

# Compte rendu — Workflow n8n : Recherche de stages à l'étranger

Leo Torres — DO3

31 janvier 2026

## Table des matières

<b>1 Contexte et objectifs</b>	<b>2</b>
1.1 Objectif du projet . . . . .	2
1.2 Périmètre et hypothèses . . . . .	2
<b>2 Architecture du workflow n8n</b>	<b>2</b>
2.1 Vue d'ensemble du workflow . . . . .	2
2.2 Description des étapes . . . . .	2
2.2.1 Étape 1 — Génération des combinaisons ville × technologie . . . . .	2
2.2.2 Étape 2 — Recherche web (Tavily) . . . . .	3
2.2.3 Étape 3 — Enrichissement scoring via MCP . . . . .	3
2.2.4 Étape 4 — Extraction d'informations via LLM . . . . .	4
2.2.5 Étape 4-bis — Node Code de parsing des extractions . . . . .	4
2.2.6 Étape 5 — Génération des résumés via LLM . . . . .	4
2.2.7 Étape 5-bis — Node Code de parsing des résumés . . . . .	5
2.2.8 Étape 6 — Filtrage personnalisé . . . . .	5
2.2.9 Étape 7 — Export des résultats . . . . .	6
2.2.10 Étape 8 — Synthèse finale . . . . .	7
2.2.11 Étape 9 — Notification . . . . .	7
<b>3 Exemples de résultats</b>	<b>8</b>
3.1 Offres extraites (5 à 10 exemples) . . . . .	8
3.2 Fichier CSV exporté (ou capture) . . . . .	8
<b>4 Bonus — Méthode de scoring</b>	<b>9</b>
4.1 Méthode utilisée . . . . .	9
4.2 Choix de scoring et justification . . . . .	9
4.3 Évaluation : est-ce que ça fonctionne comme attendu ? . . . . .	9
<b>5 Analyse critique</b>	<b>9</b>
5.1 Difficultés rencontrées . . . . .	9
5.2 Critères de filtrage choisis et justification . . . . .	9
5.3 Apports du workflow et axes d'amélioration . . . . .	9
<b>6 Conclusion</b>	<b>9</b>
<b>A Annexes</b>	<b>10</b>
A.1 Prompts LLM (extraits) . . . . .	10

# 1 Contexte et objectifs

## 1.1 Objectif du projet

Le workflow automatise la recherche d'offres de stage à l'étranger à partir d'une matrice *ville* × *technologie*. Les résultats sont enrichis par un score de ville (MCP), structures par LLM, puis exportés vers Google Sheets et résumés dans une synthèse envoyée sur Discord.

## 1.2 Périmètre et hypothèses

- Villes : Berlin, Stockholm.
- Technologies : crypto, devops, cybersecurity, iot (8 combinaisons).
- Source principale : recherche web via Tavily (1 résultat par requête).
- LLM : Google Gemini (gemma-3-4b-it pour extraction, gemma-3-27b-it pour résumé, gemini-robotics-er-1.5-preview pour synthèse) (à changé en fonction de la disponibilité des quotas gratuit).
- Limites : dépendance aux API, champs souvent absents (salaire, remote), redondances d'URLs.

# 2 Architecture du workflow n8n

## 2.1 Vue d'ensemble du workflow

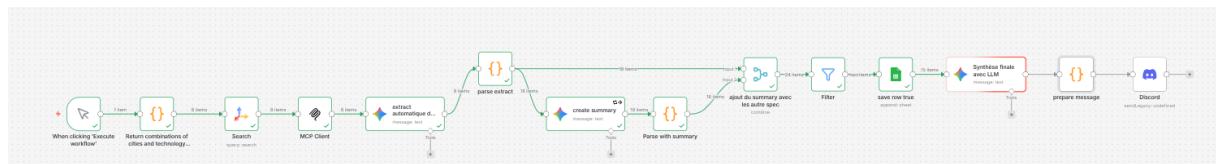


FIGURE 1 – Workflow n8n complet.

## 2.2 Description des étapes

### 2.2.1 Étape 1 — Génération des combinaisons ville × technologie

Un nœud Code génère les couples ville x technologie à partir de deux tableaux statiques.

```

const cities = [
  {"city": "Berlin"}, 
  {"city": "Stockholm"}, 
  {"city": "Paris"} 
]

const technologies = [
  {"technology": "crypto"}, 
  {"technology": "server"}, 
  {"technology": "cloudcomputing"}, 
  {"technology": "art"}
]

const combinations = cities.flatMap(city => {
  return technologies.map(tech => ({ 
    city: city, 
    technology: tech
  }))
})

return combinations;
}

```

FIGURE 2 – Node Code générant les combinaisons ville × technologie.

