

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

Leo Torres — DO3

21 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) . . . . .	2
2.2.3 Étape 3 — Enrichissement scoring via MCP . . . . .	2
2.2.4 Étape 4 — Extraction d'informations via LLM . . . . .	2
2.2.5 Étape 5 — Génération des résumés via LLM . . . . .	3
2.2.6 Étape 6 — Filtrage personnalisé . . . . .	3
2.2.7 Étape 7 — Export des résultats . . . . .	3
2.2.8 Étape 8 — Synthèse finale . . . . .	3
2.2.9 Étape 9 — Notification . . . . .	3
<b>3 Exemples de résultats</b>	<b>4</b>
3.1 Offres extraites (5 à 10 exemples) . . . . .	4
3.2 Fichier CSV exporté (ou capture) . . . . .	5
3.3 Workflow n8n (JSON) dans le repo . . . . .	5
<b>4 Bonus — Méthode de scoring</b>	<b>5</b>
4.1 Méthode utilisée . . . . .	5
4.2 Choix de scoring et justification . . . . .	5
4.3 Évaluation : est-ce que ça fonctionne comme attendu ? . . . . .	5
<b>5 Analyse critique</b>	<b>6</b>
5.1 Difficultés rencontrées . . . . .	6
5.2 Critères de filtrage choisis et justification . . . . .	6
5.3 Apports du workflow et axes d'amélioration . . . . .	6
<b>6 Conclusion</b>	<b>6</b>
<b>A Annexes</b>	<b>6</b>
A.1 Prompts LLM (extraits) . . . . .	6

## 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  $\text{ville} \times \text{technologie}$ . Les résultats sont enrichis par un score de ville (MCP), structures par LLM, dédoublonnés, 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

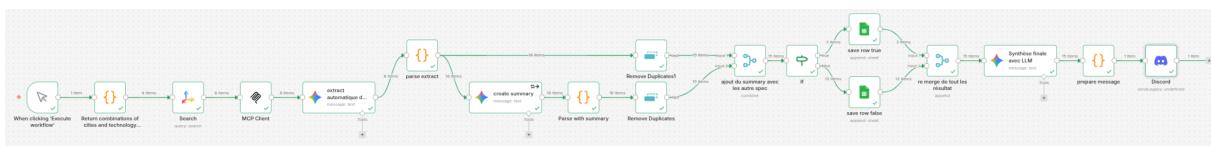


FIGURE 1 – Workflow n8n complet.

## 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.

### 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`.

### 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`.

### 2.2.5 Étape 5 — Génération des résumés via LLM

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

### 2.2.6 Étape 6 — Filtrage personnalisé

Le filtrage est minimal : dedoublonnage par `source_url`, puis routage selon la présence d'un salaire pour séparer deux onglets de stockage.

### 2.2.7 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.

### 2.2.8 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).

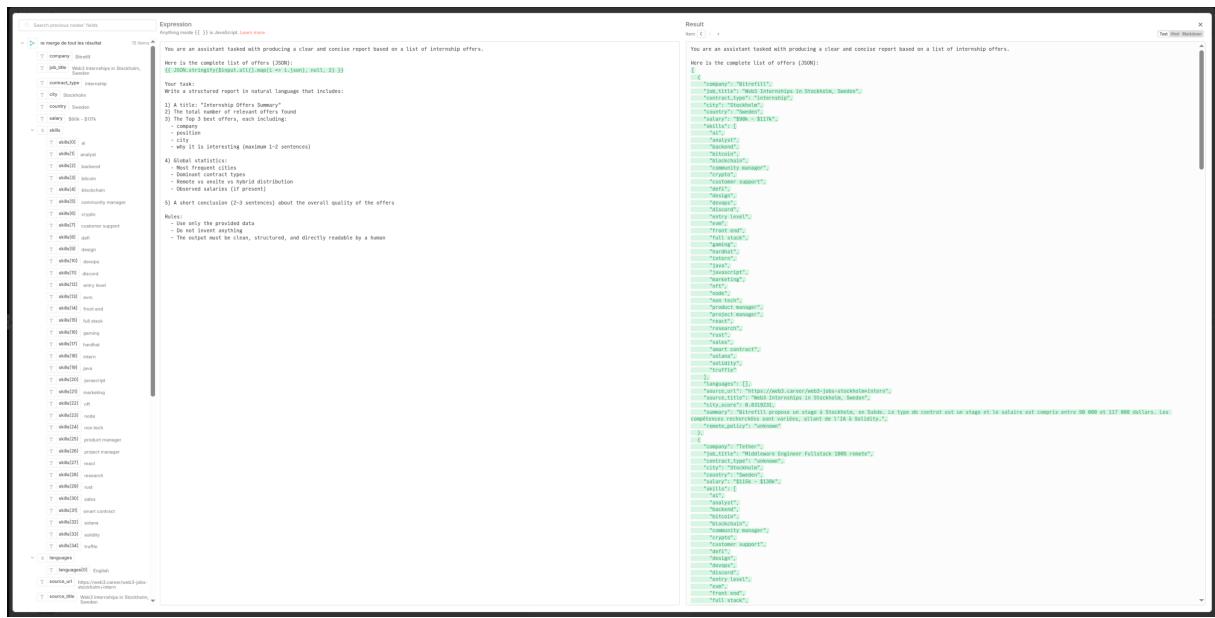


FIGURE 2 – Synthèse finale générée par LLM.

