ORGANIKK · SEO TECHNIQUE

5 agents SEO technique

01

Indexation Check

indexation-check.md

COPIER LE SKILL

Skill — Audit d'indexation Claude

Quand déclencher

- Audit ponctuel d'un site (avant call commercial, après une refonte, après un déploiement de cluster)
- Monitoring mensuel récurrent sur un client en accompagnement
- Diagnostic d'une chute de trafic non expliquée
- Pré-flight d'un projet pSEO (vérifier que la base technique est saine avant de scaler)

Input requis

SOURCE	OBLIGATOIRE
Liste des URLs à monitorer (fichier dans le repo : `urls.txt`, `src/data/wiki.ts`, `src/data/articles.ts`) OU URL du	Oui

SOURCE	OBLIGATOIRE
sitemap.xml public du site	
Domaine cible (ex : `site-client.fr`)	Oui
Périmètre du maillage à crawler (pages hubs : `/`, `/glossaire`, `/wiki`, footer, articles)	Recommandé
Service account GSC API (upgrade fiabilité)	Optionnel

Minimum viable : une liste d'URLs et le domaine. Le reste se déduit.

Architecture (3 couches)

```
1. Charger la liste des URLs (source de vérité humaine)
        ↓
2. COUCHE A — Audit technique (causes potentielles)
        ↓
3. COUCHE B — Statut d'indexation observable
        ↓
4. COUCHE C — Reporting markdown
```

La séparation causes / statut est volontaire. Le check v1 ne regardait que le statut, donc disait "OK 200" même quand la page portait un `noindex` accidentel. Le check v2 sépare strictement les deux.

Pipeline (9 checks)

COUCHE A — Audit technique (causes)

1. HTTP check (404/500, redirects) Pour chaque URL :

Note le code final ET la chaîne de redirects. 200 attendu. Toute redirection 301→200 systématique = dette à signaler (trailing slash, http→https).

2. robots.txt non bloquant

Pour chaque path, vérifie qu'aucune directive `Disallow:` ne le bloque pour `User-agent: *` ni `User-agent: Googlebot`.

3. Pas de balise noindex (HTML + en-tête HTTP)

Signaler toute présence de `noindex` ou `none`.

4. Sitemap : présence + fraîcheur

Pour chaque slug attendu, vérifie présence d'une `<loc>` et lis le `<lastmod>` associé. Flagger tout `lastmod` > 6 mois.

5. Cohérence sitemap ↔ source de vérité Diff entre les slugs de la SOT (fichier humain) et les `<loc>` du sitemap :

- Slugs en source mais absents du sitemap → à ajouter au build
- `<loc>` en sitemap mais absente de la source → orpheline sitemap

6. Maillage interne entrant Crawl les pages hubs (`/`, `/glossaire`, `/wiki`, footer, sample d'articles) et compte les liens internes entrants par URL cible.

- 0 lien entrant = orpheline (critique)
- 1 seul lien entrant = sous-maillée (à signaler)

7. Longueur de contenu (proxy thin content) Pour chaque URL, extrais le texte du `<main>` ou `<article>` (à défaut du `<body>` moins `<nav>` et `<footer>`) et compte les mots.

- < 300 mots = thin (à creuser)
- 300-800 = court mais acceptable pour fiche atomique
- > 800 = OK

COUCHE B — Statut d'indexation (sortie)

8. Indexation Google estimée (scraping `site:`)

Garde-fous obligatoires :

- Espacer les requêtes de 3 à 5 secondes (`sleep 4`)
- Détecter le blocage : `grep -E 'sorry/index|recaptcha|unusual traffic'`
- Si bloqué : marquer "non testable — rate limit", **pas** "non indexée"
- Retry une fois si réponse vide avant de conclure
- Marquer "indexée" si l'URL apparaît dans le HTML de réponse

Fiabilité ~40-60 % à cause du rate-limit. Pour 100 % de fiabilité, voir la variante GSC API.

COUCHE C — Reporting

9. Rapport markdown Structure obligatoire :

- Synthèse en tête (X/N par dimension, ✅ pour validé, ⚠️ pour anomalie)
- Anomalies CRITIQUES (action immédiate) — `noindex` accidentel, page orpheline, 404 sur page business
- Anomalies mineures — `lastmod` > 6 mois, sous-maillage, thin content
- Recommandations priorisées (2-5 actions, classées par impact estimé)
- Section "Limites du rapport" (ce qui n'a pas pu être testé et pourquoi)

