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

Leo Torres — DO3

21 janvier 2026

Table des matières

1 Contexte et objectifs

1.1 Objectif du projet

Le workflow automatise la recherche d'offres de stage a l'etranger a partir d'une matrice *ville* × *technologie*. Les resultats sont enrichis par un score de ville (MCP), structures par LLM, dedoublonnes, puis exportes vers Google Sheets et resumes dans une synthese envoyee 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

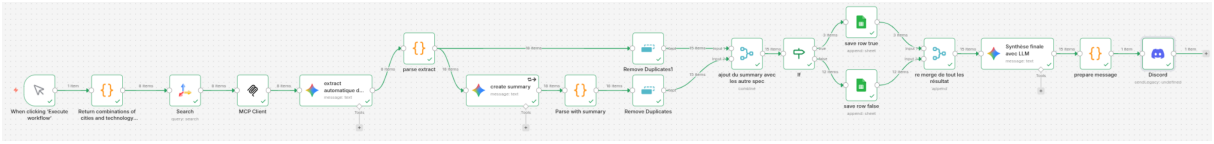

FIGURE 1 – Workflow n8n complet.

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 declenche une recherche Tavily avec la requete 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 ajoute.

2.2.4 Étape 4 — Extraction d'informations via LLM

Le LLM retourne un JSON strict, sans texte additionnel, conforme a un schema 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 forcees a `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 presence d'un salaire pour separer deux onglets de stockage.

2.2.7 Étape 7 — Export des résultats

Les offres sont ecrites dans Google Sheets (equivalent 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.

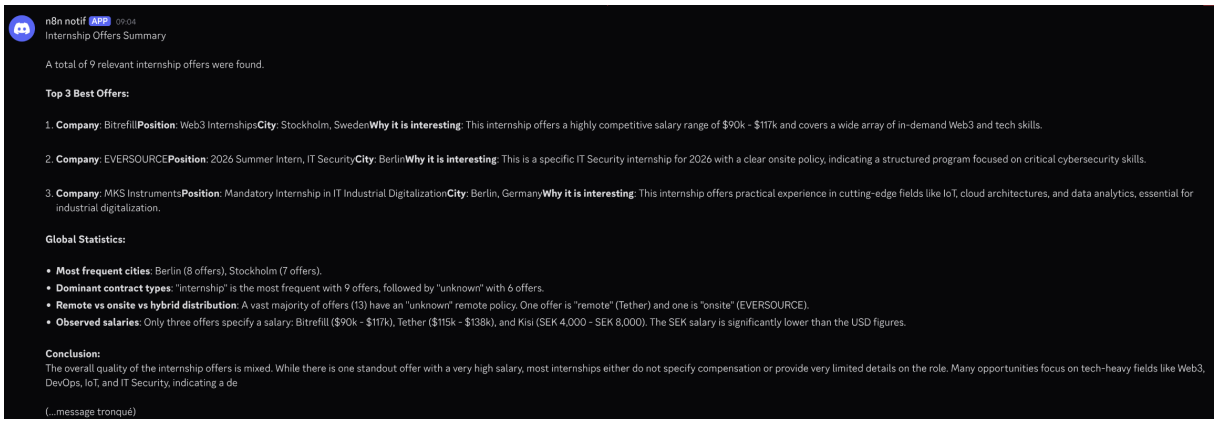

FIGURE 3 – Notification Discord via webhook.

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) :

