Conception technique détaillée pour le plugin **ApisManager Pro**, transformant WordPress en un SGBD apicole à valeur probante.

---

1) Vision & périmètre

- **Cas d’usage :**
    - **Apiculteur :** Saisie terrain (mobile-first), suivi de production, gestion légale du registre d'élevage, pilotage de la sélection génétique.
    - **TSA (Technicien Sanitaire Apicole) :** Accompagnement, saisie de rapports de visite.
    - **GDSA/GDS (Lecture) :** Surveillance épidémiologique, vérification de la conformité des traitements (AMM) sur un département donné, après consentement de l'apiculteur.
- **Hypothèses et limites :**
    - Le serveur doit supporter PHP 8.1+ (pour l'énumération et la gestion stricte des types).
    - L'immutabilité est garantie au niveau applicatif (PHP/SQL) ; une compromission "root" de la base de données reste détectable par le chaînage de hash mais ne peut empêcher techniquement une suppression physique par un administrateur système.

---

2) Modèle de données (tables custom)

L'utilisation de tables SQL personnalisées via $wpdb est obligatoire pour l'intégrité relationnelle.

- **wp_apis_entities** : Ruchers et Emplacements. (ID, NAPI, Coordonnées GPS, Type d'environnement).
- **wp_apis_colonies** : Ruches et Reines. (ID, Entity_ID, Code_Souche, Date_Introduction, Généalogie_JSON).
- **wp_apis_registry (La table critique)** :
    - id (BIGINT), user_id, timestamp_utc, type (Traitement, Observation, Récolte).
    - payload_json (Données métier brutes).
    - status (DRAFT, FINALIZED, RECTIFIED).

    - hash_prev (SHA-256 de l'entrée précédente).
    - hash_current (SHA-256 de l'entrée actuelle).
    - rectified_id (FK vers l'ID original si c'est un correctif).
- **wp_apis_audit_log** : (ID, Timestamp, User_ID, Action, Object_ID, Hash_Snapshot).
- **Stratégie de suppression :**
    - Tables Ruchers/Colonies : **Soft-delete** autorisé (colonne deleted_at).
    - Table Registre (finalized) : **Suppression INTERDITE**. Toute ligne avec un hash calculé doit rester en base pour maintenir la continuité de la chaîne.

---

3) Mécanisme d'immutabilité

- **Workflow :**
    1. DRAFT : Modification possible. Pas de hash généré.
    2. VALIDATION : L'utilisateur signe électroniquement (clic de validation).
    3. FINALIZED : Le système calcule le hash, verrouille la ligne (lecture seule applicative) et génère l'entrée dans l'audit log.
- **Rectificatifs :** Un rectificatif est une **nouvelle ligne** avec status = RECTIFIED. Elle contient un champ rectified_id pointant vers l'erreur. L'UI affiche la donnée rectifiée mais permet de consulter l'original (barré) pour l'audit.
- **Canonicalisation (C14N) :** Avant le hash, le payload_json est :
    1. Trié par clés alphabétiques.
    2. Dates converties au format ISO-8601 (UTC).
    3. Nombres normalisés (précision fixe).
    4. Suppression des champs volatils (ID auto-incrément, timestamps système).
- **Hash Chain :**
- Hn=SHA256(Payloadn+Hn−1+Timestampn)H_n = SHA256(Payload_n + H_{n-1} + Timestamp_n)Hn=SHA256(Payloadn+Hn−1+Timestampn)

.

- **Vérification :** Une routine integrity-check (CLI WP-CLI ou Cron) recalcule la chaîne complète. Si
- Hcalc≠HstoredH_{calc} \neq H_{stored}Hcalc•=Hstored

, une alerte critique est levée (altération détectée).