Sortie obligatoire

```
# Indexation check — {domaine} — {date}

## Synthèse

Vérifications validées :
✅ {check} : X / N
✅ ...

Vérifications avec anomalies :
⚠️ {check} : X / N ({Y détails})
⚠️ ...

## Anomalies CRITIQUES (action immédiate)

⚠️ {URL} — {anomalie détectée}
   {contexte : depuis quand, impact estimé en impressions ou clics}
   → {action concrète à mener}

## Anomalies mineures

- {liste compactée des points secondaires}

## Recommandations priorisées

1. {action 1} — {effort estimé}, {impact estimé}
2. {action 2} — ...
3. {action 3} — ...
```

```
## Limites de ce rapport

- {ce qui n'a pas pu être testé et pourquoi : rate-limit Google, JS rendu
```

Garde-fous (NON-NÉGOCIABLES)

- **Lecture seule** sur le web public. Aucune requête authentifiée non explicitement autorisée.
- **Aucune action de forçage d'indexation.** Pas de soumission GSC, pas de force-crawl.
- **Pas de modification du site.** Pas de PR sur les fichiers source. Sortie limitée à `reports/` ou directement dans la conversation.
- **Distinction stricte** "non indexée" vs "non testable" dans le rapport.
- **Priorité au signalement** : un `noindex` accidentel sur une page business doit remonter en TOP du rapport, pas en ligne 47 du tableau.
- Si un check échoue, continuer les autres et signaler dans la section "Limites".

> *Discipline : observation côté agent, décision côté humain. Un agent qui peut "réparer" peut aussi tout casser un dimanche à 3 h du matin.*

Variante haute fiabilité — GSC URL Inspection API

Si l'utilisateur dispose d'un service account Google Cloud + propriété GSC :

1. Lire la clé JSON via secret stocké côté projet
2. Signer un JWT, échanger contre access token
3. POST sur `https://searchconsole.googleapis.com/v1/urlInspection/index:inspect`
4. Statuts officiels disponibles : `INDEXED`, `DISCOVERED_NOT_INDEXED`, `CRAWLED_NOT_INDEXED`, `URL_IS_UNKNOWN_TO_GOOGLE`

Passe le check #8 de ~50 % à 100 % de fiabilité. Setup ~30 minutes. Lève le rate-limit Google (limite officielle 2 000 inspections / jour / propriété).

Variante site JS rendu côté client (SPA)

Si le contenu principal est rendu côté client (Single Page Application), un simple fetch HTTP ne verra qu'une coquille HTML quasi vide. Brancher Playwright headless :

1. Lancer un browser headless
2. Charger l'URL, attendre `networkidle`
3. Comparer le HTML rendu vs le HTML servi par le serveur
4. Signaler tout contenu uniquement présent post-JS (= invisible pour les crawlers qui n'exécutent pas le JS)

À réserver aux sites SPA ou aux clusters critiques (~30 s par URL).

Récurrence — programmation via `/schedule`

Pour transformer l'audit ponctuel en monitoring continu :

```
/schedule
```

Choix structurants :

- **Récurrence** : `cron_expression: "0 1 1 * *"` (1er de chaque mois, 01h UTC)
- **Alternative one-shot** : `run_once_at` à J+14 après un déploiement de cluster
- **Modèle** : `claude-sonnet-4-6` (Opus inutile pour ce job)
- **Tools** : `Bash, Read, Write, Edit, Glob, Grep`
- **Environment** : Default Anthropic Cloud

À chaque exécution : 5 à 10 min de tourner-en-fond, rapport posté dans la conversation (ou push d'une PR si configuré côté repo).

Réutilisation client

Pour appliquer ce skill à un client en accompagnement :

1. Créer un fichier `urls-prio.txt` dans son repo (top 30 pages business) OU pointer le sitemap public
2. Adapter le périmètre du maillage (hubs spécifiques au site)
3. Schedule cron mensuel

4. Inclure le rapport mensuel dans la note de suivi

Coût marginal par client : ~15 min de setup, 0 € récurrent. À facturer dans la rétro mensuelle ou comme bonus différenciant en pitch.

Sources internes

- `wiki/briefs/check-indexation-claude.md` (brief de référence v2, 2026-05-11)
- `wiki/concepts/grounding-score.md`
- `wiki/concepts/passage-ranking.md`
- `raw/articles/brouillons/préparation-newsletter-indexation-2026-05-11.md` (newsletter dérivée)

02

Données Structurées

seo-donnees-structurees.md

COPIER LE SKILL

Skill — Données structurées JSON-LD automatiques

Quand déclencher

Ajouter, refondre ou auditer le balisage structuré d'un site. Objectif : éligibilité aux rich results Google ET consolidation d'une entité unique exploitable par les moteurs de réponse (AEO). Pas du schema décoratif, du schema qui sert le Grounding et la citation.

Principe non-négociable

Trois règles, dans cet ordre. Si l'une saute, le skill a échoué.

