



Comment les développeurs intègrent l'optimisation de tournées

Comment celles et ceux qui développent ou intègrent eux-mêmes la solution traitent l'optimisation de tournées aujourd'hui : pratiques, difficultés, langages et outils, et ce qu'ils attendent d'une API.



Qui a été interrogé, et comment lire ces chiffres

Kardinal a interrogé 145 professionnels, majoritairement de la logistique et de l'édition logicielle, sur l'optimisation de tournées : comment ils la traitent, ce qui coince, les langages et outils qu'ils utilisent, et ce qu'ils attendent d'une API. Ce rapport se concentre sur celles et ceux qui développent ou intègrent eux-mêmes la solution.

99répondants
techniques**145**

répondants au total

15

questions présentées

Été 2026

1 juil → 8 sept

Ce qu'il faut retenir

Les enseignements principaux, avec les chiffres marquants. Le détail question par question suit ensuite.

56 %**Le problème, c'est le terrain**

Le décalage avec les conditions réelles est la difficulté n°1 (56 %), devant la performance / temps de calcul (47 %) et les besoins métier mouvants (43 %).

45 %**Le fait maison, mais qui coûte cher**

L'auto-hébergé reste le modèle privilégié (45 %) et le développement interne domine (56 %). Or, parmi ceux qui opèrent leur propre stack, la moitié déclare une maintenance régulière ou lourde.

1 sur 3**Les agents IA, déjà là**

Un tiers des répondants utilise un agent dans son workflow quotidien, et seuls 17 % n'en utilisent aucun. Claude Code est de loin le plus cité.

≈ 2 sur 3**Le prix public compte**

Près des deux tiers sont fortement gênés ou refusent même d'évaluer une API sans tarification publique. Documentation et exemples de code priment aussi.

Qui a répondu

Type d'organisation, rôle, expérience et implication dans l'optimisation de tournées.

Q1 Pour quel type d'organisation travaillez-vous ?

Choix unique · base 99

Opérateur logistique / transport		39 %	39
Éditeur de logiciels / SaaS		36 %	36
Agence / conseil / intégrateur		10 %	10
Autre		14 %	14

→ L'échantillon technique est réparti entre **opérateurs logistiques** et **éditeurs de logiciels**, au coude à coude, devant les agences et les autres organisations.

Q2 Quel est votre rôle principal ?

Choix unique · base 99

Développeur / ingénieur		34 %	34
CTO / responsable ingénierie		27 %	27
Tech lead / architecte		18 %	18
Autre rôle technique		20 %	20

→ **Développeurs et ingénieurs forment le premier groupe**, devant les CTO et les tech leads.

Q3 Avez-vous déjà travaillé sur l'optimisation de tournées ?

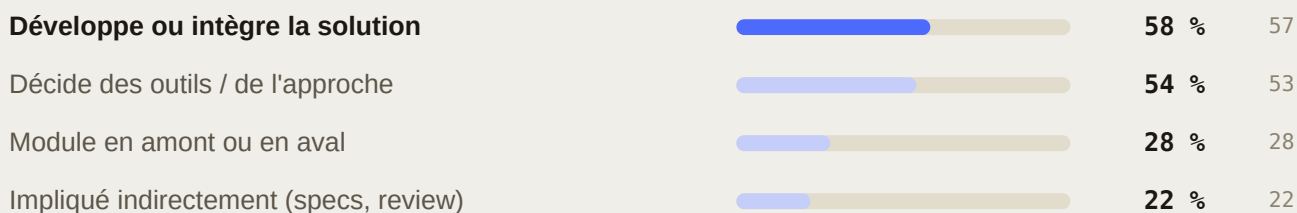
Choix unique · base 99

Régulièrement		44 %	44
Occasionnellement		32 %	32
Pas directement, mais familier		16 %	16
Exploré, jamais implémenté		7 %	7

→ La grande majorité travaille **régulièrement ou occasionnellement** sur le sujet.