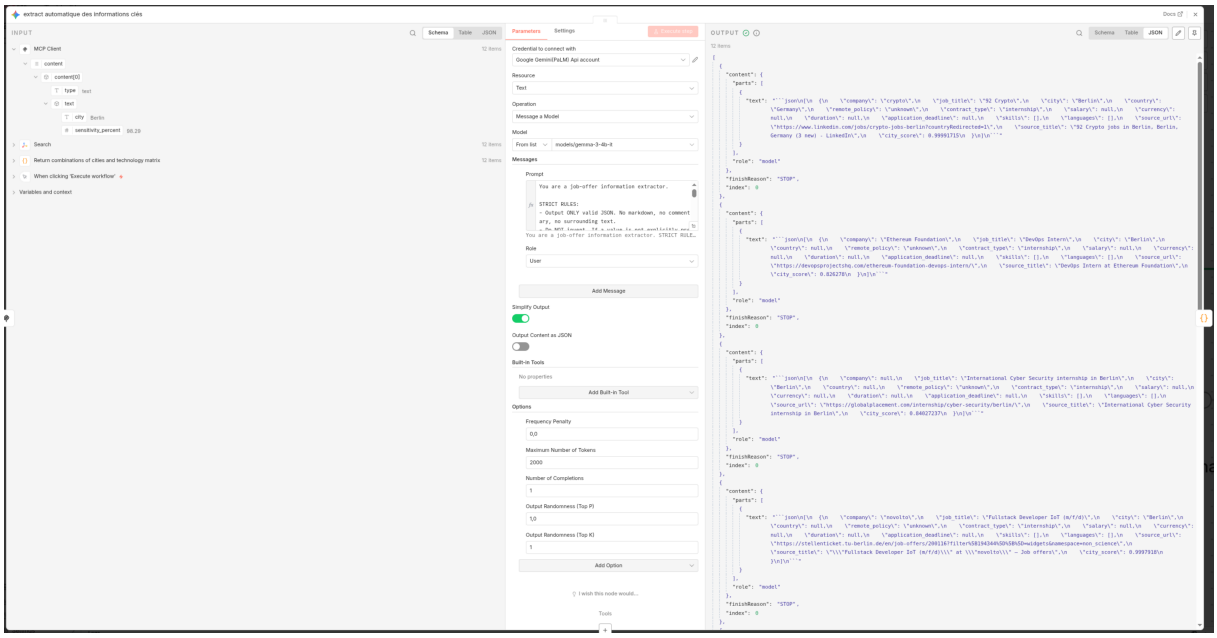

FIGURE 4 – Exemple de 5–10 offres extraites (JSON).

3.2 Fichier CSV exporté (ou capture)

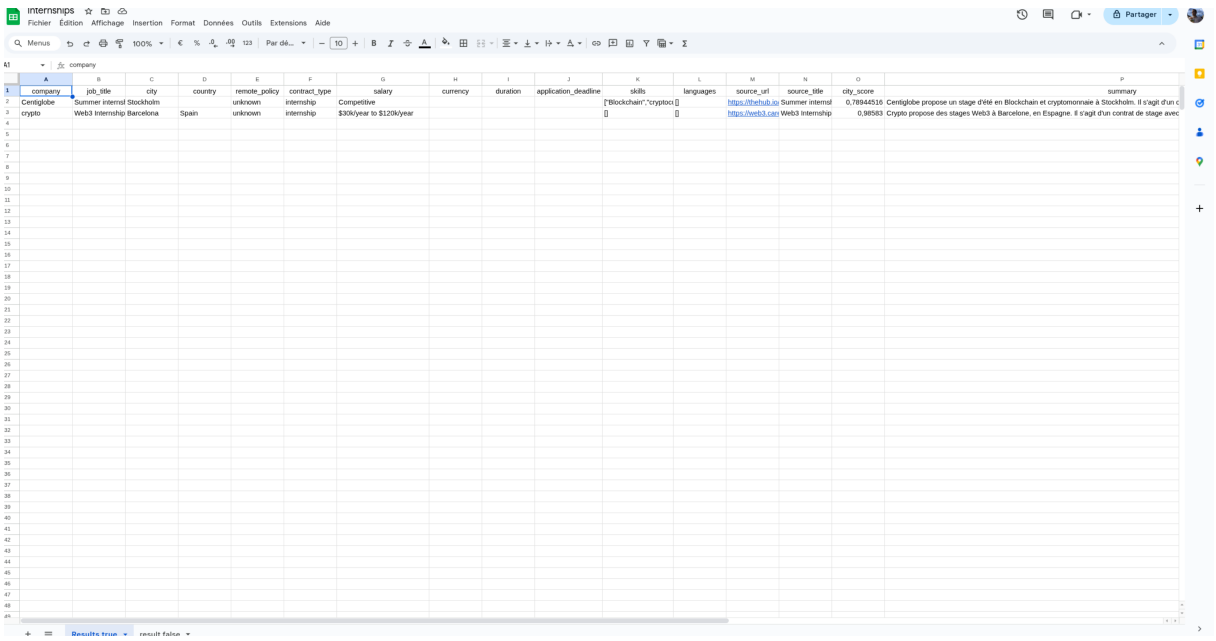

FIGURE 5 – Aperçu de l'export CSV / Google Sheets.

3.3 Workflow n8n (JSON) dans le repo

```
repo/
  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 a la casse, renvoie `sensitivity_percent` si la ville est connue, sinon une erreur et le score est laisse a `null`. Ce score est ensuite injecte dans le champ `city_score` des offres (export CSV et synthese) afin d'apporter un signal rapide d'attractivite sans dependre des donnees extraites.

4.2 Choix de scoring et justification

— Pertinence : indicateur externe pour comparer rapidement l'attractivite des villes.
— Simplicite : table pre-calculée, sans besoin de recalcul complexe.
— Robustesse : le score est stable et ne depend pas des offres collecte es.

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

Pour les villes presentes dans la table, le score est coherent et stable. Les limites viennent des villes absentes ou des libelles atypiques (ex. quartiers, regions). Un enrichissement geographique serait utile.

5 Analyse critique

5.1 Difficultés rencontrées

— Choix d'un LLM gratuit : trouver un modele avec un plan gratuit, assez rapide et suffisamment performant pour des extractions fiables.
— Parsing et fusion : difficultes a parser les sorties LLM puis a 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 prevu avant de pouvoir ecrire dans un tableur.

5.2 Critères de filtrage choisis et justification

Le filtrage est volontairement leger pour maximiser la couverture : dedoublonnage par URL et separation selon `salary` present 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.
— Ameliorations : augmenter `max_results`, ajouter des sources, filtrer par `city_score`, enrichir la deduplication, meilleure gestion d'erreurs et retries.

6 Conclusion

Le workflow automatise efficacement la recherche et la synthese d'offres de stage. L'extraction structuree et l'export permettent une analyse rapide, tandis que le scoring MCP apporte un signal supplementaire. Les principaux axes d'amelioration concernent la couverture des sources et la robustesse face aux donnees incompletes.

A Annexes

A.1 Prompts LLM (extraits)

