FORMATION SUP2TECH

Maîtrisez n8n L'Automatisation Workflow

Guide complet pour créer des automatisations puissantes avec n8n, des bases aux fonctionnalités avancées

Niveau : Débutant à Avancé

14 Modules • Glossary • Workflows • APIs • Webhooks • Data Handling • RAG • Vector DBs

Table des Matières

Module 1 : Introduction à l'Automatisation

🎯 **Objectifs de ce module**

- Comprendre ce qu'est l'automatisation et pourquoi elle est essentielle
- Découvrir les outils d'automatisation populaires
- Apprendre les concepts fondamentaux : workflows, déclencheurs et actions

Qu'est-ce que l'Automatisation ?

L'**automatisation** consiste à utiliser la technologie pour effectuer des tâches avec une intervention humaine minimale. Elle permet de connecter différentes applications, de transférer des données et d'exécuter des actions automatiquement.

💡 **Analogie du Restaurant**

Imaginez un restaurant où le serveur prend votre commande et la transmet à la cuisine. L'automatisation fonctionne de la même manière : elle prend une "commande" (données ou événement) et la transmet au "cuisinier" (traitement) pour produire un résultat.

Pourquoi Automatiser ?

- **Gain de temps :** Les tâches répétitives sont exécutées automatiquement
- **Réduction des erreurs :** Moins d'intervention humaine = moins d'erreurs
- **Évolutivité :** Gérez plus de travail sans embaucher plus de personnel
- **Productivité :** Concentrez-vous sur le travail important pendant que l'automatisation gère le reste

Outils d'Automatisation Populaires

Outil	Modèle	Idéal pour
Zapier	Basique, payant	Débutants, intégrations simples
Make	Visuel, payant	Workflows complexes, visualisation
n8n	Open source + Cloud	Flexibilité, contrôle total, coût

💡 **Bonne Pratique**

Commencez par identifier les tâches répétitives dans votre travail quotidien. Ces tâches sont les meilleures candidates pour l'automatisation.

✅ Récapitulatif

- L'automatisation exécute des tâches automatiquement avec une intervention minimale
- Elle fait gagner du temps, réduit les erreurs et améliore la productivité
- n8n est un outil puissant et flexible, idéal pour les workflows complexes

Module 2 : Glossaire des Termes Clés

🎯 Objectifs de ce module

- Maîtriser le vocabulaire essentiel de n8n
- Comprendre les termes techniques utilisés dans l'automatisation
- Pouvoir lire et comprendre la documentation n8n

Termes Fondamentaux

Terme	Définition
Workflow	Un workflow est une séquence d'étapes automatisées qui connectent des applications et traitent des données. C'est votre "recette" d'automatisation.
Node (Nœud)	Un nœud représente une étape unique dans un workflow. Chaque nœud effectue une action spécifique (récupérer des données, envoyer un email, etc.).
Trigger (Déclencheur)	Le nœud qui démarre un workflow. Il surveille les événements et lance l'exécution quand quelque chose se produit.
Action	Une opération effectuée par un nœud après le déclenchement (envoyer un message, créer une ligne, etc.).
Connection	L'autorisation donnée à n8n pour accéder à une application externe (ex: Google Sheets, Slack).
Credential	Les informations d'identification (clé API, token) permettant à n8n de se connecter à des services externes.
Execution	Une exécution est une instance de workflow qui a été lancée et complétée.
Expression	Une expression dynamique utilisée pour accéder ou manipuler des données dans un workflow (ex: `{{ $json.name }}`).
Item	Un élément de données unique traité par un nœud. Si vous avez 3 lignes dans Google Sheets, vous avez 3 items.
JSON	Format de données utilisé par n8n pour structurer et transmettre les informations entre les nœuds.
Webhook	Un mécanisme permettant à une application d'envoyer des données en temps réel à n8n dès qu'un événement se produit.
API	Interface de Programmation Applicative - permet aux applications de communiquer entre elles.
HTTP Request	Un nœud permettant d'envoyer des requêtes HTTP pour communiquer avec des APIs externes.
Polling	Vérification régulière d'une source pour détecter des changements (opposé aux webhooks).
Static Data	Données fixes saisies manuellement dans un workflow (ex: "Bonjour!").

Dynamic Data	Données qui changent à chaque exécution, provenant d'autres nœuds ou expressions.

💡 **À Retenir**

Chaque workflow n8n commence par un **Trigger** et contient une ou plusieurs **Actions**. Les données circulent entre les nœuds au format **JSON**.

✅ Récapitulatif

- **Workflow** = séquence automatisée d'étapes
- **Node** = étape individuelle du workflow
- **Trigger** = point de départ du workflow
- **JSON** = format de données standard
- **Expression** = syntaxe pour accéder aux données dynamiquement

Module 3 : Comprendre les LLMs

🎯 **Objectifs de ce module**

- Comprendre ce qu'est un LLM et comment il fonctionne
- Connaître les LLMs populaires et leurs cas d'usage
- Savoir comment utiliser les LLMs dans n8n

Qu'est-ce qu'un LLM ?

Un **LLM (Large Language Model)** ou Grand Modèle de Langage est un type d'intelligence artificielle entraînée sur d'énormes quantités de texte pour comprendre et générer du langage naturel.

💡 **Analogie de la Prédiction**

Un LLM fonctionne comme une fonction de saisie automatique ultra-avancée. Il prédit le mot suivant le plus probable en fonction du contexte. Par exemple : "Le ciel est..." → "bleu".

Comment fonctionne un LLM ?