Q4 Comment êtes-vous impliqué dans l'optimisation ?

Choix multiple • base 99



→ Une majorité **développe ou intègre elle-même** la solution, et plus de la moitié décide des outils ou de l'approche.

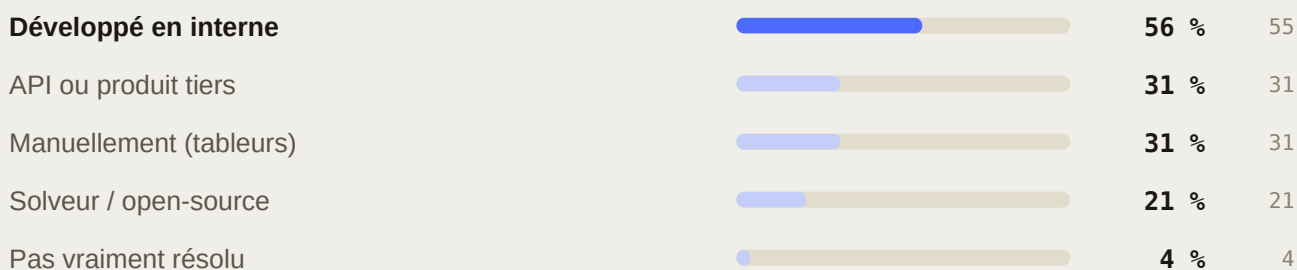
Un échantillon pertinent et qualifié : des experts techniques qui, pour les trois quarts, travaillent eux-mêmes sur l'optimisation de tournées, en choisissent les outils ou en intègrent les solutions, et connaissent donc très bien le sujet et ses difficultés.

Comment le problème est traité, et ce qui coince

La façon dont l'optimisation de tournées est traitée aujourd'hui, les difficultés rencontrées et la charge de maintenance.

Q5 Comment le problème est-il traité aujourd'hui ?

Choix multiple • base 99



→ Le **développement interne domine** ; l'API tierce et le manuel suivent à égalité, devant l'open-source. Un tiers des répondants (33 %) combine au moins deux approches ; parmi ceux qui développent en interne, près de la moitié s'appuient aussi sur une autre approche.

Q5 Qui combine quoi : un tiers combine plusieurs approches

Co-occurrence des approches · lecture par ligne

UTILISE...	+ INTERNE	+ API TIERCE	+ MANUEL	+ OPEN-SRC	UNE SEULE
Développé en interne (55)	—	20 %	24 %	18 %	53 %
API ou produit tiers (31)	35 %	—	19 %	26 %	48 %
Manuel (31)	42 %	19 %	—	16 %	42 %
Open-source (21)	48 %	38 %	24 %	—	33 %

→ Se lit par ligne : parmi ceux qui **développent en interne**, 53 % ne font que ça. Le groupe le plus hybride est l'**open-source** : seul un tiers s'y limite.

Q6 Quel(s) outil(s) ou bibliothèque(s) en particulier ?

Champ libre · 37 réponses

Trois familles d'outils reviennent nettement, citées ici par ordre d'occurrence :

