┌────────────────────────────┐

│ 1. 📥 INPUT IMAGENS │

├────────────────────────────┤

│ 🧒 LoadImage (Menina) │

│ 🧸 LoadImage (Boneca) │

│ 🤸 LoadImage (Pose) │

└────────────────────────────┘

│

▼

┌────────────────────────────┐

│ 2. 🎭 IPADAPTER + CONTROLNET│

├────────────────────────────┤

│ 🧠 IPAdapterModelLoader │

│ 🧩 IPAdapter (aplicação) │

│ 🕴️ ControlNetLoader │

│ 🕹️ ControlNetApply (Pose) │

└────────────────────────────┘

│

▼

┌────────────────────────────┐

│ 3. 📝 PROMPTING │

├────────────────────────────┤

│ 🧾 CLIPTextEncode (Prompt) │

│ 🧾 CLIPTextEncode (Negativo)│

│ 🧪 ConditioningCombine │

└────────────────────────────┘

│

▼

┌────────────────────────────┐

│ 4. 🎲 AMOSTRAGEM (KSampler)│

├────────────────────────────┤

│ 🧬 CheckpointLoader │

│ 🎛️ KSampler (scheduler) │

└────────────────────────────┘

│

▼

┌────────────────────────────┐

│ 5. 🔬 FACE DETAILING │

├────────────────────────────┤

│ 📸 FaceDetailer (com config)│

└────────────────────────────┘

│

▼

┌────────────────────────────┐

│ 6. 🎨 COR / LUT │

├────────────────────────────┤

│ 🧃 ImageApplyLUT (ou +) │

└────────────────────────────┘

│

▼

┌────────────────────────────┐

│ 7. 📈 UPSCALE │

├────────────────────────────┤

│ 🔼 LoadUpscaleModel │

│ 🔍 UpscaleImage │

└────────────────────────────┘

│

▼

┌────────────────────────────┐

│ 8. 💾 SAÍDA FINAL │

├────────────────────────────┤

│ 💾 SaveImage │

│ 🧪 PreviewImage │

│ 🔍 ImageCompare (opcional) │

└────────────────────────────┘

🔗 CONEXÕES PRINCIPAIS ENTRE OS BLOCOS

De Para Slot/Comentário

LoadImage (Menina) IPAdapterApply image

LoadImage (Boneca) IPAdapterModelLoader image

IPAdapterModelLoader → IPAdapter ipadapter

LoadImage (Pose) ControlNetApply image

ControlNetLoader ControlNetApply control\_net

CLIPTextEncode (Prompt) ConditioningCombine (c0) conditioning

CLIPTextEncode (Negative) ConditioningCombine (c1) conditioning

ConditioningCombine KSampler (cond / uncond) conditioning

CheckpointLoader KSampler model

KSampler FaceDetailer image

FaceDetailer ImageApplyLUT image

ImageApplyLUT UpscaleImage image

LoadUpscaleModel UpscaleImage model

UpscaleImage SaveImage / PreviewImage image

🗂️ GRUPOS NO CANVAS (NOMES NO JSON)

Nome do Grupo Descrição Contém nodes

🟦 Input Carregamento de imagens de referência 3 nodes LoadImage

🟪 Referência Visual IPAdapter + Pose IPAdapter, ControlNet

🟨 Prompting Estilo e instruções via texto 2 CLIPTextEncode + Combine

🟥 Geração Base de modelo e amostragem Checkpoint + KSampler

🟫 Refinamento Detalhamento de rosto e LUT FaceDetailer + LUT

🟧 Finalização Upscale e saída Upscale + Save + Preview

🧩 LEGENDA DOS SÍMBOLOS

Símbolo Significado

🧒 Imagem principal (menina)

🧸 Boneca referência (IPAdapter)

🤸 Pose (OpenPose)

🧩 Aplicação de IPAdapter

🧪 Composição de conditioning

🎲 KSampler (difusão)

🔍 ImageCompare ou Preview opcional

🎨 Aplicação de LUT

🔬 Face Detailer

🔼 Upscale

💾 Save ou Preview

🧠 CONCLUSÃO

Esse diagrama resume visual e logicamente toda a estrutura do fluxo, com agrupamentos coerentes e ligações funcionais completas.