1. **Entraînement :** Le modèle est entraîné sur des milliards de pages de texte (livres, articles, sites web)
2. **Tokenisation :** Le texte est découpé en "tokens" (morceaux de mots)
3. **Prédiction :** Le modèle prédit le token suivant en fonction du contexte
4. **Génération :** En répétant ce processus, le LLM génère du texte cohérent

LLMs Populaires

LLM	Entreprise	Forces	Cas d'usage
GPT-4 / GPT-5	OpenAI	Polyvalent, conversationnel	Chatbots, rédaction, analyse
Claude	Anthropic	Contexte long (200K), sûr	Documents longs, analyse juridique
Gemini	Google	Multimodal, intégration Google	Vision, Workspace, recherche
Llama	Meta	Open source, gratuit	Auto-hébergement, confidentialité
DeepSeek	DeepSeek-AI	Raisonnement, économique	Mathématiques, codage

Utiliser les LLMs dans n8n

n8n intègre nativement plusieurs LLMs via le nœud **OpenAI** et d'autres intégrations :

- **Génération de texte :** Créer du contenu, résumer des documents
- **Classification :** Catégoriser des emails, tickets support
- **Extraction :** Extraire des informations structurées de textes
- **Traduction :** Traduire automatiquement des messages
- **Analyse de sentiment :** Détecter si un texte est positif/négatif

🧠 Tokens et Limites

Les LLMs ont des limites de **tokens** (morceaux de texte). Par exemple, GPT-4 gère environ 8 000 tokens (~6 000 mots). Claude peut gérer jusqu'à 200 000 tokens (~150 000 mots) !

✅ Récapitulatif

- Un LLM est une IA entraînée pour comprendre et générer du texte
- Il prédit le mot suivant en fonction du contexte
- Les LLMs populaires incluent GPT-4, Claude, Gemini, Llama
- n8n permet d'intégrer les LLMs dans vos workflows d'automatisation

Module 4 : Workflows, Triggers, Actions & Conditions

🎯 **Objectifs de ce module**

- Comprendre la structure d'un workflow n8n
- Maîtriser les différents types de triggers
- Savoir utiliser les actions et conditions
- Créer des workflows logiques et efficaces

Structure d'un Workflow

Un **workflow** n8n est une séquence de nœuds connectés qui s'exécutent dans un ordre défini. Chaque workflow suit cette structure :

📋 **Structure Type**

Trigger → **Action 1** → **Action 2** → **Condition** → **Action 3A** ou **Action 3B**

Les Triggers (Déclencheurs)

Un **trigger** est le nœud qui démarre votre workflow. Sans trigger, le workflow ne s'exécute jamais !

Type de Trigger	Description	Exemple
Manuel	Vous cliquez sur "Execute Workflow" pour lancer	Test, exécution ponctuelle
Schedule	S'exécute à intervalles réguliers	Tous les jours à 9h, toutes les heures
Webhook	Déclenché par une requête HTTP externe	Stripe envoie un paiement, formulaire soumis
Application	Déclenché par un événement dans une app	Nouvel email Gmail, ligne Sheets ajoutée
Error Trigger	Déclenché quand un autre workflow échoue	Notification d'erreur, backup

> 💡 **Analogie de la Sonnette**
>
> Un trigger est comme une sonnette d'entrée. Quelqu'un sonne (événement) → la porte s'ouvre (workflow démarre). Sans sonnette, personne ne sait qu'il faut ouvrir !

Les Actions

Les **actions** sont les nœuds qui effectuent des opérations après le déclenchement. Elles peuvent :

- **Récupérer des données :** Lire depuis Google Sheets, une base de données, une API
- **Créer des données :** Ajouter une ligne, créer un contact, envoyer un email
- **Modifier des données :** Transformer, filtrer, formater
- **Envoyer des notifications :** Slack, Email, SMS
- **Appeler des APIs :** Communiquer avec des services externes

Les Conditions (IF Node)

Le nœud **IF** permet de créer des branches conditionnelles dans votre workflow. Selon une condition, le workflow emprunte différents chemins.

> 🔄 **Exemple de Condition**
>
> **Condition :** Le montant du paiement > 100€

SI VRAI : Envoyer email de remerciement VIP

SI FAUX : Envoyer email standard

Types de Conditions

Opérateur	Description	Exemple
Equal	Égal à	Statut = "completed"
Not Equal	Différent de	Type ≠ "test"
Contains	Contient	Email contient "@gmail.com"
Greater Than	Supérieur à	Montant > 50
Less Than	Inférieur à	Quantité < 10
Exists	Existe	Champ "téléphone" existe
Empty	Vide	Nom est vide

Exemple Complet de Workflow

📧 Workflow : Notification de Nouveau Client

1. **Trigger :** Webhook Stripe (nouveau paiement)
2. **Action :** Créer contact dans HubSpot
3. **Condition :** Montant > 500€ ?
4. **Si OUI :** Envoyer email VIP au sales
5. **Si NON :** Envoyer email standard
6. **Action :** Ajouter ligne dans Google Sheets
7. **Action :** Notification Slack au channel #sales

💡 Bonnes Pratiques

- Commencez toujours par identifier votre trigger avant de construire le workflow

- Utilisez des noms descriptifs pour vos nœuds (ex: "Envoyer Email Client" au lieu de "Gmail 1")
- Testez chaque étape individuellement avant de tester le workflow complet
- Documentez vos workflows avec des notes pour les autres utilisateurs

✅ Récapitulatif

- Un **workflow** = Trigger + Actions (+ Conditions)
- Les **triggers** démarrent le workflow (manuel, schedule, webhook, app)
- Les **actions** effectuent les opérations (créer, lire, modifier, envoyer)
- Les **conditions** créent des branches logiques (IF/ELSE)

Module 5 : Comprendre les Nœuds dans n8n

🎯 **Objectifs de ce module**

- Comprendre ce qu'est un nœud et son rôle dans n8n
- Connaître les différents types de nœuds disponibles
- Savoir configurer et utiliser les nœuds efficacement
- Maîtriser les connexions entre nœuds

Qu'est-ce qu'un Nœud ?

Un **nœud (node)** est l'unité fondamentale de construction dans n8n. Chaque nœud représente une étape spécifique dans votre workflow et effectue une action particulière : récupérer des données, transformer des informations, envoyer une notification, etc.

