# 📘 Gatto — Document de vision & architecture

**But** : agréger des *sources de vérité* et des signaux (éditoriaux, plateformes, données locales) pour **identifier, qualifier et classer les meilleurs POIs** d’une ville — pas seulement les nouveautés ou les tendances, mais **le “meilleur état de l’art”** par catégorie, quartier et moment. Gatto détecte aussi **les nouvelles ouvertures** et **les tendances émergentes**, attribue des **badges/tags**, et génère des **collections dynamiques** (façon “Spotify des lieux”), pouvant être éditorialisées par un **agent IA**.

## 1) Vision produit

### 1.1 Positionnement

* **Agrégateur de vérité** : croise guides, presse, blogs, listes d’experts, plateformes (via API) et signaux sociaux.
* **Qualité > quantité** : met en avant *les meilleurs lieux* (stables) et *les vraies tendances* (dynamiques), avec transparence des sources.
* **Curation hybride** : scoring automatique + interventions éditoriales + **agents IA** pour scénariser des collections.

### 1.2 Périmètre

* **Best-of structurés** (catégories, quartiers, budgets, styles).
* **Nouveautés & buzz** (openings, relances, pop-ups, prix reçus).
* **Évolution dans le temps** (momentum, récence des mentions, saisonnalité).
* **Explicabilité** : pourquoi ce lieu est classé ? quelles sources ? quelles dates ?

### 1.3 Non-objectifs (court terme)

* Avis UGC non vérifiés (hors agrégats de plateformes officielles via API).
* Scraping sauvage instable/illégal.

## 2) Principes de classement

### 2.1 Dimensions

1. **Autorité** de la source (poids éditorial, ex. Michelin > blog niche).
2. **Récence / Décroissance** (mentions récentes pondérées par *decay\_tau\_days*).
3. **Diversité** (pluralité des sources vs. répétitions d’une même source).
4. **Cohérence sémantique** (matching POI précis, désambiguisation).
5. **Preuves fortes** (prix, classements officiels, listes annuelles).
6. **Signals plateformes** (notes & volumes d’avis **via API** autorisées).

### 2.2 Badges & tags

* **Badges (forts, binaires)** : *New Opening*, *Awarded*, *Hot Trend*, *Editor’s Pick*, *Neighborhood Gem*, *Top Brunch*, *Best Cocktail Bar*, etc.
* **Tags (attributs souples)** : cuisine, budget, ambiance, features (terrasse, végétarien, natural wine, coffee geek…).

### 2.3 Explicabilité

* Chaque score/collection garde la **trace des contributeurs** (source → mention → score partiel → score final).
* Affichage : *“Classé #2 car cité par Le Fooding (2025-06), Télérama (2025-07), note Google 4.6/2300 (2025-08)”*.

## 3) Sources & gouvernance

### 3.1 Catalogue de sources (DB-driven)

Table **source\_catalog** (exemples de colonnes) : - source\_id (pk), label, type (guide/press/blog/local/reviews/api),  
- authority\_weight, fetch\_method (cse|rss|html|api), base\_url,  
- rate\_limit\_daily, ttl\_default\_days, is\_active, decay\_tau\_days.

**Règles** : - Pas d’URLs/sélecteurs en dur dans le code.  
- Ajout/édition de sources via la DB (activation, poids, méthode…).  
- Séparation **éditorial** vs **reviews/API** (les plateformes d’avis sont consommées via leurs APIs dédiées, pas via CSE).

### 3.2 Conformité & robots

* Respect robots.txt, TOS des sites, et usage d’APIs officielles.
* Journalisation des méthodes (cse/rss/html/api) et des limites (quotas/429/backoff).

## 4) Modèle de données (vue logique)

**POI** (id, name, city\_slug, lat, lng, place\_id, categories, price\_level, …)  
**Source** (source\_id, label, type, authority\_weight, decay\_tau\_days, fetch\_method, base\_url, …)  
**Mention** (id, poi\_id, source\_id, url, title, published\_at, matched\_score, extracted\_at, raw\_snippet, …)  
**Rating Snapshot** (id, poi\_id, provider, rating, reviews\_count, created\_at)  
**Photo** (id, poi\_id, provider, url, width, height, checksum, created\_at)  
**Badge/Tag** (id, poi\_id, badge\_type, tag, source\_expl, created\_at)  
**Collection** (id, slug, title, type, rules\_json, curator (human/agent), updated\_at)

## 5) Pipeline & services

### 5.1 Ingestion POIs (Google Places API)

* **Ciblée** (par nom/Place ID) et **par ville**.
* Incrémentale (upsert), snapshots de note/volume, normalisation d’adresses.

