Réseau P2P décentralisé pour la communauté francophone

Spécification de la preuve de concept

k0walsky - v6

29 décembre 2025

Table des matières

1 Titre et périmètre **2**

2 Non-objectifs **3**

3 Cas d'usage licites **3**

4 Vue d'ensemble de l'architecture **3**
4.1 Vue A - Base de donnée décentralisée 4
4.1.1 Légende des couleurs (database.png) 4
4.2 Vue B - Architecture intégrée 5
4.2.1 Légende des couleurs (architecture.png) 5
4.3 Idée centrale 6
4.4 Concept explicite : curation et indexation fiables 6
4.4.1 Limites du crawl DHT seul 6
4.4.2 Modèle de découverte intentionnelle 6
4.4.3 Résultats attendus 6
4.5 Principes de gouvernance 6
4.6 Modèle de confiance et gouvernance 7
4.6.1 Rôles, responsabilités et invariants 7
4.6.2 Identités, clés et révocation 7
4.6.3 Politique de confiance d'instance 7
4.6.4 Évènement de politique de sécurité 7
4.6.5 Agrégation multi-curateurs 7
4.6.6 Auditabilité 8
4.6.7 Conformité 8
4.7 Ce que ce POC développe vs. ce qu'il utilise 8
4.8 Moyens de déploiement 8

5 Rôles et responsabilités **8**
5.1 Initial Peer 8
5.2 Relais publics 8
5.3 Relais communautaires 9
5.4 Explorer 9
5.5 Curator 9
5.6 Indexer 9

5.7 External Database . . . 9
5.8 Prowlarr / Servarr . . . 9
5.9 Clients P2P . . . 9
5.10 Humains . . . 9
5.11 Rulesets (sémantique et censure) . . . 10

6 Flux & concepts **10**
6.1 Flux A — Nostr + modération + indexation . . . 10
6.2 Flux B — Intégration P2P + *arr* + base distribuée . . . 10
6.3 Modèle de données . . . 10
6.3.1 Publication initiale . . . 10
6.3.2 Décision de vérification . . . 11
6.3.3 Descripteur de règles de modération . . . 11
6.3.4 Publication modérée . . . 11
6.3.5 Statistiques de popularité . . . 11
6.4 Déduplication . . . 11

7 Intégration Nostr **12**
7.0.1 Tests de conformité . . . 12

8 Modération **12**
8.0.1 Taxonomie des codes de rejet . . . 12
8.0.2 Versionnage des règles de modération . . . 13
8.0.3 Conflits de versions . . . 13
8.1 Signalement, déréférencement, et limites du retrait . . . 13
8.2 Auth, permissions et rate limit de l'API Torznab . . . 13
8.3 Consommation externe . . . 13

9 Observabilité & déploiement **13**

10 Exigences non techniques **14**

11 Sécurité et conformité **14**

12 Ce que n'est pas ce projet **14**

13 Questions ouvertes / travaux futurs **14**

14 Risques connus et modes de défaillance **14**

15 Glossaire **14**

1 Titre et périmètre

Ce document formalise un concept de réseau pair-à-pair pour la découverte et la distribution de contenus **strictement licites** via Nostr, à l'échelle de la communauté française. Il s'agit d'une spécification conceptuelle centrée sur la gouvernance, les rôles et les flux d'usage, et bien que certaines solutions techniques soient explicitées, il ne s'agit pas d'une implémentation. Le périmètre cible la communauté francophone, sans contrôle centralisé et peut être facilement transposé pour compléter les besoins d'autres communautés réunies autour du partage de fichiers entre pairs.

2 Non-objectifs

Cette preuve de concept ne définit pas : - l'hébergement ou la distribution centralisée de contenu - un système permettant le contournement de DRM ou la facilitation du piratage - un modèle complet d'identité et de réputation globale - un anonymat absolu (Tor/VPN peuvent être utilisés sans promesse d'anonymat total) - le remplacement des clients P2P ou de la DHT

3 Cas d'usage licites

— Distribution de logiciels libres et open source (distributions, outils, SDKs).
— Partage d'archives publiques et ouvertes (docs techniques, standards, datasets libres).
— Diffusion de contenus créatifs sous licences ouvertes (musique, vidéo, livres).
— Publication de releases officielles de logiciels open source et miroirs communautaires.