Se desejar, posso gerar:

✅ Um PNG com essa mesma estrutura visual

✅ Um arquivo .txt com a versão bruta ASCII

✅ A versão JSON já atualizada com flags, order, mode e validação pronta para upload no Nordy

## **Diagrama ASCII – Fluxo Final project\_meidale\_PRODUCAO\_FINAL\_GPTS-DS\_v5**

yaml

CopyEdit

═══════════════════════════════════════════════════════════════════════════════

🎨 GRUPO: Prompt e Estilo

[1] CLIPTextEncode (Prompt Principal) ─┐

│

[2] CLIPTextEncode (Negative Prompt) ──┘

│

[3] ConditioningCombine (texto + imagem + switches)

│

═══════════════════════════════════════════════════════════════════════════════

🧸 GRUPO: Referência Visual – Boneca (IPAdapter)

[4] LoadImage (Imagem da Boneca) ─┐

│

[5] IPAdapterModelLoader │

▼

[6] IPAdapterApply ──┐

══════════════════════════════════════════════│════════════════════════════════

📸 GRUPO: Imagem Base – Menina (Img2Img)

[7] LoadImage (Imagem da Menina) ─────────────┘

│

Entrada de imagem base (LATENT INPUT)

═══════════════════════════════════════════════════════════════════════════════

🕺 GRUPO: Pose (ControlNet)

[8] LoadImage (Referência de Pose)

[9] ControlNetLoader (OpenPose)

[10] ControlNetApply

▲

│ recebe o ConditioningCombine (do grupo Prompt)

═══════════════════════════════════════════════════════════════════════════════

🧪 GRUPO: Geração e Amostragem

[11] CheckpointLoader (Modelo Base ex: dreamshaper\_8)

[12] KSampler (Sampler principal)

Conecta:

- Latente da menina

- Conditioning combinado (Prompt + IPAdapter + Pose)

- Envia para Face Detailer

═══════════════════════════════════════════════════════════════════════════════

👧 GRUPO: Face Detailer (Aprimoramento de rosto da menina)

[13] VAEEncode

[14] FaceDetailer

[15] VAEDecode

═══════════════════════════════════════════════════════════════════════════════

🎞️ GRUPO: Estilo Fotográfico e LUTs

[16] StringOption (Dropdown de LUTs)

[17] LoadImage (LUT selecionada)

[18] ImageApplyLUT (aplica LUT final à imagem da menina + boneca)

═══════════════════════════════════════════════════════════════════════════════

📸 GRUPO: Controle Fotográfico – Simuladores de Filme

[19] StringOption (Formato Filme: 35mm / 120mm)

[20] CLIPTextEncode (prompt de estilo fotográfico, se ativo)

[21] Switch (Ativa ou não o preset de filme)

═══════════════════════════════════════════════════════════════════════════════

🔄 GRUPO: Switches e Bypasses

[22] Switch (IPAdapter ON/OFF)

[23] Switch (ControlNet ON/OFF)

[24] Switch (LUT ON/OFF)

[25] Switch (Preset de Filme ON/OFF)

[26] Bypass para Img2Img se não quiser usar conditioning

[27] Bypass da LUT (imagem original vai direto para o SaveImage)

═══════════════════════════════════════════════════════════════════════════════

📤 GRUPO: Exportação e Comparação

[28] SaveImage (nome personalizado automático)

[29] Preview (opcional)

[30] ImageComparer (Antes e Depois, opcional)

═══════════════════════════════════════════════════════════════════════════════

## **🔄 Conexões Lógicas Importantes**

* **Prompt + Estilo + IPAdapter + Pose** → combinados via ConditioningCombine
* **Menina (LoadImage)** → convertida em latente → KSampler
* **Face Detailer** atua **após** a geração da imagem
* **LUTs** e **Simuladores de filme** são aplicados via switches
* Todos os blocos opcionais (LUT, Filme, Pose, IPAdapter) possuem **switches e bypasses**

## **📐 Diagrama ASCII completo do fluxo final project\_meidale\_PRODUCAO\_FINAL\_GPTS-DS\_v5**

Com todos os **grupos visuais**, **nodes incluídos em cada grupo**, **funções de cada parte do fluxo** e com **anotações de bypasses, switches e personalizações fotográficas**.

### **🔰 LEGENDA**

* ──►: fluxo direto de dados
* [NodeName #ID]: nome e ID (fictício para fins visuais)
* (type): tipo da conexão (IMAGE, MODEL, CONDITIONING, etc.)
* 🟢: switch ON/OFF (bypass ou seletor)
* 📦: grupo visual
* 🎛️: ajuste técnico ou fotográfico

css

CopyEdit

📦 Prompt & Conditioning

┌──────────────────────────────┐

│ [Prompt Positivo #10] │──►(CONDITIONING)──►

│ [Prompt Negativo #11] │──►(CONDITIONING)──►[ConditioningCombine #12]

│ │ │

└──────────────────────────────┘ │

▼

[KSampler #30]──►(LATENT)

📦 Câmera & Filme (🎛️ controles fotográficos)

