Kong

Powering the API world

livre électronique

Guide du leader pour

Pourquoi les initiatives API échouent

KongHQ.com

Guide du leader sur les raisons de l'échec des initiatives API

Les initiatives API existent depuis longtemps, que ce soit sous le couvert de la « transformation numérique » ou de la modernisation technologique.

Ces initiatives commencent souvent de la même manière : avec les meilleures intentions, un programme de travail est lancé pour introduire des API RESTful — généralement aux côtés de microservices — dans une nouvelle façon de travailler, en se concentrant sur la réduction du délai de mise sur le marché et l'augmentation de la productivité des développeurs grâce à la réutilisation.

Une plateforme API est développée par une équipe plateforme afin d'offrir un accélérateur de production aux équipes développant des microservices. L'objectif est de fournir des fonctionnalités et des enjeux transversaux sous forme de service, permettant aux équipes de développement de s'appuyer sur ces fonctionnalités, réduisant ainsi le travail informatique et leur permettant de se concentrer sur la logique applicative (et, par extension, sur la valeur métier).

Kong collabore fréquemment avec des clients qui ont développé une plateforme, mais qui n'obtiennent pas les bénéfices escomptés. Bien sûr, chaque entreprise est confrontée à ses propres défis, mais nous constatons des raisons courantes expliquant l'échec d'un programme d'API. Dans ce rapport, nous examinerons ces éléments courants, leur impact et, surtout, les solutions possibles.

1. Manque de normes API et de bonnes pratiques

En adoptant une stratégie API-first, il est admis que les API constituent la partie centrale de la nouvelle entreprise, et toutes les applications doivent avoir un contrat et un SLA associé pour former cette entreprise composable.

Cependant, le plus souvent, les normes API sont une idée fugace destinée à rendre hommage à la gouvernance d'entreprise plutôt qu'un véritable ensemble de normes auxquelles tous les membres de l'organisation adhéreront.

L'objectif des normes est de fournir une base fiable permettant aux utilisateurs de partager les mêmes attentes concernant un produit ou un service. Lorsqu'une organisation souhaite que les API deviennent les éléments constitutifs d'une entreprise composable, c'est une nécessité absolue.

Pour ceux qui ont défini des normes, se pose bien sûr la question de leur application et de leur gouvernance. Par exemple, chaque API de la plateforme est-elle soumise à un contrôle de conformité à ces normes lors d'une publication ? Et en cas d'infraction grave, l'API est-elle bloquée ?

Un autre défi lié aux normes API définies au niveau du groupe réside dans le fait que les développeurs sont perçus comme des « architectes de leur tour d'ivoire » qui créent des attentes déraisonnables en établissant des normes perçues comme peu pratiques ou trop rigides. Cela conduit rapidement à une avalanche de normes ignorées, au lieu de créer des boucles de rétroaction appropriées entre architectes et développeurs pour intégrer les retours et s'accorder sur ces normes.

Impact:

- Pas de cohérence organisationnelle des API (demandez au question : cela serait-il acceptable sur un site Web ?) • Des approches très différentes des principales approches architecturales telles que le versionnage des API (une API utilise le versionnage basé sur les routes, une autre utilise les en-têtes) • Les API sont mal conçues et deviennent difficiles à consommer, ce qui diminue la réutilisation des API et conduit à la duplication

