

CERTIFICATION RS6601

Transformation digitale et outillage no-code

CAHIER DE RECETTES

Automatisation n8n — Contrôle des doublons de réservation

Projet : Open Spaces — Gestion des Réservations (4 sites)
Workflow recetté : WF01_OPENSPACES_Controle_Doublons
Stack : Airtable · n8n · Softr

Version	1.1.0
Date	2026-07-22
Auteur	Gregory Tudela
Statut	Recette fonctionnelle validée

Historique des versions

1.0.0	Rédaction initiale — CT-01 à CT-05, A-01 et A-02 documentées, A-03 non tracée (correctif appliqué avec un second outil IA, non relu).
1.1.0	Relecture du code final : cause racine de A-03 identifiée et documentée (sérialisation du champ de liaison ressource) ; mise à jour du §9.2 avec les points restants observés dans ce code (_debug_logs, break, homogénéisation du nom de champ).

Sommaire

Sommaire	2
1. Objet du document	3
2. Références documentaires	3
3. Périmètre de la recette	3
3.1 Fonctionnalité testée	3
3.2 Hors périmètre de cette recette (rappel du cadrage)	3
4. Conditions et environnement de test	3
5. Convention de lecture	4
6. Cas de test exécutés	4
CT-01 — Exécution isolée du nœud « Filtre nouvelle réservation »	4
CT-02 — Vérification de la structure de sortie du nœud « Airtable Trigger »	4
CT-03 — Exécution complète du workflow depuis le déclencheur	5
CT-04 — Re-exécution après correction de l'accès à l'identifiant (item.json.id)	5
CT-05 — Validation finale sur données réelles	5
7. Anomalies détectées et corrections apportées	6
A-01 — Hypothèse erronée sur l'ordre du premier enregistrement	6
A-02 — Confusion entre item.id et item.json.id	6
A-03 — Détection de chevauchement toujours négative malgré des données concordantes	6
8. Synthèse des résultats	7
9. Restes à construire et à tester	7
9.1 Automatisations n8n restantes	7
9.2 Corrections techniques à finaliser avant mise en production	8
9.3 Tests fonctionnels à planifier	8
9.4 Interfaces Softr (non construites à ce stade)	8
9.5 Rappel — hors périmètre projet	8
10. Validation	8

1. Objet du document

Ce cahier de recettes présente la démarche de validation fonctionnelle de la première automatisation n8n du projet Open Spaces : le contrôle des doublons de réservation (WF01_OPENSACES_Controle_Doublons). Il documente, de façon chronologique et traçable, les cas de test exécutés, les anomalies détectées, les corrections apportées et les résultats obtenus sur un jeu de données réel dans la base Airtable de production du projet.

Ce document constitue une preuve de la démarche de recette itérative attendue dans le cadre de la certification RS6601 : cartographie du besoin, implémentation, test, détection d'anomalie, correction, re-test jusqu'à validation.

2. Références documentaires

Document	Description
Note de cadrage Open Spaces	Périmètre fonctionnel du projet (in/out), acteurs, architecture cible
WF01_OPENSACES_Controle_Doublons	Export JSON du workflow n8n recetté dans ce document
Base Airtable « Open Spaces - Gestion des Réservations »	Table Réservations utilisée comme jeu de données de test réel

3. Périmètre de la recette

3.1 Fonctionnalité testée

Automatisation n8n n°1 — détection et blocage automatique des demandes de réservation en chevauchement de créneau sur une même ressource (salle, matériel), avec :

- déclenchement à la création/modification d'un enregistrement dans la table Réservations (Airtable Trigger) ;
- récupération de l'ensemble des réservations existantes sur la ressource (Search records) ;
- calcul du chevauchement de créneau en JavaScript (nœud Code « Filtre nouvelle réservation ») ;
- orientation du flux selon le résultat : réservation refusée automatiquement (doublon), ou poursuite vers le contrôle de la valeur assurée (> 5 000 €) pour validation manuelle ou confirmation automatique.

3.2 Hors périmètre de cette recette (rappel du cadrage)

Les éléments suivants ne font pas partie du périmètre testé ici et sont rappelés uniquement pour mémoire :

- le plafond de 2 réservations actives par adhérent ;
- l'automatisation de suppression RGPD ;
- le tableau de bord indicateurs du coordinateur réseau ;
- les interfaces Softr (Adhérent, Gestionnaire de site, Coordinateur réseau).