■ Solveurs open-source

1. OR-Tools
2. VROOM
3. Timefold
4. pgRouting

■ Cartographie et routage

1. Google Maps
2. ESRI / ArcGIS
3. GraphHopper
4. OpenRouteService
5. OSRM
6. HERE

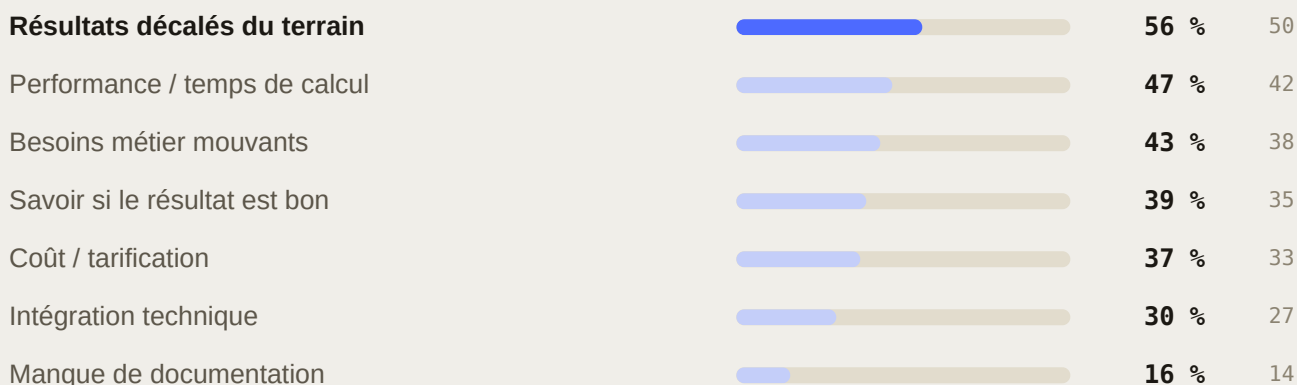
■ Plateformes / TMS

1. Kardinal*
2. Nomadia
3. Ortec
4. PTV
5. Urbantz

* Cette enquête étant diffusée par Kardinal, ses clients sont possiblement surreprésentés parmi les répondants.

Q7 Qu'est-ce qui a été le plus difficile ?

Choix multiple • base 89



→ Le **décalage avec les conditions réelles** domine nettement, devant la performance et les besoins métier mouvants.

“

Le principal angle mort des solutions actuelles est la validation terrain : il est difficile de savoir si une tournée optimisée est réellement meilleure qu'une tournée construite empiriquement par un chauffeur expérimenté. Des métriques d'écart au réel seraient plus utiles que des métriques purement algorithmiques.

Un répondant, en réponse libre

Q8 Quelle charge de maintenance demande votre stack ?

Choix unique • conditionnelle • base 60 • opère sa propre stack



→ Posée uniquement à ceux qui opèrent leur propre stack. La charge reste soutenue : **la moitié la déclare régulière ou lourde.**

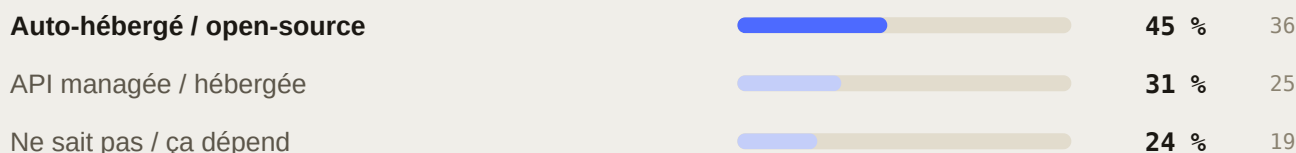
Un vrai point de douleur. La majorité des équipes développe sa propre solution (56 %), près d'une fois sur deux en complément d'une autre approche, alors que la maintenance est régulière ou lourde pour la moitié de celles qui opèrent leur stack. Les difficultés touchent une large part des répondants (jusqu'à 56 %), quelle que soit l'approche : l'optimisation de tournées reste un sujet complexe, mal adressé par les solutions existantes, maison, open-source ou API tierces.

Ce que privilégient les équipes de développement

Le modèle de déploiement préféré, les langages utilisés, et ce qui compte pour adopter une API.

Q9 Vers quel modèle penchez-vous ?