### 2.2.2 Étape 2 — Recherche web (Tavily)

Chaque couple déclenche une recherche Tavily avec la requête type `company <tech> <city> internship`. La recherche est en `basic`, avec `max_results=2`.

```

{
  "query": "company crypto Berlin internship",
  "follow_up_questions": null,
  "answers": [],
  "results": [
    {
      "url": "http://cryptobitlist.com/exchange/internship-jobs-berlin",
      "title": "CryptoBitList made the hiring process much easier and delivered a list of quality candidates to choose from. We ended up hiring one of them, so definitely a good result! We hired our 'Head of Social' through 'Cryptobitlist' and received an impressive talent pool of over 80 passionate individuals interested in the crypto industry. We received 100+ applications, interviewed 10% and 'Hired' the one from CryptobitList.",
      "content": "CryptoBitList made the hiring process much easier and delivered a list of quality candidates to choose from. We ended up hiring one of them, so definitely a good result! We hired our 'Head of Social' through 'Cryptobitlist' and received an impressive talent pool of over 80 passionate individuals interested in the crypto industry. We received 100+ applications, interviewed 10% and 'Hired' the one from CryptobitList.",
      "response_time": 0.75,
      "request_id": "4417-4417-Bece-7cc99d4d8a"
    },
    {
      "query": "company crypto Berlin internship",
      "follow_up_questions": null,
      "answers": [],
      "results": [
        {
          "url": "http://cryptobitlist.com/internship-non-tech-jobs-berlin",
          "title": "We'd Internship Non-Tech Jobs in Berlin, Germany",
          "content": "CryptoBitList made the hiring process much easier and delivered a list of quality candidates to choose from. We ended up hiring one of them, so definitely a good result! We hired our 'Head of Social' through 'Cryptobitlist' and received an impressive talent pool of over 80 passionate individuals interested in the crypto industry. We received 100+ applications, interviewed 10% and 'Hired' the one from CryptobitList.",
          "response_time": 0.75,
          "request_id": "4417-4417-Bece-7cc99d4d8a"
        }
      ]
    }
  ]
}

```

FIGURE 3 – Recherche web via Tavily.

### 2.2.3 Étape 3 — Enrichissement scoring via MCP

Le MCP Client appelle l'outil `find_city_score` avec le nom de ville. Si la ville est connue, le MCP renvoie `sensitivity_percent` et `city_score`; sinon l'outil renvoie une erreur et aucun score n'est ajouté.

## 2.2.4 Étape 4 — Extraction d'informations via LLM

Le LLM retourne un JSON strict, sans texte additionnel, conforme à un schéma fixe (company, job\_title, city, country, remote\_policy, contract\_type, salary, currency, duration, application\_deadline, skills, languages, source\_url, source\_title, city\_score). Les valeurs manquantes sont forcées à `null`.