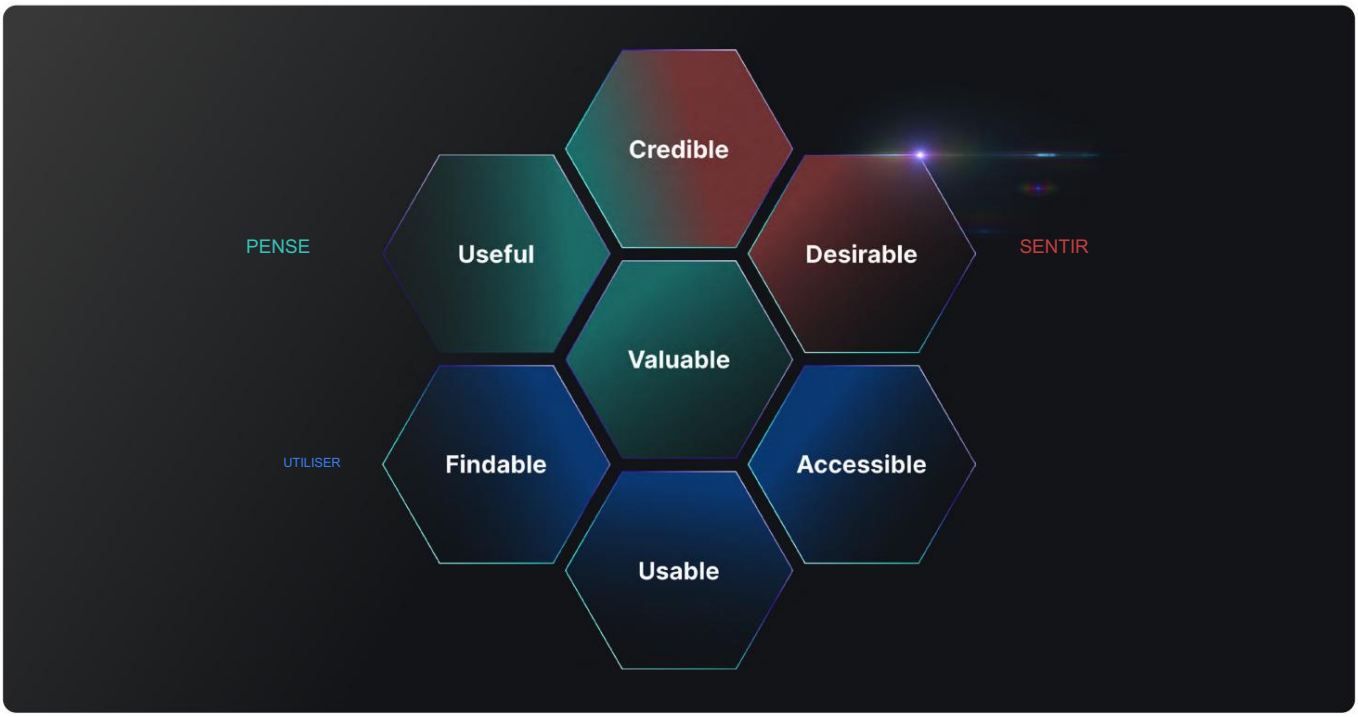
Remède:

- Mettre en œuvre des normes API qui répondent aux besoins organisationnels • Publier des normes et commencer à les appliquer pipelines (envisagez d'utiliser Kong Insomnia (écrire des ensembles de règles spectrales) • Implémenter MosCow pour la violation des normes avec des obligations entraînant le blocage d'un déploiement d'API via un APIOps pipeline
- Mettre en place des forums de commentaires pour solliciter les commentaires des développeurs d'API, et permettre de contester les normes et d'intégrer les commentaires

2. Absence d'un catalogue de services API unique avec une documentation de support riche

Le moyen le plus rapide de tuer une initiative API est peut-être de s'assurer que les API sont incroyablement difficiles à trouver et à consommer, que ce soit par le biais de catalogues d'API cloisonnés, d'outils médiocres ou de la non-publication automatique des API dans un catalogue, ce qui entraîne une dérive entre le service en direct et ce que la documentation indique.

Lorsque l'on considère ce qui rend une API précieuse, il est utile d'appliquer la perspective du nid d'abeilles UX développé par Peter Morville.



Trois de ces éléments en nid d'abeille sont remplis par un catalogue d'API organisationnel.

1. Le premier est la possibilité de trouver des API. Si un contrat d'API n'existe que dans le dépôt Git de l'équipe de développement, on peut supposer que cette API ne sera jamais utilisée. De plus, si un modèle de métadonnées n'est pas respecté dans une organisation de grande envergure, il peut être difficile de trouver des API.
   Un modèle de métadonnées permet des recherches de paires clé:valeur sur l'ensemble du domaine.

2. La seconde est utilisable. Lorsque l'API REST
   Il y a environ 10 ans, les spécifications étaient en compétition pour devenir la norme de l'industrie. Il y avait l'idée entre les différentes spécifications qu'elles serait entièrement auto-documenté — un développeur en créerait un et d'autres développeurs seraient alors capables de les comprendre intuitivement. 10 ans d'expérience nous ont montré que ce n'est pas le cas, et c'est l'une des principales raisons pour lesquelles les API finissent par être jetées dans les intégrations — d'autres développeurs consommateurs ne peuvent pas comprendre pourquoi ils utiliseraient une API particulière.

