MESR

Etude métadonnées Crawler Web

SCANESR/ETU\_001/1.0

Entité : COEXYA/SSL

Contrat/Projet : ScanESR

Identifiant :

Date de réf.(1) : 03/07/2023

État :  à valider  validé

Classification : 3 - Restreint

Diffusion :

Libre

Interne

Contrôlée : … (Préciser BU, Client…)

(1) Date d’approbation (cf. circuit de validation interne).

Coexya

9 avenue Charles de Gaulle

69370 Saint-Didier-au-Mont-d’Or

France

CIRCUIT DE VALIDATION

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Version | Rédaction | | | Vérification | | | Approbation | | | |
|  | Nom | Date | Visa | Nom | Date | Visa | Nom | Date | Visa |
| 1.0 | Kevin GROSJEAN | 29/06/2023 | KGN | Yilei PAN | 03/07/2023 | YPN | Carmen INDRECAN | 03/07/2023 | CIN |

Figure 1 – Circuit de validation

HISTORIQUE DES EVOLUTIONS

|  |  |
| --- | --- |
| Version | Objet de la version  (citer les fiches de revue de document prises en compte) |
| 1.0 | Initialisation du document |

Figure 2 – Historique des évolutions

LISTE DE DIFFUSION

|  |  |  |  |
| --- | --- | --- | --- |
| Destinataire | Fonction | Nombre d’exemplaires | Support |
| MESR |  |  | mail |

Figure 3 – Liste de diffusion

DOCUMENTS ASSOCIÉS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Origine | N° | Titre | Référence | Usage (\*) |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

(\*) : Indiquer le contexte de citation du document : à lire au préalable, documents de référence, documents complémentaires, ...

Figure 4 – Documents associés

Table des matières

[1 Introduction 5](#_Toc139287032)

[1.1 Objet du document 5](#_Toc139287033)

[1.2 Rappel du contexte 5](#_Toc139287034)

[1.3 Organisation du document 5](#_Toc139287035)

[2 RGAA (accessibilité) 6](#_Toc139287036)

[2.1 Exigences fonctionnelles 6](#_Toc139287037)

[2.2 Solution retenue 6](#_Toc139287038)

[2.2.1 Lighthouse (catégorie « accessibility ») 6](#_Toc139287039)

[2.3 Solutions étudiées mais non retenues 7](#_Toc139287040)

[3 Bonnes pratiques web 8](#_Toc139287041)

[3.1 Exigences fonctionnelles 8](#_Toc139287042)

[3.2 Solution retenue 8](#_Toc139287043)

[3.2.1 Lighthouse (catégorie « best-practices ») 8](#_Toc139287044)

[3.3 Solutions étudiées mais non retenues 9](#_Toc139287045)

[4 Détection des technologies et des trackers 10](#_Toc139287046)

[4.1 Exigences fonctionnelles 10](#_Toc139287047)

[4.2 Solution retenue 10](#_Toc139287048)

[4.2.1 Wappalyzer 10](#_Toc139287049)

[4.3 Solutions étudiées mais non retenues 11](#_Toc139287050)

[5 Responsive 13](#_Toc139287051)

[5.1 Exigences fonctionnelles 13](#_Toc139287052)

[5.2 Solution retenue 13](#_Toc139287053)

[5.2.1 Google Mobile Friendly Test 13](#_Toc139287054)

[5.3 Solutions étudiées mais non retenues 14](#_Toc139287055)

[6 Empreinte CO2 15](#_Toc139287056)

[6.1 Exigences fonctionnelles 15](#_Toc139287057)

[6.2 Solutions 15](#_Toc139287058)

[6.2.1 Website Carbon Calculator 15](#_Toc139287059)

[6.2.2 Carbon Calculator 16](#_Toc139287060)

[7 Récapitulatif des solutions retenues / conseillées 19](#_Toc139287061)

Table des figures

[Figure 1 – Circuit de validation 2](#_Toc139287062)

