

Structurer ses prompts en JSON

Ingénierie des prompts ■ Architecture logicielle ■ Niveau avancé

POURQUOI UTILISER DES PROMPTS EN JSON ?

Pour les applications industrielles ou agents autonomes, le texte libre atteint ses limites (hallucinations, dérive de format). Structurer en **JSON** apporte des bénéfices fondamentaux :

- **Structure claire** : Facile à parser, versionner via Git et tester unitairement.
- **Séparation des responsabilités** : Délimitation stricte entre instructions, contexte et données.
- **Réduction de l'ambiguïté** : Le format tabulaire ou hiérarchique guide l'attention du modèle.
- **Sauvegarde et réutilisation** : Vous pouvez sauvegarder vos prompts directement dans un format compréhensible par les IA.

1. PRINCIPE FONDAMENTAL : VERROUILLER L'ESPACE DES POSSIBLES

Pour obtenir un comportement déterministe, agissez sur 5 axes majeurs :

1. **Rôle strict** : Spécifier un périmètre technique précis sans adjectifs vagues.
2. **Instructions procédurales** : Découper la tâche en étapes algorithmiques.
3. **Contraintes négatives** : Définir clairement ce que le modèle *ne doit pas* faire.
4. **Contrat de sortie formel** : Imposer un schéma strict de type JSON-first qui est une méthode de développement et de conception qui consiste à définir, structurer et valider les données au format JSON avant de concevoir l'IHM ou d'écrire la logique métier.
5. **Auto-contrôle** : Demander au modèle de valider sa sortie avant de la rendre.

2. BONNES PRATIQUES OPÉRATIONNELLES

- **Séparer instructions et données** : Isoler le corps du texte dans un objet `input` dédié plutôt que de tout mélanger.
- **Verrouiller le schéma de sortie** : Utiliser des clauses telles que `"additionalProperties": false` pour rejeter les hallucinations de clés.
- **Contraintes négatives strictes (`must_not`)** : Les LLM réagissent mieux aux interdictions explicites qu'aux recommandations vagues.

```
"constraints": { "must_not": ["Ajouter du contexte non demandé", "Expliquer les bases conceptuelles"] }
```

⚠ ATTENTION AUX PIÈGES D'INTÉGRATION API

Même rédigé en JSON, un fournisseur (OpenAI, Anthropic) n'interprète pas magiquement un texte brut comme un objet structuré natif. Deux options s'offrent à vous :

- **Option A (Texte brut)** : Passer le JSON stringifié dans une consigne globale.
- **Option B (Recommandé - Mapping API)** : Mapper les blocs logiques directement dans les messages `system` et `user` de l'API.

3. EXEMPLE DE STRUCTURE RECOMMANDÉE

Voici un modèle générique, robuste et extensible pour vos prompts structurés. Pour chacune des ligne, le premier mot est la thématique à aborder, le deuxième la valeur à affecter. Noter les possibles imbrications de données.

```
{  "role":    "personne": "...",  "scope": [...],  ,  "task":    "objective": "...",  "context": "...",  ,  "instructions": [    "step": 1, "action": "...", "details": "...  ],  "input": ,  "constraints": {    "must": [],    "must_not": []  },  ,  "output_contract": {    "type": "json",    "schema": }
```

💡 EN RÉSUMÉ

Considérer vos prompts non plus comme de la prose littéraire, mais comme des **interfaces logicielles strictes**, garantissant la fiabilité, la maintenabilité et un passage à l'échelle réussi en production et une réutilisabilité des prompts.