### **2.2.9 Étape 9 — Notification**

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

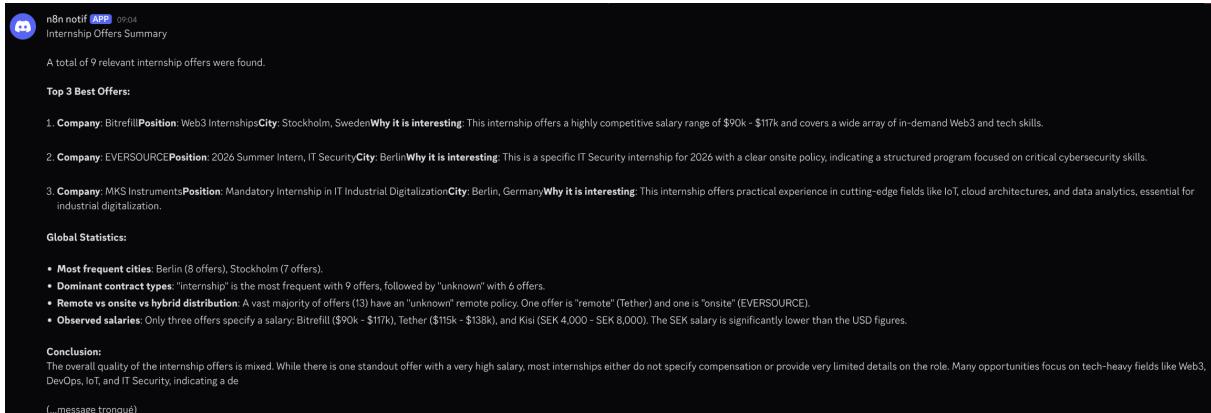


FIGURE 3 – Notification Discord via webhook.

## 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 MCP Client interface for 'extract automatique des informations clés'. The 'INPUT' section shows a tree view with 'MCP Client' selected, containing 'content' (with 'contentID' and 'text' nodes, the latter containing 'city: berlin' and 'sensitivity\_percent: 00.20'), 'Search', 'Return combination of cities and technology matrix', 'When clicking "Execute workflow"', and 'Variables and context'. The 'Parameters' tab is active, showing 'Credentialement pour se connecter à Google GeminiPaLM API account', 'Resource' set to 'Text', 'Operation' set to 'Message à Modèle', and 'Model' set to 'From list' with 'models/gpt-3-4b-t'. The 'Messages' section contains a prompt: 'You are a job-offer information extractor. /r/STREET\_RULES: - Output ONLY valid JSON. No markdown, no comment tags, no links. Be brief. If a value is not available, use "N/A". You are a job-offer information extractor. STREET RULES: Role: User'. The 'Add Message' button is visible. The 'Output' section shows 'Simplifier le résultat' is turned on, 'Output Content as JSON' is selected, and 'Bufalo Tools' are listed. The 'Options' section includes 'Frequency Penalty: 0.0', 'Maximum Number of Tokens: 2000', 'Number of Completions: 1', 'Output Randomness (Top P): 1.0', 'Output Randomness (Top K): 1', and 'Add Option'. The 'Tools' section has a note: 'I wish this node would...'. The 'OUTPUT' section displays 12 items of extracted JSON data, each representing an internship offer with fields like 'company', 'city', 'contract\_type', 'duration', 'language', 'source\_url', and 'salary'.

FIGURE 4 – Exemple de 5–10 offres extraites (JSON).

### 3.2 Fichier CSV exporté (ou capture)

FIGURE 5 – Aperçu de l'export CSV / Google Sheets.

### 3.3 Workflow n8n (JSON) dans le repo

```
1 | repo/  
2 |   workflow-final.json
```

## 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 : dédoublonnage par URL et 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`, enrichir la déduplication, 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
76   type, and salary if present.
77 - If a field is missing, omit it (do not guess).
78 - Write the summary in French.
79
80 INPUT JSON (array of extracted offers):
81 [
82   {
83     "company": "crypto",
84     "job_title": "Director of Venture Studio",
85     "city": "Berlin",
86     "country": "Germany",
87     "remote_policy": "unknown",
88     "contract_type": "internship",
89     "salary": null,
90     "currency": null,
91     "duration": null,
92     "application_deadline": null,
93     "skills": [],
94     "languages": [],
95     "source_url": "https://www.linkedin.com/jobs/crypto-jobs-berlin?
96       countryRedirected=1",
97     "source_title": "95 Crypto jobs in Berlin, Berlin, Germany (1 new)",
98     "city_score": 9.67
99   },
100   ...
101 ]
102
103 OUTPUT SCHEMA (per item):
104 {
105   "source_url": string|null,
106   "summary": string
107 -----
108
109 Prompt de synthese finale (Gemini) :
110 You are a report writer for internship search results.
111
112 STRICT RULES:
113 - Output ONLY plain text. No JSON, no markdown, no commentary.
114 - Write in French.
115 - Keep the message under 1900 characters.
116 - Do not invent numbers or facts.
117
118 INPUT:
119 You will receive a JSON array of extracted offers. Each item may include
120   :
121   company, job_title, city, country, remote_policy, contract_type, salary,
122     currency, duration, application_deadline, skills, languages,
123     source_url, source_title, city_score, summary.
124
125 OUTPUT FORMAT:

```

```
123 - Title on the first line.  
124 - Then 3 to 5 short paragraphs:  
125   1) Nombre total d'offres valides.  
126   2) Top 3 des offres les plus interessantes avec 1 phrase de  
       justification chacune (score ville, salaire, remote, contrat,  
       pertinence tech).  
127   3) Statistiques globales (villes les plus presentes, types de contrats  
       , remote, salaires disponibles).  
128  
129 RULES:  
130 - If fewer than 3 offers, list all.  
131 - If a field is missing, skip it.  
132 - Use source_url only if needed to disambiguate.
```