4. Conditions et environnement de test

Paramètre	Valeur
Instance n8n	n8n-demo.kindmotion.tech
Base Airtable	Open Spaces - Gestion des Réservations (app7HtuYEsqlyxlfI)

Paramètre	Valeur
Table testée	Réservations (tblqjk7fg4eKjHzHd)
Jeu de données	9 enregistrements réels constitués au fil des essais (voir §6 et annexe)
Champ déclencheur	date_reservation (Airtable Trigger, poll every minute)

Point d'attention — adresses email de test

Les adresses email des acteurs (Adhérents, Gestionnaires) étant à ce stade fictives (aucun compte réel créé), l'adresse email personnelle de l'auteur a été utilisée à la place de l'adresse du gestionnaire de site afin de pouvoir vérifier concrètement la réception des notifications (accusé de réception, email de confirmation, demande de validation).

Le nœud « Envoi demande de validation » contient actuellement une adresse email en dur (gregory.tudela@gmail.com) plutôt qu'un champ dynamique — ceci est une substitution volontaire et temporaire à des fins de recette, à corriger avant toute mise en production (voir §9.2).

5. Convention de lecture

Chaque cas de test est qualifié selon le code couleur suivant :

Statut	Signification
OK	Résultat obtenu conforme au résultat attendu
KO	Résultat obtenu non conforme — anomalie bloquante ou fonctionnelle
INFO	Test de vérification technique, sans attendu fonctionnel binaire

6. Cas de test exécutés

Les cas de test ci-dessous sont présentés dans l'ordre chronologique réel de la recette (approche itérative test → anomalie → correction → re-test), conformément à la démarche attendue en conditions d'examen.

CT-01 — Exécution isolée du nœud « Filtre nouvelle réservation »

Champ	Détail
Préconditions	Nœud Code exécuté seul (sans ré-exécution du workflow complet depuis le déclencheur), à la suite du nœud « Search records » (1 item en entrée : id_reservation 3, Salle Eiffel).
Données d'entrée	1 enregistrement Airtable en entrée du nœud Code.
Résultat attendu	Le nœud retourne un JSON enrichi des champs is_doublon, conflicting_record_id, is_sensible, valeur_assuree.
Résultat obtenu	« No output data returned ». Le nœud ne produit aucune sortie : n8n stoppe l'exécution faute de donnée en sortie.
Statut	KO

CT-02 — Vérification de la structure de sortie du nœud « Airtable Trigger »

Champ	Détail
Préconditions	Déclenchement manuel du nœud via « Fetch Test Event ».
Données d'entrée	Aucune (test de vérification technique de structure).
Résultat attendu	Confirmer que le nœud Airtable Trigger expose un champ id au même niveau que fields, à l'identique du nœud Search records.
Résultat obtenu	Structure confirmée : { id, createTime, fields } — id = rechYNZbnVI4lqrFS, identique au format retourné par Search records.
Statut	INFO

CT-03 — Exécution complète du workflow depuis le déclencheur

Champ	Détail
Préconditions	Workflow ré-exécuté dans son intégralité depuis « Airtable Trigger » (et non plus le nœud Code isolément), après première correction (recherche de l'enregistrement déclencheur par id).
Données d'entrée	Réservation id_reservation = 3, Salle Eiffel, déclenchée via Airtable Trigger.
Résultat attendu	Le nœud « Filtre nouvelle réservation » retrouve l'enregistrement déclencheur dans les résultats de Search records et calcule is_doublon.
Résultat obtenu	Erreur bloquante : « Enregistrement déclencheur introuvable dans Search records. triggerId=rechYNZbnVI4lqrFS. ids disponibles dans allRecords=[,] ».
Statut	KO

CT-04 — Re-exécution après correction de l'accès à l'identifiant (item.json.id)

Champ	Détail
Préconditions	Correction appliquée : remplacement de item.id par item.json.id dans toutes les comparaisons du nœud Code.
Données d'entrée	Idem CT-03 — réservation en chevauchement de créneau avec un enregistrement existant sur la même ressource.
Résultat attendu	is_doublon = true pour la réservation en chevauchement ; statut mis à jour en conséquence.
Résultat obtenu	Le workflow s'exécute sans erreur technique, mais is_doublon est retourné à false alors qu'un chevauchement de créneau existait sur la ressource testée. Un tableau de diagnostic (_debug) est ajouté au code pour comparer ressource et dates enregistrement par enregistrement.
Statut	KO

CT-05 — Validation finale sur données réelles

Champ	Détail
Préconditions	Correction finale appliquée (voir anomalie A-03, §7). Jeu de données réel : 9 réservations sur 3 ressources distinctes (Salle Eiffel, Salle Prado, Kit Visio Bordeaux).
Données d'entrée	9 enregistrements réels dans la table Réservations, dont un groupe de 3 réservations en chevauchement de créneau sur « Kit Visio Bordeaux » (id_reservation 8, 9 et 13 — pages 27/07 au 30/07 se recouvrant).