1. **Source unique.** Tout le JSON-LD vient d'un seul module (`lib/schema.ts`). Aucun bloc schema écrit en dur dans une page.
2. **Une entité, référencée par `@id`.** Le graphe site-wide (Organization, WebSite, Person) est émis une seule fois dans le layout racine. Chaque page de contenu pointe vers ces nœuds par `@id` (author / publisher / isPartOf), elle ne redéclare jamais l'auteur ni l'éditeur. Google fusionne tous les blocs d'une page et consolide une seule entité. C'est ça qui construit le knowledge graph du site, pas la répétition.
3. **Le schema est dérivé du contenu, jamais saisi à la main.** Un champ schema qui n'est pas calculé depuis la donnée de la page est une dette : il se désynchronise au premier edit. Si on doit le taper deux fois, c'est cassé.

Architecture de référence

Graphe site-wide (`siteGraph()`, émis dans le layout racine) :

- `Organization` (+ `ProfessionalService` si activité de service) avec `@id = {SITE_URL}/#organization`
- `WebSite` avec `@id = {SITE_URL}/#website`, `publisher` → ref Organization
- `Person` (fondateur / auteur) avec `@id = {SITE_URL}/#person-...`, `worksFor` -> ref Organization
- `ImageObject` logo avec `@id = {SITE_URL}/#logo`, partagé par `logo` et `image`
- `sameAs` : profils sociaux réels (LinkedIn, YouTube, X, Substack). Pas d'URL inventée.

Refs légères exportées et réutilisées partout : `AUTHOR_REF = {'@id': PERSON_ID}`, `PUBLISHER_REF = {'@id': ORG_ID}`, `WEBSITE_REF = {'@id': WEBSITE_ID}`.

Règle de dérivation automatique (le cœur du skill)

Chaque type de schema se déduit d'un signal présent dans la donnée de la page. Tableau de correspondance à appliquer :

SIGNAL DANS LE CONTENU	SCHEMA ÉMIS	DÉRIVATION
Toute page article / post	`Article` ou `BlogPosting`	titre, excerpt, date, section ; `wordCount` calculé par `countWords()` ; `keywords` depuis les highlights ; author/publisher/isPartOf = refs
Toute page (toujours)	`BreadcrumbList`	`breadcrumb()` à partir du chemin (Accueil > rubrique > titre)
Bloc `faq` présent dans les sections	`FAQPage`	`mainEntity` = map des items q/a en Question/Answer. Zéro saisie
Section `video` présente	`VideoObject`	ID YouTube extrait par regex de l'URL ; thumbnail + embedUrl déduits
H2 dont le titre matche un prédicat (ex. contient "prompt", "étapes", "tutoriel")	`HowTo`	collecte les H3 numérotés suivants + leur contenu (p / code / listes) en `HowToStep`
Page produit / offre avec prix	`Product` / `Offer`	depuis les champs prix/dispo de la donnée. Jamais de prix inventé

Principe : on ne demande jamais à l'utilisateur "quel schema veux-tu". On lit la structure de la page et on émet ce que la structure prouve. Un schema sans preuve dans le contenu = suppression.

Procédure

1. **Détecter le terrain.** Framework et version. Si Next.js : App Router ou Pages ? Lire la doc du projet (sur organikk-next : `node_modules/next/dist/docs/`, conventions imposées par AGENTS.md) avant d'écrire une ligne. Ne jamais présumer l'API metadata.
2. **Cartographier le modèle de données.** Où vivent les articles, pages, FAQ, vidéos (ex. `src/data/*.ts`). Identifier la forme des sections (le schema se dérive de là).

3. **Créer / refondre `lib/schema.ts`** : config d'identité en tête (SITE_URL, identité Organization/Person/sociaux), les nœuds, `siteGraph()`, les refs `@id`, les helpers `breadcrumb()` et `countWords()`. Partir de `references/schema-reference.ts`.
4. **Câbler le layout racine** : un seul `<script type="application/ld+json">` avec `siteGraph()`.
5. **Câbler chaque template de contenu** : construire un tableau `jsonLd[]` (Article + breadcrumb toujours, puis FAQ / Video / HowTo conditionnels selon les signaux), sérialisé dans UN script en tête de page.
6. **Dédupliquer.** Supprimer tout ancien bloc Organization/Person redéclaré dans les pages : tout passe par les refs `@id`.
7. **Valider** (voir checklist). Corriger jusqu'à zéro erreur.

Émission

- Site-wide : un `<script type="application/ld+json">` dans le layout racine, contenu = `siteGraph()`.
- Par page : un seul `<script type="application/ld+json">` en tête, contenu = `JSON.stringify(jsonLd)` où `jsonLd` est le tableau des nœuds de la page.
- App Router : injection via `dangerouslySetInnerHTML` dans un Server Component. Ne pas utiliser de lib tierce, ce sont des objets JS purs.