3. Enfin, un catalogue de services contribue à l'utilité d'une API.
   L'ensemble des ressources étant cataloguées et des règles de gouvernance globales (telles que le linting) appliquées à ces spécifications, associées à une documentation complète expliquant comment et pourquoi utiliser l'API, l'utilité de
   une API pour un développeur est considérablement augmentée.

Impact:

- Plusieurs sources de vérité cloisonnées, ce qui rend difficile la découverte d'API à partir d'un catalogue central, ce qui conduit à une duplication des efforts et à la prolifération des API ; les API sont difficiles à consommer car les développeurs ont du mal à comprendre le but et la fonctionnalité d'une API, ce qui conduit également à des doublons
- Faible visibilité des API dans l'ensemble de l'organisation, augmentant le risque d'incidents de sécurité car les API non gérées et inutilisées ne sont pas retirées ou corrigées lorsque des problèmes ou des vulnérabilités sont identifiés

Remède:

- Mettre en œuvre un catalogue de services qui est le source unique de vérité pour tous les actifs API de l'organisation — cela permet aux services d'être plus facilement suivis et gérés et permet aux équipes d'atténuer de manière plus proactive les vulnérabilités en éliminant les API non découvertes ou inutilisées au sein de l'organisation • Adopter une approche axée sur le produit, exigeant

  les développeurs pour créer une documentation plus riche pour leurs API qui s'étend au-delà de la simple spécification technique

- Mettre en œuvre un modèle de métadonnées pour aider à découvrir les API ainsi que documenter à qui appartient une API si un consommateur potentiel a des questions à ce sujet

3. Trop d'accent mis sur la réutilisation

Étant donné que l'une des principales raisons pour lesquelles les API sont créées est d'encourager les développeurs à réutiliser les ressources existantes plutôt qu'à les créer eux-mêmes, trop se concentrer sur la réutilisation est peut-être la raison la plus paradoxale pour laquelle les programmes API échouent.

Cependant, se concentrer uniquement sur la réutilisation peut conduire à confondre les moyens avec les fins des API. L'accent doit être mis sur la consommabilité. Si une API est consommable, elle est précieuse et la réutilisation en sera naturellement le résultat.

Il convient également de souligner que toutes les API ne sont pas nécessairement conçues pour être réutilisées. Par exemple, le back-end d'un modèle front-end est une API très spécifique qui ne sera utilisée que par le front-end qui l'utilise ; mesurer la réutilisation et considérer l'API comme non performante serait une erreur dans ce cas, comme pour de nombreux autres modèles.

Impact:

- API conçues à un niveau de granularité incorrect, ce qui augmente les cycles de développement • Les indicateurs clés de performance signalent une faible réutilisation des API non conçues pour être réutilisées, ce qui déroute les parties prenantes de l'entreprise • Ne pas se concentrer sur la consommabilité, ce qui entraîne un écosystème d'API de faible valeur

Remède:

- Se concentrer sur la consommabilité et mesurer la réutilisation comme un sous-produit des API à haute valeur ajoutée (voir UX honeycomb pour voir à quoi devrait ressembler une « bonne » API)
- Reconnaître que toutes les API ne se prêtent pas intrinsèquement à la réutilisation et mettre en place un filtrage approprié pour les supprimer des KPI

4. Aucun test de contrat API

Heureusement, l'époque où le code était envoyé en production sans aucune couverture de tests unitaires est révolue. Cependant, la plupart du temps, les API ne disposent pas de tests automatisés au niveau des contrats ou des fonctions pour prouver leur bon fonctionnement de bout en bout une fois déployées sur une plateforme. Toutes les API devraient au minimum disposer de tests contractuels.
Cela permet de confirmer automatiquement qu'une API remplit son contrat lorsqu'elle est envoyée à travers l'ensemble de l'écosystème, y compris une passerelle.

L'un des problèmes les plus courants liés aux API peut être atténué par l'introduction de tests mieux automatisés : le manque de confiance et les efforts nécessaires pour modifier l'API. Si un développeur doit effectuer manuellement des tests de régression de bout en bout à chaque modification de l'infrastructure sous-jacente ou de l'API elle-même (même s'il s'agit d'une modification non destructive), deux situations se produisent généralement :

1. Le développeur effectue des tests manuels, ce qui ouvre la voie à des risques de bugs et impacte gravement l'expérience du développeur (peu de développeurs souhaitent exécuter manuellement les commandes curl).