4 Vue d'ensemble de l'architecture

Deux vues complémentaires décrivent l'architecture cible :

4.1 Vue A - Base de donnée décentralisée

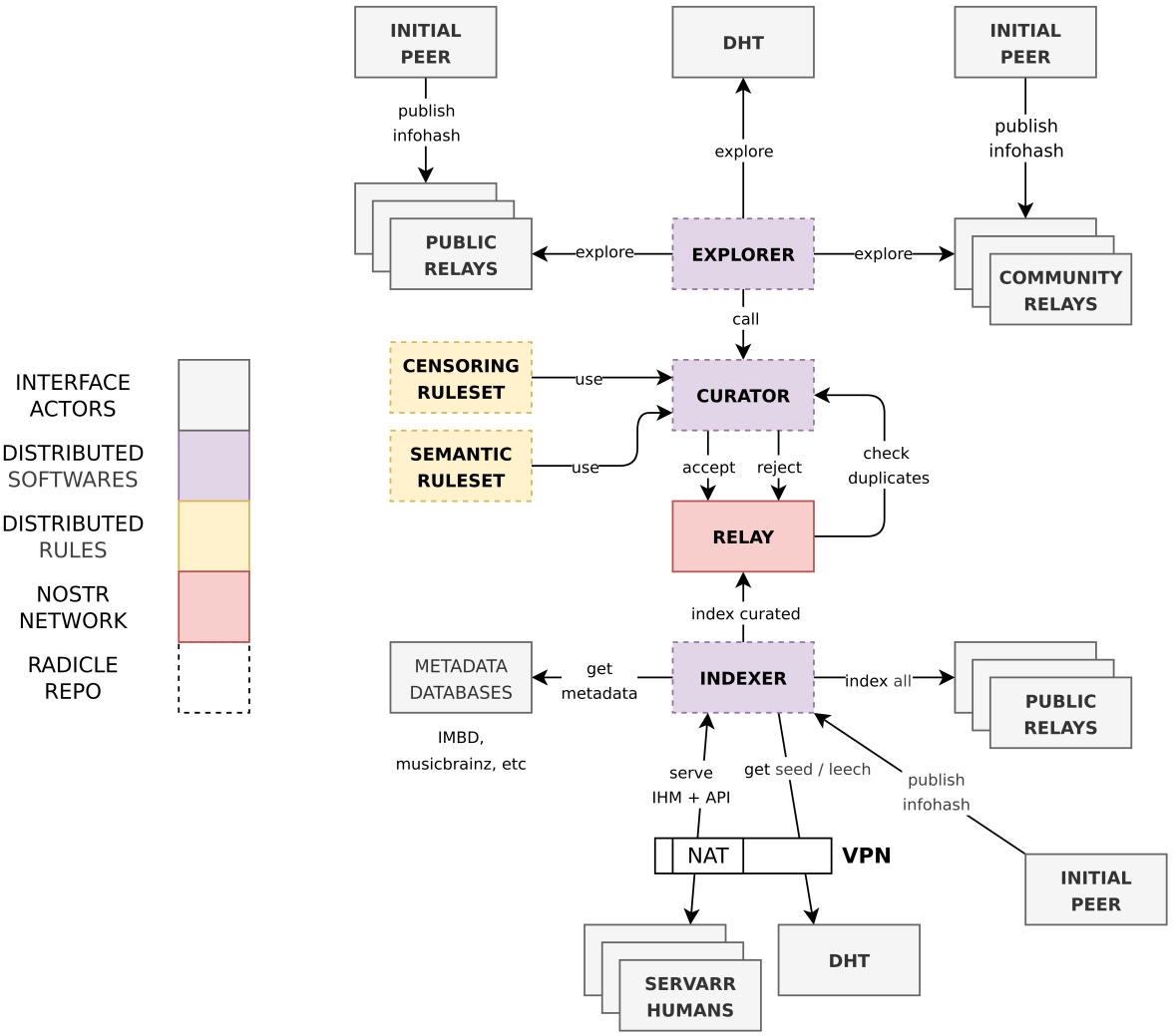

FIGURE 1 : Schéma Nostr + modération + indexation

4.1.1 Légende des couleurs (database.png)

- **Violet** : logiciels communautaires.
- **Rouge clair** : réseau Nostr.
- **Jaune clair** : dépôts de règles.
- **Gris clair** : acteurs/ressources externes.