Champ	Détail
Résultat attendu	Les réservations en chevauchement sur une même ressource sont automatiquement marquées « Refusée » ; les réservations sans chevauchement conservent un statut cohérent (« En attente » ou « Confirmée »).
Résultat obtenu	Conforme. Les 3 réservations « Kit Visio Bordeaux » en chevauchement (id_reservation 8, 9, 13) sont toutes au statut « Refusée ». Les 6 autres réservations (Salle Eiffel, Salle Prado, Kit Visio Bordeaux non chevauchant) affichent des statuts « En attente » ou « Confirmée » cohérents avec l'absence de conflit.
Statut	OK

7. Anomalies détectées et corrections apportées

A-01 — Hypothèse erronée sur l'ordre du premier enregistrement

Champ	Détail
Cause racine	Le code supposait que l'enregistrement ayant déclenché le workflow était systématiquement le premier élément du tableau (allRecords[0]). Or l'ordre des résultats de « Search records » dépend de la vue Airtable configurée, pas de l'ordre de déclenchement du trigger.
Correction apportée	Récupération explicite de l'identifiant de l'enregistrement déclencheur via \$('Airtable Trigger').first().json.id, puis recherche de cet enregistrement précis (par id) dans les résultats de « Search records », plutôt que de supposer sa position dans le tableau.
Cas de test	CT-01 → CT-03
Statut	OK

A-02 — Confusion entre item.id et item.json.id

Champ	Détail
Cause racine	Dans un nœud Code n8n, chaque item est enveloppé dans une structure { json: {...} }. L'identifiant Airtable est accessible via item.json.id et non item.id (qui n'existe pas à ce niveau et vaut undefined). Le code comparait donc des valeurs undefined entre elles, provoquant un échec silencieux du matching.
Correction apportée	Remplacement de toutes les occurrences de item.id / currentRecord.id par item.json.id / currentRecord.json.id dans le nœud « Filtre nouvelle réservation ».
Cas de test	CT-03 → CT-04
Statut	OK

A-03 — Détection de chevauchement toujours négative malgré des données concordantes

Champ	Détail
Cause racine	Le champ ressource est un champ de liaison Airtable (lookup vers la table Ressources). Le code initial supposait qu'il se présentait toujours comme une simple chaîne ou un tableau de chaînes (String(res[0])). Or, selon le nœud d'origine, la même donnée peut être sérialisée différemment : le nœud Airtable Trigger et le nœud Search records ne renvoient pas nécessairement le champ ressource sous la même forme (chaîne d'identifiant brute d'un côté, objet { id, name } de l'autre selon la version/le mode du nœud Airtable). En appliquant String() directement sur un objet, la comparaison produisait la chaîne littérale « [object Object] » ou des valeurs non concordantes entre

Champ	Détail
	l'enregistrement déclencheur et les enregistrements comparés : la condition « même ressource » n'était donc jamais vraie, quelles que soient les dates.
Correction apportée	Deux corrections combinées, relues dans le code final : (1) la fonction parseRessource extrait désormais explicitement l'identifiant quelle que soit la forme reçue (chaîne, tableau de chaînes, tableau d'objets ou objet unique, via <code>res[0]?.id res[0] ""</code>), et tolère en plus un nom de champ alternatif (Ressources) ; (2) changement d'architecture — l'enregistrement « courant » n'est plus recherché par id dans les résultats de Search records (source de l'anomalie A-02), mais utilisé directement depuis la charge utile du nœud Airtable Trigger ; Search records ne sert plus qu'à fournir la liste des AUTRES réservations à comparer. Cela supprime définitivement la classe d'erreur « enregistrement déclencheur introuvable ».
Cas de test	CT-04 → CT-05
Statut	OK

Suivi requis

Le champ `_debug_logs` est toujours renvoyé en sortie du nœud dans le code final, et le `break` qui arrêta la boucle dès le premier doublon trouvé a été volontairement retiré à des fins de diagnostic (la boucle continue désormais jusqu'au bout, donc `conflicting_record_id` peut désigner le dernier conflit trouvé plutôt que le premier). Avant mise en production : (1) retirer `_debug_logs` de la sortie ; (2) réintroduire un `break` après `isDoublon = true` pour un résultat déterministe et une exécution plus légère ; (3) harmoniser dans la base le nom du champ (ressource vs Ressources) pour ne conserver qu'une seule source de vérité plutôt que de gérer les deux en défensif ; (4) formaliser ce cas comme test de non-régression permanent (voir §9.3).