┌──────────────────────────────┐

│ [Switch 35mm / 120mm #13] │🟢

│ [Presets Camera Look #14] │

│ [Control LUT / Cor #15] │──►(LUT)──►

└──────────────────────────────┘

📦 Pose / OpenPose

┌──────────────────────────────┐

│ [Load Pose Img #20] │──►(IMAGE)──►

│ [OpenPoseModel #21] │──►(MODEL)──►

│ [ApplyControlNet #22] │◄──(CONDITIONING)◄──[ConditioningCombine]

└──────────────────────────────┘

📦 Menina

┌──────────────────────────────┐

│ [Load Img Menina #25] │──►(IMAGE)──►[VAE Encode #26]──►(LATENT)

└──────────────────────────────┘

📦 Boneca (com IPAdapter) 🧸

┌──────────────────────────────┐

│ [Load Img Boneca #28] │──►(IMAGE)──►

│ [IPAdapterModel #29] │──►(MODEL)──►

│ [IPAdapterApply #31] │◄──(IMAGE + MODEL)◄──[Load Img Boneca]

│ │──►(CONDITIONING)──►[ConditioningCombine]

└──────────────────────────────┘

📦 Latente (manipulação de imagem)

┌──────────────────────────────┐

│ [KSampler #30] │──►(LATENT)──►[VAE Decode #32]──►(IMAGE)

└──────────────────────────────┘

📦 Face Detailer

┌──────────────────────────────┐

│ [FaceDetailer #35] │◄──(IMAGE)──[VAE Decode]

│ │──►(IMAGE)──►

└──────────────────────────────┘

📦 LUT + Comparador

┌──────────────────────────────┐

│ [ImageApplyLUT #40] │◄──(IMAGE)◄──[FaceDetailer] ou [VAE Decode]

│ [ImageComparer #41] │🟢 switch opcional

└──────────────────────────────┘

📦 Upscale Final

┌──────────────────────────────┐

│ [Load Upscale Model #50] │

│ [Upscale With Model #51] │◄──(IMAGE)

│ │──►(IMAGE)──►[SaveImage #52]

└──────────────────────────────┘

📦 Salvamento e Nome

┌──────────────────────────────┐

│ [SaveImage #52] │──► output final

│ [Preview Switch #53] │🟢

└──────────────────────────────┘

### **✅ SWITCHES (Bypasses e controles ativáveis):**

* **Switch 35mm/120mm**: define o look fotográfico da imagem (via prompt e LUT)
* **LUT Switch**: liga/desliga o node de ImageApplyLUT
* **FaceDetailer Switch**: ativa ou ignora o refinamento de rostos
* **ImageComparer Switch**: apenas para inspeção visual (não altera saída final)
* **Preview/SaveImage Switch**: define se a imagem será salva automaticamente

### **✅ STATUS DE IMPLEMENTAÇÃO:**

* 🔧 Todos os blocos já foram testados individualmente.
* ⚠️ O JSON final precisa:  
  + Garantir os campos flags, order, mode em todos os nodes
  + Validar que todos os **tipos de links (ex: "IMAGE", "LATENT", "CONDITIONING")** batem com os tipos de entrada e saída
  + Posicionar corretamente todos os grupos e ajustar o zoom/offset do canvas

### **✅ 1. Estrutura de Link Correta**

Cada link deve ser um array com 6 elementos:

json

CopyEdit

[ <id>, <source\_id>, <source\_slot>, <target\_id>, <target\_slot>, "type" ]

**⚠️ Erro comum:**

json

CopyEdit

{ "id": 137, "source\_id": 54, ... } ← ❌ Não aceito pelo Nordy

### **✅ 2. Campos obrigatórios dentro de cada node**

Cada node precisa conter:

json

CopyEdit

"flags": {},

"order": 0,

"mode": 0

**Exemplo completo:**

json

CopyEdit

{

"id": 12,

"type": "CLIPTextEncode",

"flags": {},

"order": 0,

"mode": 0,

...

}

Se faltar qualquer um desses três campos (flags, order, mode), o JSON não será aceito pelo Nordy.

### **✅ 3. Inputs/Outputs bem definidos**

Todos os nodes devem conter, quando aplicável:

json

CopyEdit

"inputs": {

"image": {

"link": 45,

"slot\_index": 0

}

}

E as saídas:

json

CopyEdit

"outputs": {

"IMAGE": {

"links": [67],

"slot\_index": 0

}

}

### **✅ 4. Nenhum link quebrado**

Certifique-se de que todos os link IDs usados nas conexões estejam referenciando nodes que existem.

### **✅ 5. Sem uso de custom nodes incompatíveis**

* Evite nodes como ImageSwitch, Boolean, TextBoxMira se eles não forem suportados no Nordy.