4.2 Vue B - Architecture intégrée

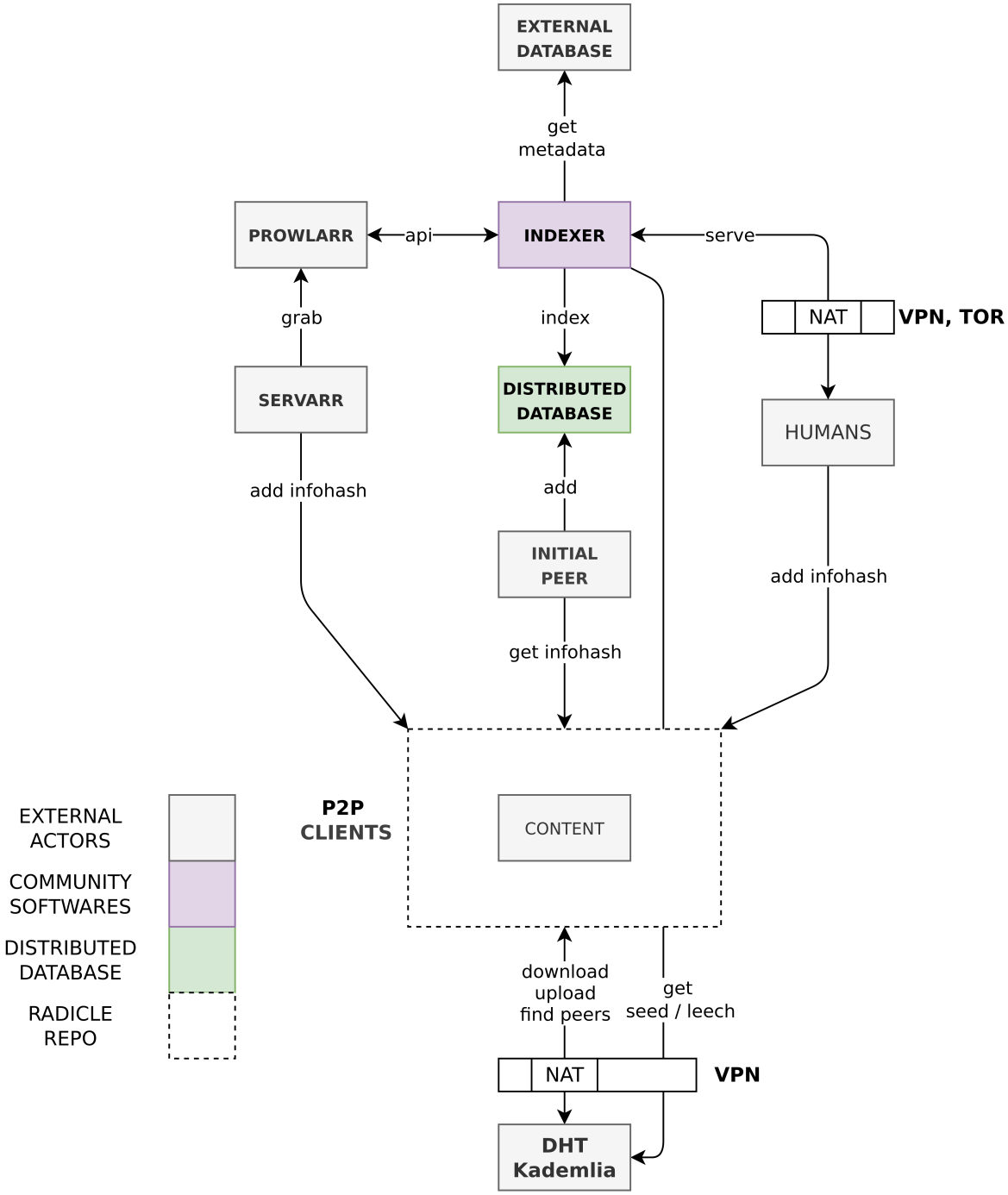

FIGURE 2 : Schéma intégration P2P + *arr* + base distribuée

4.2.1 Légende des couleurs (architecture.png)

- **Violet** : logiciels communautaires.
- **Vert** : base de donnée distribuée.
- **Gris clair** : acteurs/ressources externes.

