```
{
  "contexto": {
    "titulo": "Instrucciones de Construcción — Pre-Procesar → Constructor de Estructuras (desde código)",
    "descripcion": "Guía operativa para generar un Cognitive Node a partir de un paquete de pre-proceso de código, asegurando que el resultado nazca con la forma canónica {contexto, nodes, metaforas}. Estas instrucciones especifican entradas, pasos, restricciones, validaciones y trazabilidad.",
    "principio": "El cómo determina el qué: la creación es inseparable de la estructura canónica.",
    "operador": "E (Estructuración determinista)",
    "formalizacion": "CognitiveNode := E(PreprocessedPackage) ⇒ {contexto, nodes, metaforas}",
    "invariants": ["evidence-only", "composition", "adjacency", "ordering", "provenance"]
  },

  "nodes": [
    {
      "id": "intake.preprocessed-package",
      "rol": "Ingesta del paquete",
      "entrada": "PreprocessedPackage = {Artifacts.ndjson, CodeManifest.json, ProvenanceTrace}",
      "procedimiento": [
        "Validar existencia y legibilidad de los tres artefactos.",
        "Rechazar si falta alguno o hay hash inconsistente (ProvenanceTrace)."
      ],
      "salida": "Fuente validada con índices a PARTS/RELS/ORDER"
    },

    {
      "id": "structure.mapper",
      "rol": "Mapeo a esqueleto canónico",
      "entrada": "Artifacts + Manifest",
      "procedimiento": [
        "Para cada file|chunk construir un objeto base con llaves: {contexto, nodes, metaforas}.",
        "contexto := {path, tipo, propósito inferido con evidencia, contratos (si existen), restricciones observadas}.",
        "nodes := descomponer en subunidades tipadas (p.ej., controller, service, dto, enum, file, script) con edges {composition|uses|depends} extraídos del Manifest.",
        "metaforas := inicializar con 0..n entradas; cada entrada debe referenciar evidencia (línea|chunk|path)."
      ],
      "salida": "SkeletonList[{contexto, nodes, metaforas}]"
    },

    {
      "id": "content.instantiator",
```

```
      "rol": "Relleno de contenido",
      "entrada": "SkeletonList",
      "procedimiento": [
        "Completar 'contexto' con: resumen objetivo (máx. 3 líneas), ubicación (path), rol en el sistema y límites conocidos.",
        "Completar 'nodes' con subnodos tipados y edges evidenciales; incluir 'why' breve por subnodo.",
        "Completar 'metaforas' con imágenes técnicas (p.ej., imprenta/molde/puerto/andamio) vinculadas a rasgos estructurales detectados."
      ],
      "salida": "StructuredElements[{contexto, nodes, metaforas}]"
    },

    {
      "id": "validator.coherence",
      "rol": "Validación de coherencia e invariantes",
      "entrada": "StructuredElements",
      "checks": [
        "EVIDENCE: todo enunciado debe tener referencia (path+range o chunkId).",
        "COMPOSITION/ADJACENCY/ORDER: no romper relaciones declaradas en CodeManifest.",
        "FORMA: las tres secciones {contexto, nodes, metaforas} deben existir y no estar vacías.",
        "UNICIDAD: IDs estables por elemento y subnodo (derivar de path+hash corto)."
      ],
      "salida": "ValidationReport{ok|violations[], coverage%}"
    },

    {
      "id": "graph.integrator",
      "rol": "Integración al grafo cognitivo",
      "entrada": "StructuredElements + ValidationReport(ok)",
      "procedimiento": [
        "Asignar namespace y ruta lógica del nodo (cnode:<namespace>/<name>@<version>).",
        "Publicar edges externos permitidos (solo los sustentados por evidencia).",
        "Registrar Provenance (decisiones, mapeos, versiones de herramienta)."
      ],
      "salida": "CognitiveNodePackage{contexto, nodes, metaforas, ids, edges, provenance}"
    },

    {
      "id": "orchestrator.preprocess-to-structure",
      "rol": "Orquestación del flujo",
      "describe_how_nodes_interact": [
        "1) intake.preprocessed-package valida fuentes y hashes.",
        "2) structure.mapper crea el esqueleto canónico por file|chunk.",
        "3) content.instantiator rellena contexto/nodos/metáforas con evidencia.",
```

```
        "4) validator.coherence aplica invariantes y forma canónica.",
        "5) graph.integrator emite el Cognitive Node listo para el grafo."
      ],
      "acceptance": [
        "El paquete final contiene SOLO elementos en forma {contexto, nodes, metaforas}.",
        "Ninguna afirmación sin evidencia; relaciones preservan PARTS/RELS/ORDER.",
        "Provenance presente (hash, toolVersion, timestamps)."
      ]
    }
  ],

  "metaforas": [
    { "orden": 1, "texto": "Manual de encuadernación: cada capítulo (archivo) sale ya con portada (contexto), índice (nodes) y glosario visual (metáforas)." },
    { "orden": 2, "texto": "Moldeo por inyección: la materia (chunks) se inyecta en la cavidad canónica y solidifica con la misma geometría cognitiva." },
    { "orden": 3, "texto": "Puerto con estándar ISPS: cualquier barco atraca, descarga con el mismo protocolo y queda trazado en bitácora (provenance)." }
  ]
}
```