SAKIRA

ARTICLE

Le programme d’abord, Jev pour le oui ou non

Saray Hach · 4 novembre 2026

© SAKIRA

sakira.fr
Texte de Saray Hach, © 2026 Saray Hach, publié par SAKIRA SAS
SAKIRA SAS · 6 rue d'Armaille, 75017 Paris · SIREN 108 493 933

Le programme d'abord, Jev pour le oui ou non

Saray Hach, fondateur de SAKIRA

SOMMAIRE

1. 1. Délimiter ce que Jev doit décider
2. 2. Essayer les quatre issues sans appeler Jev
3. 3. Prévoir le cas où Jev ne sait pas répondre
4. 4. Éprouver la décision avant de s'en servir

Le programme d'abord, Jev pour le oui ou non

Tout problème n'a pas besoin d'un modèle qui raisonne : le programme règle ce qu'une règle sait écrire, et Jev, le modèle de TypeSafe, répond par oui, non ou une catégorie à la question de jugement qui reste. [Documentation TypeSafe](https://docs.typesafe.ai/introduction/quickstart).

Nous avons fait classer 102 alertes internes en « stress » ou « calme » : un agent qui raisonne a mis 80 secondes et consommé 138 000 jetons, Jev a pris 0,4 seconde par cas pour moins d'un centime, et rendu le même verdict que l'agent sur 93 des 102.

Voici comment brancher Jev dans un programme, avec une demande d'aide à classer et un code à essayer avec quatre réponses simulées.

1. Délimiter ce que Jev doit décider

Dans notre exemple, nous voulons préparer le classement d'une demande. La question est : « Ce message décrit-il une difficulté à utiliser le produit ? » Le texte à examiner est : « Le bouton de lecture ne lance rien. »

Cette question appelle un oui ou un non. TypeSafe nomme ce type de question **Noul**. Sa réponse contient une valeur entre 0 et 1 : la probabilité estimée du oui. Jev propose aussi des questions à choix parmi des options, et des évaluations sur des niveaux définis. [Types de questions](https://docs.typesafe.ai/introduction/quickstart).

Le résultat sert ici à proposer une rubrique de traitement. Il ne rédige pas une réponse au client et n'autorise aucune action sur son compte. Le texte reçu reste une donnée à examiner : une instruction qui s'y trouverait ne change pas les règles du programme.

Avant de choisir l'outil, trois voies sont possibles :

Une règle écrite dans le programme

Exemple et moyens nécessaires. Un formulaire demande directement « Aide à l'utilisation » ou « Question sur l'offre ». Il faut concevoir et tester ce formulaire.

Gain recherché et limite. Le traitement est prévisible et sans appel à un modèle. Le client doit choisir la rubrique.

Une question fermée à Jev

Texte de Saray Hach, © 2026 Saray Hach, publié par SAKIRA SAS · sakira.fr
SAKIRA SAS · 6 rue d'Armaille, 75017 Paris · SIREN 108 493 933

Exemple et moyens nécessaires. Le client écrit librement ; un développeur prépare la question, branche le service et teste ses erreurs. Un budget d'appels est nécessaire.

Gain recherché et limite. Le classement peut tenir compte des formulations. Il peut aussi être faux ou incertain.

Une lecture humaine

Exemple et moyens nécessaires. Une personne lit chaque demande et choisit la suite. Il faut réserver du temps de traitement.

Gain recherché et limite. Elle peut demander des précisions et traiter les cas nouveaux. Sa disponibilité limite le débit.

Ces voies peuvent se compléter. Le temps économisé et le coût par demande doivent être relevés sur le cas choisi.

2. Essayer les quatre issues sans appeler Jev

Cet essai utilise des réponses fabriquées. Vous pouvez lire les quatre cas ci-dessous ou exécuter le code Python sans appeler Jev.

Le programme propose une rubrique, demande une lecture humaine ou signale un problème technique. Les quatre cas et leurs noms sont expliqués ci-dessous.

Les quatre situations ci-dessous sont **simulées** et choisies pour expliquer le programme.

0,93 : Aide à l'utilisation proposée

La valeur dépasse le seuil du oui choisi pour l'exemple. Dans le code : `aide_utilisation_proposee`.

0,06 : Accueil général proposé

La valeur passe sous le seuil du non. Aucune autre catégorie n'est déduite. Dans le code : `accueil_general_propose`.

0,52 : Lecture humaine

La règle refuse de choisir une rubrique. Dans le code : `lecture_humaine`.

Aucune réponse exploitable : Contrôle technique

Il faut vérifier l'appel ou son résultat. Ce n'est pas une hésitation du modèle. Dans le code : `controle_technique`.