[Figure 2 – Historique des évolutions 2](#_Toc139287063)

[Figure 3 – Liste de diffusion 2](#_Toc139287064)

[Figure 4 – Documents associés 2](#_Toc139287065)

[Figure 5 – Exemple de retour de Lighthouse pour la catégorie « accessibility » 7](#_Toc139287066)

[Figure 6 – Exemple de retour de Lighthouse pour la catégorie « best-practices » 9](#_Toc139287067)

[Figure 7 – Exemple de retour de Wappalyzer 11](#_Toc139287068)

[Figure 8 – Exemple de retour de l’API Google Mobile Friendly Test 14](#_Toc139287069)

[Figure 9 – Exemple de retour de l'API Website Carbon Calculator 16](#_Toc139287070)

[Figure 10 – Exemple de retour de la librairie carbon-calculator 18](#_Toc139287071)

# Introduction

## Objet du document

Ce document constitue l’étude de sélection des outils permettant de recenser diverses métadonnées pour l’évolution « Crawler » demandé par le Ministère de l’Enseignement supérieur et de la Recherche (MESR).

## Rappel du contexte

Lancée en 2016, ScanR est la première déclinaison de ScanESR, plateforme de moteurs de recherche de la recherche et de l’innovation, de l’enseignement supérieur et de l’insertion professionnelle.

Le but de cette application est de permettre aux utilisateurs de mieux comprendre les orientations scientifiques, les partenariats, l’activité des structures publiques et des entreprises dans le domaine de la R&D et de l’innovation.

Pour cela, ScanR utilise deux principales sources d’informations :

* Les informations issues de bases de données structurées
* Les informations crawlées sur des sites internet de structures publiques et privées du périmètre de ScanR

Depuis mars 2018, Coexya a en charge la maintenance et l’évolution de l’application ScanR ainsi que de son backoffice d’administration associé aux workflows de publication des données.

Dans le cadre de ce projet, Coexya a notamment réalisé les travaux suivants :

* Mise en œuvre d’une nouvelle version (V2) de ScanR permettant de rechercher plus largement, dans les projets, les publications ainsi que les auteurs
* Évolution du modèle de données afin d’accueillir les brevets dans l’application
* Évolution technique (montée de version d’Elasticsearch) ainsi que mise en œuvre de tableaux de bord de suivi des données, et de l’usage des API (Kibana, Filebeat)

## Organisation du document

Le présent document précise, pour chaque métadonnée, les attendus en terme fonctionnel, la solution retenue et les autres solutions étudiées mais non retenues. En fin de document il y a un récapitulatif des solutions retenues par métadonnée.

# RGAA (accessibilité)

## Exigences fonctionnelles

Récupération du score et des critères d’accessibilité à partir d’une url.

## Solution retenue

### Lighthouse (catégorie « accessibility »)

Lighthouse est un outil permettant d’auditer des pages web et de retourner un rapport ainsi qu’un score sur plusieurs facteurs de qualité et de performance. Cette solution a été retenue car elle permet de couvrir plusieurs métadonnées (RGAA et Bonne pratiques Web), est gratuite et open-source. Elle s’intègrera bien dans le processus de récupération des métadonnées.

[Lien vers la documentation Lighthouse](https://github.com/GoogleChrome/lighthouse#using-the-node-cli)

#### Avantages

* Gratuit
* Une seule solution pour plusieurs métadonnées
* Solution toujours maintenue

#### Inconvénients

* Nécessite d’embarquer la solution

#### Récupération des données

lighthouse <url> --only-categories <categories (array)> --output <output-format> --output-path <output-path>

Exemple de commande permettant de récupérer le score ainsi que les critères d’accessibilité et de best-practice pour le site https://www.unistra.fr/ et de stocker le résultat dans un fichier json nommé « lighthouse.json » :

lighthouse https://www.unistra.fr/ --only-categories accessibility best-practices --output json --output-path ./lighthouse.json