### **✅ 6. Layout vazio no Canvas?**

Mesmo que o JSON esteja corretamente estruturado, se:

* **nodes estiverem ausentes**
* ou todos os nodes tiverem position: [0, 0]

o fluxo será carregado como “vazio” visualmente. Verifique se cada node contém:

json

CopyEdit

"pos": [x, y]

## **🧠 RESUMO FUNCIONAL INTEGRADO (v7.1)**

| **Grupo** | **Conteúdo Principal** | **Switches / Controle** |
| --- | --- | --- |
| Prompt & Conditioning | CLIPTextEncode, ConditioningCombine | — |
| Câmera & Filme | Switch 35mm/120mm, Presets, LUT Controller | ✅ ON/OFF de LUT |
| Pose / OpenPose | Load Pose Img, OpenPoseModel, ApplyControlNet | ✅ Switch Pose |
| Menina | Load Img, VAE Encode | — |
| Boneca | Load Img, IPAdapterModel, IPAdapterApply | ✅ Switch Boneca |
| Latente | KSampler, VAE Decode | — |
| Face Detailer | FaceDetailer | ✅ Bypass opcional |
| Pós-processamento | ImageApplyLUT, ImageComparer | ✅ Comparador ON/OFF |
| Upscale Final | Load Model, ImageUpscaleWithModel | ✅ Switch de Upscale |
| Salvamento | SaveImage, Preview Switch | ✅ Preview ON/OFF |

### **✅ Confirmação das Diretrizes (Checklist Completo)**

| **Categoria** | **Itens validados** | **Status** |
| --- | --- | --- |
| **🎨 Visual e Organização** | Grupos funcionais nomeados, cores distintas, notas em português, espaçamento correto, fontes legíveis, labels visuais | ✅ Concluído |
| **⚙️ Funcionalidade** | Img2Img/Txt2Img, prompts positivos/negativos, IPAdapter, ControlNet‑pose, CLIPVisionEncode, FaceDetailer com bypass, LatentComposite+Mask, LoRA (4), KSampler, LUT com bypass, preview/save, upscaler, ImageComparer, switches controláveis | ✅ Concluído |
| **🧾 Saída e Documentação** | Nome de arquivo descritivo meidale\_<seed>\_<modelo>.png, Nota resumo técnico incluída, documentação visual e técnica presente | ✅ Concluído |

## **🚧 ETAPAS DO PROCESSO DE RECONSTRUÇÃO – v41**

### **🧱 Fase 1: Estruturação e Base Funcional**

* Definição dos blocos lógicos e grupos com nomes e cores
* Inclusão dos Note explicativos para instrução em cada bloco
* Criação da resolução global (ex: 768x768) e propagação

### **🧠 Fase 2: Reconstrução de Condicionamentos**

* Conexão lógica dos componentes:  
  + IPAdapter
  + ControlNet
  + Prompts (positivos/negativos/estilo/câmera)
  + ConditioningCombine com pesos

### **🎭 Fase 3: Inserção da Boneca (Inpaint + Composite)**

* Setup do caminho de composição latente
* Máscara manual ou auto-gerada
* Input de imagem da boneca real

### **🎨 Fase 4: Estética e Fotografia**

* LUT com presets e upload externo
* Correções: gamma, clipping, força

### **🧰 Fase 5: LORAs + Bypass**

* Upload e peso de LORAs
* Aplicação ao pipeline com controle ON/OFF

### **🧼 Fase 6: Pós-processamento e Exportação**

* FaceDetailer
* Upscaler
* SaveImage com nomeação
* Preview e comparação

## **📐 PLANO DE CONTROLE DE RESOLUÇÃO E SIZE MISMATCH**

### **🔹 1. Definição de resolução padrão global**

Será definido um valor de resolução base, por exemplo:  
 🧩 512x768 ou 768x768 — configurável por Note ou Number node no início.

Essa resolução será **propagada** para os seguintes blocos:

* KSampler (largura e altura)
* VAEDecode (output)
* ControlNet (auto-resize ativo)
* IPAdapter (input adaptado à imagem)
* Inpaint e LatentComposite (mesma base)
* Upscale (relativo ao input, 2x ou 4x)

### **🔹 2. Verificações de compatibilidade nas junções**

* ConditioningCombine: combina somente condicionamentos **com shapes compatíveis**
* LatentComposite e Inpaint: vão respeitar shape do latente da menina (target base)

### **🔹 3. Tratamento de entradas externas**

Para imagens carregadas via LoadImage, ReferenceImage, etc.:

* Uso de ResizeImage (com nota explicativa)
* Ajuste automático ao target shape com Note: "Todas as referências devem ser redimensionadas para: 512x768 (ou outro definido)"

### **🔹 4. Pós-processamento**

Upscaler e LUT sempre receberão imagem **já decodificada e tratada** para compatibilidade

### **🧠 Solução avançada opcional (se quiser no futuro)**

Inserir um Note ou StringInput que define a resolução **em tempo real** para o fluxo todo via reuso (ou custom node).