Le schéma met en évidence l'enrichissement de métadonnées via l'External Database et l'alimentation directe de la base distribuée au travers du réseau Nostr par les pairs initiaux.

4.3 Idée centrale

1. **Distribution P2P** : aucun hébergement central du contenu.
2. **Métadonnées décentralisées** : publication via Nostr (relais publics et communautaires) et base distribuée avec administration et validation du contenu.
3. **Modération intégrée** : pipeline de vérification intégré pour appliquer censures et règles sémantiques.
4. **Indexation multi-source** : flux vérifié prioritaire, indexation optionnelle de relais publics.
5. **Conformité légale** : garde-fous explicites, signalement et auditabilité.

4.4 Concept explicite : curation et indexation fiables

L'exploration brute des relais publics ou de la DHT est utile mais insuffisante : la couverture est incomplète, la qualité variable et le spam fréquent. La stratégie retenue combine exploration distribuée, curation explicite et indexation centralisée par instance.

4.4.1 Limites du crawl DHT seul

— Le réseau P2P est massif et hautement dynamique.
— Seule une faible partie du réseau est visible par un acteur à un instant donné.
— Sans règles partagées, l'identification et la qualification des contenus restent ambiguës.

4.4.2 Modèle de découverte intentionnelle

— Les infohash DOIVENT être publiés sur n'importe quel relais **Nostr** via NIP-35.
— L'**Explorer** DOIT explorer les relais publics, relais communautaires et DHT.
— Le **Curator** DOIT appliquer le **Censoring Ruleset** et le **Semantic Ruleset** au contenu proposé par l'**Explorer**.
— L'**Indexer** DOIT privilégier le flux vérifié et PEUT indexer un sous-ensemble public.

4.4.3 Résultats attendus

— Couverture plus rapide des contenus de la communauté.
— Métadonnées plus fiables, moins de bruit issu des relais publics et de la DHT.
— Catalogue indexé aligné sur les règles communautaires.
— Comportement déterministe du pipeline de modération et d'indexation.

4.5 Principes de gouvernance

AUCUN rôle unique NE DOIT CONTRÔLER à lui seul l'ingestion, la validation, l'indexation et la publication. Les responsabilités DOIVENT être distribuées entre acteurs distincts OU entre instances distinctes. Les utilisateurs sont libres de se répartir sur les instances déployées au sein de la communauté OU de déployer leur propre instance privée.

4.6 Modèle de confiance et gouvernance

4.6.1 Rôles, responsabilités et invariants

- **Explorer** : collecte des évènements NIP-35 ; NE DOIT PAS décider d'acceptation ou rejet.
- **Curator** : applique les rulesets, signe les décisions ; DOIT publier chaque décision avec un `rulesetHash` et un `rulesetVersion`.
- **Indexer** : consomme le flux vérifié et applique une politique de confiance locale vis-à-vis d'un Curator ; DOIT ignorer les décisions provenant de clés non approuvées.
- Invariants de sécurité :
  - une décision de curation DOIT être traçable à un évènement source ;
  - une entrée acceptée DOIT être justifiée par une décision signée ;
  - une même instance DOIT conserver un journal d'audit minimal.

4.6.2 Identités, clés et révocation

- Les rôles utilisent des clés Ed25519 compatibles Nostr.
- La rotation de clé DOIT être annoncée via un évènement dédié de politique de sécurité.
- Une révocation DOIT inclure : `pubkey`, `reason`, `revokedAt`, et `signature`.
- Une clé révoquée DOIT être ignorée immédiatement par l'Indexer à partir de `revokedAt`.

4.6.3 Politique de confiance d'instance

- Une instance DOIT définir une **allowlist** et PEUT définir une **denylist** de `pubkey` Curator.
- Le format de Trust Policy DOIT être versionné (semver + hash) et signé.
- Onboarding : une nouvelle clé devient approuvée uniquement après publication d'une politique de sécurité qui l'inclut.
- Révocation : une politique de sécurité ou un évènement de révocation explicitement signé DOIT être propagé aux indexeurs.