Validation (checklist, zéro erreur tolérée)

- [ ] Chaque `@id` site-wide est unique et stable, réutilisé par ref dans les pages
- [ ] Aucune page ne redéclare un nœud Organization ou Person en entier
- [ ] `Article` : headline non vide, `datePublished` valide, `wordCount` calculé (pas codé en dur), author/publisher en ref
- [ ] `FAQPage` émis seulement si un vrai bloc FAQ existe, et chaque réponse est du texte réel
- [ ] `HowTo` émis seulement si des étapes ordonnées existent réellement
- [ ] `VideoObject` : `contentUrl` / `embedUrl` résolus, thumbnail valide
- [ ] `BreadcrumbList` cohérent avec l'URL réelle
- [ ] JSON-LD valide et passe le test Rich Results de Google + le validateur schema.org
- [ ] Aucun champ inventé (prix, note, avis, premier sur X) sans preuve : un faux signal dégrade la note Google de la page

Anti-patterns (à ne jamais faire)

- Redéclarer l'auteur et l'éditeur sur chaque page au lieu de référencer par `@id`
- Écrire du JSON-LD en dur dans le JSX d'une page
- Émettre un `FAQPage` ou `HowTo` "au cas où" sans contenu correspondant visible sur la page (risque de pénalité, et c'est du mensonge à Google)
- Mettre `aggregateRating` / `Review` / prix non vérifiables
- Multiplier les `<script ld+json>` redondants au lieu d'un tableau unique par page
- Hardcoder `wordCount`, des dates ou des compteurs au lieu de les dériver de la donnée
- Présumer l'API Next.js sans lire la doc de la version du projet

Référence d'implémentation

`references/schema-reference.ts` : module généralisé prêt à adapter (config d'identité en tête, nœuds, `siteGraph`, refs `@id`, `breadcrumb`, `countWords`, builders `articleSchema` / `faqSchema` / `videoSchema` / `howToFromSections`). C'est la version dé-Organikk-isée du système en production sur `organikk-next/src/lib/schema.ts`. L'adapter au modèle de données du site cible, ne pas le copier tel quel.

Concepts liés

`aeo` · `grounding-score` · `entites-vectorielles` · `passage-ranking` · `knowledge-graph` · `e-e-a-t`

03

Cannibalisation SEO

seo-cannibalisation.md

COPIER LE SKILL

Skill — Cannibalisation SEO

Quand déclencher

Deux pages se concurrencent sur les mêmes mots-clés ou intentions. Chutes de positions inexpliquées, CTR qui stagne malgré le volume.

Input requis

SOURCE	OBLIGATOIRE
Export GSC Requêtes — filtré par URL, 90j	Oui
Liste des URLs (scraping ou sitemap)	Recommandé
Contexte stratégique (pilier vs satellite)	Recommandé

Pipeline (5 étapes)

1. **Identifier les conflits** — requêtes qui déclenchent 2+ URLs dans la GSC
2. **Classifier le type** :

- **(A) Mot-clé exact** — deux pages sur la même requête précise
- **(B) Même intention** — deux pages répondent à la même intention
- **(C) Proximité sémantique** — sujets proches sans conflit direct
- **(Triade SERP)** — opportunité, pas conflit

1. **Analyser les métriques** — position, impressions, clics, CTR par page
2. **Évaluer l'architecture** — pilier vs satellite, objectif business
3. **Recommander l'action** :

SITUATION	ACTION
Type A + perdante faible	Redirection 301

SITUATION	ACTION
Type A + deux fortes	Fusion + 301
Type B + micro-intentions distinctes	Différenciation + maillage croisé
Type C	Renforcement maillage vers pilier
Triade SERP	Aucune action — optimiser chaque angle

Output obligatoire

```
CANNIBALISATION DÉTECTÉE
Requête : '[requête]' — Type : (A/B/C/Triade)
| URL | Position | Impressions | Clics | CTR | Statut |

→ Diagnostic : [explication]
→ Action : [action précise + 3 étapes d'implémentation]
```

Règles absolues

- Ne pas recommander une 301 sans analyser les métriques des deux pages
- Ne pas traiter toutes les cannibalisations de la même façon
- Ne pas confondre duplication de contenu et cannibalisation
- Ne pas fusionner des pages avec micro-intentions distinctes
- Identifier les Triades SERP comme opportunités, pas comme problèmes

Sauvegarde

Output dans `wiki/cannibalisation/YYYY-MM-DD-slug.md` selon hook §7 AGENTS.md.

Concepts liés

`triade-serp` · `rrf` · `maillage-interne` · `intention-recherche` · `gsc-export`

04

Maillage Interne

maillage-systeme + maillage-interne-gsc.md

COPIER LE SKILL

maillage-systeme.md

COPIER

Skill — Maillage Interne : Architecture, Ancres et Audit

Rôle

Construire et auditer le graphe de liens internes d'un site à partir du contenu existant ou planifié, sans dépendre de la Search Console. Le skill produit trois livrables : une **architecture en piliers**, un **plan d'ancres diversifiées** par lien, et un **rapport d'audit** des trous structurels.