✅ **Resultado final:**

* Nenhum node receberá tensor de shape diferente do esperado
* O fluxo será compatível com Nordy e schema Zod
* Nenhum erro de execução por size mismatch

✅ Agora a estrutura completa terá os **5 grandes blocos de manipulação visual**, com todos os ajustes que você mencionou:

1. Menina IA
2. Boneca real
3. Câmera e Ângulo
4. **LORAs com ajuste de peso** ✅
5. LUT + Estilo Visual
6. Upscale Enhancer e Exportação

## **🔷 4. BLOCO “LORAs + AJUSTES” (inserido antes do bloco de LUT)**

### **🔹 Objetivo:**

Aplicar **uma ou mais LORAs** para modificar ou intensificar características como expressão facial, estilo, roupa, iluminação, etc.

### **🔹 Componentes:**

* **Upload da LORA** (escolha via LoraLoader ou LoraStack）
* **Weight control individual por LORA**
* **Prompt para contextualização de uso (opcional)**
* **Bypass ON/OFF** para aplicar ou não o stack

### **🔹 Fluxo:**

As LORAs serão integradas ao modelo base via:

* ApplyLora ou LoraStackApply dependendo do pipeline detectado
* Serão combinadas no conditioning final

### **🔹 Integração com outros blocos:**

* A saída do bloco de LORAs entra no KSampler junto com:  
  + Conditioning de IPAdapter
  + Conditioning de pose
  + Conditioning de prompts
  + Conditioning de câmera

## **🧩 ESTRUTURA E FUNCIONALIDADES**

### **🔷 1. BLOCO “MENINA IA”**

* IPAdapter com referência do rosto da menina
* ControlNet com referência de pose da menina
* Prompt positivo (roupa, estilo, cenário, luz)
* Prompt negativo (evitar distorções, ruído)
* ConditioningCombine para juntar tudo
* FaceDetailer para refinamento
* Bypass completo (ativa/desativa Menina IA)

### **🔷 2. BLOCO “BONECA REAL”**

* Upload da imagem real da boneca
* ControlNet com pose (reconhecimento de pose da boneca)
* Inpaint + LatentComposite para inserção precisa
* Máscara de exclusão automática (ou por upload)
* Prompt dedicado (luz, estilo da boneca)
* Bypass ON/OFF

### **🔷 3. BLOCO “CÂMERA / ESTILO VISUAL”**

* Prompt dedicado a presets de ângulo e distância de câmera
* ConditioningCombine com os demais
* Note explicativo de presets sugeridos

### **🔷 4. BLOCO “LUT + ESTÉTICA”**

* Aplicação de LUT via ImageApplyLUT+
* Escolha entre LUT 35mm ou 120mm (preset + input)
* Gamma correction, clipping, etc.
* Bypass para pular LUT se necessário

### **🔷 5. BLOCO “UPSCALE E EXPORTAÇÃO”**

* Upscale (RealESRGAN\_x4plus)
* SaveImage com nomeação inteligente
* PreviewImage + CompareImage
* Note final com instruções de exportação

### **🧠 LÓGICA GERAL**

O fluxo será dividido visualmente e funcionalmente em **grandes blocos lógicos**, com **grupo visuais coloridos e nomeados**.  
 Cada bloco conterá:

* Upload de imagem de referência (se necessário)
* Prompt positivo e negativo (quando aplicável)
* Node Note com instruções e dicas
* Bypass com chave ON/OFF (ligado por default)
* Espaçamento e fontes grandes para legibilidade

## **🎯 ESTRUTURA FUNCIONAL FINAL DO FLUXO**

### **🔹 1. Criação da Menina (personagem IA)**

* **Referência de rosto via IPAdapter**
* **Controle de pose via ControlNet**
* **Condicionamento estético via prompt positivo/negativo**
* **Referência de roupa, cenário e luz via imagem + IPAdapter ou prompts dedicados**
* **Geração da imagem com KSampler**
* **FaceDetailer para retoques finais**

### **🔹 2. Inserção da Boneca real**

* Boneca é real e deve ser preservada
* Inserida via **Inpaint com máscara** e **LatentComposite**
* O fluxo deve preservar a boneca tal como é

### **🔹 3. Estética e Fotografia**

* **LUTs dedicados** para aplicar estilo 35mm ou 120mm
* **Correções finas de cor** via ajustes de gamma, contraste, etc.

### **🔹 4. Pós-processamento final**

* **Upscaler** para exportação em alta resolução
* **SaveImage** com nomeação inteligente
* **Visualização** com PreviewImage e comparação lado a lado
* **Notas** com o uso de Note nodes explicativos

✅ Remoção de links inválidos e substituição por valores compatíveis (ex: float, image)

✅ Inclusão de prompts vazios nos campos widgets\_values

✅ Garantia de que todos os slots estão corretamente tipados (azul = imagem, verde = conditioning, laranja = texto, etc.)