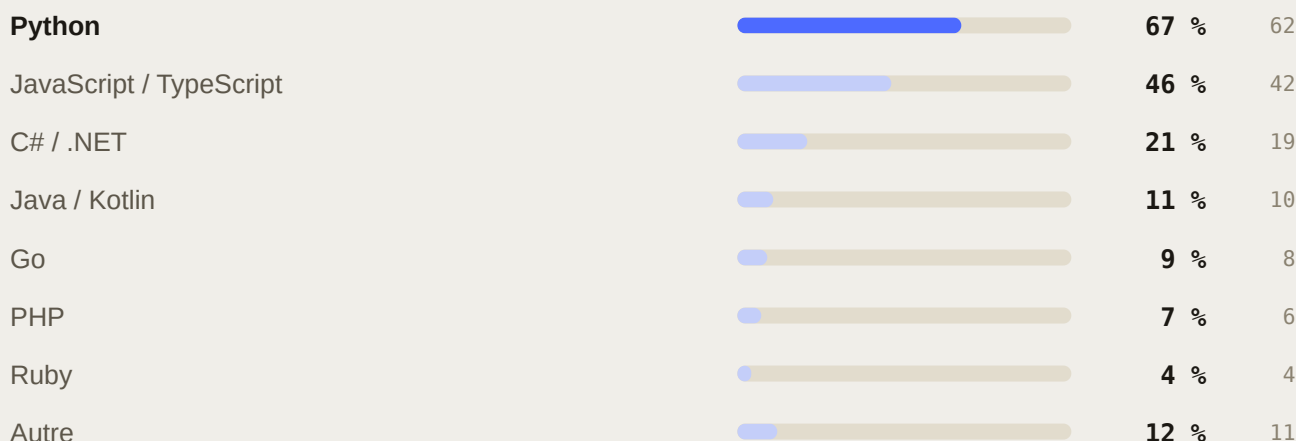
Choix unique • base 80



→ L'auto-hébergé reste le modèle privilégié, devant l'API managée, avec une part notable d'indécis.

Q10 Quels langages utilisez-vous ?

Choix multiple • base 92



→ Python domine largement, JavaScript / TypeScript suit.

Q11 Au moment d'adopter une API, qu'est-ce qui compte le plus ?

Classement forcé de 7 items · base 92 · par rang moyen croissant

CRITÈRE	RANG MOYEN	CLASSÉ 1ER	TOP 3
1 Documentation claire et complète	3,37	26 %	59 %
2 Exemples de code / templates	3,60	14 %	58 %
3 Conseils pour bien modéliser	3,85	18 %	39 %
4 Sandbox pour tester sans engagement	3,98	7 %	46 %
5 Tarification prévisible et transparente	3,99	18 %	39 %
6 SDK bien conçu dans mon langage	4,50	10 %	35 %
7 Temps réduit jusqu'au premier appel	4,72	7 %	25 %

→ **Documentation et exemples de code** dominent les critères d'adoption ; le classement reste peu polarisé (rangs moyens resserrés).

Le choix d'une API se fait d'abord sur des éléments de réassurance essentiels : une documentation claire, des exemples de code, des conseils pour bien modéliser son problème. Autrement dit : l'API est-elle vraiment pensée pour des développeurs ?

La place des agents IA

Usage réel des agents de développement et outils cités.

Q12 Quelle place occupent les agents IA dans votre workflow ?

Choix unique · base 92

Pas du tout		17 %	16
Expérimenté, rien en prod		20 %	18
Occasionnellement		23 %	21
Dans le workflow quotidien		35 %	32
Produits où des agents appellent des APIs		5 %	5

→ Les agents IA sont bien installés : **un tiers les utilise au quotidien**, et moins d'un cinquième n'en utilise aucun.

Q13 Quels outils d'agents utilisez-vous ?

Choix multiple • base 83 • a cité au moins un outil

Claude Code		81 %	67
GitHub Copilot		45 %	37
Google Antigravity / Gemini CLI		28 %	23
OpenAI Codex		25 %	21
Cursor		20 %	17
OpenCode		12 %	10
Cline		1 %	1
Aider		1 %	1

→ Parmi les utilisateurs d'agents, **Claude Code est très largement cité**. 65 % en citent au moins deux, soit 2,1 en moyenne.

