

# 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 . . . . .	6
2.2.11 Étape 9 — Notification . . . . .	7
<b>3 Exemples de résultats</b>	<b>7</b>
3.1 Offres extraites (5 à 10 exemples) . . . . .	7
3.2 Fichier CSV exporté (ou capture) . . . . .	8
<b>4 Bonus — Méthode de scoring</b>	<b>8</b>
4.1 Méthode utilisée . . . . .	8
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>9</b>
A.1 Prompts LLM (extraits) . . . . .	9

## 1 Contexte et objectifs

## 1.1 Objectif du projet

Le workflow automatisé 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 resultat par requete).
  - LLM : Google Gemini (gemma-3-4b-it pour extraction, gemma-3-27b-it pour resume, gemini-robotics-er-1.5-preview pour synthese) (à changé en fonction de la dispo des quotas gratuit).
  - Limites : dependance aux API, champs souvent absents (salaire, remote), redondances d'URLs.

## 2 Architecture du workflow n8n

## 2.1 Vue d'ensemble du workflow

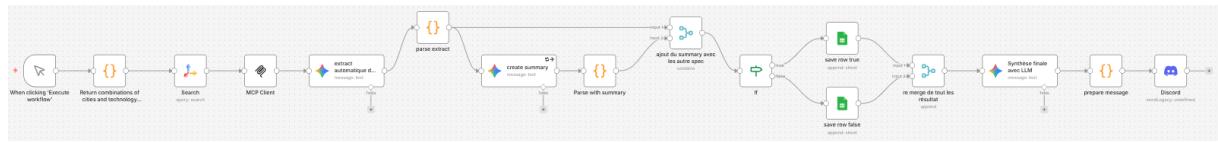


FIGURE 1 – Workflow n8n complet.