L'objectif n'est pas de générer des liens à la chaîne. C'est de construire un graphe où chaque lien est justifié par trois signaux : topique, intention et autorité.

Réflexion appliquée (méthode Boussardon)

- Le maillage interne est **un système, pas une passe**. Trois axes simultanés : topique, intention, autorité.
- Une ancre, ce n'est pas un mot-clé. C'est **une promesse de continuité** entre deux pages, lue par Google, par les LLM (en vecteur), et par l'humain (en désir de cliquer).
- **5 ancres possibles vers la même page = 5 ancres différentes.** Un seul exact match. Le reste est partial / sémantique / contextuel long.
- **Test à voix haute** : si la phrase tombe juste sans le lien, l'ancre est bien intégrée. Si elle clopine, l'ancre est plaquée.

- **Le maillage Know→Do passe avant le maillage Know→Know.** Une page qui explique un concept doit toujours pointer vers la page qui permet de l'exécuter (outil, audit, démo).
- **Pas de "Voir aussi" en bas d'article.** Le contexte de lien est dilué. Liens contextuels in-body uniquement.
- **Une page mère n'est pas un titre de catégorie.** C'est l'article le plus stratégique du pilier, celui qui définit le vocabulaire et reçoit le plus de liens internes.
- **Le cross-pillar pollination compte autant que le maillage intra-cluster.** 1 lien sortant sur N doit pointer vers un autre pilier pour éviter la siloïsation.

Données requises

SOURCE	DESCRIPTION	OBLIGATOIRE
Liste des URLs/articles	Titre, slug, catégorie, excerpt, mots-clés cibles	Oui
Contenu intégral (markdown ou HTML)	Pour détecter les opportunités contextuelles	Recommandé
Mots-clés piliers (3-5)	Le vocabulaire métier business du client	Recommandé
Pages "Do" identifiées	URLs des outils, audits, formulaires, simulateurs	Recommandé

Minimum viable : la liste des articles avec titre + excerpt + mots-clés. Sans le contenu intégral, le skill produit l'architecture mais pas les ancres précises.

Raisonnement de l'agent (étapes obligatoires)

L'agent DOIT suivre ces étapes **dans l'ordre** avant de répondre.

Étape 1 — Classifier chaque page en intention

Pour chaque article, déterminer son intention dominante :

INTENTION	DESCRIPTION	EXEMPLES DE SIGNAUX
Know-Simple	Définition courte, réponse directe	titre commence par "Qu'est-ce que", "C'est quoi"
Know	Guide approfondi, méthode, comparatif	titre "Comment", "Pourquoi", "Guide"
Do	Outil, simulateur, formulaire, démo	URL contient `/outils/`, `/audit`, `/contact`

Une page peut avoir une intention dominante + une intention secondaire. Noter les deux.

Étape 2 — Identifier les piliers

Regrouper les articles par cohérence sémantique (pas par catégorie technique). Cibler **3 à 5 piliers max**. Pas plus. Pas moins de 3.

Critères pour qu'un cluster forme un vrai pilier :

- Au moins 3 articles dans le cluster
- Un mot-clé business central qui revient dans tous les titres ou excerpts
- Une page-hub naturelle : l'article le plus complet ou le plus stratégique du cluster

Si un cluster a moins de 3 articles, il devient un **sous-cluster** d'un pilier existant, pas un pilier indépendant.

Étape 3 — Désigner le hub de chaque pilier

Pour chaque pilier, identifier la page-hub :

- Article le plus complet (ou prévu pour l'être)
- Recouvre les concepts secondaires des autres articles du pilier
- Idéalement : positionné sur le mot-clé pilier exact

Le hub reçoit des liens entrants depuis tous les satellites. Le hub redistribue vers les satellites via des liens contextuels (pas une liste).

Étape 4 — Cartographier les liens existants

Pour chaque article, lister :

- **Inbound links** : combien d'articles pointent vers lui ?
- **Outbound links** : vers combien d'articles pointe-t-il ?
- **Click depth** : combien de clics depuis la home ?

Détecter les anomalies :

- **Orphan pages** : 0 inbound link
- **Dead-end pages** : 0 outbound link
- **Hub sous-maillé** : moins de 5 inbound depuis ses satellites

Étape 5 — Sélectionner les ancres pour chaque lien proposé

Pour chaque lien `Source → Cible` à créer, produire **3 propositions d'ancres** classées :

1. **Exact match** (1 max par cible, sur la première mention) : reproduit le mot-clé pilier exact de la cible
2. **Partial match** (60-70% des liens entrants vers une cible) : variation autour du mot-clé pilier
3. **Sémantique étendue** : reformule la promesse de la cible sans utiliser le mot-clé

Pour chaque ancre, vérifier les 5 critères :

CRITÈRE	QUESTION À SE POSER
Promesse de la cible	L'ancre reflète-t-elle ce que l'utilisateur va trouver, pas le titre H1 ?
Phrase porteuse	La phrase reste-t-elle fluide à voix haute sans le lien ?
Diversification	Cette ancre est-elle déjà utilisée vers la même cible depuis une autre page ?
Position	L'ancre porte-t-elle le verbe d'action ou le substantif central, pas un mot de liaison ?