```
Prompt d'extraction (Gemini) :
You are a job-offer information extractor.

STRICT RULES:
- Output ONLY valid JSON. No markdown, no commentary, no surrounding text.
- Do NOT invent. If a value is not explicitly present, use null.
- Empty lists must be [].
- Extract ONE output object per offer in the input array, preserving the same order and length.

OUTPUT FORMAT:
Return a JSON array of objects. Each object must follow EXACTLY this schema:

{
  "company": string|null,
  "job_title": string|null,
```

```
  "city": string|null,
  "country": string|null,
  "remote_policy": "onsite"|"hybrid"|"remote"|"unknown"|null,
  "contract_type":   "internship"|"apprenticeship"|"full_time"|"
     part_time"|"unknown"|null,
  "salary": string|null,
  "currency": string|null,
  "duration": string|null,
  "application_deadline": string|null,
  "skills": string[],
  "languages": string[],
  "source_url": string|null,
  "source_title": string|null,
  "city_score": number|null
}

MAPPING RULES:
- source_url = offer.url if present, else null
- source_title = offer.title if present, else null
- city_score = input.content[0].text.city_score if present (MCP score),
    else null
- remote_policy:
  - "remote" if clearly remote
  - "hybrid" if clearly hybrid
  - "onsite" if clearly onsite/on-site
  - otherwise "unknown"
- contract_type:
  - "internship" if stage/internship/intern
  - "apprenticeship" if alternance/apprenticeship
  - otherwise "unknown"
- salary: keep original text (e.g. "800 EUR/month", "$22k-$62k")
- currency: "EUR", "USD", "GBP" if clearly identifiable, else null

INPUT:
You will receive a JSON object containing offers in an array field. Use:
- offers = input.result OR input.results (whichever exists)

Input JSON :
{
  "city": string,
  "sensitivity_percent": number,
}

-------------------------------------------------------------------------------


Prompt de resume (Gemini) :
Generate a short readable summary for each extracted offer.

STRICT RULES:
- Output ONLY valid JSON. No markdown, no commentary.
- Return a JSON array with the same length/order as the input array.
- Do not include any other keys.

OUTPUT SCHEMA (per item):
{
  "source_url": string|null,
  "summary": string
```

```
}

SUMMARY RULES:
- 2 to 3 sentences maximum.
- Include: company (if known), role, location, remote policy, contract type, and salary if present.
- If a field is missing, omit it (do not guess).
- Write the summary in French.

INPUT JSON (array of extracted offers):
[
  {
    "company": "crypto",
    "job_title": "Director of Venture Studio",
    "city": "Berlin",
    "country": "Germany",
    "remote_policy": "unknown",
    "contract_type": "internship",
    "salary": null,
    "currency": null,
    "duration": null,
    "application_deadline": null,
    "skills": [],
    "languages": [],
    "source_url": "https://www.linkedin.com/jobs/crypto-jobs-berlin?countryRedirected=1",
    "source_title": "95 Crypto jobs in Berlin, Berlin, Germany (1 new)",
    "city_score": 9.67
  },
  ...
]

OUTPUT SCHEMA (per item):
{
  "source_url": string|null,
  "summary": string
}

--------------------------------------------------------------------------------


Prompt de synthese finale (Gemini) :
You are a report writer for internship search results.

STRICT RULES:
- Output ONLY plain text. No JSON, no markdown, no commentary.
- Write in French.
- Keep the message under 1900 characters.
- Do not invent numbers or facts.

INPUT:
You will receive a JSON array of extracted offers. Each item may include :
company, job_title, city, country, remote_policy, contract_type, salary, currency, duration, application_deadline, skills, languages, source_url, source_title, city_score, summary.

OUTPUT FORMAT:
```

```
- Title on the first line.
- Then 3 to 5 short paragraphs:
  1) Nombre total d'offres valides.
  2) Top 3 des offres les plus interessantes avec 1 phrase de
     justification chacune (score ville, salaire, remote, contrat,
     pertinence tech).
  3) Statistiques globales (villes les plus presentes, types de contrats
     , remote, salaires disponibles).

RULES:
- If fewer than 3 offers, list all.
- If a field is missing, skip it.
- Use source_url only if needed to disambiguate.
```