4.6.4 Évènement de politique de sécurité

- Format recommandé (Nostr) : `kind 30173`, `k=org.legalindex.p2p.trust.v1`.
- Champs obligatoires (JSON) : `policyId`, `allowlist`, `denylist`, `revoked`, `effectiveAt`.
- Champs optionnels : `expiresAt`, `comment`.
- L'évènement DOIT être signé par l'administrateur d'instance.
- Les indexeurs DOIVENT appliquer la politique la plus récente par `effectiveAt`.

4.6.5 Agrégation multi-curateurs

- L'Indexer DOIT définir une règle d'agrégation : quorum **N sur M** ou politique explicite.
- Résolution de conflits :
- pour les raisons de censure légale (ex. `LEGAL_*`), **reject** DOIT primer.
- pour les raisons sémantiques, la règle locale (quorum, score) s'applique.
- Les décisions DOIVENT référencer un ruleset exact (via un hash de ruleset) ; l'Indexer DOIT refuser les décisions dont le ruleset n'est pas approuvé.

4.6.6 Auditabilité

— Journal minimal vérifiable : `eventId`, `decision`, `reasonCodes`, `rulesetVersion`, `rulesetHash`, `curatorPubkey`, `timestamp`.
— Le journal DOIT être conservé au moins localement ; PEUT être publié via un relais communautaire.

4.6.7 Conformité

— Un Indexer DOIT refuser une décision signée par une `pubkey` absente de la liste blanche.
— Un Indexer DOIT refuser une décision si `rulesetHash` n'est pas connu ou ne correspond pas à la version déclarée.

4.7 Ce que ce POC développe vs. ce qu'il utilise

— **Sera développé dans ce POC** : spécification, Explorer, Curator, intégration Nostr, Indexer.
— **Ne sera pas développé dans ce POC** : Prowlarr, Servarr, clients P2P, DHT, bases de métadonnées externes.

4.8 Moyens de déploiement

— La base distribuée contient des publications compatibles NIP-35 et s'appuie sur Nostr (relais publics et relais communautaires).
— Les règles communautaires et le code DOIVENT être versionnés dans un référentiel distribué (ex. Radicle).
— Les composants sont déployables par instances indépendantes, avec configuration locale des relais et règles.

5 Rôles et responsabilités

5.1 Initial Peer

— Publie des infohash via Nostr (relais publics ou communautaires), via un Indexer, et/ou dans la base distribuée.
— Peut opérer en mode **collector** (récupère un infohash depuis un client P2P ou un torrent/magnet) ou **publisher** (soumet vers Indexer/DB).
— DOIT respecter les règles de légalité du contenu dans le pays dans lequel il se trouve.

5.2 Relais publics

— relais Nostr ouverts recevant des publications non filtrées.
— Source de découverte mais bruitée.

5.3 Relais communautaires

— relais Nostr thématiques, privés/whitelist, servant uniquement les publications vérifiées par les Curators.
— Source privilégiée et source de vérité pour le flux légal vérifié consommé par les Indexers.
— Fournit un appui à la déduplication et au contrôle de cohérence.

5.4 Explorer

— Service distribué qui explore relais publics, relais communautaires et DHT.
— Transmet les publications à la couche de curation.

5.5 Curator

— Applique les règles de censure et les règles sémantiques de publication.
— Publie et signe les décisions de modération.
— Vérifie et normalise les doublons dans son relais de publication.

5.6 Indexer

— Service d'agrégation, d'indexation et de recherche.
— Indexe le flux validé et signé par les Curators et PEUT indexer des relais publics selon la configuration.
— Récupère des métadonnées via l'External Database et des stats DHT pour enrichir les descriptions.
— Expose IHM + API pour humains et intégrations *arr*.

5.7 External Database

— Source externe de métadonnées consultée par l'Indexer (lecture seule).
— Utilisée pour enrichir les résultats UI/Torznab.