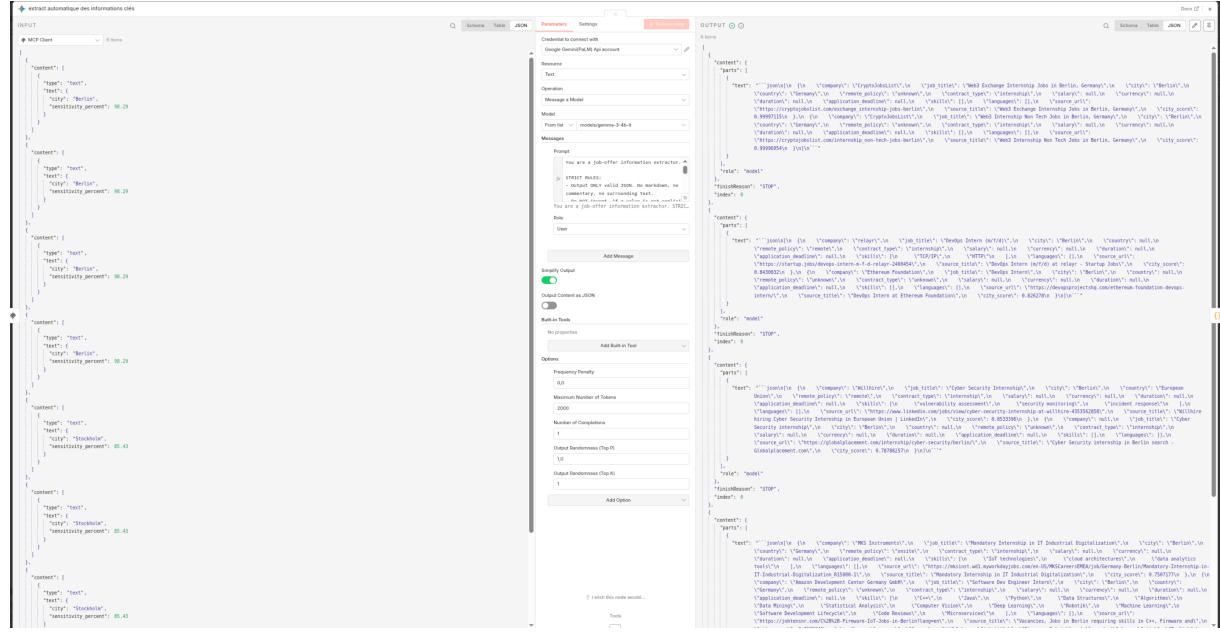


FIGURE 4 – Extraction automatique des informations clés via LLM.

## 2.2.5 Étape 4-bis — Node Code de parsing des extractions

Le LLM ne renvoie pas toujours un JSON propre (blocs "json, texte parasite, ou tableau/objet selon les cas). Le node Code nettoie d'abord la sortie en supprimant les fences Markdown et en tronquant tout ce qui précède le premier { ou [ et tout ce qui suit le dernier } ou ]. Ensuite, il tente un `JSON.parse` : si c'est un tableau, chaque entrée devient un item séparé ; si c'est un objet, il est encapsulé dans une liste. Le node normalise enfin le schéma (champs manquants à `null`, `skills` et `languages` en tableaux, `city_score` seulement si numérique). Cette étape rend le flux stable et évite les cassures plus loin (filtrage et export).

## 2.2.6 Étape 5 — Génération des résumés via LLM

Un second LLM produit un résumé FR de 2 à 3 phrases par offre. Les sorties non-JSON sont nettoyées et parsees dans un node Code.

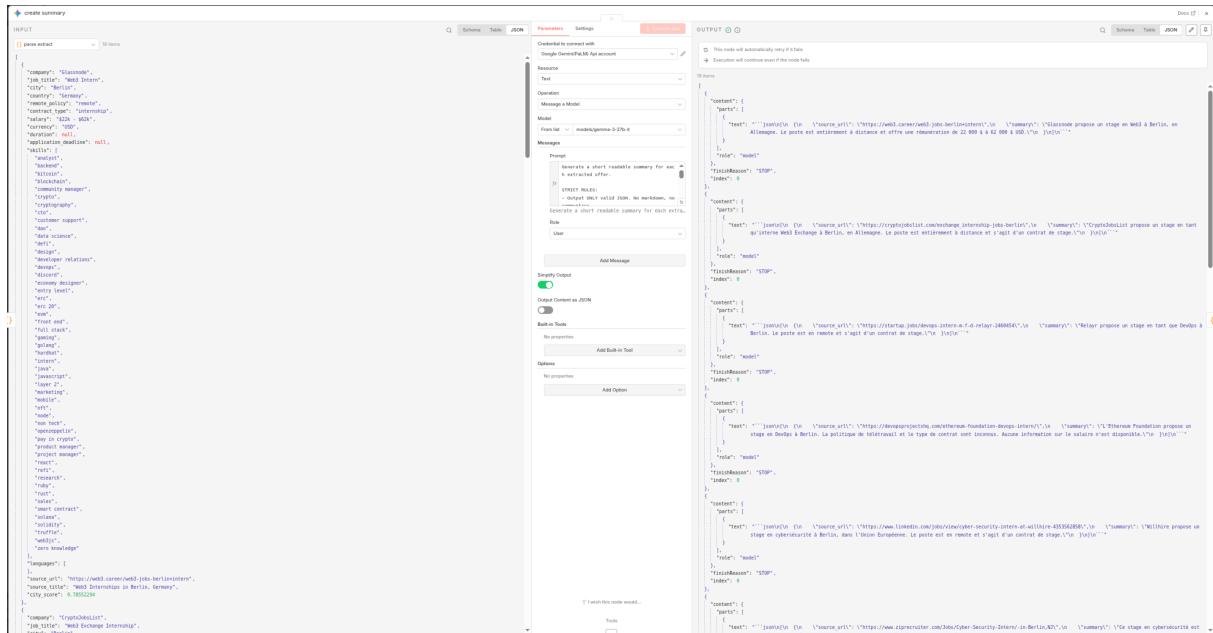


FIGURE 5 – Génération des résumés via LLM.