💡 **Analogie des Briques LEGO**

Les nœuds sont comme des briques LEGO. Chaque brique a une fonction spécifique, et en les connectant ensemble, vous construisez une structure (votre workflow) qui réalise quelque chose de complexe.

Types de Nœuds dans n8n

Type	Description	Exemples
Trigger Nodes	Démarrent le workflow quand un événement se produit	Webhook, Schedule, Gmail Trigger
Action Nodes	Effectuent des opérations sur des données	HTTP Request, Gmail, Google Sheets
Logic Nodes	Contrôlent le flux d'exécution	IF, Switch, Merge, Wait
Data Nodes	Manipulent et transforment des données	Code, Set, Function, Aggregate
AI Nodes	Intègrent l'intelligence artificielle	OpenAI, Embeddings, Vector Store

Anatomie d'un Nœud

Chaque nœud dans n8n contient plusieurs éléments :

- **Nom :** Identifiant unique du nœud (modifiable)
- **Paramètres :** Configuration spécifique à l'action
- **Credentials :** Authentification pour les services externes
- **Options :** Paramètres avancés supplémentaires
- **Connecteurs :** Points d'entrée et de sortie pour lier les nœuds

🔧 Configuration d'un Nœud

1. Cliquez sur le nœud pour ouvrir le panneau de configuration
2. Remplissez les paramètres requis (en rouge)
3. Configurez les credentials si nécessaire
4. Testez avec "Execute Node"
5. Enregistrez avec "Save"

Connexions entre Nœuds

Les nœuds se connectent entre eux via des **connexions** (lignes). Les données circulent d'un nœud à l'autre à travers ces connexions.

🔄 Flux de Données

Le nœud A produit des données → Les données passent à travers la connexion → Le nœud B reçoit et traite ces données

Type de Connexion	Description
Main	Connexion principale pour le flux normal de données
True/False	Branches conditionnelles du nœud IF
Loop	Connexion pour les boucles et itérations
Error	Connexion pour gérer les erreurs

💡 Bonnes Pratiques

- Donnez des noms descriptifs à vos nœuds pour faciliter la compréhension
- Organisez vos nœuds logiquement (de gauche à droite, haut en bas)
- Utilisez les notes pour documenter les nœuds complexes
- Testez chaque nœud individuellement avant de connecter

✅ Récapitulatif

- Un nœud est une unité de traitement dans un workflow
- Il existe 5 types : Trigger, Action, Logic, Data, et AI Nodes
- Les nœuds se connectent pour créer un flux de données
- Chaque nœud a des paramètres, credentials et options configurables

Module 6 : Les Triggers en Détail

🎯 Objectifs de ce module

- Maîtriser tous les types de triggers disponibles dans n8n
- Savoir choisir le bon trigger pour chaque cas d'usage
- Configurer correctement les triggers
- Comprendre les différences entre polling et webhooks

Types de Triggers

Les **triggers** sont les points de départ de tout workflow. n8n propose plusieurs types de triggers pour s'adapter à différents scénarios.

Trigger	Mécanisme	Cas d'usage
Manual Trigger	Exécution manuelle par clic	Tests, exécutions ponctuelles
Schedule Trigger	Exécution programmée (cron)	Rapports quotidiens, backups
Webhook	Déclenché par requête HTTP externe	Paiements, formulaires, intégrations
App Triggers	Événements dans des applications	Nouveaux emails, messages Slack
Error Trigger	Déclenché par l'échec d'un workflow	Alertes, notifications d'erreur

1. Manual Trigger

Le **Manual Trigger** est le plus simple. Il démarre le workflow quand vous cliquez sur "Execute Workflow".

⚡ Quand l'utiliser ?

- Pendant le développement et les tests
- Pour les workflows exécutés occasionnellement

- Quand vous avez besoin de contrôler quand le workflow s'exécute

2. Schedule Trigger

Le **Schedule Trigger** exécute le workflow à des intervalles réguliers définis.

Option	Description	Exemple
Interval	Toutes les X minutes/heures	Toutes les 30 minutes
Once	Une seule fois à une date précise	Le 25 décembre à 9h
Every Day	Chaque jour à une heure précise	Tous les jours à 8h00
Every Week	Chaque semaine, jours sélectionnables	Tous les lundis à 9h
Every Month	Chaque mois, jour sélectionnable	Le 1er de chaque mois
Custom (Cron)	Expression cron personnalisée	0 9 * * 1-5 (lundi-vendredi 9h)

💡 **Analogie du Réveil**

Un Schedule Trigger est comme un réveil programmable. Vous définissez l'heure et la fréquence, et il se déclenche automatiquement sans intervention.

3. Webhook Trigger

Le **Webhook Trigger** crée une URL unique qui écoute les requêtes HTTP entrantes. Quand une requête est reçue, le workflow démarre.

🌐 **Configuration Webhook**

- **Authentication :** Aucune, Basic Auth, ou Header
- **HTTP Method :** GET, POST, PUT, DELETE
- **Response Mode :** Last Node, First Entry, ou JSON
- **Path :** URL unique générée automatiquement

💡 **Sécurité Webhook**

Toujours activer l'authentification pour les webhooks en production. Utilisez des tokens secrets ou des signatures pour vérifier que les requêtes proviennent de la bonne source.

4. App Triggers (Polling vs Webhook)

Les **App Triggers** surveillent les événements dans des applications externes. Ils fonctionnent selon deux mécanismes :

Mécanisme	Fonctionnement	Avantages	Inconvénients
Polling	Vérifie régulièrement les changements	Fonctionne partout	Délai, consomme des ressources
Webhook	Notification instantanée de l'app	Temps réel, efficace	Nécessite support webhook

5. Error Trigger

Le **Error Trigger** est spécial : il démarre quand un autre workflow échoue. Parfait pour les alertes et la gestion des erreurs.