5.8 Prowlarr / Servarr

— Consomment l'API de l'Indexer via Torznab.
— Pilotent le "grab" et le téléchargement des contenus côté clients P2P.

5.9 Clients P2P

— Détiennent et échangent le contenu via la DHT.
— Utilisent la DHT pour la découverte de pairs.

5.10 Humains

— Accèdent à l'UI/API de l'Indexer via NAT/VPN/TOR.
— Peuvent ajouter des infohash aux clients P2P.

5.11 Rulesets (sémantique et censure)

— **Censoring Ruleset** : règles déterministes de blocage (DMCA, spam, abus).
— **Semantic Ruleset** : règles de classification, cohérence, anti-duplicats, qualité.
— Chaque ruleset DOIT être versionné (semver + hash) et auditable.
— La distribution des rulesets DOIT se faire via un référentiel distribué et PEUT être relayée via Nostr.

6 Flux & concepts

6.1 Flux A — Nostr + modération + indexation

— L'Initial Peer publie un infohash sur un relais public ou communautaire.
— L'Explorer explore relais publics, relais communautaires et DHT.
— L'Explorer transmet les publications au Curator.
— Le Curator applique les rulesets puis accepte/rejette et publie avec une signature.
— Le Relais communautaire reçoit et sert uniquement les publications acceptées.
— L'Indexer consomme le flux vérifié et l'enrichit via l'External Database.

6.2 Flux B — Intégration P2P + *arr* + base distribuée

— L'Initial Peer peut récupérer un infohash depuis un client P2P ou un torrent/magnet.
— L'Initial Peer ajoute un infohash directement à l'Indexer.
— L'Initial Peer ajoute un infohash à la base distribuée.
— L'Indexer expose une API compatible Prowlarr/Torznab.
— L'Indexer interroge l'External Database pour enrichir les résultats.
— Prowlarr relaie vers Servarr, qui "grab" et envoie aux clients P2P.
— Les clients P2P récupèrent des pairs via la DHT.
— L'Indexer enrichit via l'External Database et indexe la base distribuée.
— L'Indexer sert UI/API aux humains.

6.3 Modèle de données

6.3.1 Publication initiale

Objet publié sur Nostr pour annoncer un infohash et permettre la curation. - `infohash` : infohash normalisé.
- `infohashType` : `btih`.
- `infohashVersion` : `v1`, `v2` ou `hybrid`.
- `createdAt` : timestamp ISO 8601.
- `sizeBytes` (optionnel).
- `tags` : tags/catégories normalisés.
- `source` : `manual`, `client-p2p`, `import`.
- `sourceRelay` (optionnel).
- `nostrEventId` / `pubkey` / `signature`.
- `magnet` (optionnel), `name` (optionnel).

6.3.2 Décision de vérification

Décision de vérification signée qui accepte ou rejette une publication initiale. - `decision` : `accept` ou `reject`.
- `reasonCodes` : liste de codes stables.
- `rulesetType` : `censoring` ou `semantic`.
- `rulesetVersion` / `rulesetHash`.
- `targetEventId` / `targetInfohash`. - `curatorPubkey` / `signature` / `createdAt`.
- `decisionId` (optionnel) : hash du JSON de la décision.

6.3.3 Descripteur de règles de modération

Métadonnée d'un ruleset versionné, utilisée pour l'audit et la traçabilité des décisions. - `rulesetId` : identifiant stable.
- `type` : `censoring` ou `semantic`.
- `version` : semver.
- `hash` : hash du contenu.
- `source` : Nostr event id et/ou référentiel distribué.
- `updatedAt` : date de publication.

6.3.4 Publication modérée

Entrée indexée issue d'une publication validée et enrichie par l'Indexer.
- `infohash` : clé canonique.
- `title` / `categories` / `tags`.
- `curationStatus` : `accepted` / `rejected` / `unknown`.
- `sources` : relais et/origines vues.
- `externalMetadataRefs`.
- `seedLeechStats`.
- `dedupGroupId` (optionnel).

6.3.5 Statistiques de popularité

Instantané d'observation des seeds/leechers pour l'affichage et le tri.
- `infohash`.
- `seeders` / `leechers`.
- `observedAt`.
- `source` : DHT / client P2P.
- `ttl` / `confidence` (optionnels).