#### Résultats

La réponse renvoyée par Lighthouse contient notamment les propriétés suivantes :

|  |  |
| --- | --- |
| Propriété | Description |
| audits | Liste des propriétés auditées |
| categories | Section concernant la catégorie examinée |
| categories.auditRefs | Liste d’objets comportant notamment les clefs permettant de faire le lien entre les audits et la catégorie pour laquelle ils sont vérifiées. |
| categories.score | Score (entre 0 et 1) calculé par Lighthouse pour la catégorie correspondante |

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 5 – Exemple de retour de Lighthouse pour la catégorie « accessibility »

#### Performances

Temps de réponse moyen pour une exécution de Lighthouse sur les catégories « accessibilty » et « best-practices » uniquement : Entre 5 et 15 secondes

## Solutions étudiées mais non retenues

|  |  |
| --- | --- |
| Solution | Remarques |
| Axe | * Ne teste que l’accessibilité, contrairement à Lighthouse qui permet d’auditer à la fois l’accessibilité et les bonnes pratiques * Pas de support RGAA (norme Française) donc non adapté au besoin |
| Wave | * Ne teste que l’accessibilité, contrairement à Lighthouse qui permet d’auditer à la fois l’accessibilité et les bonnes pratiques * Pas de cli (Command Line Interface), d’api ou de solution embarquable. Donc aucun moyen de l’intégrer dans le processus automatisé de récupération des métadonnées. |
| Accessibility Developer tools | * Ne teste que l’accessibilité, contrairement à Lighthouse qui permet d’auditer à la fois l’accessibilité et les bonnes pratiques |

1. Bonnes pratiques web

## Exigences fonctionnelles

Récupération du score, et des critères de tests de bonnes pratiques web à partir d’une URL.

## Solution retenue

### Lighthouse (catégorie « best-practices »)

Lighthouse est un outil permettant d’auditer des pages web et de retourner un rapport ainsi qu’un score sur plusieurs facteurs de qualité et de performance. Cette solution a été retenue car elle permet de couvrir plusieurs métadonnées (RGAA et Bonne pratiques Web), est gratuite et open-source. Elle s’intègre bien dans le processus de récupération des métadonnées.

[Lien vers la documentation Lighthouse](https://github.com/GoogleChrome/lighthouse#using-the-node-cli)

#### Avantages

* Gratuit
* Une seule solution pour deux métadonnées
* Seule solution identifiée intégrable pour l’audit des bonnes pratiques
* Solution toujours maintenue

#### Inconvénients

* Nécessite d’embarquer la solution

#### Récupération des données

lighthouse <url> --only-categories <categories (array)> --output <output-format> --output-path <output-path>

Exemple de commande permettant de récupérer le score ainsi que les critères d’accessibilité et de bonnes pratiques pour le site https://www.unistra.fr/ et de stocker le résultat dans un fichier json nommé « lighthouse.json » :

lighthouse https://www.unistra.fr/ --only-categories accessibility best-practices --output json --output-path ./lighthouse.json

#### Résultats

La réponse renvoyée par Lighthouse contient notamment les propriétés suivantes :

|  |  |
| --- | --- |
| Propriété | Description |
| audits | Liste des propriétés auditées |
| categories | Section concernant la catégorie examinée |
| categories.auditRefs | Liste d’objets comportant notamment les clefs permettant de faire le lien entre les audits et la catégorie pour laquelle ils sont vérifiées. |
| categories.score | Score (entre 0 et 1) calculé par Lighthouse pour la catégorie correspondante. |

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

Figure 6 – Exemple de retour de Lighthouse pour la catégorie « best-practices »

#### Performances

Temps de réponse moyen pour une exécution de Lighthouse sur les catégories « accessibilty » et « best-practices » uniquement : Entre 5 et 15 secondes

## Solutions étudiées mais non retenues

|  |  |
| --- | --- |
| Solution | Remarques |
| Akcces | * Pas de cli (Command Line Interface), d’api ou de solution embarquable. Donc aucun moyen de l’intégrer dans le processus automatisé de récupération des métadonnées. |

1. Détection des technologies et des trackers

## Exigences fonctionnelles

Récupération des technologies utilisées par un site web à partir d’une url

## Solution retenue

### Wappalyzer

Wappalyzer est une solution permettant de trouver quelles technologies sont utilisées par un site web. Cette solution est gratuite et s’intègre dans le processus de récupération des métadonnées. C’est la seule qui a été identifiée répondant aux besoins.

