"I am building the **PlayNAC-KERNEL** with the **EarnedPath Engine (EP)**. Here are the specification documents. I need to design the data model for **GERP** and the Python classes for the EPIR-Q calculation logic. Please advise on the architecture and generate the core pseudocode."

1. Architectural Advice for PlayNAC-KERNEL & EarnedPath

The PlayNAC-KERNEL should be architected as a **TETRA-CERT Control Loop** with three primary, communicating services, all secured by GtC=ME attestations and governed by the P3 visibility policy.

Architectural Layer	Core Service / Component	Function	Technology Recommendation
Data Plane / Inputs	**ERI/ARI Data Ingestion**	Handles massive, real-time streaming data (Biochemical, Physiological, Behavioral Resonance) to calculate the human-environment coherence index.	Real-time Stream Processing (e.g., Apache Kafka / Google Pub/Sub), Time-Series Database (e.g., InfluxDB / Google Cloud Bigtable).
State Plane / Model	**GERP / BERC Graph Model**	The unified, authoritative model for all resources, their current state, and their efficiency (Ephemeralization). Updates based on actions and **NBERS** targets.	Graph Database (e.g., Neo4j / Google Cloud Knowledge Graph) to model **BERC** entities and relationships (resource nodes, flow paths, consumers).
Execution Plane / Output	**EarnedPath Engine (EP)**	The governance service that executes the merit-based economy. It consumes **NBERS** and **EPIR-Q** to calculate and disburse resources, rights, and benefits (VACATIONOMICS).	Python/Go Microservice architecture deployed on a scalable platform (e.g., Google Cloud Run/Kubernetes). Requires robust Transactional Database for EP disbursement ledgers.

Security/Audit	**GtC=ME Attestation Layer**	Cryptographic layer that wraps all transactions, linking them to physical reality via Energy/Matter attestations (E) and enforcing P3 access policies.	Separate Signing/Verification Microservice utilizing post-quantum hash functions, possibly managed by a Hardware Security Module (HSM).

Export to Sheets

---

2. GERP Data Model Design (Conceptual)

The **Generalized Ephemeralization Resource Proxy (GERP)** acts both as a metric and as the foundational model for the **Bio-Ecologic Resource Cyberspace (BERC)**. A **Graph Data Model** is the most suitable architecture to track resources, their connections, and their efficiency/velocity.

Core Entities (Nodes)

Entity Name	Description	Key Properties
ResourceNode	A fundamental unit of resource (e.g., 1 unit of clean water, 1 MWh of solar energy, 1 hour of labor).	`ResourceType`, `Quantity`, `Location_GERP_XYZ`, `Current_NBERS_Alignment`, `Entropy_Gradient`
SystemNode	A consuming or producing entity (e.g., a Smart City Zone, a Farm, a Manufacturing Facility).	`SystemType`, `TETRA_CERT_Status`, `Current_ARI_Score`, `Target_NBERS_Delta`, `Responsible_Entity_ID ($GtC=ME$)`
HumanAgent	An individual or collective participating entity (always linked to a GtC=ME identity).	`AgentID ($GtC=ME$ Key)`, `Current_EPIR_Q_Score`, `EarnedPath_Balance`, `Cognition_Resonance_Profile (C Key)`

Export to Sheets

Key Relationships (Edges)

Relationship Name	Source Node	Target Node	Edge Properties (Data)
CONSUMES	`SystemNode` or `HumanAgent`	`ResourceNode`	`QuantityConsumed`, `Timestamp`, `Efficiency_Metric (GERP)`, `EPIR_Q_of_Action`
PRODUCES	`SystemNode` or `HumanAgent`	`ResourceNode`	`QuantityProduced`, `Timestamp`, `Quality_Score`, `Sustainability_Factor`
IS_GOVERNED_BY	`ResourceNode` or `SystemNode`	`HumanAgent`	`Responsibility_Level`, `Policy_ID ($P^3$)`

Export to Sheets

---

3. EPIR-Q Calculation Logic (Python Pseudocode)

The **Emotional Personal IQ Real and Quantum (EPIR-Q)** component converts discrete choices and outcomes into a quantifiable rating, which then feeds into the NBERS calculation and the EarnedPath disbursement logic.

The core formula described in the ERES documents is:

$$\text{EPIR-Q} = (\sum P^3 \text{ Scores}) \times \prod (\text{Alignment/Scaling Factors})$$

The Python class should encapsulate the collection of P3 scores and the application of required scaling factors (e.g., Ecological, AI Alignment).

Python

```
# EPIR-Q Calculation Logic - Class Pseudocode

class EPIR_Q_Calculator:
    """
    Calculates the Emotional Personal IQ Real and Quantum (EPIR-Q) score
    for a proposed action or project, based on P^3 visibility and
    systemic alignment factors. This score is an input to the NBERS control loop.
    """
    def __init__(self):
        # Initialize default alignment factors
        # These factors adjust the score based on long-term systemic goals (e.g., 1000-Year Map)
        self.ai_alignment_factor = 1.0     # e.g., 1.2 for strong alignment
        self.ecological_factor = 1.0       # e.g., 1.5 for high sustainability
```

```
        self.equity_factor = 1.0          # e.g., 0.8 for poor equity outcome

    def calculate_p3_scores(self, project_data: dict) -> dict:
        """
        Simulates or retrieves the P3 scores (Personal, Private, Public)
        for a given project from the ERI/ARI and social data streams.
        Scores should be normalized (e.g., 0.0 to 1.0).
        """
        # In a real system, this would call a data service (ERI/ARI, social audit)
        scores = {
            'personal': project_data.get('personal_wellbeing_score', 0.0), # e.g., access, stress
reduction
            'private': project_data.get('ethical_procurement_score', 0.0),  # e.g., internal ethics,
supply chain audit
            'public': project_data.get('public_trust_score', 0.0)         # e.g., transparency, community
consensus
        }
        return scores

    def set_alignment_factors(self, ai_factor: float, eco_factor: float, eq_factor: float):
        """Allows setting the factors based on current NBERS targets/policy goals."""
        self.ai_alignment_factor = ai_factor
        self.ecological_factor = eco_factor
        self.equity_factor = eq_factor

    def calculate_epir_q(self, project_data: dict) -> float:
        """
        Calculates the final EPIR-Q score.
        Formula: (P_personal + P_private + P_public) * (Alignment_Factors product)
        """
        p3_scores = self.calculate_p3_scores(project_data)

        # Step 1: Sum the P^3 scores
        p3_sum = sum(p3_scores.values())

        # Step 2: Calculate the product of all alignment factors
        alignment_product = (self.ai_alignment_factor * self.ecological_factor * self.equity_factor)

        # Step 3: Apply the product to the sum
        epir_q_final = p3_sum * alignment_product

        return epir_q_final

# --- Example Usage (Mirroring the Bus Network Example from Docs) ---
```

```
# bus_network_project_data = {
#     'personal_wellbeing_score': 0.6,
#     'public_trust_score': 0.7,
#     'ethical_procurement_score': 0.4
# }

# calculator = EPIR_Q_Calculator()
# # Set factors based on the document example: AI Factor 1.2, Eco Factor 1.5, Equity Factor
1.0 (implied)
# calculator.set_alignment_factors(ai_factor=1.2, eco_factor=1.5, eq_factor=1.0)
# score = calculator.calculate_epir_q(bus_network_project_data)
# print(f"Calculated EPIR-Q Score: {score}") # Expected: (0.6 + 0.7 + 0.4) * 1.2 * 1.5
```