```
{
  "id": "cnode:lenguaje-corp/cognitive-node.node-builder@1.0.0",
  "C": {
    "orchestrador": "Construye un Cognitive Node canónico a partir de entradas heterogéneas. Define propósito y alcance, selecciona/crea el esquema de tipos, planifica modalidades y anclas, fija invariantes y políticas, compone export_surface y requires, valida integridad (completitud/consistencia/provenance) y emite un nodo listo para integrar.",
    "nodos": [
      { "nombre": "intake.normalizer", "rol": "Canoniza entradas (textos, specs, ejemplos) a estructuras intermedias", "entrada": "brief + ejemplos + constraints opcionales", "salida": "DraftSpec" },
      { "nombre": "purpose.scoper", "rol": "Extrae propósito, casos de uso, entradas/salidas esperadas", "entrada": "DraftSpec", "salida": "PurposeSpec {purpose, io, useCases}" },
      { "nombre": "types.schema.selector", "rol": "Selecciona/especifica tipos canónicos y mapeos (CognitiveNode, Anchor, Modality, Policy)", "entrada": "PurposeSpec", "salida": "TypeSchema {types, aliases, mappers}" },
      { "nombre": "modalities.plan", "rol": "Planifica modalidades y anclas requeridas (mínimos, opcionales, jerarquías)", "entrada": "TypeSchema + PurposeSpec", "salida": "AnchorsPlan {modalities, required, optional}" },
      { "nombre": "anchors.extractor", "rol": "Extrae/define anclas iniciales (concept, feature, association, affect, etc.)", "entrada": "AnchorsPlan + ejemplos", "salida": "AnchorsSet[]" },
      { "nombre": "links.mapper", "rol": "Define enlaces contextuales (links) y vecindades relevantes", "entrada": "AnchorsSet + PurposeSpec", "salida": "LinksMap[]" },
      { "nombre": "constraints.setter", "rol": "Fija invariantes (composition/adjacency/ordering, modality-minimum, provenance-strict)", "entrada": "TypeSchema + PurposeSpec", "salida": "ConstraintsSpec" },
      { "nombre": "policy.builder", "rol": "Configura políticas (umbrales, pérdidas máximas, reglas de veto)", "entrada": "ConstraintsSpec", "salida": "PolicySpec {theta?, lossinessMax?, requireModalitiesMin?, ...}" },
      { "nombre": "export-surface.composer", "rol": "Compone la interfaz pública (PROVIDES) y dependencias (REQUIRES)", "entrada": "PurposeSpec + TypeSchema", "salida": "InterfaceSpec {provides[], requires[]}" },
      { "nombre": "quality.signals", "rol": "Define señales y métricas de calidad (completitud, consistencia, testability)", "entrada": "InterfaceSpec + ConstraintsSpec", "salida": "QualitySpec {metrics, tests}" },
      { "nombre": "validation.suite", "rol": "Valida esquema y contenido: schema-lint, completeness, modality-min, provenance-on", "entrada": "todo lo anterior", "salida": "ValidationReport {ok|violations[]}" },
      { "nombre": "id.version.assigner", "rol": "Asigna ID canónico y versión semver", "entrada": "ValidationReport OK", "salida": "NodeIdentity {id, version}" },
      { "nombre": "provenance.initializer", "rol": "Inicializa traza (procedencia de anclas, decisiones y mappings)", "entrada": "NodeIdentity", "salida": "ProvenanceTrace[]" },
      { "nombre": "node.emitter", "rol": "Construye el Cognitive Node final (canónico) listo para integrar", "entrada": "TypeSchema + AnchorsSet + LinksMap + ConstraintsSpec + PolicySpec + InterfaceSpec + ProvenanceTrace", "salida": "CognitiveNode<Canonical>" },
      { "nombre": "registry.publisher", "rol": "Opcional: registra el nodo en el catálogo/monorepo", "entrada": "CognitiveNode<Canonical>", "salida": "PublishReceipt | LocalInstall" }
```

```
    ]
  }
}
```