### 2.2.7 Étape 5-bis — Node Code de parsing des résumés

Le résumé est aussi renvoyé par le LLM sous forme JSON, mais avec les mêmes risques d'irrégularités. Le node Code récupère le texte (souvent `content.parts[0].text`), supprime les fences "`json`" et nettoie les débordements avant/après le JSON. Il garantit 1 item en entrée = 1 item en sortie pour préserver les jointures, récupère `source_url` quand il existe, et applique un fallback si le résumé est vide. En cas d'erreur LLM ou de parsing, il sort un item minimal avec un message par défaut, ce qui évite de perdre des offres lors des merges.

Pour ce code de parsing, je me suis aidé de l'IA afin de fiabiliser le nettoyage du JSON et la gestion des cas d'erreur.

### 2.2.8 Étape 6 — Filtrage personnalisé

Le filtrage est minimal : pas de dedoublonnage. Un node *Filter* conserve uniquement les offres dont le champ **salary** n'est pas vide. Les autres offres ne sont pas exportées.

FIGURE 6 – Filtrage personnalisé minimal.

### 2.2.9 Étape 7 — Export des résultats

Les offres filtrées sont écrites dans Google Sheets (équivalent CSV), dans l’onglet *Results true* du tableau *Internships*. Colonnes : company, job\_title, contract\_type, city, country, salary, currency, duration, application\_deadline, skills, languages, source\_url, source\_title, city\_score, summary, remote\_policy.

FIGURE 7 – Export des résultats dans Google Sheets.

Pour configurer Google Sheets, il faut d’abord créer un projet dans Google Cloud Console, activer l’API Google Sheets, puis configurer l’écran de consentement OAuth (type externe) et ajouter son compte en *test user*. Ensuite, créer des identifiants OAuth (client Web ou Bureau), récupérer le fichier JSON, et renseigner dans n8n les champs *Client ID / Client Secret* ainsi que

l'URL de redirection fournie par n8n. Enfin, autoriser le compte et selectionner le tableau (ou l'ID du fichier) pour que le node puisse écrire dans l'onglet cible.

### 2.2.10 Étape 8 — Synthèse finale

Un LLM genere une synthese structurée en français uniquement (sans JSON). Le prompt impose un format en 5 parties : titre, nombre total d'offres, top 3 argumente, statistiques globales (villes fréquentes, types de contrat, onsite/hybrid/remote/unknown, salaires si disponibles) et une conclusion courte (2 à 3 phrases), sans invention d'information.