[Lien vers la documentation de Wappalyzer](https://github.com/wappalyzer/wappalyzer/blob/master/README.md#quick-start)

#### Avantages

* Gratuit
* Retour complet
* Performant
* Solution toujours maintenue

#### Inconvénients

* Nécessite d’embarquer la solution

#### Récupération des données

node src/drivers/npm/cli.js <url>

Exemple de commande permettant de récupérer la liste des technologie utilisées par le site <https://www.unistra.fr/>:

node src/drivers/npm/cli.js https://www.unistra.fr/

#### Résultats

La réponse renvoyée par Wappalizer contient la liste des technologies utilisées par le site web.

Une technologie contient les propriétés suivantes :

|  |  |  |
| --- | --- | --- |
| Propriété | Type | Description |
| slug | String | Identifiant de la technologie |
| name | String | Nom de la technologie |
| description | String | Description de la technologie |
| confidence | Number | Pourcentage de confiance de détection de la technologie |
| version | String | Version de la technologie |
| icon | String | Nom de l’icône et extension |
| website | String | Site de la technologie |
| cpe | String | Nommage technique de la technologie |
| categories | Object | Catégorie d’outil, composé :   * D’un identifiant (type : Number) * D’un nom (type : String) * Du nom court (type : String) |
| rootPath | Boolean | Page d’accueil du site (oui ou non) |

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure 7 – Exemple de retour de Wappalyzer

#### Performances

Temps de réponse moyen pour une exécution de Wappalyzer : Entre 2 et 4 secondes

## Solutions étudiées mais non retenues

|  |  |
| --- | --- |
| Solution | Remarques |
| Wappalyzer API | * Nécessite une clé API payante |
| BuiltWith Free API | * Pas assez d’informations renvoyées * Limité à 1 requête par seconde |
| WhatRuns | * Seulement disponible en tant qu’extension de navigateur donc pas d’intégration possible pour le processus de récupération des métadonnées |
| BuiltWith API | * Nécessite une clé API payante |

**Note :** Beaucoup de solutions existantes pour lister les technologies utilisées par un site web, mais la grande majorité est payante. Pour la plupart il s’agit d’extensions pour navigateur ou d’API nécessitant un accès payant.

1. Responsive

## Exigences fonctionnelles

Récupération du caractère « responsive » d’un site web à partir d’une url.

## Solution retenue

### Google Mobile Friendly Test

Cette solution est gratuite, dispose d’une API et s’intègre facilement au processus de récupération des métadonnées.

[Lien vers la documentation de Google Mobile Friendly Test](https://developers.google.com/webmaster-tools/search-console-api/reference/rest/v1/urlTestingTools.mobileFriendlyTest/run?hl=fr)

#### Avantages

* Gratuit
* Simplicité de mise en place
* Solution toujours maintenue

#### Inconvénients

* Lent (entre 1 et 2 minutes pour une réponse)

#### Récupération des données

urllib2.urlopen(url= 'https://searchconsole.googleapis.com/v1/urlTestingTools/mobileFriendlyTest:run', data=urllib.urlencode({‘url’:<url>, ‘key’: <API\_KEY>}).read()

