



CHECKLIST IMPRIMABLE

Rendre les relances de paiement sans risque de doublon

Utilisez cette fiche pour concevoir, vérifier et tester une API idempotente pour les paiements ou les synchronisations. L'objectif : qu'une même opération métier reste une seule opération, même après une relance ou un traitement concurrent.

Au sommaire

- 1 Les repères à garder en tête
- 2 Concevoir le contrat de l'API
- 3 Mettre en œuvre le traitement côté serveur
- 4 Choisir le bon niveau d'idempotence
- 5 Tester avant la mise en production

Les repères à garder en tête

Une clé d'idempotence identifie une opération métier unique, comme la création d'un paiement ou d'une commande.

Le serveur conserve la clé, l'empreinte de la requête et la réponse produite dans un stockage persistant.

L'idempotence réduit les doublons, mais ne remplace ni une transaction locale ni le rapprochement avec le prestataire de paiement.

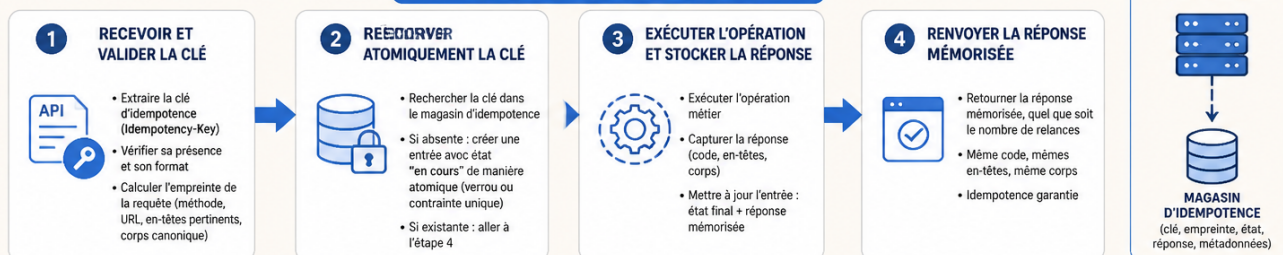
Une relance doit réutiliser la même clé : elle ne doit pas être interprétée comme une nouvelle opération.

L'idempotence, l'unicité en base de données et l'atomicité sont trois mécanismes différents et complémentaires.

TRAITEMENT D'UNE RELANCE IDEMPOTENTE CÔTÉ SERVEUR

Objectif : garantir qu'une même requête (même intention) ne produit qu'un seul effet, même en cas de relances réseau ou de timeouts.

FLUX DE TRAITEMENT CÔTÉ SERVEUR



ÉTATS POSSIBLES D'UNE CLÉ D'IDEMPOTENCE



EN COURS

La clé a été réservée, l'opération est en cours d'exécution. Aucune réponse finale n'est encore disponible.



RÉUSSE

L'opération s'est terminée avec succès. La réponse est stockée.



ÉCHOUÉ DÉFINITIVEMENT

L'opération a échoué de manière non transitoire (ex. : erreur de validation). La réponse d'erreur est stockée.



RÉSULTAT INCONNU

Le serveur ne sait pas si l'opération a abouti (ex. : timeout après exécution, crash avant persistance).

Réponse aux relances : 409 Conflict ou 425 Too Early (Réessayez plus tard)

Réponse aux relances : 200/201/... + réponse mémorisée

Réponse aux relances : Même code d'erreur + réponse mémorisée

Réponse aux relances : 409 Conflict (ou 425 Too Early) Une compensation peut être nécessaire côté métier.

RÈGLE FONDAMENTALE



EMPREINTE IDENTIQUE = RÉPONSE MÉMORISÉE



EMPREINTE DIFFÉRENTE = REJET

DÉTAILS

- L'empreinte est calculée à partir des éléments significatifs de la requête (méthode, URL, en-têtes pertinents, corps canonique).
- Si la même clé est réutilisée avec une empreinte différente, la requête est rejetée (400 Bad Request ou 409 Conflict).



BONNES PRATIQUES



TTL / RÉTENTION
Conserver les entrées pendant une durée définie (ex. : 24h, 7 jours).



ATOMICITÉ
Utiliser une contrainte unique ou un verrou distribué.