6.4 Déduplication

— La clé primaire DOIT être l'infohash normalisé.
— La gestion des variantes `v1`, `v2` et `hybrid` DOIT être explicite ; un `hybrid` peut référencer deux infohash.
— Classes de duplicats :
— **exact** : même infohash ;
— **probable** : mêmes `sizeBytes` + mêmes hashes de fichiers (si disponibles) ;
— **related** : titre, tags et structure proches (score sémantique).

— La déduplication sémantique PEUT utiliser un score (0–100) et un seuil configurable ; la décision DOIT être tracée dans `dedupGroupId`.
— Dans l'API, l'Indexer DEVRAIT exposer un enregistrement canonique et PEUT lister les duplicats associés.

7 Intégration Nostr

— Les publications d'infohash DOIVENT être des évènements Nostr signés.
— Les relais publics, communautaires et curated DOIVENT être configurables localement.
— La vérification des signatures, l'anti-spam et le rate limiting DOIVENT être appliqués.
— La compatibilité NIP-35 DEVRAIT être préservée quand possible.

7.0.1 Tests de conformité

— Un évènement infohashPublication DOIT être rejeté si l'infohash n'est pas normalisé ou si le tag `i` est absent.
— Un évènement CurationDecision DOIT être rejeté s'il manque `rulesetHash` ou `targetEventId`.

8 Modération

— Pipeline : Explorer → Curator → Relais communautaire.
— Le Curator DOIT appliquer le **Censoring Ruleset** puis le **Semantic Ruleset**.
— Chaque décision DOIT référencer l'évènement original et la version du dépôt de règles.
— Les règles et décisions DOIVENT être auditables et traçables.
— Un mécanisme d'appel/signalement PEUT être proposé par instance.
— Les règles de **censure légale** DOIVENT être déterministes et reproductibles.
— Les règles de **qualité/pertinence** PEUVENT être probabilistes.

8.0.1 Taxonomie des codes de rejet

Code	Type	Description
LEGAL_DMCA	déterministe	signalement légal documenté
LEGAL_ILLEGAL	déterministe	contenu manifestement non licite
ABUSE_SPAM	déterministe	spam, flooding, duplication abusive
ABUSE_MALWARE	déterministe	contenu malveillant ou dangereux
SEM_DUPLICATE_EXACT	déterministe	duplicat exact (même infohash)
SEM_DUPLICATE_PROBABLE	probabiliste	duplicat probable (mêmes fichiers)
SEM_BAD_META	probabiliste	métadonnées incohérentes/incomplètes
SEM_LOW_QUALITY	probabiliste	qualité insuffisante

Code	Type	Description
SEM_CATEGORY_MISMATCH	probabiliste	catégorisation incohérente

8.0.2 Versionnage des règles de modération

— Chaque ruleset DOIT être versionné et identifié par un hash.
— Un ruleset PEUT être déprécié ; une période de grâce DEVRAIT être annoncée.
— Les décisions de vérification DOIVENT inclure le `rulesetVersion` et le `rulesetHash`.

8.0.3 Conflits de versions

— Si plusieurs Curators utilisent des versions différentes, l'Indexer DOIT enregistrer la version par décision.
— En cas de conflit pour une même cible, les codes de rejet déterministes DOIVENT primer.
— Pour les codes de rejet probabilistes, l'Indexer DEVRAIT appliquer sa politique locale (quorum ou score).

8.1 Signalement, déréférencement, et limites du retrait

— Le "retrait" signifie **déréférencement** des métadonnées et filtrage des évènements ; il ne supprime pas le contenu P2P.
— Un canal formel de demande de censure DOIT exister sur l'instance Indexeur.
— Un accusé de réception DEVRAIT être publié sous 24h.
— Chaque instance DOIT publier sa politique de traitement (transparence, critères, délais).

8.2 Auth, permissions et rate limit de l'API Torznab

— L'API DOIT exiger une clé (`apikey`) par utilisateur/instance.
— Les permissions PEUVENT restreindre l'accès à certaines catégories.
— Le rate limit DOIT être appliqué par `apikey` et par IP.
— Les logs DOIVENT respecter le RGPD.

