C'est un projet d'ingénierie pédagogique passionnant. Pour transformer CaSS d'un simple registre en un **instrument de mesure décisionnel**, il ne faut pas modifier le référentiel lui-même (les objets Competency), mais travailler sur la couche **Assertions** et **Evidence**.

Puisque vous utilisez Coolify sur un VPS, la méthode la plus robuste et automatisable est de déployer un **Script de Traitement (Logic Engine)** qui interagit avec l'API de CaSS.

Voici la stratégie d'implémentation pour intégrer la couche de calcul CEREDIS :

### 1. Structure des données (Le format d'import)

Pour automatiser, vous devez utiliser des **Assertions** enrichies. CaSS utilise le format JSON-LD. Chaque preuve envoyée vers https://cass.ceredis.net doit suivre cette structure pour que le moteur de calcul puisse la traiter :

JSON

{  
 "@type": "Assertion",  
 "competency": "https://cass.ceredis.net/api/data/.../1.1",  
 "subject": "ID\_APPRENANT",  
 "agent": "ID\_EVALUATEUR",  
 "evidence": ["URL\_OU\_CONTENU\_PREUVE"],  
 "ceredis": {  
 "evidenceType": "P2",  
 "rawScore": 85.0,  
 "timestamp": "2026-01-08T10:00:00Z"  
 }  
}

### 2. L'Architecture du Moteur de Calcul (Logic Layer)

CaSS ne gère pas nativement les agrégations pondérées complexes (sommes croisées avec seuils planchers). Vous devez exécuter un script (Python ou Node.js) que vous pouvez héberger comme un service supplémentaire sur votre Coolify.

#### Le Script d'Automatisation (Algorithme)

Le script doit effectuer ces 4 opérations de manière séquentielle :

| **Niveau** | **Opération** | **Logique à implémenter** |
| --- | --- | --- |
| **Lvl 1 & 2** | **Score Compétence** | Filtrer les assertions par subject et competency. Appliquer les poids : $S\_{comp} = \sum (rawScore\_i \times poids\_{P\_x})$. |
| **Lvl 3** | **Agrégation Domaine** | Grouper les $S\_{comp}$ selon votre table de mapping (ex: D1 = 1.1, 1.2, 1.3). |
| **Lvl 4** | **Score Global** | $ceredisScore = \sum (domainScore\_k \times poidsD\_k) \times 6$. |
| **Lvl 5** | **Décision CECRL** | Appliquer les seuils (200, 300, 400, 500) **ET** vérifier les conditions de blocage (seuils planchers par domaine). |

### 3. Automatisation de l'implémentation

Pour éviter le travail manuel, je vous recommande de ne pas utiliser l'interface graphique de CaSS pour les scores, mais de passer par un **Fichier de Configuration de Calcul (JSON)**.

#### Étape A : Créer la table de référence (ceredis-config.json)

Ce fichier définit les règles "autour" de CaSS :

JSON

{  
 "weights": { "P1": 0.15, "P2": 0.30, "P3": 0.35, "P4": 0.20 },  
 "domains": {  
 "D1": { "competencies": ["1.1", "1.2", "1.3"], "weight": 0.20, "floor": 50 },  
 "D2": { "competencies": ["2.1", "2.2", "2.3"], "weight": 0.20, "floor": 50 }  
 // ... etc pour D3, D4, D5  
 }  
}

#### Étape B : Exécution

Vous n'avez pas besoin d'importer un fichier XML/JSON *dans* CaSS pour le calcul. La méthode la plus propre est :

1. **Stocker** les assertions brutes dans CaSS (via API).
2. **Exécuter** un script "Cron" (via Coolify) qui lit les données, calcule les scores CEREDIS, et **publie** le résultat final (CECRL\_Level et ceredisScore) dans un profil de l'apprenant dédié dans CaSS.

### 4. Modèle de calcul des seuils (Niveau 5)

Voici comment traduire votre règle décisionnelle renforcée en code (pseudo-code) :

Python

def derive\_cecrl(score\_global, scores\_domaines, preuves\_p4):  
 # Mapping de base  
 if score\_global < 200: return "A1"  
 elif 200 <= score\_global < 300: niveau\_cible = "A2"  
 elif 300 <= score\_global < 400: niveau\_cible = "B1"  
 elif 400 <= score\_global < 500: niveau\_cible = "B2"  
 else: niveau\_cible = "C1"  
  
 # Vérification des conditions de blocage (CEREDIS)  
 if any(s < seuil\_plancher for s in scores\_domaines):  
 return descendre\_niveau(niveau\_cible)  
   
 if (niveau\_cible in ["B2", "C1"]) and not preuves\_p4:  
 return "B1+" # Niveau inférieur conservé si manque de preuves réflexives  
   
 return niveau\_cible

### Prochaines étapes suggérées

Souhaitez-vous que je vous génère :

1. Le **script Node.js complet** prêt à être déployé sur votre Coolify pour effectuer ces calculs via l'API de votre instance ?
2. Ou le **schéma JSON-LD précis** pour les Assertions CEREDIS afin de préparer vos futurs imports de données ?