### 5.2 Scanner de mentions (éditorial)

* **POI-centré** : requêtes SERP *site:domain “poi”* + hints (ville, CP, arr.), + variantes normalisées.
* Modes :
  + CSE\_DOMAIN (par défaut) ;
  + RSS/HTML (quand confirmé en DB) ;
  + fallback/combinaison selon la source.
* **Matching & scoring** :
  + filtrage par nom exact/variantes ;
  + pondération par autorité × récence ;
  + déduplication (fenêtre, canonical URL, hash titre+date).

### 5.3 Classification (Intelligent Classifier)

* Produit le **Gatto Score** multi-facteurs et déclenche **badges/tags**.
* Alimente l’index de recherche et les collections.

### 5.4 Photos (Photo Processor)

* Fetch multi-provider (API / URLs publiques autorisées), optimisation (formats, tailles), cache.

### 5.5 Collections dynamiques (Builder)

* **Règles** : requêtes déclaratives (ex. *“Top 20 néo-bistrots du 11e, budget $$, dernier article ≤ 90j, score ≥ 75”*).
* **Playlists** : ordonnées par *Gatto Score* + recettes de diversité (éviter 5 lieux d’un même groupe).
* **Agent IA curator** :
  + propose titres, descriptions, couvertures ;
  + ajuste la playlist (diversité quartiers/cuisines),
  + explique ses choix (*“on garde 1 étoilé max par collection”*).

### 5.6 Orchestration & monitoring

* **run\_pipeline** : full | mentions | classify | photos | seed→scan.
* **Monitoring service** : planifie, expose une API santé/metrics (latence, quotas, taux de match, mentions/jour).
* **Logs** structurés par étape + métriques clé : *URLs scannées, candidats, upserts, 429, coût estimé*.

## 6) Qualité, quotas & résilience

* **Rate limiting & backoff** : limiter QPS, re-tenter sur 429 (jitter).
* **Caching SERP** (clé = domain+query, TTL configuré).
* **Idempotence** : upserts sur POI/Mention/Photo.
* **Tests ciblés** : filtres --poi-id / --poi-name pour valider une chaîne bout-à-bout sur un *POI d’autorité*.
* **Explainability** : conserver *pourquoi* un résultat a été accepté/refusé (seuils HIGH/MID/LOW, règles de matching).

## 7) Parcours utilisateur (exemples)

* **“Je veux le meilleur de Paris maintenant”** → Collections *Best-of Paris* (stables) + *Hot Right Now* (dynamiques).
* **“Montre-moi les ouvertures dans le 11e à <40€”** → Collection *New & Good 11e* selon badges *New Opening* + score mini + budget.
* **“Un guide week-end”** → Agent IA génère *48h dans le Marais* (mix brunch, café, bar, dîner) + cartes & explications.

## 8) API & surfaces d’exposition

* **API REST** (lecture) :
  + /pois?city=paris&sort=score
  + /pois/{id}/mentions
  + /collections/{slug}
  + /search?q=natural wine paris
* **Backoffice** : ajustement des poids, activation de sources, curation humaine, diff/rollback.

## 9) KPIs & garde-fous

* **Qualité** : taux de mentions pertinentes, précision du matching, couverture sources.
* **Fraîcheur** : âge médian des mentions, part des contenus ≤ 90 jours.
* **Stabilité** : churn des top‑N d’une collection, variance intra‑semaine.
* **Coûts & quotas** : requêtes/jour, 429/s, coût/API.

## 10) Roadmap

* **Sources** : élargir guides locaux, RSS/HTML confirmés par domaine.
* **ML/NLP** : classification d’articles (sentiment, topics), NER pour menus/chefs/adresses, détection d’événements.
* **Collections** : contraintes éditoriales avancées (diversité quartiers/cuisines), *mixing rules* (anti‑doublons de groupe), *time windows*.
* **UX** : fiches POI avec explication du score, historique des mentions, playlists partageables.
* **Ops** : cache distribué, observabilité (OpenTelemetry), alerts quotas.

## 11) Glossaire rapide

* **POI** : Point d’Intérêt (lieu).
* **Mention** : occurrence d’un POI dans une source.
* **Authority weight** : poids éditorial de la source.
* **Decay** : décroissance temporelle appliquée aux mentions.
* **Badge** : label fort (binaire), **Tag** : attribut souple.
* **Collection** : liste dynamique régie par des règles + curation IA/humaine.

### TL;DR

Gatto = **le graphe de vérité des lieux** d’une ville : - il agrège et *explique* pourquoi un lieu est parmi les meilleurs,  
- détecte les **ouvertures** et **tendances**,  
- attribue **badges/tags**,  
- et génère des **collections dynamiques** orchestrées par un **agent IA**.