolved problems. They can come from network request failures, insufficient security controls, and other browser concerns. Open up the Issues panel in Chrome DevTools for more details on each issue.

Page has valid source maps

Source maps translate minified code to the original source code. This helps developers debug in production. In addition, Lighthouse is able to provide further insights. Consider deploying source maps to take advantage of these benefits. Learn more about source maps.

NOT APPLICABLE (1) Hide

Fonts with `font-display: optional` are preloaded

Preload `optional` fonts so first-time visitors may use them. Learn more about preloading fonts

SEO

These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search Essentials.

ADDITIONAL ITEMS TO MANUALLY CHECK (1) Hide

Structured data is valid

Run the Structured Data Testing Tool and the Structured Data Linter to validate structured data. Learn more about Structured Data.

Run these additional validators on your site to check additional SEO best practices.

PASSED AUDITS (12) Hide

Has a `<meta name="viewport">` tag with `width` or `initial-scale`

A `<meta name="viewport">` not only optimizes your app for mobile screen sizes, but also prevents a 300 millisecond delay to user input. Learn more about using the viewport meta tag. TBT

Document has a `<title>` element

The title gives screen reader users an overview of the page, and search engine users rely on it heavily to determine if a page is relevant to their search. Learn more about document titles.

Document has a meta description

Meta descriptions may be included in search results to concisely summarize page content. Learn more about the meta description.

Page has successful HTTP status code

Pages with unsuccessful HTTP status codes may not be indexed properly. Learn more about HTTP status codes.

Links have descriptive text

Descriptive link text helps search engines understand your content. Learn how to make links more accessible.

Links are crawlable

Search engines may use `href` attributes on links to crawl websites. Ensure that the `href` attribute of anchor elements links to an appropriate destination, so more pages of the site can be discovered. Learn how to make links crawlable

Page isn't blocked from indexing

Search engines are unable to include your pages in search results if they don't have permission to crawl them. Learn more about crawler directives.

Image elements have `[alt]` attributes

Informative elements should aim for short, descriptive alternate text. Decorative elements can be ignored with an empty alt attribute. Learn more about the `alt` attribute.

Document has a valid `hreflang`

hreflang links tell search engines what version of a page they should list in search results for a given language or region. Learn more about `hreflang`.

Document uses legible font sizes — 100% legible text

Font sizes less than 12px are too small to be legible and require mobile visitors to "pinch to zoom" in order to read. Strive to have >60% of page text ≥12px. Learn more about legible font sizes.

Source	Selector	% of Page Text	Font Size
Legible text		100.00%	≥ 12px

Document avoids plugins

Search engines can't index plugin content, and many devices restrict plugins or don't support them. Learn more about avoiding plugins.

Tap targets are sized appropriately — 100% appropriately sized tap targets

Interactive elements like buttons and links should be large enough (48x48px), or have enough space around them, to be easy enough to tap without overlapping onto other elements. Learn more about tap targets.

NOT APPLICABLE (2) Hide

robots.txt is valid

If your robots.txt file is malformed, crawlers may not be able to understand how you want your website to be crawled or indexed. Learn more about robots.txt.

Document has a valid `rel=canonical`

Canonical links suggest which URL to show in search results. Learn more about canonical links.

PWA

These checks validate the aspects of a Progressive Web App. Learn what makes a good Progressive Web App.

INSTALLABLE

Web app manifest or service worker do not meet the installability requirements — 1 reason

Service worker is the technology that enables your app to use many Progressive Web App features, such as offline, add to homescreen, and push notifications. With proper service worker and manifest implementations, browsers can proactively prompt users to add your app to their homescreen, which can lead to higher engagement. Learn more about manifest installability requirements.

Failure reason
Page has no manifest <link> URL

PWA OPTIMIZED

▲ Is not configured for a custom splash screen Failures: No manifest was fetched.

A themed splash screen ensures a high-quality experience when users launch your app from their homescreens. Learn more about splash screens.

▲ Does not set a theme color for the address bar.
Failures: No manifest was fetched, No `<meta name="theme-color">` tag found.

The browser address bar can be themed to match your site. Learn more about theming the address bar.

Content is sized correctly for the viewport

If the width of your app's content doesn't match the width of the viewport, your app might not be optimized for mobile screens. Learn how to size content for the viewport.

Has a `<meta name="viewport">` tag with `width` or `initial-scale`

A `<meta name="viewport">` not only optimizes your app for mobile screen sizes, but also prevents a 300 millisecond delay to user input. Learn more about using the viewport meta tag. TBT

▲ Manifest doesn't have a maskable icon No manifest was fetched

A maskable icon ensures that the image fills the entire shape without being letterboxed when installing the app on a device. Learn about maskable manifest icons.

ADDITIONAL ITEMS TO MANUALLY CHECK (3) Hide

○ Site works cross-browser

To reach the most number of users, sites should work across every major browser. Learn about cross-browser compatibility.

○ Page transitions don't feel like they block on the network

Transitions should feel snappy as you tap around, even on a slow network. This experience is key to a user's perception of performance. Learn more about page transitions.

○ Each page has a URL

Ensure individual pages are deep linkable via URL and that URLs are unique for the purpose of shareability on social media. Learn more about providing deep links.

These checks are required by the baseline PWA Checklist but are not automatically checked by Lighthouse. They do not affect your score but it's important that you verify them manually.

- Captured at Oct 25, 2023, 6:10 PM GMT+2
- Initial page load
- Emulated Moto G Power with Lighthouse 11.0.0
- Slow 4G throttling
- Single page load
- Using Chromium 118.0.0.0 with devtools

Generated by **Lighthouse** 11.0.0 | File an issue

V. Vérification de la taille des fichiers des pages Web avant et après modifications

https://www.seoptimer.com/fr/web-page-size-check