## 2.2 Description des étapes

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

Un node Code genere les couples ville x technologie a 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-7cc99d1408a"
    },
    {
      "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-7cc99d1408a"
        }
      ]
    }
  ]
}

```

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`.

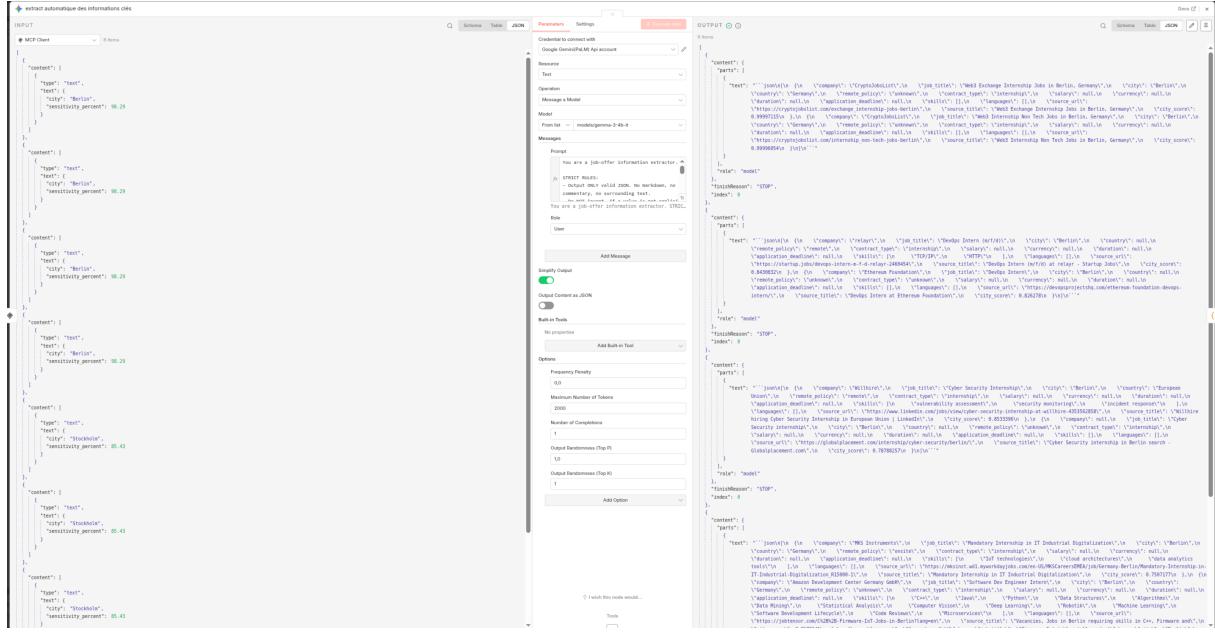


FIGURE 4 – Extraction automatique des informations clés via LLM.

## 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.

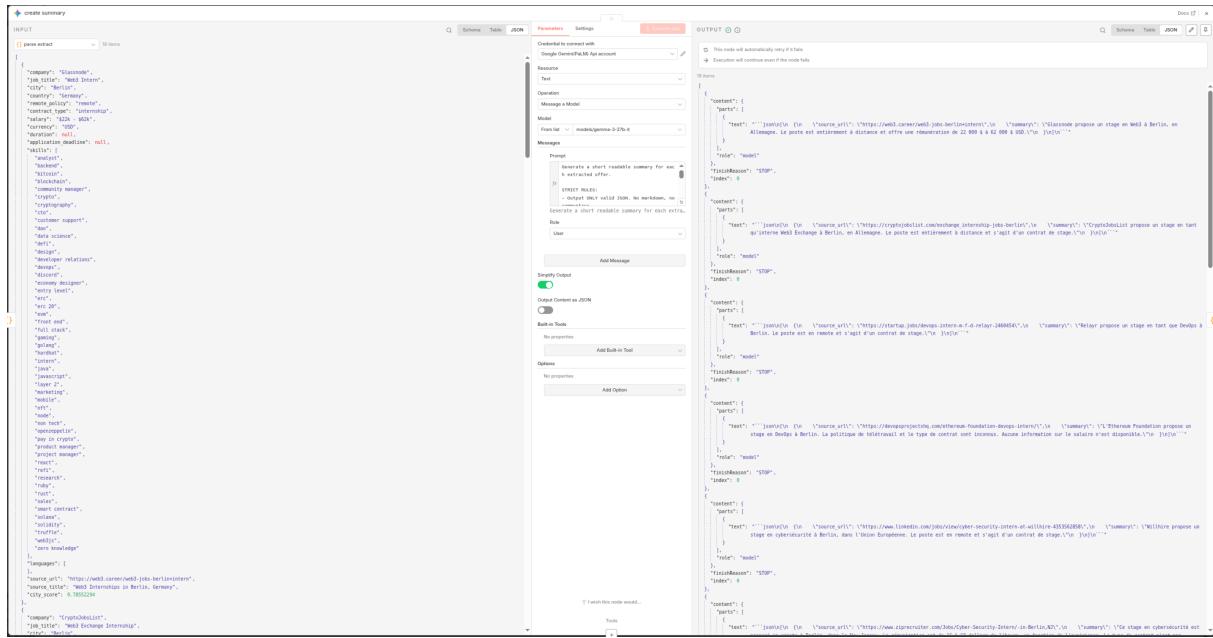


FIGURE 5 – Génération des résumés via LLM.

### 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, seulement un routage selon la présence d’un salaire pour séparer deux onglets de stockage.

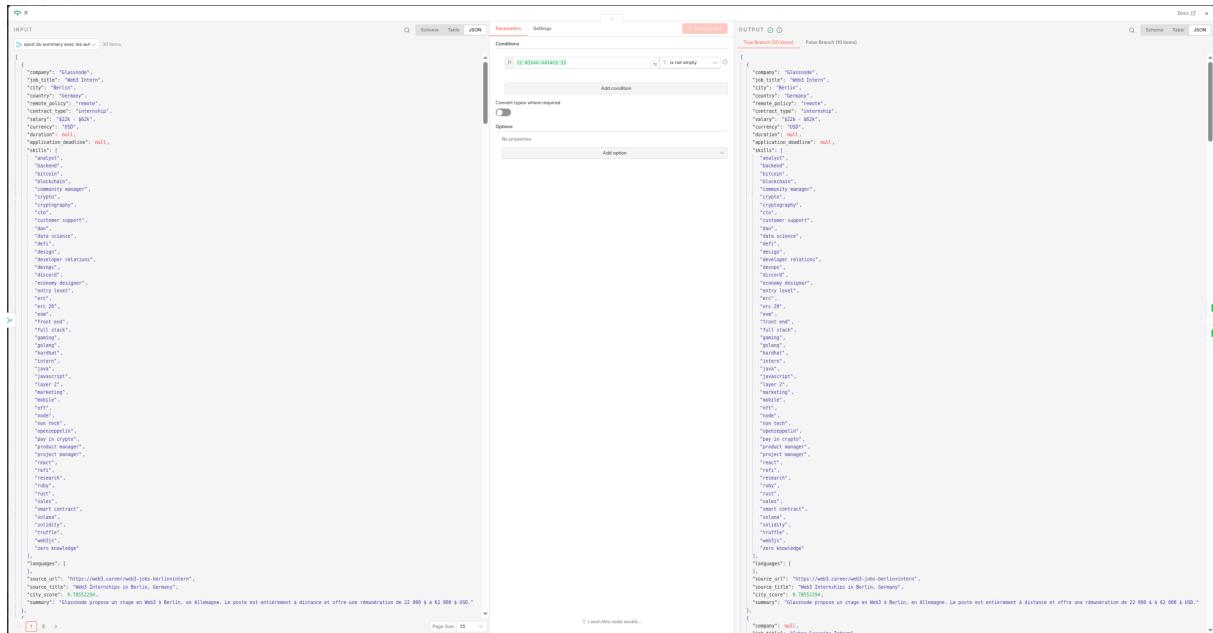


FIGURE 6 – Filtrage personnalisé minimal.

### 2.2.9 Étape 7 — Export des résultats

Les offres sont écrites dans Google Sheets (équivalent CSV), dans deux onglets en fonction du filtre (pour les offres ok selon le filtre et pour les autres). 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.