RÉPONSE COMPLÈTE
Mémoriser le code, les en-têtes et le corps pour garantir une réponse strictement identique.



OBSERVABILITÉ
Tracer les clés, états, empreintes et décisions pour diagnostiquer.



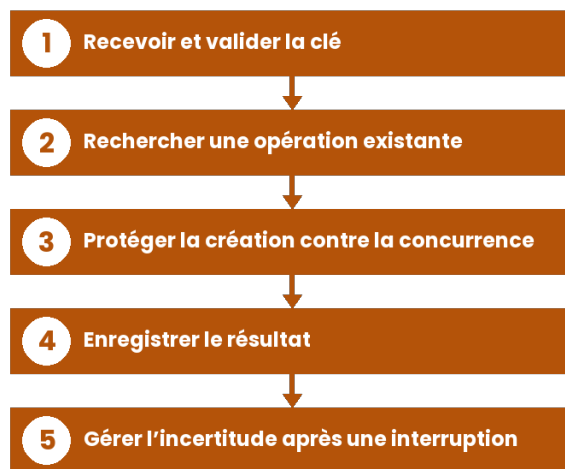
NETTOYAGE
Purger les entrées expirées pour maîtriser la taille du magasin.

INFOGRAPHIE Schéma du traitement d'une relance par une API idempotente

Concevoir le contrat de l'API

- Définir l'opération métier rendue idempotente**
Par exemple : créer un paiement, une commande ou une synchronisation.
- Générer une clé distincte pour chaque opération métier**
Une nouvelle opération doit avoir une nouvelle clé ; une relance doit conserver la clé initiale.
- Transmettre la clé avec la requête**
Le client doit pouvoir la réutiliser après un délai d'attente ou une erreur réseau.
- Associer la clé à une empreinte de la requête**
Cela permet de détecter qu'une même clé est réutilisée avec des paramètres différents.
- Documenter le comportement d'une relance**
Préciser la réponse attendue lorsqu'une opération est déjà enregistrée ou encore en cours.

Mettre en œuvre le traitement côté serveur



1 Recevoir et valider la clé

Vérifiez que la requête contient une clé d'idempotence exploitable et que ses paramètres sont cohérents avec l'opération demandée.

2 Rechercher une opération existante

Consultez le stockage persistant avant de lancer un nouveau traitement. Si la clé correspond déjà à une réponse produite, restituez cette réponse au lieu de recréer l'opération.

3 Protéger la création contre la concurrence

Empêchez deux instances, workers ou requêtes simultanées de traiter la même clé comme deux opérations distinctes. Une contrainte d'unicité seule ne suffit pas à restituer la réponse initiale.

4 Enregistrer le résultat

Conservez la clé, l'empreinte de la requête et le résultat produit afin qu'une relance puisse retrouver le même état.