⚠️ **Workflow de Gestion d'Erreurs**

Workflow Principal échoue → **Error Trigger** détecte → **Workflow Alertes** envoie notification (email, Slack, SMS)

✅ **Récapitulatif**

- **5 types de triggers** : Manual, Schedule, Webhook, App, Error
- Le **Schedule** est idéal pour les tâches récurrentes
- Le **Webhook** offre des déclenchements en temps réel
- Le **Polling** vérifie régulièrement, le **Webhook** notifie instantanément
- Le **Error Trigger** gère les échecs de workflows

Module 7 : Les Nœuds Essentiels (Core Nodes)

🎯 **Objectifs de ce module**

- Connaître les nœuds fondamentaux de n8n
- Savoir manipuler et transformer des données
- Maîtriser les nœuds de logique et de contrôle
- Créer des workflows robustes avec les core nodes

Qu'est-ce qu'un Core Node ?

Les **Core Nodes** sont les nœuds natifs de n8n qui ne nécessitent pas de connexion externe. Ils permettent de manipuler des données, contrôler le flux d'exécution et effectuer des opérations de base.

💡 **Analogie de la Boîte à Outils**

Les Core Nodes sont comme les outils de base dans une boîte à outils : marteau, tournevis, clés. Ils sont toujours disponibles et essentiels pour tout projet.

Nœuds de Manipulation de Données

Nœud	Fonction	Cas d'usage
Set	Créer ou modifier des valeurs	Définir des variables, transformer des données
Code	Exécuter du JavaScript/Python	Transformations complexes, logique personnalisée
Function	Exécuter du code sur chaque item	Traitement item par item
Split Out	Séparer un tableau en items individuels	Traiter chaque élément séparément
Aggregate	Regrouper plusieurs items en un seul	Créer des résumés, statistiques
Remove Duplicates	Supprimer les doublons	Nettoyer des listes d'emails, contacts

Nœuds de Logique et Contrôle

Nœud	Fonction	Cas d'usage
IF	Condition binaire (vrai/faux)	Branches conditionnelles simples
Switch	Multiples branches selon valeur	Routing complexe (statuts, types)
Merge	Combiner plusieurs branches	Attendre plusieurs résultats
Wait	Mettre en pause le workflow	Délais, attente de réponse
No Operation	Ne fait rien (placeholder)	Documentation, organisation

Nœud IF en Détail

Le nœud **IF** est l'un des plus utilisés. Il crée deux branches selon qu'une condition est vraie ou fausse.

🔄 **Exemple IF**

Condition : `{{ $json.amount }}` > 100

Branche TRUE : Envoyer email VIP

Branche FALSE : Envoyer email standard

Opérateur	Description
Equal	Égal à
Not Equal	Différent de
Contains	Contient
Not Contains	Ne contient pas
Greater Than	Supérieur à
Less Than	Inférieur à
Starts With	Commence par
Ends With	Termine par
Exists	Existe (non null/undefined)
Empty	Vide ou null

Nœud Code en Détail

Le nœud **Code** permet d'écrire du JavaScript ou Python pour des transformations complexes.

💻 Exemple JavaScript

```
const items = $input.all();
const result = items.map(item => ({
  json: {
    name: item.json.name.toUpperCase(),
    total: item.json.price * item.json.quantity
  }
}));
```

```
return result;
```

💡 **Bonnes Pratiques Code**

- Utilisez le nœud Code uniquement quand les autres nœuds ne suffisent pas
- Commentez votre code pour faciliter la maintenance
- Testez le code avec différents jeux de données
- Préférez les nœuds natifs pour les opérations simples

Nœud HTTP Request

Le nœud **HTTP Request** est essentiel pour communiquer avec des APIs externes. Il permet d'envoyer des requêtes GET, POST, PUT, DELETE, etc.

Méthode	Usage
GET	Récupérer des données
POST	Créer une ressource
PUT	Mettre à jour complètement
PATCH	Mettre à jour partiellement
DELETE	Supprimer une ressource

✅ **Récapitulatif**

- Les Core Nodes sont natifs à n8n, toujours disponibles
- Set pour créer/modifier des données
- IF pour les conditions, Switch pour les branches multiples
- Code pour les transformations complexes
- HTTP Request pour communiquer avec des APIs

Module 8 : Introduction aux APIs

🎯 **Objectifs de ce module**

- Comprendre ce qu'est une API et comment elle fonctionne
- Connaître les concepts fondamentaux des APIs (endpoints, méthodes, réponses)
- Savoir lire et utiliser une documentation d'API
- Effectuer vos premières requêtes API dans n8n

Qu'est-ce qu'une API ?

Une **API (Application Programming Interface)** est une interface qui permet à différentes applications de communiquer entre elles. C'est comme un serveur dans un restaurant qui prend les commandes et apporte les plats.

💡 **Analogie du Restaurant**

Vous (application) → Serveur (API) → Cuisine (serveur/serveur de données)

Vous demandez quelque chose via le serveur, la cuisine prépare, et le serveur vous apporte le résultat.

Concepts Clés des APIs

Concept	Description	Exemple
Endpoint	URL spécifique pour une fonction	https://api.example.com/users
Méthode HTTP	Type d'action demandée	GET, POST, PUT, DELETE
Headers	Métadonnées de la requête	Authorization, Content-Type
Body	Données envoyées (POST/PUT)	{"name": "John", "email": "..."}
Response	Réponse du serveur	JSON avec données ou erreur
Status Code	Code indiquant le résultat	200 OK, 404 Not Found

Méthodes HTTP

🌐 Les 4 Méthodes Principales (CRUD)

- **GET** → Read (Lire des données)
- **POST** → Create (Créer une ressource)
- **PUT/PATCH** → Update (Modifier une ressource)
- **DELETE** → Delete (Supprimer une ressource)

Codes de Statut HTTP

Code	Signification	Description
200	OK	Requête réussie
201	Created	Ressource créée avec succès
400	Bad Request	Requête mal formée
401	Unauthorized	Authentification requise
403	Forbidden	Accès interdit
404	Not Found	Ressource introuvable
500	Server Error	Erreur serveur interne

Format JSON

La plupart des APIs utilisent le format **JSON (JavaScript Object Notation)** pour échanger des données. C'est un format lisible et structuré.