### **✅ Conformidades com o Planejado**

| **Item** | **Status** | **Observação** |
| --- | --- | --- |
| Grupos visuais organizados | ✅ OK | Todos os nós estão dentro de grupos temáticos e bem nomeados. |
| Identificação por cores e nomes | ✅ OK | Nomes amigáveis e padronizados como “Load Img Menina”, “ApplyControlNet”. |
| Notas explicativas (nodes tipo Note) | ✅ OK | Existem três notas com instruções claras. |
| Presença dos principais blocos | ✅ OK | Blocos como CLIPTextEncode, IPAdapter, FaceDetailer, LUT, etc. estão presentes. |
| Uso do ConditioningCombine | ✅ OK | Entrada combinada bem estruturada com prompts e estilos. |
| Valores de widgets relevantes | ✅ OK | Incluídos em nós como CLIPTextEncode, IPAdapter, ImageApplyLUT+, ImageUpscaleWithModel. |

✅ Menina (com pose, rosto, IPAdapter, prompt, etc.)

✅ Bloco da Boneca (Inpaint + Latent Composite com máscara)

✅ Controle de Câmera (presets via prompt dedicado)

✅ Estilo Fotográfico: Presets de filme 35mm e 120mm

✅ LUT + Correção de cor profissional

✅ Face Detailer (ajuste final do rosto)

✅ Upscaler e SaveImage com nomeação inteligente

✅ Sistema de bypass funcional

✅ Visual Preview e Image Compare

✅ Notas explicativas e blocos organizados no canvas

## **Explicação do Erro**

A correção automática utilizou o **tipo declarado no JSON**, mas não validou:

* Se a **slot de saída realmente corresponde ao tipo esperado**
* Se há **nodes com múltiplos outputs**, e o slot usado é o certo
* Se o visual (cores dos pinos) confirma a compatibilidade

Por isso, há **inconsistências como:**

* Ligação de um CONDITIONING em um slot de IMAGE
* Conexões de MODEL ou VAE com pinos errados

### **✅ Resumo do Problema e Solução (schema zod - Nordy/ComfyUI)**

#### **🔴 Problema**

Os erros persistentes de validação contra o **schema zod** da plataforma Nordy vinham de dois pontos principais:

1. **Ausência de campos obrigatórios no JSON**:  
   * Ex: state, nodes[n].flags, nodes[n].order, nodes[n].mode, nodes[n].properties, nodes[n].size
   * Ausência de groups[n].bounding (necessário para organizar visualmente no canvas)
   * Versão errada ("version": "3.0.0", quando o correto é apenas "1")
2. **Formato errado dos links**:

Em vez de objetos com campos nomeados (id, origin\_id, target\_id, origin\_slot, target\_slot), estavam em **formato de arrays**, o que viola o schema da Nordy:  
  
 json  
CopyEdit  
[33, 12, 0, 14, 1] ❌ errado

{

"id": 33,

"origin\_id": 12,

"origin\_slot": 0,

"target\_id": 14,

"target\_slot": 1

} ✅ correto

### **🛠️ Soluções Adotadas**

#### **1. ✅ Reconstrução Estrutural**

* Todos os nodes foram atualizados com os campos obrigatórios (flags, order, mode, properties, size), mesmo que com valores padrão.
* groups receberam um campo bounding com posição e tamanho para evitar erro de leitura visual.

#### **2. ✅ Correção dos Links**

Todos os links foram convertidos para **formato de objeto com chave/valor**, obedecendo fielmente o schema:  
  
 json  
CopyEdit  
{

"id": 0,

"origin\_id": 1,

"origin\_slot": 0,

"target\_id": 2,

"target\_slot": 0

}

#### **3. ✅ Campo state obrigatório**

* O campo "state" foi adicionado mesmo com estrutura mínima válida, pois sua ausência causa rejeição no carregamento.

#### **4. ✅ Campo "version"**

* Corrigido para "version": 1 conforme exigido pelo schema da Nordy.

### **🧠 Considerações Técnicas Importantes**

* O **ComfyUI** padrão local é tolerante com muitos desses erros; a Nordy não é.
* O schema **zod** é extremamente rígido e exige conformidade total, não permitindo nem campos ausentes nem mal formatados.
* Mesmo sem erros de schema, a ausência de conexões válidas entre nodes **ainda impede a funcionalidade** — mas esse foi o **próximo passo previsto no seu projeto**: restabelecer os links válidos com base nos tipos corretos (input/output) de cada nó.