---

4) Sécurité & permissions WordPress

- **Rôles & Capabilities :**
  - apis_beekeeper : apis_edit_own_drafts, apis_finalize_registry.
  - apis_institution : apis_view_consented_data (uniquement lecture).
- **Protection :**
  - **Prepared Statements** ($wpdb->prepare) sur toutes les requêtes.
  - **Nonce WP** systématique sur chaque action de validation.
  - **Immutabilité SQL** : Trigger SQL (si possible sur l'environnement) ou BEFORE UPDATE en PHP qui wp_die() si l'état est FINALIZED.

---

5) RGPD & consentements

- **Table wp_apis_consents :**
  - user_id, receiver_id (ID de l'organisation GDSA), scope (Sanitaire, Sélection), expires_at.
- **Journalisation des accès :** Chaque GET sur un endpoint API institutionnel crée une ligne dans wp_apis_access_log : "L'utilisateur [ID] a consulté le registre de [ID_Apiculteur] le [Date] pour motif [Surveillance]".
- **Minimisation :** Les exports institutionnels masquent le nom de l'apiculteur au profit du NAPI et de la localisation par code postal (pseudonymisation).

---

6) API REST & JSON Schemas versionnés

- **Endpoints :**
  - POST /apis/v1/registry/draft : Créer un brouillon.
  - PUT /apis/v1/registry/finalize/{id} : Verrouiller une entrée.

    - GET /apis/v1/institution/export?dept=77 : Extraction GDS (filtrée par consentement).
- **Versioning :** Inclus dans le header Accept: application/vnd.apis.v1+json.
- **Schéma JSON (Exemple simplifié) :**

code JSON

downloadcontent_copy

expand_less

```
{
  "$schema": "https://apismanager.io/schemas/v1/treatment.json",
  "type": "treatment",
  "data": {
    "amm_code": "FR/V/1234/2010",
    "product": "Apivar",
    "lot_number": "LOT-2024-001",
    "dosage": "2 lanières"
  }
}
```

---

7) Exports “valeur probante”

- **PDF Horodaté :** Utilisation de la librairie *TCPDF* ou *mPDF*. Chaque page contient un footer avec le hash de l'entrée et l'ID de la transaction.
- **Empreinte numérique :** Un QR Code en fin de document contient l'URL de vérification publique (/verify/{hash}) permettant de comparer le PDF avec la donnée en base.
- **Réconciliation :** Le fichier JSON d'export contient un bloc proof incluant : timestamp, merkle_root (si export massif) et hash_chain_status.

---

8) UI / UX (Gutenberg widgets + écrans)

- **Widget "État Sanitaire" :** Bloc Gutenberg affichant les indicateurs varroa récents (couleur Vert/Orange/Rouge selon seuils Arista).

- **UX Anti-erreur :**
    - Bouton "Valider le registre" avec une double confirmation : *"Je certifie que ces informations sont exactes. Cette action est irréversible."*
    - Saisie assistée pour les médicaments (recherche par base AMM intégrée).

---

9) Performance & scalabilitÃ©

- **Volumes :** Pour 10 000 apiculteurs, le registre peut atteindre plusieurs millions de lignes.
- **Optimisation :** Index B-Tree sur (user_id, timestamp_utc) et (status, rectified_id).
- **Stockage :** Le payload_json est stocké en LONGTEXT pour la flexibilité, mais les champs de filtrage (Date, Type, User) sont des colonnes SQL indexées.

---

10) Plan de tests & critères d'acceptation

- **Test d'Intégrité (Critique) :**
    - Scénario : Modifier manuellement une valeur de traitement directement en base de données.
    - Résultat attendu : La fonction check_chain_integrity() doit retourner FALSE et identifier la ligne corrompue.
- **Test RGPD :**
    - Scénario : Révoquer un consentement. Tenter un appel API institutionnel.
    - Résultat attendu : Erreur 403 Forbidden.
- **Critères de Go/No-Go :**

1. La chaîne de hash fonctionne sur 1000 entrées successives (< 500ms).
2. Aucun utilisateur (même admin) ne peut éditer une ligne FINALIZED via l'interface WP.
3. Export PDF généré avec hash vérifiable.