📋 Exemple de Réponse JSON

```
{
  "id": 123,
  "name": "John Doe",
  "email": "john@example.com",
  "active": true,
  "orders": [
    {"id": 1, "total": 50},
    {"id": 2, "total": 100}
  ]
}
```

Authentification API

La plupart des APIs nécessitent une **authentification** pour identifier qui fait la requête.

Méthode	Description
API Key	Clé unique envoyée dans le header ou l'URL
Bearer Token	Token JWT envoyé dans le header Authorization
OAuth 2.0	Protocole sécurisé pour l'autorisation (Google, etc.)
Basic Auth	Username et password encodés en base64

💡 **Sécurité API**

- Ne stockez jamais de clés API en dur dans vos workflows
- Utilisez les Credentials de n8n pour stocker les secrets
- Utilisez HTTPS pour toutes les communications API
- Limitez les permissions des clés API au minimum nécessaire

✅ Récapitulatif

- Une API permet aux applications de communiquer
- Endpoint = URL, Méthode = action, Response = résultat
- CRUD : GET (lire), POST (créer), PUT/PATCH (modifier), DELETE (supprimer)
- Les codes de statut indiquent le résultat (200, 404, 500...)
- Le JSON est le format standard d'échange de données

Module 9 : APIs en Profondeur

🎯 **Objectifs de ce module**

- Maîtriser le nœud HTTP Request dans n8n
- Savoir configurer les requêtes API avancées
- Gérer les erreurs et les réponses complexes
- Implémenter la pagination et les workflows multi-étapes

Le Nœud HTTP Request

Le nœud **HTTP Request** est l'outil le plus puissant de n8n pour interagir avec des APIs externes. Il offre un contrôle total sur vos requêtes.

💡 **Analogie du Téléphone**

Le nœud HTTP Request est comme un téléphone qui vous permet d'appeler n'importe quel service. Vous composez le numéro (URL), choisissez le type d'appel (méthode), et parfois envoyez un message (body).

Configuration Complète

Paramètre	Description	Exemple
Method	Type de requête HTTP	GET, POST, PUT, DELETE
URL	Endpoint de l'API	https://api.example.com/users
Authentication	Méthode d'authentification	Generic Credential, Header Auth
Headers	En-têtes HTTP supplémentaires	Content-Type: application/json
Query Parameters	Paramètres dans l'URL	?page=1&limit=10
Body	Données envoyées (POST/PUT)	JSON avec les données
Options	Paramètres avancés	Timeout, redirects, etc.

Exemples de Requêtes

📥 Requête GET (Récupérer des données)

URL : `https://jsonplaceholder.typicode.com/users`

Method : GET

Authentication : None (API publique)

Résultat : Liste des utilisateurs en JSON

📤 Requête POST (Créer une ressource)

URL : `https://api.example.com/users`

Method : POST

Headers : Content-Type: application/json

Body :

```
{
  "name": "{{ $json.name }}",
  "email": "{{ $json.email }}"
```

```
}
```

Gestion des Erreurs

Les APIs peuvent retourner des erreurs. Il est essentiel de les gérer pour créer des workflows robustes.

Erreur	Cause	Solution
401 Unauthorized	Credentials invalides	Vérifier la clé API/token
403 Forbidden	Permissions insuffisantes	Vérifier les droits d'accès
404 Not Found	Ressource inexistante	Vérifier l'URL/ID
429 Too Many Requests	Rate limit dépassé	Ajouter un délai entre les requêtes
500 Server Error	Erreur serveur	Réessayer plus tard

💡 **Gestion d'Erreurs avec IF**

Utilisez un nœud IF après l'HTTP Request pour vérifier le statut :

Condition : `{{ $json.status }}` >= 200 AND `{{ $json.status }}` < 300

TRUE : Continuer le workflow normal

FALSE : Envoyer alerte d'erreur

Pagination des Résultats

Quand une API retourne beaucoup de données, elle les divise en **pages**. Vous devez récupérer chaque page.

🔄 **Types de Pagination**

- **Offset/Limit :** ?offset=0&limit=100, ?offset=100&limit=100...
- **Page/Size :** ?page=1&size=100, ?page=2&size=100...
- **Cursor :** ?cursor=xyz123 (token pour la page suivante)

⚙️ Workflow de Pagination

1. Faire la première requête
2. Vérifier s'il y a une page suivante
3. Si oui, mettre à jour le paramètre de pagination
4. Répéter jusqu'à la dernière page
5. Agréger tous les résultats

Expressions dans les Requêtes

Utilisez les **expressions** pour rendre vos requêtes dynamiques.

Expression	Description
`{{ $json.id }}`	Accéder au champ id du JSON
`{{ $json.name.toUpperCase() }}`	Transformer en majuscules
`{{ Date.now() }}`	Timestamp actuel
`{{ $env.API_KEY }}`	Variable d'environnement

✅ Récapitulatif

- Le nœud HTTP Request offre un contrôle total sur les APIs
- Configurez Method, URL, Headers, Query Params, Body
- Gérez les erreurs avec des conditions IF
- Implémentez la pagination pour les grandes quantités de données
- Utilisez les expressions pour des requêtes dynamiques

Module 10 : Les Webhooks

🎯 Objectifs de ce module

- Comprendre ce qu'est un webhook et comment il fonctionne
- Savoir configurer des webhooks dans n8n
- Sécuriser vos webhooks
- Déboguer et tester les webhooks

Qu'est-ce qu'un Webhook ?

Un **webhook** est un mécanisme qui permet à une application d'envoyer des données en temps réel à une autre application dès qu'un événement se produit. C'est une "notification push".

💡 Analogie de la Sonnette

Au lieu de vérifier constamment si quelqu'un est à la porte (polling), la sonnette vous alerte immédiatement quand quelqu'un arrive. Le webhook est cette sonnette pour vos applications.