Les agents IA deviennent la norme. Mesurés à l'été 2026, ces chiffres sont sans doute déjà obsolètes.

Tarification et prix caché

Les irritants de pricing et la réaction à une API sans prix public.

Q14 Quels aspects de la tarification vous dérangent le plus ?

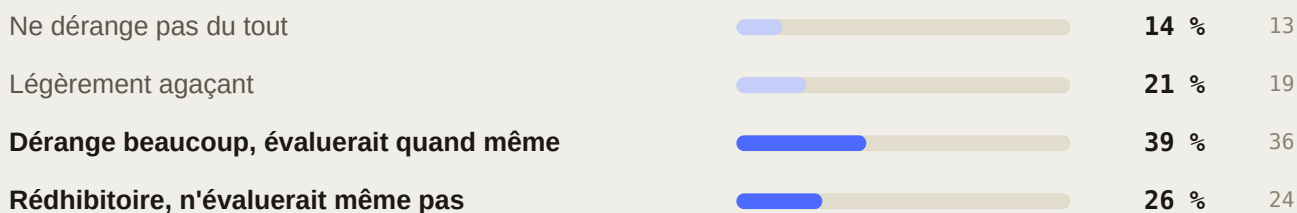
Choix multiple • base 92

Facture variable qui pourrait exploser		41 %	38
Service coupé en cours de mois		37 %	34
Surveiller son usage pour changer de plan		35 %	32
Estimer son volume à l'avance		27 %	25
Volume / crédits qui expirent		18 %	17

→ La **facture imprévisible** arrive en tête, devant le service coupé en cours de mois et la surveillance de l'usage.

Q15 Une API sans tarification publique : à quel point cela dérange ?

Choix unique · base 92



→ Près des **deux tiers** sont fortement gênés ou refusent même d'évaluer une API sans prix public.

Le marché attend un vrai self-service, à un coût maîtrisable. Documentation et exemples priment à l'adoption, près des deux tiers sont fortement gênés par une API sans prix public, et la facture imprévisible est le premier irritant. C'est cohérent avec le fait que près de la moitié privilégie l'auto-hébergé / open-source (45 %), qui coche a priori ces deux cases.

Ce que révèlent les réponses croisées

Les enseignements qui n'apparaissent qu'en croisant les réponses entre elles.

01 Faire soi-même ne règle pas le problème du terrain

Le décalage avec les conditions réelles touche autant les équipes qui développent en interne que les autres (55 % contre 58 %) : faire soi-même n'élimine pas la difficulté n°1, et expose davantage aux problèmes de performance (53 % contre 37 %) et d'intégration (35 % contre 24 %).

02 Claude Code est presque toujours accompagné d'un autre outil.

76 % des utilisateurs de Claude Code citent aussi un autre outil (2,4 en moyenne) : Copilot (40 %), Gemini (30 %), Codex (28 %), Cursor (22 %). Le multi-outillage est la norme.

03 Chaque secteur a sa signature de langages.

Python domine partout (logistique 56 %, éditeurs 70 %), mais le C# / .NET marque la logistique (33 % contre 12 %), quand les éditeurs sont deux fois plus sur JS / TS (64 % contre 31 %).

04 La sensibilité au prix caché n'est pas uniforme.

Les éditeurs sont les plus rebutés par une API sans prix public (76 % contre 53 % dans la logistique). La logistique est plus sensible à l'estimation de son volume à l'avance (28 % contre 18 %), les éditeurs à la surveillance de leur usage (45 % contre 25 %).

Une API pensée à partir de ces constats

Ces résultats orientent directement la façon dont nous construisons notre API d'optimisation de tournées.

LE CONSTAT

Le décalage avec les conditions réelles est la première difficulté (56 %), devant la performance et des besoins métier qui bougent sans cesse.