CRITÈRE	QUESTION À SE POSER
Link context	Les 5 mots avant/après parlent-ils du sujet de la cible ?

Si une ancre rate un critère, la rejeter.

Étape 6 — Prioriser les liens à créer

Score d'urgence par lien proposé : **Score = (impressions cible × poids_intention) + (gain_authority × 0.4)**

Où :

- `poids_intention` : Do = 1.0, Know-décisionnel = 0.8, Know = 0.5, Know-Simple = 0.3
- `gain_authority` : 1 si la source est un hub, 0.5 si la source est un satellite mailé, 0.2 sinon

Prioriser dans cet ordre :

1. **Liens manquants Hub → Satellite** dans un pilier (pour activer le cocon)
2. **Liens Know → Do** (pour orienter le funnel)
3. **Liens cross-pillar** (1 par pilier minimum vers un autre pilier)
4. **Liens vers pages orphelines** identifiées en étape 4

Étape 7 — Vérifier les règles de conservation

Avant de finaliser le plan, valider :

- Aucune page orpheline restante (chaque page reçoit ≥ 1 inbound)
- Aucune page dead-end (chaque page contient ≥ 2 outbound)
- Chaque hub reçoit ≥ 5 inbound depuis ses satellites
- Aucune cible ne reçoit la même ancre 2 fois
- Densité raisonnable : 2 à 5 liens internes par 1000 mots, jamais plus

NE PAS répondre avant d'avoir complété chaque étape.

Format de sortie OBLIGATOIRE

Bloc 1 — Architecture détectée

```
PILIER 1 — [Nom thématique] (mot-clé pilier : "...")
├── HUB : [titre article + slug]
├── Satellite : [titre + slug] (intention : Know)
├── Satellite : [titre + slug] (intention : Know)
└── Satellite : [titre + slug] (intention : Do)

PILIER 2 — [Nom]
...
```

Bloc 2 — Audit du graphe existant

Tableau :

ARTICLE	INBOUND	OUTBOUND	CLICK DEPTH	STATUT
...	3	4	2	OK
...	0	2	3	**ORPHELINE**
...	5	0	1	**DEAD-END**

Bloc 3 — Plan de liens à créer (priorisé)

Pour chaque lien proposé :

```
PRIORITÉ : HAUTE | Score : 8.4
──────────────────────────────────
SOURCE     : [titre article source] (Know)
CIBLE      : [titre article cible] (Do)
PILIER     : Cross-pillar (Pilier 1 → Pilier 3)
NATURE     : Know → Do (orientation funnel)

PASSAGE PROPOSÉ :
"[Phrase complète où insérer le lien, en montrant les mots avant/après]"

ANCRES PROPOSÉES (choisir 1) :
  [exact]    "mot-clé pilier exact"
  [partial]  "variation naturelle du mot-clé"
```

```
   [sémant.]  "reformulation de la promesse cible"

JUSTIFICATION : [1 phrase sur pourquoi ce lien crée de la valeur]
```

Bloc 4 — Règles de gouvernance

Une checklist finale que le client suit à chaque nouvelle publication :

- [ ] Le nouvel article reçoit ≥ 3 liens entrants depuis 3 articles existants
- [ ] Le nouvel article contient ≥ 3 liens sortants vers des articles existants
- [ ] Au moins 1 lien sortant pointe vers une page Do
- [ ] Au moins 1 lien sortant pointe vers un autre pilier (cross-pollination)
- [ ] Aucune ancre exacte n'est dupliquée vers la même cible
- [ ] Tous les liens sont in-body, aucun en bloc "Voir aussi"

Points de vigilance

- **Ne pas tout automatiser.** Le skill propose, l'humain décide. Une ancre forcée détruit le naturel d'un texte.
- **Le contexte vaut plus que l'ancre.** Une ancre parfaite dans une phrase qui parle d'autre chose = lien faible. Réécris la phrase.
- **Le hub n'est pas figé.** Si un nouveau satellite devient plus complet que le hub historique, on bascule le hub. La structure suit le contenu, pas l'inverse.
- **Cross-pillar ≠ liens hors-sujet.** Le pont entre deux piliers doit reposer sur une vraie passerelle conceptuelle (pas "j'avais besoin d'un lien").
- **Ne jamais lier vers la home depuis le contenu.** La home a déjà tout le PageRank, elle n'en a pas besoin. Garde le jus pour les pages business.
- **Les FAQ sont une mine d'ancres.** Chaque réponse de FAQ qui mentionne un sous-sujet déjà traité doit lier. Densité haute, contexte naturel.
- **Densité plafonnée.** Au-delà de 5 liens internes pour 1000 mots, la dilution s'installe. Google pondère chaque ancre par 1/N où N est le nombre total de liens.