#### Résultats

La réponse renvoyée par Google Mobile Friendly Test contient notamment les données suivantes :

|  |  |  |
| --- | --- | --- |
| Propriété | Type | Description |
| testStatus | Object | Objet contenant les propriétés suivantes :   * status : status du test (TEST\_STATUS\_UNSPECIFIED, COMPLETE, INTERNAL\_ERROR ou PAGE\_UNREACHABLE) * details : détails de l’erreur si applicable |
| MobileFriendliness | String | Résultat du test (MOBILE\_FRIENDLY\_TEST\_RESULT\_UNSPECIFIED, MOBILE\_FRIENDLY ou NOT\_MOBILE\_FRIENDLY) |

Une image contenant texte, Police, blanc, capture d’écran

Description générée automatiquement

Figure 8 – Exemple de retour de l’API Google Mobile Friendly Test

#### Performances

Temps de réponse moyen pour une exécution : Entre 1 et 2 minutes

## Solutions étudiées mais non retenues

|  |  |
| --- | --- |
| Solution | Remarques |
| Responsive Design Checker | * Outil graphique non utilisable dans le processus de récupération des métadonnées |
| MattKersley.com | * Réponse graphique ne permettant pas d’interpréter les résultats lors du processus de récupération des métadonnées |

**Note :** La très grande majorité des outils permettant de vérifier le caractère responsive d’un site web qui ont été identifiés sont des outils graphiques n’ayant pas de support pour une intégration dans le processus de récupération des métadonnées.

1. Empreinte CO2

## Exigences fonctionnelles

Récupération de l’empreinte carbone d’un site web à partir d’une url.

## Solutions

### Website Carbon Calculator

C’est la solution la plus simple à mettre en place car elle met à disposition une API. Elle s’intègre facilement au processus de récupération des métadonnées. Selon le site, il est possible d’obtenir un accès gratuit à cette API. Cependant l’appel de récupération de données fonctionne sans clé API.

Nous conseillons cette solution car elle ne dépend pas d'autres outils. Si les méthodes de calcul de l'empreinte carbone évoluent, ça sera pris en compte également car l'API est toujours maintenue. Néanmoins le calcul n’est pas personnalisable.

[Lien vers la documentation de Website Carbon Calculator](https://www.websitecarbon.com/how-does-it-work/)

#### Avantages

* Facilité de mise en place et d’utilisation (API)
* Gratuit
* Indépendant d’autres outils
* Solution toujours maintenue
* À jour avec les méthodes de calcul les plus récentes

#### Inconvénients

* Peu d’informations
* Le détail du calcul n’est pas retourné
* Pas de modification du calcul possible

#### Récupération des données

urllib2.urlopen(url=’ https://api.websitecarbon.com/site?url=https%3A%2F%2Fwww.unistra.fr%2F’)

#### Résultats

La réponse renvoyée par Website Carbon Calculator contient notamment les propriétés suivantes :

|  |  |  |
| --- | --- | --- |
| Propriété | Type | Description |
| green | Boolean|String | Information concernant la « propreté » du site web. Renvoie « unknown » (oui ou non) si cela n’a pas pu être établi. |
| bytes | Integer | Nombre d’octets transférés lors du chargement de la page. |
| cleanerThan | Number | Valeur numérique (de 0 à 1) représentant le pourcentage de site testés auxquels celui-testé est plus « propre » |
| statistics | Object | Un objet contenant les propriétés suivantes :  adjustedBytes (Number) : Nombre approximatif d’octets transférés lors du chargement de la page, ajusté pour prendre en compte les premiers chargements et les réouvertures (moyenne sur le long terme)  energy (Number) : Montant d’énergie approximatif nécessaire pour l’affichage de chaque page (en KWh)  co2(Object) : Objet contenant des informations sur la quantité de CO2 généré pour chaque chargement de page. 2 options (énergie renouvelable (renewable) ou grille nationale (grid)). Chacune de ces options contient un objet contenant 2 propriétés : La quantité approximative de CO2 générée par chaque chargement de page en grammes (grams) ou en litre (litres) |

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

Figure 9 – Exemple de retour de l'API Website Carbon Calculator

#### Performance

Temps d’exécution moyen : Entre 5 et 20 secondes

### Carbon Calculator

Il s’agit d’une librairie Python gratuite reprenant le mode de fonctionnement de Website Carbon Calculator. Elle nécessite la mise en place de Lighthouse (outil utilisé pour d’autres métadonnées) ainsi que d’une base de données récupérée gratuitement depuis The Green Web Foundation. Elle s’intègre parfaitement au processus de récupération des métadonnées.