NOTRE RÉPONSE

Le décalage vient surtout de contraintes réelles que le modèle ne sait pas représenter. Notre moteur en couvre une part inhabituelle : 93 % d'un référentiel de 149 contraintes métier sur 8 secteurs, là où aucun des dix autres moteurs comparés n'atteint 75 % (benchmark d'août 2026). Nous publions aussi un guide et des exemples issus de nos cas réels.

LE CONSTAT

La documentation et les exemples de code priment au moment d'adopter une API, et près des deux tiers sont rebutés par une API sans tarification publique.

NOTRE RÉPONSE

Une API REST documentée, avec des schémas d'entrée / sortie explicites, des exemples de requêtes exécutables et une grille tarifaire publique. De quoi lire les specs, tester une requête et chiffrer une intégration sans parler à un commercial.

LE CONSTAT

Les principaux irritants de tarification : une facture imprévisible qui peut s'envoler, et le risque de voir le service coupé en cours de mois.

NOTRE RÉPONSE

Une tarification indexée sur une unité lisible et mesurable, le nombre de stops uniques par tranche de 24 h. Le comportement au-delà du plan est défini à l'avance : pas de coupure du service en cours de mois.

LE CONSTAT

Un tiers des répondants fait déjà tourner des agents IA au quotidien, et beaucoup peinent à juger si un résultat d'optimisation est réellement bon.

NOTRE RÉPONSE

Une API callable programmatiquement, qui renvoie des sorties structurées (JSON) accompagnées d'indicateurs sur la qualité de la solution et les contraintes respectées ou violées. Un agent peut ainsi juger un résultat et enchaîner l'action suivante.

LE CONSTAT

Beaucoup développent en interne ou s'auto-hébergent, mais la maintenance de cette stack reste régulière ou lourde pour une bonne moitié.

NOTRE RÉPONSE

Nous opérons l'optimisation en SaaS managé : l'API est appelée, nous prenons en charge le solveur, le tuning et les mises à jour d'algorithme. La maintenance de la brique d'optimisation ne repose plus sur vos équipes.

Enquête « Comment les développeurs intègrent l'optimisation de tournées » · profils techniques (99 sur 145) · collecte du 1er juillet au 8 septembre 2026 · publication septembre 2026 · données déclaratives, échantillon auto-sélectionné. Chiffres API et périmètre : communiqué de presse Kardinal du 28 septembre 2026.

L'API en libre accès

Tout ce qu'il faut pour passer de l'enquête à votre première optimisation.

Testez l'API sur vos propres tournées

Documentation claire, schémas d'entrée / sortie, exemples de requêtes exécutables et grille tarifaire publique. Créez un compte et lancez votre premier appel, sans passer par un commercial.

[Lire la documentation](#)[Ouvrir la console](#)

[developers.kardinal.ai](#) · [console.kardinal.ai](#)

1000

stops optimisés
offerts

dès 0,012 €

par stop optimisé

1x / 24 h

un stop facturé une
seule fois

Self-service

sans contact
commercial

Ce qui est inclus

- ✓ SDK Python et console développeur
- ✓ Clés API et suivi de consommation
- ✓ Documentation pensée pour la lecture machine
- ✓ Guides de modélisation sur cas industriels réels
- ✓ Exemples de requêtes exécutables
- ✓ Tarification publique et transparente



Dix ans d'optimisation logistique, à portée d'API.

Kardinal développe depuis 2015 des technologies d'optimisation de tournées pour les acteurs du transport et de la logistique, combinant recherche opérationnelle et machine learning pour modéliser les contraintes métier les plus complexes.

Après avoir accompagné transporteurs et chargeurs via des déploiements sur mesure, Kardinal ouvre aujourd'hui cette même expertise en API self-service, pensée pour être exploitée aussi bien par des équipes techniques que par des agents IA.

Reconnu Representative Vendor dans le Gartner Market Guide for Vehicle Routing and Scheduling 2025, renouvelé en 2026.