Cas particulier — Site sans donnée GSC

Si le site est nouveau (moins de 3 mois ou pas d'accès GSC) :

- Étape 4 (cartographie inbound/outbound) se fait par parsing du contenu (markdown/HTML)
- Étape 6 (scoring) utilise un proxy : `position_business` (1 si Do, 0.7 si Know-décisionnel, 0.5 si Know, 0.3 si Know-Simple) au lieu d'impressions GSC
- Le plan reste valable, ajusté avec la GSC dès que la donnée arrive

---

Cas particulier — Refonte de site existant

Si le site existe et a ≥ 6 mois de GSC :

- Chaîner ce skill avec **maillage-interne-gsc** : ce skill définit l'architecture, l'autre injecte la donnée comportementale
- Identifier les pages qui rankent déjà sur le pilier et les promouvoir hub si pertinent
- Préserver les liens existants qui marchent (pas de refonte aveugle)

Rappels méthode

> *"Le maillage interne, c'est un système, pas une passe." "5 liens entrants vers la même page = 5 ancres différentes. Un seul exact match." "Test à voix haute : si la phrase tombe juste sans le lien, l'ancre est bonne." "Une page Know doit toujours pointer vers une page Do." "Pas de Voir aussi. Liens in-body uniquement." "Un hub n'est pas une catégorie. C'est l'article le plus stratégique du pilier." "Le contexte des 5 mots avant/après l'ancre vaut plus que l'ancre elle-même."*

`maillage-interne-gsc.md` COPIER

Skill — Maillage Interne GSC

Quand déclencher

Analyse et optimisation du maillage interne depuis les données GSC. Hiérarchie page mère/fille/petite-fille.

Philosophie

> *"Le maillage interne, c'est la puissance. Et ça part de tes mots-clés."*

- Page mère = au moins 5 citations depuis des pages filles/petites-filles
- Le maillage part de la stratégie de mots-clés → le cocon est la conséquence
- Priorité : transactionnel > décisionnel > informationnel
- Maillage par intention (Know → Do) en plus du maillage sémantique

Input requis

SOURCE	OBLIGATOIRE
Export GSC Pages — URL, Clics, Impressions, CTR, Position	Oui
Export GSC Requêtes par page	Recommandé
Période 3-6 mois	Recommandé

Pipeline (5 étapes)

1. **Récupérer les données GSC** — export Pages + Requêtes par URL
2. **Diagnostiquer la structure** :

- Pages mères potentielles (impressions élevées, pos 4-15, requête transactionnelle)
- Pages sous-maillées (bonne position mais CTR faible)
- Pages orphelines (aucune thématique secondaire dans GSC)

1. **Construire le plan de maillage** — hiérarchie mère/fille/petite-fille + règles Know→Do

2. **Prioriser** — score urgence = (Impressions × 0.4) + (Potentiel position × 0.4) + (Business value × 0.2)
3. **Générer les recommandations** — page source + page destination + ancre + contexte + priorité

Structure hiérarchique

```
Page Mère (mot-clé principal business)
├── Page Fille 1 (requête secondaire transactionnelle)
│   ├── Page Petite-Fille A (longue traîne / micro-intention)
│   └── Page Petite-Fille B
└── Page Fille 2
```

Règle Know → Do : chaque page Know doit pointer vers au moins 1 page Do thématiquement reliée.

Output obligatoire

Pour chaque action :

- Page source (intention Know/Do/Know+Do)
- Page destination + intention
- Nature du lien (sémantique ou intentionnel Know→Do)
- Ancre recommandée (jamais "cliquez ici")
- Contexte d'insertion
- Priorité (Haute/Moyenne/Faible)

Règles absolues

- Ne pas automatiser à 100% — le maillage part de la stratégie, pas des outils
- Ne pas mailler sans tenir compte de l'intention (sémantique ≠ intentionnel)
- Ne pas répéter la même ancre sur toutes les pages filles
- 10 citations minimum pour une page mère "active"

Sauvegarde

Output dans `wiki/maillage/YYYY-MM-DD-slug.md` selon hook §7 AGENTS.md.

Concepts liés

`cocon-semantique` · `pagerank-interne` · `intention-recherche` · `cannibalisation` · `gsc-export`

05

Core Web Vitals

seo-core-web-vitals.md

COPIER LE SKILL

Skill — Audit Core Web Vitals (Lighthouse local)

Quand déclencher

Diagnostic de performance SEO d'un site complet, ou avant intervention technique (refonte, optimisation perf, migration). Toujours mobile (Google = mobile-first indexing).