8. Synthèse des résultats

Cas	Objet	Statut final	Itérations
CT-01	Exécution isolée du nœud Code	KO	1
CT-02	Vérification structure Airtable Trigger	INFO	1
CT-03	Exécution complète depuis le déclencheur	KO	1
CT-04	Re-exécution après correction A-02	KO	1
CT-05	Validation finale sur données réelles	OK	3

Sur les 5 cas exécutés, la fonctionnalité a nécessité 3 corrections successives (A-01, A-02, A-03) avant d'atteindre une validation fonctionnelle complète sur données réelles (CT-05). Cette itération est cohérente avec une démarche de recette rigoureuse : chaque anomalie a été identifiée par un résultat obtenu explicite (message d'erreur ou constat fonctionnel), tracée, corrigée puis re-testée.

9. Restes à construire et à tester

Le tableau ci-dessous liste, par catégorie, ce qui doit encore être construit et/ou recetté au-delà du périmètre couvert par ce document.

9.1 Automatisations n8n restantes

- Automatisation n°2 — validation manuelle des réservations « sensibles » (valeur assurée > 5 000 €) : les nœuds existent déjà dans le workflow (Valeur assurée >5000?, En attente, Confirmée, envois d'email) mais n'ont pas fait l'objet d'une recette dédiée dans ce document (déclenchement du circuit de validation, notification du gestionnaire, mise à jour du statut après décision manuelle).
- Automatisation de suppression RGPD (article 17) : hors périmètre du projet à ce stade, non planifiée.

9.2 Corrections techniques à finaliser avant mise en production

- Remplacer l'adresse email en dur du nœud « Envoi demande de validation » (gregory.tudela@gmail.com) par un champ dynamique (email du gestionnaire de site concerné), une fois ce champ disponible dans la base.
- Retirer le champ de diagnostic `_debug_logs` de la sortie du nœud Code — encore présent dans le code final relu (voir A-03).
- Réintroduire un break dans la boucle de comparaison dès qu'un doublon est détecté (`isDoublon = true`) : il a été volontairement retiré pour le diagnostic, ce qui rend `conflicting_record_id` non déterministe (peut pointer vers le dernier conflit trouvé plutôt que le premier) et fait tourner la boucle plus longtemps que nécessaire.
- Harmoniser dans la base Airtable le nom du champ ressource (vs Ressources) afin de ne conserver qu'une seule source de vérité, plutôt que de gérer les deux noms en défensif dans le code (`parseRessource`).
- Réintroduire une garde de type `isNaN` sur les dates (`creneau_debut` / `creneau_fin`) pour distinguer explicitement une date invalide d'une simple absence de chevauchement — actuellement, une date invalide produit un NaN qui échoue silencieusement dans le test de chevauchement sans être signalé.
- Ajouter un filtre (`filterByFormula`) sur le nœud « Search records » pour exclure au minimum les réservations déjà « Refusée », afin de limiter la charge à mesure que la table grossit.

9.3 Tests fonctionnels à planifier

- Cas limite : chevauchement exact à la frontière (`creneau_fin` d'une réservation = `creneau_debut` d'une autre) — la formule actuelle (opérateurs stricts `<` et `>`) considère ce cas comme non chevauchant ; à valider avec le métier.
- Cas de 3 réservations ou plus en chevauchement simultané sur une même ressource — observé empiriquement en CT-05 sur « Kit Visio Bordeaux » (`id_reservation` 8, 9 et 13, toutes refusées) : la règle métier de départage n'est pas formalisée et doit faire l'objet d'un cas de test dédié.
- Réservation avec le champ ressource vide ou non renseigné.
- Réservation en conflit avec un enregistrement déjà au statut « Refusée » — vérifier que ce dernier est bien ignoré par le contrôle (comportement actuellement attendu du code, à confirmer par un test explicite).
- Test de volumétrie : comportement du workflow avec un nombre important de réservations historiques dans la table.

9.4 Interfaces Softr (non construites à ce stade)

- Interface Adhérent (dépôt de demande de réservation).
- Interface Gestionnaire de site (validation des réservations sensibles, annulation exceptionnelle).
- Interface Coordinateur réseau (tableau de bord indicateurs).

9.5 Rappel — hors périmètre projet

- Plafond de 2 réservations actives par adhérent.

10. Validation

La fonctionnalité de contrôle des doublons de réservation (WF01_OPENSPACES_Controle_Doublons) est validée fonctionnellement sur données réelles à l'issue du cas de test CT-05, sous réserve des actions de suivi listées au §9.2 avant toute mise en production.

Rédacteur	Date	Statut
Gregory Tudela	2026-07-22	OK