### 2.2.10 Étape 8 — Synthèse finale

Un LLM genere une synthese lisible avec un titre, le nombre d'offres, un top 3 argumente et des statistiques globales (villes, contrats, remote, salaires).

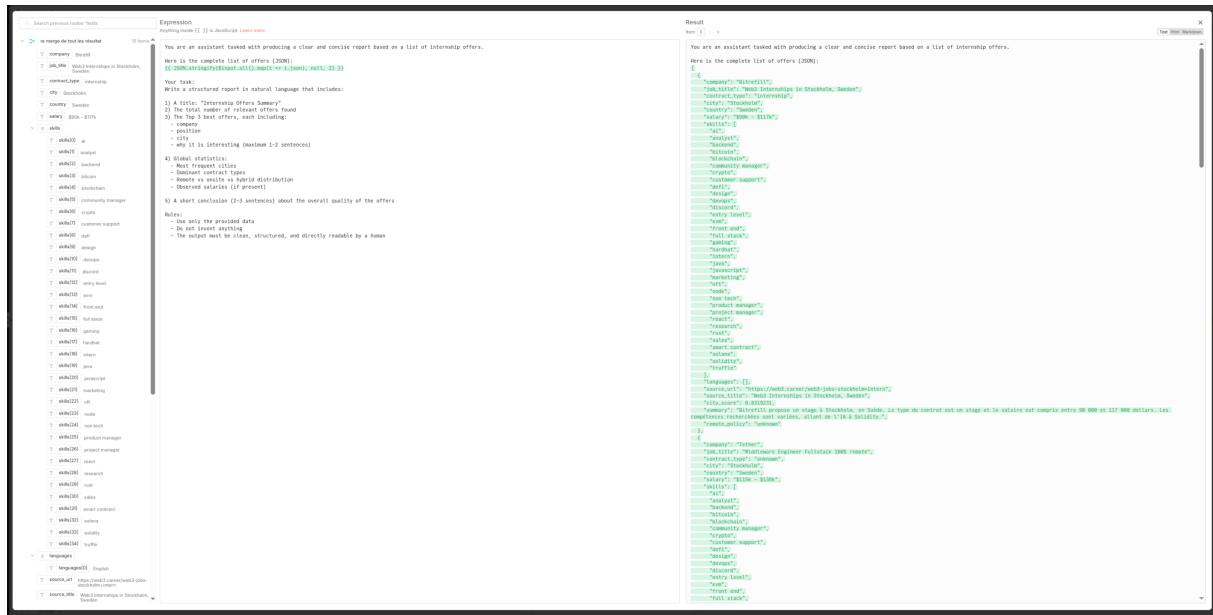


FIGURE 8 – Synthèse finale générée par LLM.

### 2.2.11 Étape 9 — Notification

La synthese est envoyee sur Discord via webhook. Un node Code tronque le message a 1900 caracteres pour respecter la limite.

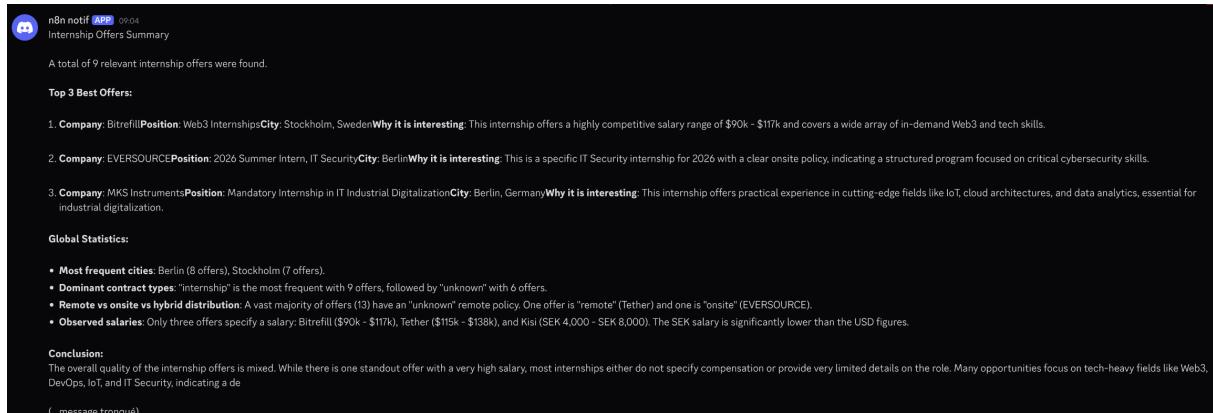


FIGURE 9 – Notification Discord via webhook.

### 3 Exemples de résultats

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

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

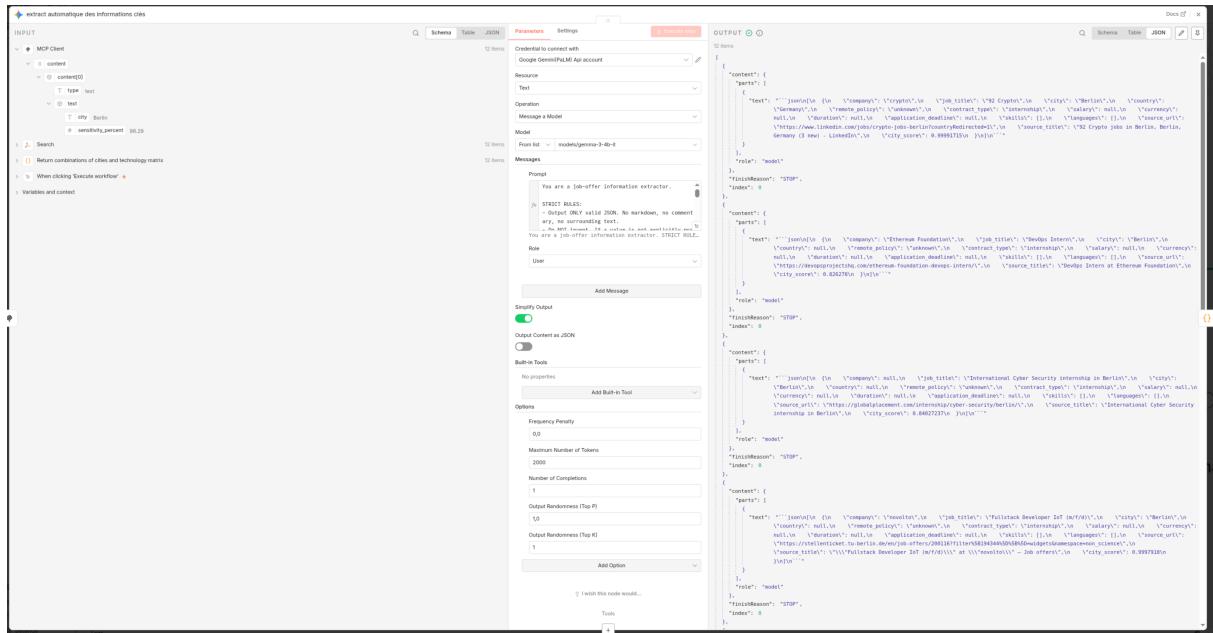


FIGURE 10 – Exemple de 5-10 offres extraites (JSON).

### 3.2 Fichier CSV exporté (ou capture)

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 synthese) 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?
95       countryRedirected=1",
96     "source_title": "95 Crypto jobs in Berlin, Berlin, Germany (1 new)",
97     "city_score": 9.67
98   },
99   ...
100 ]
101 OUTPUT SCHEMA (per item):
102 {
103   "source_url": string|null,
104   "summary": string
105 }

```

```
107 | -----  
108  
109 Prompt de synthese finale (Gemini) :  
110 Rules:  
111     - Use only the provided data  
112     - Do not invent anything  
113     - The output must be clean, structured, and directly readable by a  
114         human  
115 You must write the final answer in French.  
116  
117 Task:  
118 Summarize the internship offers provided in the JSON below.  
119  
120 Produce:  
121 1. Title: "Synthese des offres de stage"  
122 2. Total number of offers  
123 3. Top 3 offers (company, job_title, city, short reason based only on  
124     the data)  
125 4. Statistics:  
126     - Most frequent cities  
127     - Contract types distribution  
128     - Remote / onsite / hybrid distribution  
129     - Salaries if present  
130 5. Short conclusion (2-3 sentences)  
131  
132 Rules:  
133     - Use only the provided data  
134     - Do not invent missing information  
135     - Be concise and structured  
136 Data:
```