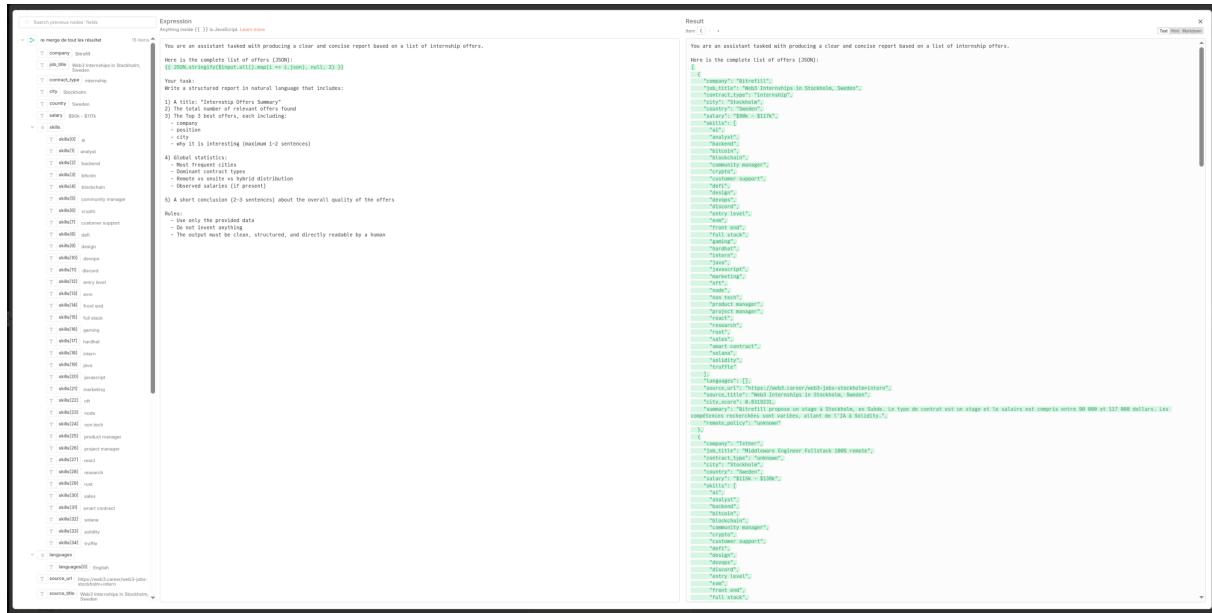


FIGURE 8 – Synthèse finale générée par LLM.

### 2.2.11 Étape 9 — Notification

La synthese est envoyee sur Discord via webhook. Un node Code extrait le texte renvoye par le LLM (par ex. `content.parts[0].text`), tronque a 1900 caracteres si necessaire, et ajoute un marqueur *message tronque* avant envoi.

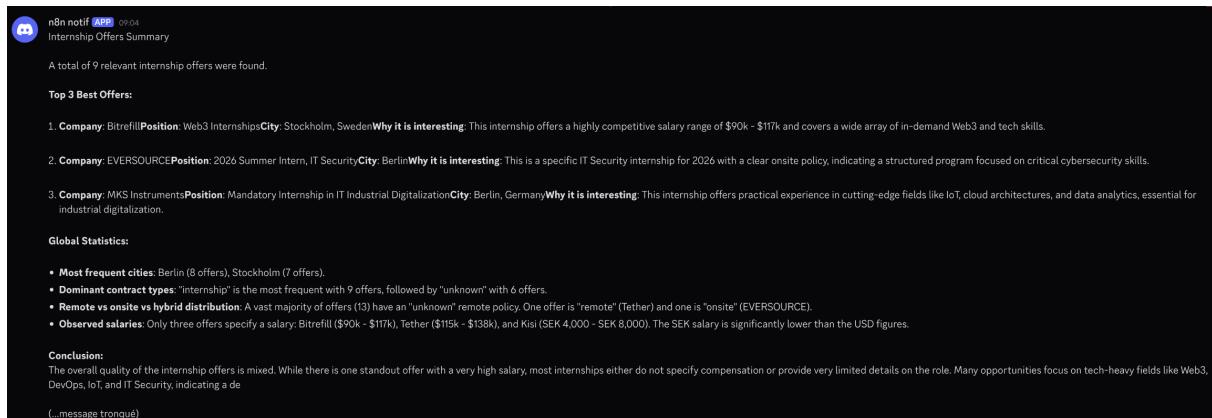


FIGURE 9 – Notification Discord via webhook.

### 3 Exemples de résultats

#### 3.1 Offres extraites (5 à 10 exemples)

Exemple de 5 a 10 offres extraites (toutes les infos dans le JSON) :

The screenshot shows the 'extract automatique des informations clés' tool interface. On the left, the 'INPUT' section displays a schema for 'MCP Client' with fields like 'content[]' and 'text'. The 'OUTPUT' section shows the extracted JSON for 10 job listings. The 'Parameters' tab includes settings for 'Credential to connect with' (Google GeminiAPI API account), 'Resource' (Test), 'Operation' (Message & Model), 'Model' (From list - models/gemini-3-4b-8), and 'Role' (User). The 'Messages' section contains a prompt for extracting job offers from text. The 'Output Content as JSON' toggle is on. The 'Options' section includes Frequency Penalty (0.0), Maximum Number of Tokens (2000), Number of Completions (1), Output Randomness (Top P) (1.0), and Output Randomness (Top K) (1). The 'Tools' section has an 'Add Option' button.

FIGURE 10 – Exemple de 5-10 offres extraites (JSON).