[Lien vers la documentation de carbon-calculator](https://pypi.org/project/carbon-calculator/)

#### Avantages

* Utilisation simple
* Gratuit
* Rapidité
* Détail de la quantité d’octet retourné par technologie
* Basé sur Lighthouse, outil déjà sélectionné pour d’autres métadonnées

#### Inconvénients

* Dernière version datant de fin 2021
* Nécessite plus de temps pour la mettre en place
* Nécessite la mise à jour de la base de données des hébergeurs verts de The Green Web Foundation.
* Le détail du calcul n’est pas retourné
* Si les méthodes de calcul évoluent, celui-ci nécessitera une mise à jour.

#### Récupération des données

Exemple de script permettant de récupérer l’impact carbone du site https://www.unistra.fr/ :

from carbon.calculator import CarbonCalculator

from carbon.services import LighthouseService, GreenWebService

# If lighthouse tool is installed globally the following row can be omitted

lighthouse = LighthouseService(lighthouse = PATH\_OF\_LIGHTHOUSE\_TOOL)

# It loads the Green Web Dataset DB (must be a SQL3Lite file)

greenweb = GreenWebService(greenweb = PATH\_URL\_OF\_GREEN\_DB)

# It calculates CO2 emissions

carbon = CarbonCalculator(lighthouse=lighthouse, greenweb=greenweb)

carbon.footprint("https://www.unistra.fr/")

#### Résultats

La réponse renvoyée par la librairie carbon-calculator contient notamment les informations suivantes :

|  |  |  |
| --- | --- | --- |
| Propriété | Type | Description |
| hosting\_green | Boolean | Information concernant la propreté du site web (oui ou non) |
| co2\_grams | Number | Emission CO2 en grammes |
| energy\_kWh | Number | La quantité d’énergie consommée en KWh |
| resources | Array | Tableau contenant le détail de la quantité d’octets transféré au chargement de la page web. Contient notamment le total d’octet, le total d’octet en prenant en compte le cache ainsi que le détail pour chaque type de fichier (html, css, javascript, image, etc) |

Une image contenant texte, lettre, Police, menu

Description générée automatiquement

Figure 10 – Exemple de retour de la librairie carbon-calculator

#### Performances

Temps d’exécution moyen : Entre 5 et 15 secondes

# Récapitulatif des solutions retenues / conseillées

|  |  |  |  |
| --- | --- | --- | --- |
| Métadonnée | Solution retenue | Pourquoi ? | Alternatives |
| RGAA | Lighthouse | * Gratuit * Couvre plusieurs métadonnées * Solution toujours maintenue * Seule solution intégrable identifiée pour les bonnes pratiques |  |
| Bonnes pratiques |
| Technologies et trackers | Wappalyzer (CLI) | * Gratuit * Seule solution gratuite identifiée retournant suffisamment de données pour le traitement * Solution toujours maintenue | * Wappalyzer API (payant) * Builtwith Api (Payant) |
| Responsive | Google Mobile Friendly Test | * Gratuit * Seule solution intégrable dans le processus identifiée * Solution toujours maintenue |  |
| Empreinte Carbone | Website Carbon Calculator (API) | * Gratuit * Indépendant d’autres technologies * Simplicité et rapidité d’intégration * Solutionmaintenue donc à jour avec les méthodes de calculs les plus répandues | * Carbon-calculator API (retour plus détaillé, mais non maintenu donc potentiellement déprécié) |