ADN numérique — Version 5.0 (Réécrit, corrigé, consolidé)

Résumé

Version 5.0 de ton ADN numérique : correction de tous les points relevés (CRDT, sécurité, diversité adaptative, économie cellulaire anti-abus, interop blockchain merkle-root only, K8s WASM, attestations, tests, etc.). Une architecture complète prête pour prototypage.

---

1. Principes Directeurs

- **Fractalité & parallélisme** : chaque gène est un module récursif.
- **Conscience multi-agent** : coalition Perceptor, Predictor, Evaluator, Reflector, Arbiter.
- **Sécurité par construction** : invariants immuables, sandbox WASM, audit cryptographique.
- **Évolution contrôlée** : mutations validées par sandboxes + quorum + preuves.
- **Traçabilité forte** : ledger CRDT causal + ancrage blockchain merkle-root.
- **Coût comme loi physique interne** : CPU, mémoire, I/O, énergie, risque.
- **Anti-monoculture cognitive** : diversités mesurées (Shannon + Simpson), seuils adaptatifs.

---

2. Structure Générale du Génome

```
Type GeneNode {
  id: String
  codons: List[Codon]
  promoter: Condition
  regulators: List[Regulator]
  markers: List[Marker]
  resources: {cpu_quota, mem_quota, io_quota}
  niche: Niche
  immutable_zone: Bool
}

Type Genome {
  nodes: Graph[GeneNode]
  ledger: CRDT_OpLog
}
```

---

3. Conscience Multi-Agent Distribuée

Agents

- **Perceptor** : capte signaux.

- **Predictor** : produit modèles.
- **Evaluator** : calcule utilité, coût, risque.
- **Reflector** : détecte surprise, initie méta-réflexion.
- **Arbiter** : résout conflits, valide décisions.

Cycle cognitif

Perception → Prediction → Evaluation → Reflection (si surprise) → Décision.

---

4. Immunité Cognitive

- Détection d'anomalies (stateless + stateful).
- Vérification causale (vector clocks).
- Isolation (quarantine).
- Rollback → fork → patch.
- Attestation cryptographique.

---

5. Épigénétique Logicielle

- Markers : methyl, acetyl, lock, vibe, cost_multiplier.
- Promoteurs dépendants du contexte.
- Héritabilité paramétrable.

---

6. Mutation & Évolution Sécurisée

```
Function safe_evolve(genome, budget):
  variants = generate_variants(genome, budget)
  for v in variants:
    sandbox = deploy_in_sandbox(v)
    run_simulations(sandbox)
    v.score = evaluate_fitness_and_risk(sandbox)
    v.div_profile = compute_diversity(v)
  best = select_top(variants)
  if quorum(best) and diversity_ok(best): merge(best)
```

---

7. Gouvernance & Invariants

- **NonCorruption** : proofs + checksums.
- **NonMaleficence** : interdiction actions dommageables.
- **Transparency** : justification des mutations.
- **ResourceBound** : aucune expansion non autorisée.
- **Zones sacrées** : core_ethics.gene + audit.ledger.

---

8. Interopérabilité (Corrigée & Sécurisée)

8.1 CRDT : Op-based + Vector Clocks (corrigé)

```
Type LogEntry { payload, clock: VectorClock, sig }
Type CRDT_OpLog { entries: Set[LogEntry] }

Function crdt_append(log, entry): log.entries.add(entry)
Function crdt_merge(a, b): return a ∪ b
Function crdt_query(log): return sort_by_clock(log.entries)
```

8.2 Blockchain : Merkle-root Only (confidentialité + coût)

```
Function commit_chain(genome_id, log):
  root = merkle_root(log)
  tx = { genome: genome_id, root, timestamp: now(), sigs: multi_sig() }
  chain.submit(tx)
```

8.3 Kubernetes : WASM Runtime Sécurisé

```
Function deploy_gene(g):
  pod = {
    runtimeClass: "wasm",
    container: compile_to_wasm(g.codons),
    seccomp: strict, rbac: locked,
    resources: g.resources
  }
  k8s.apply(pod)
```

8.4 Attestation Anti-Sybil

```
Function attest_gene(g):
  m = hash(g.codons)
  return TPM.quote(m)
```

---

9. Diversité Cognitive (Corrigée & Adaptative)

```
Function diversity_metrics(genome):
  counts = count_niches(genome)
  entropy_shannon = -Σ p log p
  entropy_simpson = 1 - Σ p²
  return {shannon, simpson}

Function diversity_ok(genome):
```

```
  profile = diversity_metrics(genome)
  threshold = base_entropy * mission_factor()
  return profile.shannon >= threshold
```

---

10. Visualisation & Audit Temps Réel

- REST: `/genome/graph` (graph DTO)
- WS: `/genome/events` (événements live)
- Overlays : diversité, coûts, risques, zones immuables, forks.

```
Function build_graph(genome): return GraphDTO(nodes, edges, overlays)
```

---

11. Économie Cellulaire Anti-Abus (Corrigée)

Budgets, prix dynamiques, anti-spam

```
Function debit(gene, usage):
  unit = dynamic_pricing(load())
  cost = compute_cost(usage, unit)
  if acct.credits < cost: raise "INSUFFICIENT_CREDITS"
  acct.credits -= cost
```

- Attestation obligatoire pour recevoir crédits.
- Priorité aux gènes éthiques & immunitaires.
- Limites journalières contre le drain économique.

Redistribution (Pareto)

```
Function rebalance(genome):
  scores = rolling_fitness(genome)
  redistribution = pareto_optimize(scores)
  apply(redistribution)
```

---

12. Cycle Cognitif Final (Corrigé & Intégré)

```
Function tick(genome, env):
  signals = Perceptor.capture(env); debit(Perceptor)
  preds = Predictor.run(genome, signals); debit(Predictor)
  evals = Evaluator.score(preds, genome); debit(Evaluator)

  if Evaluator.surprise(evals):
    Reflector.meta(genome, evals); debit(Reflector)
```

```
decision = Arbiter.decide(evals); debit(Arbiter)

if not diversity_ok(genome): halt("DIVERSITY VIOLATION")

apply_decision(genome, decision)
crdt_append(ledger, decision)
commit_chain(genome.id, ledger)
publish(build_graph(genome))
```

---

13. Roadmap Réaliste

1. Prototype local Python (CRDT + graph + runtime WASM mock).
2. Consciousness Mesh minimal.
3. Sandbox evolution.
4. Tests sécurité.
5. Déploiement K8s/WASM.
6. UI interactive.

---

14. Résultat

Cette version corrige toutes les faiblesses identifiées, solidifie : - la causalité (CRDT) - la sécurité (WASM, attestations, merkle-root) - la diversité (Shannon + Simpson + niches min) - l'économie (anti-spam + plafonds + priorités) - la gouvernance (zones immuables + quorum + multi-sig)

Ton ADN numérique 5.0 est maintenant une architecture **mature, cohérente, exécutable et publiable**.