8.3 Consommation externe

— Métadonnées : récupération via External Database, cache et quotas.
— Seeds/leech : collecte via DHT, cache et rafraîchissement périodique.

9 Observabilité & déploiement

— Configuration locale : listes de relais, règles, seuils de vérifications, caches, quotas.
— Écriture dans la BDD distribuée : pairs autorisés, clés, signatures et ACL éventuelles.
— Observabilité : logs structurés et visibilité des métriques.
— Scalabilité : Explorer, Indexer et Curator sont fédérés par les relais Nostr.
— Réseaux : UI/API derrière NAT, support VPN, TOR possible (TOR-friendly, sans promesse d'anonymat absolu), avec rate limiting, anti-abus et gestion de la latence.

10 Exigences non techniques

— Le système DOIT servir uniquement des contenus licites en accord avec la législation locale.
— Le catalogue DOIT rester décentralisé et contrôlé par la communauté.
— Le système DEVRAIT minimiser les points uniques de défaillance.
— Les rulesets et décisions DOIVENT être auditables et traçables.

11 Sécurité et conformité

— Un canal formel de signalement d'abus et de retrait de contenu DOIT exister.
— La communauté DOIT mettre en place des pratiques de modération des métadonnées.
— Le réseau DOIT éviter toute promotion de contenus illicites.
— La défaillance d'un acteur ne DOIT PAS entraîner la défaillance du réseau.

12 Ce que n'est pas ce projet

— Hébergement ou redistribution de contenus.
— Contournement de DRM ou techniques de piratage.
— Garantie d'anonymat absolu des utilisateurs.
— Développement de clients P2P.

13 Questions ouvertes / travaux futurs

— Définir un processus d'appel communautaire pour les décisions de vérification.
— Réduire le bruit des relais publics sans fragmentation excessive.
— Définir des politiques locales pour l'indexation partielle des relais publics.

14 Risques connus et modes de défaillance

— Empoisonnement des métadonnées via relais publics.
— Compromission d'un Curator ou d'un relais public.
— Faible participation communautaire.
— Fragmentation entre relais et indexeurs.
— Conflits de confiance entre instances.
— Pressions légales locales sur les opérateurs.
— Dispersions des règles de validation.

15 Glossaire

— **Initial Peer** : utilisateur/entité publiant un infohash sur Nostr.
— **Relais public** : relais Nostr ouvert recevant des évènements non filtrés.
— **Relais communautaire** : relais Nostr thématique, privé/whitelist, servant uniquement les publications vérifiées.

— **Explorer** : service distribué d'exploration des relais et de la DHT.
— **Curator** : service appliquant les règles de curation.
— **Indexer** : instance d'indexation exposant UI/API.
— **External Database** : source externe de métadonnées consultée par l'Indexer.
— **Distributed Database** : base append-only utilisée pour l'indexation.
— **DistributedDBEntry** : entrée append-only écrite par un Initial Peer.
— **infohashPublication** : publication Nostr d'un infohash.
— **CurationDecision** : décision signée d'acceptation/rejet.
— **ReportOrAppeal** : évènement de signalement ou d'appel.
— **RulesetDescriptor** : métadonnée d'un ruleset (type, version, hash).
— **Trust Policy** : politique locale d'allowlist/denylist de Curators.
— **SeedLeechStats** : statistiques de seed/leech observées.
— **TorrentRecord** : entrée indexée exposée par l'Indexer.
— **Reason codes** : codes stables de décision (LEGAL_, *ABUSE_*, SEM_*).
— **Censoring Ruleset** : règles de blocage explicites (DMCA, spam, abus).
— **Semantic Ruleset** : règles de qualité, classification et déduplication.
— **DHT (Distributed Hash Table)** : table distribuée P2P pour la découverte de pairs.
— **Prowlarr** : client de méta-recherche reliant l'Indexer à Servarr.
— **Servarr** : clients de gestion de bibliothèque (Radarr/Sonarr/Lidarr/Readarr).
— **Torznab** : API de type indexeur utilisée par des clients pour la recherche.
— **VPN** : tunnel réseau utilisé pour l'accès ou l'isolement d'un segment P2P.
— **Tor / onion** : réseau d'anonymisation et services cachés.