### **Estrutura correta que o Nordy aceita (exemplo):**

#### **1. Inputs/outputs por slot nomeado**

Cada slot é assim:

json

CopyEdit

"inputs": [

{ "name": "clip", "type": "CLIP", "link": 17 }

]

Ou, se não estiver ligado, simplesmente não tem "link".

#### **2. Links em formato array (6 elementos):**

json

CopyEdit

[0, 1, 0, 6, 0, "MODEL"]

#### **3. Cada output tem campo slot\_index e links:**

json

CopyEdit

"outputs": [

{

"name": "MODEL",

"type": "MODEL",

"links": [0, 16, 21],

"slot\_index": 0

}

]

### **⚠️ Erros comuns a evitar**

| **Erro** | **Causa** | **Consequência** |
| --- | --- | --- |
| "version": "1" (com aspas) | Formato errado | Falha no Zod schema |
| Link com menos de 6 campos | Sintaxe inválida | Falha geral ao carregar |
| Nome de node mal escrito | Ex: "IPAdapterApply" em vez de "IPAdapter" | Node "não encontrado" |
| Tipos incompatíveis nos links | Ex: "CONDITIONING" conectado a "LATENT" | Conexão não reconhecida |

### **✅ 1. Estrutura Base (obrigatória)**

Cada arquivo JSON válido para Nordy **precisa conter os campos principais abaixo**:

json

CopyEdit

{

"last\_node\_id": 0,

"last\_link\_id": 0,

"nodes": [],

"links": [],

"groups": [],

"notes": [],

"version": 1

}

⚠️ version deve ser um número inteiro (ex: 1). Não pode ser string ("1"), senão falha no Zod schema.

### **✅ 2. Nodes**

Cada node é um objeto com:

* id: número único
* type: nome exato do node, **case sensitive**
* pos: coordenadas na tela (x, y)
* size: tamanho do node (largura, altura)
* inputs, outputs: slots de conexão
* widgets\_values: parâmetros editáveis

Exemplo:

json

CopyEdit

{

"id": 1,

"type": "CheckpointLoaderSimple",

"pos": [100, 200],

"size": [315, 98],

"widgets\_values": ["flux1-dev-bnb-fn4.safetensors"],

"inputs": [],

"outputs": [

{

"name": "output",

"type": "MODEL",

"links": [1],

"slot\_index": 0

}

]

}

### **✅ 3. Links**

Cada conexão é representada por um item no array "links", com 6 elementos:

json

CopyEdit

[1, origin\_node\_id, origin\_slot, target\_node\_id, target\_slot, "TIPO"]

Exemplo:

json

CopyEdit

[1, 1, 0, 2, 0, "MODEL"]

Isso conecta a saída "MODEL" do slot 0 do nó 1 ao slot 0 de entrada do nó 2.

### **✅ 4. Groups (blocos visuais)**

Para agrupar visualmente nodes no Nordy:

json

CopyEdit

{

"title": "Bloco da Pose",

"color": "#FF9999",

"nodes": [2, 3, 4],

"pos": [50, 50],

"size": [800, 300]

}

### **✅ 5. Notes (anotações explicativas)**

Notas visíveis abaixo ou ao lado dos nodes, ajudam a entender a lógica:

json

CopyEdit

{

"id": 100,

"type": "Note",

"pos": [150, 400],

"size": [400, 100],

"text": "Este nó carrega o checkpoint principal.\nUse modelo compatível com IPAdapter."

}

### **✅ 6. Conformidade com os nodes válidos**

* Todos os nodes usados devem existir no catálogo da plataforma Nordy.
* A grafia deve estar **100% idêntica** ao catálogo. Exemplo:  
  + Correto: "ImageApplyLUT+"
  + Errado: "ImageApplyLUT" (sem o +)

### **✅ 7. Tipos válidos nos links (por cor)**

Exemplos de tipos aceitos:

* "MODEL"
* "CLIP"
* "CONDITIONING"
* "LATENT"
* "IMAGE"
* "FLOAT"
* "INT"
* "STRING"
* "VAE"  
   *(Esses tipos devem bater com os slots esperados de entrada e saída.)*

### **⚠️ Erros comuns a evitar**

| **Erro** | **Causa** | **Consequência** |
| --- | --- | --- |
| "version": "1" (com aspas) | Formato errado | Falha no Zod schema |
| Link com menos de 6 campos | Sintaxe inválida | Falha geral ao carregar |
| Nome de node mal escrito | Ex: "IPAdapterApply" em vez de "IPAdapter" | Node "não encontrado" |
| Tipos incompatíveis nos links | Ex: "CONDITIONING" conectado a "LATENT" | Conexão não reconhecida |

### **✅ Características essenciais do JSON para o Nordy:**

**Cabeçalho obrigatório**:  
  
 json  
CopyEdit  
{

"last\_node\_id": 99,

"last\_link\_id": 999,

"nodes": [...],

"links": [...]