Webhook vs Polling

Aspect	Webhook	Polling
Temps de réponse	Instantané	Délai (minutes/heures)
Efficacité	Se déclenche uniquement si événement	Vérifie constamment
Ressources	Faible consommation	Consommation élevée
Configuration	Nécessite endpoint accessible	Fonctionne partout
Fiabilité	Peut manquer si endpoint down	Plus fiable mais lent

Configuration du Webhook dans n8n

Le nœud **Webhook** dans n8n crée une URL unique qui écoute les requêtes entrantes.

⚙️ Paramètres de Configuration

- **Path :** Chemin unique (auto-généré ou personnalisé)
- **Method :** GET, POST, PUT, DELETE
- **Response Mode :**
  - *Last Node :* Retourne la sortie du dernier nœud
  - *First Entry :* Retourne l'entrée du webhook
  - *Response Node :* Utilise un nœud Respond to Webhook
- **Response Data :** Données à retourner immédiatement

Sécurité des Webhooks

Les webhooks exposent une URL publique. Il est crucial de les sécuriser.

Méthode	Description	Configuration
Header Auth	Vérifier un header spécifique	X-Webhook-Secret: votre-secret
Basic Auth	Username et password	Credentials dans n8n
Signature	Vérifier la signature HMAC	Stripe, GitHub utilisent cette méthode
IP Whitelist	Restreindre aux IPs connues	Configurer dans le firewall

⚠️ Attention

Ne partagez jamais l'URL de votre webhook. Si elle est compromise, régénérez-la immédiatement. Utilisez toujours HTTPS pour les webhooks en production.

Tester un Webhook

🧪 Méthodes de Test

- **curl :** `curl -X POST https://votre-webhook-url -d '{"test":"data"}'`
- **Postman :** Interface graphique pour tester les APIs
- **Webhook.site :** Service pour inspecter les webhooks
- **n8n Test :** Cliquez "Listen For Test Event" puis envoyez une requête

Cas d'Usage Courants

Service	Événement Webhook	Action n8n
Stripe	payment_intent.succeeded	Créer client, envoyer email
GitHub	push, pull_request	Déployer, notifier l'équipe
Shopify	orders/create	Synchroniser inventaire
Typeform	form.submit	Enregistrer réponse, notifier
Zapier	Custom webhook	Intégrer avec d'autres services

💡 Bonnes Pratiques Webhook

- Toujours valider l'authentification des webhooks entrants
- Répondez rapidement (dans les 5 secondes) pour éviter les timeouts
- Utilisez un nœud Respond to Webhook pour des réponses personnalisées
- Loggez toutes les requêtes webhook pour le débogage
- Implémentez un mécanisme de retry pour les échecs

✅ Récapitulatif

- Un webhook envoie des données en temps réel quand un événement se produit
- Plus rapide et efficace que le polling
- Le nœud Webhook crée une URL d'écoute dans n8n
- Sécurisez vos webhooks avec authentification
- Testez avec curl, Postman ou webhook.site

Module 11 : Gestion des Données dans n8n

🎯 Objectifs de ce module

- Comprendre comment n8n structure et transmet les données
- Maîtriser le format JSON et les items
- Savoir accéder et manipuler les données avec les expressions
- Comprendre la différence entre données statiques et dynamiques

Structure des Données dans n8n

Dans n8n, les données circulent entre les nœuds sous forme d'**items**. Chaque item contient un objet JSON avec vos données.

💡 Analogie du Convoyeur

Imaginez un convoyeur dans une usine. Chaque item est une boîte sur le convoyeur, contenant des informations. Les nœuds sont des stations qui traitent chaque boîte.

Format des Items

📦 Structure d'un Item

```
{
  "json": {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com"
  },
```

```
  "pairedItem": { "item": 0 }
}
```

Chaque item a :

- **json :** Les données réelles (objet JSON)
- **pairedItem :** Référence à l'item d'origine (pour le traçage)

Accéder aux Données avec les Expressions

Les **expressions** permettent d'accéder aux données d'autres nœuds. Elles utilisent la syntaxe `{{ }}`.

Expression	Description	Exemple de résultat
`{{ $json.name }}`	Champ name du JSON actuel	"John Doe"
`{{ $json.email }}`	Champ email du JSON actuel	"john@example.com"
`{{ $('Node Name').item.json.name }}`	Champ name d'un nœud spécifique	"John Doe"
`{{ $input.first().json.name }}`	Premier item de l'entrée	"John Doe"
`{{ $input.all()[0].json.name }}`	Premier item (index 0)	"John Doe"

Données Statiques vs Dynamiques

Type	Description	Exemple
Statique	Valeur fixe saisie manuellement	"Bonjour", 123, true
Dynamique	Valeur provenant d'une expression	`{{ $json.name }}`

🔄 **Exemple Combiné**

Message statique : "Bonjour, "

Nom dynamique : `{{ $json.name }}`

Résultat : "Bonjour, John Doe"

Transformation de Données

n8n offre plusieurs façons de transformer les données :

Méthode	Usage	Exemple
Set Node	Créer/modifier des champs	Ajouter un champ calculé
Code Node	Transformations complexes	Filtrer, mapper, réduire
Split Out	Découper un tableau	1 item avec 5 éléments → 5 items
Aggregate	Regrouper des items	5 items → 1 item avec tableau

Fonctions d'Expression Utiles

Fonction	Description	Exemple
`$now`	Date et heure actuelles	2026-03-13T10:30:00.000Z
`$today`	Date du jour (minuit)	2026-03-13T00:00:00.000Z
`$json.field?.toUpperCase()`	Majuscules	JOHN DOE
`$json.field?.toLowerCase()`	Minuscules	john doe
`$json.field?.trim()`	Supprimer espaces	"texte"
`$json.field?.split(',')`	Diviser en tableau	["a", "b", "c"]

💡 **Bonnes Pratiques**

- Utilisez l'onglet "JSON" dans le panneau de droite pour inspecter les données
- Testez vos expressions avec le bouton "Test" avant d'exécuter
- Utilisez l'opérateur `?.` pour éviter les erreurs si un champ est absent
- Préférez les nœuds natifs aux nœuds Code pour les opérations simples

✅ Récapitulatif

- Les données circulent sous forme d'**items** contenant du JSON
- Les **expressions** `{{ }}` permettent d'accéder aux données
- **Données statiques :** valeurs fixes | **Dynamiques :** expressions
- Utilisez **Set**, **Code**, **Split Out**, **Aggregate** pour transformer

Module 12 : Données Épinglées et Édition

🎯 Objectifs de ce module

- Comprendre l'utilité des données épinglées (pin data)
- Savoir épingler et gérer les données de test
- Maîtriser l'édition de la sortie des nœuds
- Créer des workflows de test efficaces

Qu'est-ce que le Pin Data ?