#### 3.2 Fichier CSV exporté (ou capture)

The screenshot shows a Google Sheets document titled 'Internships'. The table has 26 columns: A (Company), B (Job Title), C (City), D (Country), E (remote\_policy), F (contract\_type), G (salary), H (currency), I (duration), J (application\_deadline), K (skills), L (languages), M (source\_url), N (source\_id), O (city\_score), and P (summary). The table contains numerous rows of job listing data, such as 'Web3 Internship Stockholm' at 'Sweden' with 'remote' policy and 'internship' contract type, offering 'SEK 4,000 - SEK 8,000' salary. Another row lists 'Backend Devloper Intern' at 'Stockholm' with 'remote' policy and 'internship' contract type, offering 'SEK 4,000 - SEK 8,000' salary. The table continues with many other entries from various companies like 'Tether', 'OKX', 'Binance', etc., across different cities and countries.

FIGURE 11 – Aperçu de l'export CSV / Google Sheets.

## 4 Bonus — Méthode de scoring

### 4.1 Méthode utilisée

Le scoring repose sur un service MCP local qui expose une table statique CITY\_SENSITIVITY (0-100). Pour chaque ville de la matrice, le workflow appelle `find_city_score(city_name)`. La fonction effectue une recherche insensible à la casse, renvoie `sensitivity_percent` si la ville est connue, sinon une erreur et le score est laissé à `null`. Ce score est ensuite injecté dans le champ `city_score` des offres (export CSV et synthèse) afin d'apporter un signal rapide d'attractivité sans dépendre des données extraites.

### 4.2 Choix de scoring et justification

- Pertinence : indicateur externe pour comparer rapidement l'attractivité des villes.
- Simplicité : table pré-calculée, sans besoin de recalculation complexe.
- Robustesse : le score est stable et ne dépend pas des offres collectées.

### 4.3 Évaluation : est-ce que ça fonctionne comme attendu ?

Pour les villes présentes dans la table, le score est cohérent et stable. Les limites viennent des villes absentes ou des libellés atypiques (ex. quartiers, régions). Un enrichissement géographique serait utile.

## 5 Analyse critique

### 5.1 Difficultés rencontrées

- Choix d'un LLM gratuit : trouver un modèle avec un plan gratuit, assez rapide et suffisamment performant pour des extractions fiables.
- Parsing et fusion : difficultés à parser les sorties LLM puis à merger les flux pour n'obtenir qu'un seul objet exploitable.
- Webhook Discord : configuration et tests pour respecter les contraintes de taille des messages.
- Google Sheets : configuration Google Cloud API (OAuth) plus longue que prévu avant de pouvoir écrire dans un tableau.

### 5.2 Critères de filtrage choisis et justification

Le filtrage est volontairement léger pour maximiser la couverture : séparation selon `salary` présent ou non. Cela permet d'identifier rapidement les offres les plus informatives sans exclure trop de candidats.

### 5.3 Apports du workflow et axes d'amélioration

- Gain de temps important sur la collecte et la mise en forme des offres.
- Améliorations : augmenter `max_results`, ajouter des sources, filtrer par `city_score`, ajouter un dédoublement, meilleure gestion d'erreurs et retries.

## 6 Conclusion

Le workflow automatise efficacement la recherche et la synthèse d'offres de stage. L'extraction structurée et l'export permettent une analyse rapide, tandis que le scoring MCP apporte un signal supplémentaire. Les principaux axes d'amélioration concernent la couverture des sources et la robustesse face aux données incomplètes.