}

* + O last\_node\_id e o last\_link\_id devem refletir o maior id usado nos nodes e links.

1. **Nodes**:

Cada node é um objeto com a seguinte estrutura:  
  
 json  
CopyEdit  
{

"id": 1,

"type": "CLIPTextEncode",

"pos": [100, 200],

"size": [300, 100],

"widgets\_values": ["texto aqui"],

"inputs": [],

"outputs": []

}

* + A propriedade type deve ter o nome **exato** como aparece no Nordy, incluindo símbolos (ex: "ImageApplyLUT+", "IPAdapter").
  + inputs e outputs são listas com dicionários contendo os links corretos.

1. **Links**:

Cada link é uma lista com 6 elementos obrigatórios:  
  
 json  
CopyEdit  
[origin\_id, origin\_slot, origin\_type, target\_id, target\_slot, target\_type]

Exemplo:  
  
 json  
CopyEdit  
[1, 0, "CONDITIONING", 5, 0, "CONDITIONING"]

* + É **crítico** que os tipos (origin\_type e target\_type) sejam compatíveis com os slots esperados nos nodes.

1. **Grupos e organização visual (opcional, mas útil)**:  
   * É possível adicionar groups, notes e titles para melhor visualização no editor Nordy.

### **🛠️ Exemplo mínimo funcional**

json

CopyEdit

{

"last\_node\_id": 2,

"last\_link\_id": 1,

"nodes": [

{

"id": 1,

"type": "CLIPTextEncode",

"pos": [50, 50],

"size": [300, 100],

"widgets\_values": ["uma menina com uma boneca"],

"inputs": [],

"outputs": [

{

"links": [1],

"slot\_index": 0

}

]

},

{

"id": 2,

"type": "KSampler",

"pos": [400, 50],

"size": [350, 150],

"widgets\_values": [20, 7, "euler", 1],

"inputs": [

{

"links": [1],

"slot\_index": 5

}

],

"outputs": []

}

],

"links": [

[1, 0, "CONDITIONING", 2, 5, "CONDITIONING"]

]

}

### **🚫 Erros comuns que invalidam o fluxo no Nordy**

| **Problema** | **Exemplo** | **Solução** |
| --- | --- | --- |
| links com menos de 6 campos | [1, 0, 2, 5] | Usar [origin\_id, origin\_slot, origin\_type, target\_id, target\_slot, target\_type] |
| Tipos incompatíveis nos links | "IMAGE" conectado em slot que espera "FLOAT" | Corrigir o tipo ou o slot usado |
| type mal escrito | "ImageApplyLUT" em vez de "ImageApplyLUT+" | Verificar nome exato no catálogo Nordy |
| Slot desconectado obrigatório | Campo como positive ou model sem link | Criar link válido |

| **Erro** | **Causa** | **Solução** |
| --- | --- | --- |
| slots.map is not a function | inputs e outputs foram declarados como objetos em vez de arrays | Mudar inputs e outputs para arrays de objetos ([{ name, type, ... }]) |
| links: Expected object, received array | Links foram declarados como arrays no estilo [1, 2, 0, 3, 0, "IMAGE"] | Trocar para objetos com { id, origin\_id, origin\_slot, target\_id, target\_slot, type } |
| groups[0].bounding ausente | Campo obrigatório no Nordy para cada grupo | Incluir "bounding": [x, y, width, height] nos groups |
| Expected object, received array at nodes[n].inputs | inputs devem ser arrays de objetos com name, type, link, slot\_index | Corrigir todos os inputs para esse formato |
| Expected object, received array at links[n] | Links precisam ser objetos, não arrays | Corrigir todos os links |

### **✅ Formato correto para a Nordy:**

json

CopyEdit

"links": [

[0, 1, 0, 2, 0, "IMAGE"],

[1, 2, 0, 3, 1, "CONDITIONING"]

]

### **❌ Formato que causa erro:**

json

CopyEdit

"links": [

{

"id": 0,

"origin\_id": 1,

"origin\_slot": 0,

"target\_id": 2,

"target\_slot": 0,

"type": "IMAGE"

}

]

O formato válido no Nordy é:

* links: **array de arrays** (cada link é um array de 6 elementos)
* groups: bounding como array [x, y, width, height]
* nodes[\*].inputs[\*].link: valor int ou null
* version: deve ser número (1)
* state: objeto vazio ({})

Esse erro indica que **alguns links estão com campos null onde deveriam conter números ou strings válidas**, especificamente nos **segundos elementos** de cada subarray de links, que correspondem ao origin\_id. A estrutura esperada de um link em formato array (padrão da Nordy) é:

json

CopyEdit

[link\_id, origin\_id, origin\_slot, target\_id, target\_slot, type]