Le **Pin Data** (épinglage de données) permet de sauvegarder la sortie d'un nœud pour l'utiliser comme données de test. Même après avoir fermé n8n, ces données restent disponibles.

💡 Analogie du Marque-Page

Le Pin Data est comme un marque-page dans un livre. Vous sauvegardez une page spécifique pour y revenir facilement, sans avoir à relire tout le livre.

Pourquoi Épingler des Données ?

- **Développement rapide :** Pas besoin de réexécuter les nœuds précédents
- **Tests cohérents :** Utilisez les mêmes données à chaque test
- **Économie d'appels API :** Évitez de consommer des quotas
- **Workflows hors ligne :** Travaillez sans connexion aux services
- **Partage :** Partagez des workflows avec des données d'exemple

Comment Épingler des Données

📌 Étapes pour Épingler

1. Exécutez le nœud pour générer des données

2. Dans le panneau de droite (JSON view), cliquez sur l'icône d'épingle
3. Les données sont maintenant sauvegardées
4. Un indicateur "Pinned" apparaît sur le nœud

Gestion des Données Épinglées

Action	Comment faire	Résultat
Épingler	Cliquer l'icône d'épingle	Données sauvegardées
Désépingler	Re-cliquer l'icône d'épingle	Données libérées
Voir	Ouvrir le nœud, onglet JSON	Visualisation des données
Modifier	Éditer directement dans le JSON	Données personnalisées

Édition de la Sortie (Edit Output)

La fonction **Edit Output** permet de modifier manuellement les données de sortie d'un nœud. C'est utile pour :

- Créer des scénarios de test spécifiques
- Simuler des données qui seraient difficiles à obtenir
- Corriger des données pour tester des cas particuliers
- Développer sans dépendre de services externes

✏️ Comment Éditer la Sortie

1. Exécutez le nœud ou créez un nœud Set
2. Dans le panneau JSON, cliquez sur "Edit"
3. Modifiez les valeurs directement dans l'éditeur
4. Cliquez "Save" pour appliquer les changements
5. Les données modifiées sont automatiquement épinglées

Workflow de Test Idéal

🧪 Structure Recommandée

1. **Nœud de départ :** Épinglez les données de test
2. **Développement :** Travaillez sur les nœuds suivants
3. **Itération :** Testez rapidement sans réexécuter le début
4. **Validation :** Vérifiez avec des données réelles
5. **Production :** Désépinglez toutes les données

Bonnes Pratiques

💡 Recommandations

- Épinglez les données des nœuds "coûteux" (APIs avec quotas)
- Documentez les données épinglées avec des noms de nœuds clairs
- Créez des jeux de test représentatifs (cas normaux et limites)
- Désépinglez toutes les données avant la mise en production
- Utilisez l'édition pour simuler des erreurs et tester la gestion d'erreurs

⚠️ Important

Les données épinglées sont stockées dans le workflow. Ne partagez pas de workflows contenant des données sensibles épinglées. Toujours vérifier et nettoyer avant le partage.

✅ Récapitulatif

- **Pin Data :** Sauvegarde la sortie d'un nœud pour les tests
- **Edit Output :** Modifie manuellement les données de sortie
- Utilisez ces fonctionnalités pour un développement rapide
- Désépinglez avant la mise en production

Module 13 : Bases de Données Vectorielles

🎯 Objectifs de ce module

- Comprendre ce qu'est une base de données vectorielle
- Savoir pourquoi elles sont critiques pour les agents IA
- Connaître les bases de données vectorielles populaires
- Maîtriser le processus de vectorisation

Qu'est-ce qu'une Base de Données Vectorielle ?

Une **base de données vectorielle** est un type spécial de base de données conçu pour l'IA et les LLMs. Elle stocke des **vecteurs** (représentations numériques de la signification) au lieu de texte brut.

💡 Analogie de la Recherche Sémantique

C'est comme rechercher des endroits similaires à votre café préféré, pas seulement par son nom exact. La base de données comprend le *sens* et le *contexte*, pas seulement les mots-clés.

Qu'est-ce qu'un Vecteur ?

Un **vecteur** est un groupe de nombres qui représente la signification d'un texte. Un **vector store** (magasin de vecteurs) est une mémoire consultable où ces vecteurs sont sauvegardés.

🧮 Caractéristiques des Vecteurs

- **Représentation numérique :** Chaque mot/phrase devient une série de nombres
- **Similarité sémantique :** Les vecteurs proches ont des significations similaires
- **Recherche par contexte :** Trouve des informations même avec des mots différents

Pourquoi les Agents IA en ont Besoin ?