Input requis

SOURCE	OBLIGATOIRE	DÉFAUT
URL du sitemap.xml (ex. `https://site.com/sitemap.xml`)	Oui	—
Échantillon	Non	50 URLs

SOURCE	OBLIGATOIRE	DÉFAUT
Mode `all` (audit toutes les URLs du sitemap)	Non	off

Pipeline (5 étapes)

Étape 1 — Récupérer le sitemap

Si le sitemap est un index (contient `<sitemapindex>` au lieu de `<urlset>`), récupérer le premier sitemap enfant et l'utiliser :

Étape 2 — Échantillonner

Les 50 premières URLs du sitemap sont généralement les plus prioritaires (homepage, hubs, top articles). Si l'utilisateur demande `all`, auditer toutes les URLs.

Étape 3 — Lancer Lighthouse mobile en parallèle (3 workers)

Compter : ~15-30 secondes par URL avec 3 workers parallèles. 50 URLs ≈ 5-10 min.

Étape 4 — Parser les résultats

Chemins clés Lighthouse 13 :

- URL finale : `.finalDisplayedUrl` (et **PAS** `.finalUrl` qui peut être null)
- LCP element : `.audits."lcp-breakdown-insight".details.items[] | select(.type == "node")` (et **PAS** `.audits."largest-contentful-paint-element"` qui est null depuis LH 13)
- LCP breakdown (TTFB / render delay / load delay) : `.audits."lcp-breakdown-insight".details.items[] | select(.type == "table") | .items[]` — utile pour diagnostiquer **où** se passe le temps LCP

Seuils Google (à appliquer pour le verdict) :

MÉTRIQUE	GOOD	NEEDS IMPROVEMENT	POOR
LCP	< 2500 ms	2500-4000 ms	> 4000 ms

MÉTRIQUE	GOOD	NEEDS IMPROVEMENT	POOR
CLS	< 0.1	0.1-0.25	> 0.25
TBT (proxy INP)	< 200 ms	200-600 ms	> 600 ms
Score perf	≥ 90	50-89	< 50

Étape 5 — Produire le rapport markdown

Voir section "Output obligatoire" ci-dessous.

Étape 4bis — Détection automatique du pattern "redirect sitemap"

Si l'opportunity `redirects` apparaît sur **>50% des pages auditées** avec un gain moyen >300ms : c'est quasi-certainement un mismatch de trailing slash entre le sitemap et le serveur.

Ce pattern doit être mis EN TÊTE du rapport — c'est le quick win le plus rentable quand il est présent.

Output obligatoire

Règles absolues

- **Mobile uniquement** — Google indexe en mobile-first. Pas de desktop par défaut.
- **INP non mesuré** — toujours préciser dans le rapport que TBT est utilisé comme proxy.
- **Pas de score halluciné** — si une URL échoue (timeout, 404, JS error), la marquer `ERROR` dans le tableau, pas `0` ou un placeholder.
- **Pas de reco sans opportunity Lighthouse correspondante** — ne pas inventer "il faudrait optimiser X" si Lighthouse ne l'a pas remonté pour cette page.
- **Pas d'installation silencieuse** — si Lighthouse manque, demander à l'utilisateur de l'installer.
- **Échantillon honnête** — si l'échantillon est de 50/2000 URLs, le dire clairement, ne pas extrapoler "tout le site" à partir de 2,5% du contenu.

Edge cases

- **Sitemap index** (sitemapindex au lieu d'urlset) → récupérer le premier child + l'utiliser, signaler à l'utilisateur qu'il peut spécifier un sitemap enfant précis pour cibler.
- **Sitemap protégé** (auth, 403) → demander à l'utilisateur un export local des URLs.
- **Plus de 500 URLs avec mode** `all` → confirmer avec l'utilisateur (sera lent : 500 URLs × ~10s/3 workers ≈ 30 min).
- **Site en SPA ou auth-walled** → certaines pages peuvent timeout, c'est normal, les marquer ERROR.
- **Sitemap multi-langues** → garder tel quel, le rapport montrera les écarts par locale.

ET APRÈS

Pour aller plus loin

MISSION CONSULTING

Une mission SEO technique avec Organikk

Je passe les 5 agents sur ton site, je te livre un rapport priorisé en moins de 7 jours, et on attaque les correctifs ensemble. Pas de retainer flou, juste un livrable concret.

organikk.co →

RECHERCHE DE MOTS-CLÉS

Trouve tes mots-clés pour ChatGPT, YouTube et Google

Le SEO ne se joue plus seulement sur Google. Fusionn te sort les requêtes qui comptent sur les trois moteurs où ton audience cherche vraiment : ChatGPT, YouTube et Google. Un seul outil, trois sources de trafic.

fusionn.co →

Tim Boussardon · Consultant SEO IA · Organikk