Le code Python ci-dessous choisit la suite à donner. Les seuils de 0,20 et 0,80 sont **des choix pédagogiques**.

```python import math

def orienter(reponse): try: resultat = reponse["answers"]["difficulte"] p = resultat["noul"] if resultat["type"] != "noul": return "controle_technique" if type(p) not in (int, float): return "controle_technique" if not math.isfinite(p) or not 0 <= p <= 1: return "controle_technique" except (KeyError, TypeError): return "controle_technique"

if p >= 0.80: return "aide_utilisation_proposee" if p <= 0.20: return "accueil_general_propose" return "lecture_humaine"
```

Texte de Saray Hach, © 2026 Saray Hach, publié par SAKIRA SAS · sakira.fr
SAKIRA SAS · 6 rue d'Armaille, 75017 Paris · SIREN 108 493 933

Réponses fabriquées pour rejouer les quatre issues hors ligne. for p in (0.93, 0.06, 0.52, None): simulation = None if p is None else { "answers": {"difficulte": {"type": "noul", "noul": p}} } print(p, orienter(simulation)) ```

Cette simulation affiche successivement `aide_utilisation_proposee`, `accueil_general_propose`, `lecture_humaine` et `controle_technique`. Un non mène à l'accueil général : il ne permet pas de déduire quelle autre rubrique serait correcte. Entre oui et non, une personne relit la demande.

3. Prévoir le cas où Jev ne sait pas répondre

« Jev ne sait pas » désigne ici un résultat trop incertain pour notre règle. Jev n'écrit pas cette phrase : c'est le programme qui interprète le nombre et choisit de passer la main.

Pour une question Noul, une valeur proche de 0,5 indique l'incertitude ; il n'existe pas de champ `confidence` distinct. Pour les questions à choix et les évaluations, TypeSafe fournit aussi un champ `confidence`, calculé à partir de la répartition des probabilités. Ce nombre n'est pas un taux de réponses correctes relevé sur vos clients. [Probabilités et confiance](https://docs.typesafe.ai/confidence).

Si l'appel au service échoue ou dépasse le délai prévu, le programme appelant doit conserver la demande et signaler l'échec. Il ne doit pas fabriquer une valeur de 0,5 : cela masquerait une panne sous une apparence d'incertitude.

Pour les développeurs : préparer un appel réel

L'essai précédent ne nécessitait aucun accès au service. Les cinq étapes suivantes décrivent une intégration réelle, avec un accès configuré par une personne habilitée.

- Étape 1 : Préparer un texte inventé et une seule question, pour commencer sans données de clients.
- Étape 2 : Faire configurer l'accès au service par la personne habilitée. L'identifiant d'accès reste hors du code et des journaux.
- Étape 3 : Envoyer la requête ci-dessous à `https://api.typesafe.ai/v1/systemone`, par une requête HTTP POST authentifiée, au format JSON.
- Étape 4 : Lire le champ `answers.difficulte.noul` de la réponse, puis lui appliquer la fonction Python présentée plus haut.
- Étape 5 : Afficher la rubrique proposée pour vérification. Le branchement présenté ne contacte personne et ne modifie aucun compte.

Corps de la requête, avec un texte inventé pour l'exemple :

```json { "model": "jev-latest", "state": "Le bouton de lecture ne lance rien.", "questions": { "difficulte": { "type": "noul", "instructions": "Ce message décrit-il une difficulté à utiliser le produit ?" } } } ```

L'adresse, le format de requête et le champ de réponse sont documentés par TypeSafe. L'alias `jev-latest` sert à cet exemple ; avant un usage réel, il faudra choisir une version disponible, la consigner et réévaluer les résultats lors d'un changement. [Référence de l'API](https://docs.typesafe.ai/api/system-one).

Texte de Saray Hach, © 2026 Saray Hach, publié par SAKIRA SAS · sakira.fr
SAKIRA SAS · 6 rue d'Armaille, 75017 Paris · SIREN 108 493 933
```

4. Éprouver la décision avant de s'en servir

Pour un premier essai, préparez des exemples dont une personne a déjà déterminé le traitement attendu : demandes claires, formulations vagues, sujets mêlés et messages qui ne correspondent pas à la question. Gardez une partie de ces exemples pour vérifier le réglage après avoir choisi les seuils.

Relevez les erreurs de classement, les demandes renvoyées à une personne, les appels en échec, le temps de traitement et le coût. Comparez ces résultats à la règle simple ou à la lecture humaine. Un classement rapide qui oblige à corriger davantage de demandes peut être un mauvais choix.

Il faut aussi tester le programme quand la valeur manque, sort de l'intervalle prévu ou arrive dans un format inattendu. Dans l'exemple, ces cas mènent au contrôle technique. Une forte valeur ne dispense pas de ces vérifications, ni des autorisations nécessaires à une action réelle.

Cette séparation rend le fonctionnement explicable : une question de contexte pour Jev, puis des issues écrites et testées dans le programme. Elle permet également de conserver une lecture humaine lorsque le modèle ne permet pas de trancher. Pour commencer, choisissez une opération réversible, préparez vos cas d'essai et comparez les résultats avant d'élargir son rôle.

Essayez le code Python avec les quatre réponses simulées.

© SAKIRA

Texte de Saray Hach, © 2026 Saray Hach, publié par SAKIRA SAS · sakira.fr
SAKIRA SAS · 6 rue d'Armaille, 75017 Paris · SIREN 108 493 933