Besoin	Description
Contexte	Les LLMs ont besoin de contexte pour générer de meilleures réponses
Mémoire	Les vecteurs permettent de récupérer des morceaux d'information pertinents
Recherche Sémantique	Trouver des données par signification plutôt que par correspondance exacte
RAG	Utilisé dans les agents RAG (Retrieval-Augmented Generation)

Le Processus de Vectorisation

⚙️ Étapes du Processus

1. **Commencez avec vos données :** PDFs, textes, fichiers JSON
2. **Découpez le contenu :** Divisez en petits morceaux (chunks)
3. **Embeddez chaque chunk :** Transformez en vecteur avec un LLM
4. **Stockez les vecteurs :** Sauvegardez dans une base de données vectorielle
5. **Requête utilisateur :** Convertissez la question en vecteur
6. **Comparez :** Trouvez les vecteurs stockés les plus proches
7. **Générez la réponse :** Envoyez les chunks pertinents au LLM

Bases de Données Vectorielles Populaires

Base de Données	Type	Forces	Meilleur pour
Pinecone	Gérée (Cloud)	Facile à utiliser, plug-and-play	Agents rapides, chatbots
Supabase + PGVector	Hybride (SQL + Vecteur)	Open source, données structurées + vecteurs	Cas d'usage hybrides
Qdrant	Open source	Rapide, puissant, auto-hébergé	Contrôle total, coût
Weaviate	Open source	GraphQL natif, multimodal	Applications complexes
Chroma	Open source	Simple, intégration Python	Prototypes, développement

Comment Choisir ?

📊 Guide de Sélection

- **Pinecone :** Meilleur pour la recherche sémantique rapide et gérée
- **Supabase :** Meilleur pour combiner données structurées et vecteurs
- **Qdrant :** Meilleur pour les solutions auto-hébergées et économiques

💡 Bonnes Pratiques

- Choisissez la taille des chunks selon votre contenu (500-1000 caractères typiquement)
- Utilisez des embeddings de qualité (OpenAI, Hugging Face)
- Indexez régulièrement vos nouveaux documents
- Testez différents seuils de similarité pour affiner les résultats

✅ Récapitulatif

- Une **base de données vectorielle** stocke des vecteurs (significations numériques)
- Elle permet la **recherche sémantique** par contexte, pas par mots-clés

- Pinecone, Supabase, Qdrant sont les options populaires
- Essentielle pour les agents IA et le RAG

Module 14 : RAG - Retrieval-Augmented Generation

🎯 Objectifs de ce module

- Comprendre ce qu'est le RAG et comment il fonctionne
- Savoir pourquoi le RAG rend les assistants IA plus intelligents
- Connaître les composants clés d'un système RAG
- Construire un workflow RAG dans n8n

Qu'est-ce que le RAG ?

RAG (Retrieval-Augmented Generation) = Récupération + Génération. C'est une technique qui combine la recherche d'informations en temps réel avec la génération de texte par LLM.

💡 Analogie de l'Examen Ouvert

Un LLM sans RAG est comme un étudiant en examen fermé - il ne connaît que ce qu'il a appris. Avec RAG, c'est un examen ouvert - il peut consulter des sources pour répondre avec des faits à jour.

Pourquoi Utiliser le RAG ?

Problème	Solution RAG
Les LLMs ont une connaissance fixe (date de coupure)	Accès à des données en temps réel
Ils ne peuvent pas accéder aux données internes	Connexion aux documents et bases de données internes
Hallucinations (réponses inventées)	Réponses basées sur des faits réels
Pas de contexte spécifique à l'entreprise	Utilisation de documents propriétaires

Comment Fonctionne le RAG ?

🔄 Le Processus RAG

1. **L'utilisateur pose une question**
2. **Le LLM identifie les informations nécessaires**
3. **Le système RAG récupère les données pertinentes** (documents, APIs, bases de données)
4. **Le LLM utilise ces données pour générer une réponse**
5. **Résultat :** Une réponse précise, contextuelle et à jour

Composants d'un Système RAG dans n8n

Composant	Rôle	Nœud n8n
Tools Agent	Le "cerveau" de l'assistant avec instructions système	Agent (AI Tools)
Chat Model	Modèle LLM qui génère la réponse finale	OpenAI Chat Model
Chat Memory	Stocke l'historique des conversations	Postgres Chat Memory
Vector Store	Mémoire consultable par IA (documents)	Supabase Vector Store
Embeddings	Convertit le texte en vecteurs	OpenAI Embeddings
Window Buffer Memory	Gère le contexte de conversation récent	Window Buffer Memory

Ce que le RAG Permet

✅ Avantages du RAG

- **Réponses en temps réel** à partir de données internes
- **Automatisation intelligente** et contextuelle

- **Assistants qui ne devinent pas** - ils récupèrent la bonne réponse
- **Sources vérifiables** pour chaque réponse

Cas d'Usage Réels

Secteur	Application RAG
Support Client	Réponses basées sur les politiques réelles et l'historique des tickets
Chatbots	Dernières mises à jour (actualités, météo, inventaire)
Éducation	Récupération du matériel le plus pertinent
E-commerce	Infos de stock, recommandations personnalisées
Finance	Taux d'intérêt à jour, infos de compte
Santé	Détails des politiques, infos patients
Juridique	Lois récentes, termes de contrats

Workflow RAG dans n8n

⚙️ Architecture Type

1. **Trigger :** Message utilisateur (Webhook, Chat)
2. **Embeddings :** Convertir la question en vecteur
3. **Vector Store :** Rechercher les documents similaires
4. **Chat Memory :** Récupérer l'historique de conversation
5. **Agent :** Combiner tout et générer la réponse
6. **Output :** Envoyer la réponse à l'utilisateur

💡 Bonnes Pratiques RAG

- Préparez bien vos documents (nettoyage, chunking)
- Testez différents modèles d'embeddings

- Ajustez le nombre de chunks récupérés (top-k)
- Utilisez la mémoire de conversation pour le contexte
- Mettez à jour régulièrement votre base de connaissances

✅ Récapitulatif Final

- **RAG** = Récupération + Génération pour des réponses factuelles
- Permet aux LLMs d'accéder à des **données en temps réel**
- Évite les **hallucinations** en basant les réponses sur des faits
- Composants clés : Agent, LLM, Vector Store, Embeddings, Memory
- Applicable à de nombreux secteurs (support, e-commerce, finance...)

Félicitations ! Vous avez maintenant toutes les bases pour maîtriser n8n et créer des automatisations puissantes avec IA.