5 Gérer l'incertitude après une interruption

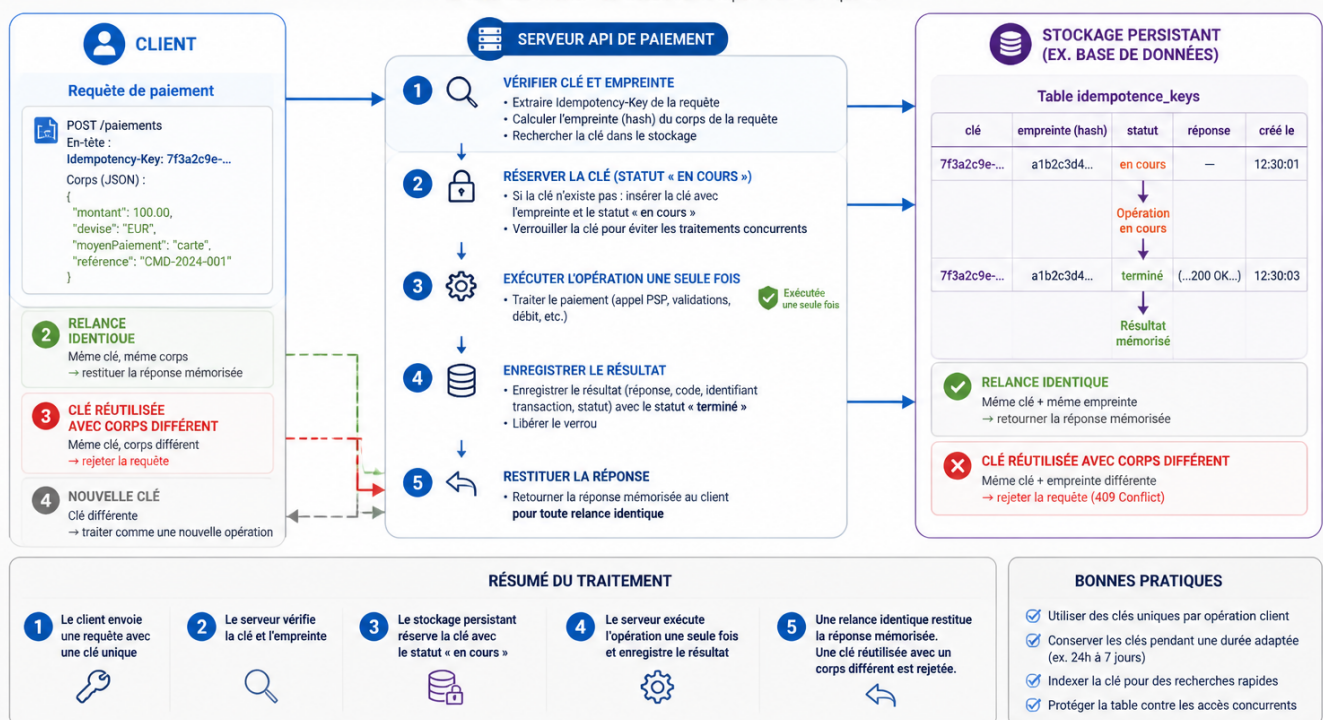
Si le prestataire a peut-être traité le paiement mais que la réponse n'est pas arrivée, ne créez pas automatiquement un second paiement : vérifiez l'état enregistré et prévoyez un rapprochement.

Choisir le bon niveau d'idempotence

Notion	Ce qu'elle garantit	Ce qu'elle ne garantit pas
Idempotence	Une répétition de la même opération métier produit un seul effet effectif.	Elle ne remplace pas la transaction de base de données ni le suivi d'un prestataire externe.
Unicité	Une valeur, comme un identifiant de commande, ne peut pas apparaître deux fois dans une table.	Elle ne garantit pas la restitution de la même réponse lors d'une relance.
Atomicité	Un ensemble d'actions locales est validé entièrement ou annulé entièrement.	Elle ne contrôle pas directement le résultat obtenu auprès d'un service de paiement externe.

TRAITEMENT D'UNE CLÉ D'IDEMPOTENCE DANS UNE API DE PAIEMENT

Le mécanisme d'idempotence garantit qu'une opération de paiement n'est exécutée qu'une seule fois, même en cas de relances réseau ou de répétitions de requêtes.



① Ce schéma est un guide pédagogique pour comprendre le traitement d'une clé d'idempotence dans une API de paiement. | PSP : Prestataire de Services de Paiement

SCHEMA La clé d'idempotence relie la requête du client, le traitement serveur et l'enregistrement durable du résultat.

Tester avant la mise en production

- Relancer une requête après un délai d'attente**
Vérifier qu'elle ne crée pas un second paiement ou une seconde ressource.
- Répéter une requête après une réponse perdue**
La réponse restituée doit refléter l'opération déjà traitée.
- Envoyer plusieurs requêtes simultanément avec la même clé**
Une seule opération doit être effective.
- Réutiliser une clé avec une requête différente**
Le serveur doit détecter l'incohérence grâce à l'empreinte de la requête.



Rejouer un message ou un webhook

Vérifier que les livraisons répétées ne déclenchent pas deux fois le même effet métier.



Tester l'expiration des clés

Contrôler le comportement prévu lorsqu'une clé n'est plus conservée.

! Point de vigilance

Une réponse HTTP absente ne prouve pas que le paiement a échoué. Le prestataire peut avoir accepté l'opération avant l'interruption réseau. Avant toute nouvelle tentative, utilisez la même clé et prévoyez un rapprochement avec l'état du prestataire.

Pour la sémantique générale des méthodes HTTP, se reporter à la RFC 9110. Une méthode HTTP classée idempotente ne garantit pas, à elle seule, que l'implémentation applicative respecte cette propriété.