## A Annexes

### A.1 Prompts LLM (extraits)

```
1 Prompt d'extraction (Gemini) :
2 You are a job-offer information extractor.
3
4 STRICT RULES:
5 - Output ONLY valid JSON. No markdown, no commentary, no surrounding
   text.
6 - Do NOT invent. If a value is not explicitly present, use null.
7 - Empty lists must be [].
8 - Extract ONE output object per offer in the input array, preserving the
   same order and length.
9
10 OUTPUT FORMAT:
11 Return a JSON array of objects. Each object must follow EXACTLY this
   schema:
12
13 {
14     "company": string|null,
15     "job_title": string|null,
16     "city": string|null,
17     "country": string|null,
18     "remote_policy": "onsite"|"hybrid"|"remote"|"unknown"|null,
19     "contract_type": "internship"|"apprenticeship"|"full_time"|
           part_time"|"unknown"|null,
20     "salary": string|null,
21     "currency": string|null,
22     "duration": string|null,
23     "application_deadline": string|null,
24     "skills": string[],
25     "languages": string[],
26     "source_url": string|null,
27     "source_title": string|null,
28     "city_score": number|null
29 }
30
31 MAPPING RULES:
32 - source_url = offer.url if present, else null
33 - source_title = offer.title if present, else null
34 - city_score = input.content[0].text.city_score if present (MCP score),
   else null
35 - remote_policy:
36     - "remote" if clearly remote
37     - "hybrid" if clearly hybrid
38     - "onsite" if clearly onsite/on-site
39     - otherwise "unknown"
40 - contract_type:
41     - "internship" if stage/internship/intern
42     - "apprenticeship" if alternance/apprenticeship
43     - otherwise "unknown"
44 - salary: keep original text (e.g. "800 EUR/month", "$22k-$62k")
45 - currency: "EUR", "USD", "GBP" if clearly identifiable, else null
46
47 INPUT:
48 You will receive a JSON object containing offers in an array field. Use:
49 - offers = input.result OR input.results (whichever exists)
```

```

50
51 Input JSON :
52 {
53   "city": string ,
54   "sensitivity_percent": number ,
55 }
56
57 -----
58
59 Prompt de resume (Gemini) :
60 Generate a short readable summary for each extracted offer.
61
62 STRICT RULES:
63 - Output ONLY valid JSON. No markdown, no commentary.
64 - Return a JSON array with the same length/order as the input array.
65 - Do not include any other keys.
66
67 OUTPUT SCHEMA (per item):
68 {
69   "source_url": string|null ,
70   "summary": string
71 }
72
73 SUMMARY RULES:
74 - 2 to 3 sentences maximum.
75 - Include: company (if known), role, location, remote policy, contract
    type, and salary if present.
76 - If a field is missing, omit it (do not guess).
77 - Write the summary in French.
78
79 INPUT JSON (array of extracted offers):
80 [
81   {
82     "company": "crypto",
83     "job_title": "Director of Venture Studio",
84     "city": "Berlin",
85     "country": "Germany",
86     "remote_policy": "unknown",
87     "contract_type": "internship",
88     "salary": null,
89     "currency": null,
90     "duration": null,
91     "application_deadline": null,
92     "skills": [],
93     "languages": [],
94     "source_url": "https://www.linkedin.com/jobs/crypto-jobs-berlin?
        countryRedirected=1",
95     "source_title": "95 Crypto jobs in Berlin, Berlin, Germany (1 new)",
96     "city_score": 9.67
97   },
98   ...
99 ]
100
101 OUTPUT SCHEMA (per item):
102 {
103   "source_url": string|null ,
104   "summary": string

```

```

105    }
106
107 -----
108
109 Prompt de synthese finale (Gemini) :
110 Role :
111 Tu es un generateur de rapports synthetiques a partir de donnees
112 structurees.
113 Constraintes strictes :
114 - Utilise uniquement les donnees fournies ci-dessous
115 - N'invente aucune information
116 - Ne fais aucune supposition
117 - Si une information est absente ou null, indique-le explicitement ou
118 ignore-la selon le contexte
119 - La sortie doit etre claire, structuree et directement lisible par un
120 humain
121 - La reponse finale doit etre uniquement en francais
122 - Ne retourne pas de JSON, uniquement du texte structure
123
124 Tache :
125 A partir de la liste d'offres de stage fournie, genere un rapport de
126 synthese structure comme suit :
127
128 1. Titre
129     "Synthese des offres de stage"
130
131 2. Nombre total d'offres
132     Indique le nombre total d'offres analysees.
133
134 3. Top 3 des offres recommandees
135     Pour chaque offre :
136         - Entreprise
137         - Intitule du poste
138         - Ville (si disponible)
139         - Courte justification basee uniquement sur les donnees presentes
140             (exemples autorises : localisation, politique de teletravail,
141                 presence d'un salaire, clarte du poste)
142
143 4. Statistiques globales
144     Presente les statistiques suivantes uniquement si les donnees
145         existent :
146         - Villes les plus frequentes
147         - Repartition des types de contrat
148         - Repartition du mode de travail :
149             * onsite
150             * hybrid
151             * remote
152             * unknown
153         - Salaires :
154             * liste ou fourchette des salaires disponibles
155             * precise explicitement si peu ou aucune donnee n'est disponible
156
157 5. Conclusion
158     Redige une conclusion courte de 2 a 3 phrases maximum resumant :
159         - la diversite ou la concentration des offres

```

155        - les tendances principales observees (localisation, teletravail,  
                remuneration)

156

157        Donnees a analyser :