2. Les tests ne sont pas effectués, ce qui entraîne des bogues dans Production et faible confiance dans l'API. Ces deux comportements peuvent conduire une organisation à aggraver la situation par inadvertance en ne publiant pas fréquemment de petits changements et en introduisant des mises à niveau massives ou des versions à forte concentration de changements, ce qui entraîne intrinsèquement davantage de risques.

De plus, en n'exécutant pas de tests d'API tout au long du cycle de vie du développement logiciel, les développeurs ont tendance à ne pas détecter les bugs pendant la phase de développement, mais plutôt dans des environnements plus complexes. Une étude classique du Systems Sciences Institute d'IBM a estimé que le coût de correction d'un bug en production est 6,5 fois plus élevé que lorsqu'il est identifié pendant le développement.

Impact:

- Les correctifs et les mises à niveau des logiciels deviennent difficile en raison de la nécessité d'effectuer des tests manuels pour ce qui est perçu comme des changements de faible valeur
- Faible confiance dans la publication des API entraînant des versions moins fréquentes, entraînant une plus grande concentration des changements qui introduit ainsi davantage de risques commerciaux
- Verrouillage du fournisseur au niveau de la passerelle API en raison de la migration étant jugée trop risquée en raison de l'impossibilité de déterminer rapidement si une API a été migrée avec succès ou non
- Mauvaise expérience client en raison de introduire des modifications à l'API qui enfreignent leur SLA
- Les bugs restent non découverts jusqu'à plus tard dans les niveaux supérieurs environnements entraînant des coûts de correction plus élevés, plutôt que d'être corrigés au cours des sprints de développement

Remède:

- Mettre en œuvre des tests de contrat à l'aide d'Insomnia à écrire des tests API
- Envisagez l'utilisation du coureur de collection Insomnia pour développer des tests fonctionnels plus complets des API qui peuvent être exécutés dans le cadre d'un Pipeline CICD
- Surveiller activement la quantité de bugs signalés par environnement d'une API pour évaluer si la couverture des tests de l'API est suffisante

5. La plateforme API est trop difficile à intégrer

Un autre défi courant est que la plateforme, conçue pour accélérer les équipes de développement, est perçue par les développeurs d'applications comme trop difficile à intégrer et à utiliser. Cela est généralement dû aux efforts manuels nécessaires pour intégrer les développeurs à la plateforme, aux parcours mal documentés et au manque d'automatisation.
Dans certains cas, cela est encore plus compliqué car une équipe a des exigences uniques que la plateforme doit prendre en compte.

Ces problèmes entraînent des retards dans l'adoption de la plateforme, retardant souvent la livraison des projets et nuisant à sa valeur. De plus, sous la pression des parties prenantes de l'entreprise pour proposer de nouvelles fonctionnalités, les équipes applicatives choisissent la solution la plus simple et créent leur propre solution pour gérer leurs API. Cela crée un environnement informatique parallèle, tant en termes de technologies différentes (et donc de coûts) utilisées que de silos avec des API inaccessibles au reste de l'organisation.

Les API fantômes augmentent également le risque d'incident de sécurité en raison du non-respect des normes de sécurité organisationnelles et de l'absence d'un volet unique pour toutes les données de conformité sur les postures de sécurité de chaque API à interroger.

Impact:

- Les API deviennent fragmentées et cloisonnées, ce qui limite la capacité des API à être socialisées au sein de l'organisation pour la consommation
- Augmentation des coûts due aux dépenses en logiciels (et à la maintenance) et au manque d'économies d'échelle avec l'infrastructure
- Les normes organisationnelles ne peuvent pas être appliquées entraînant un risque accru d'incidents de sécurité • Augmentation de la complexité de la migration pour rassembler toutes les API dans une seule plate-forme unifiée lorsque les plates-formes sont finalement consolidées

Remède:

- Améliorer la documentation et rendre l'adoption de la plateforme plus transparente pour les équipes de développement en utilisant PlatformOps pour disposer de pipelines capables d'intégrer automatiquement une équipe d'application après avoir rassemblé toutes les informations requises en une seule interaction (par exemple, un ticket Jira, une demande ServiceNow ou une intégration Slackbot)
- Si PlatformOps est suivi, toute l'infrastructure de la plateforme doit être stockée en tant qu'infrastructure en tant que code, permettant l'internalisation de la base de code ; si une équipe d'application a des exigences uniques, autorisez les développeurs à s'auto-servir une solution via une demande d'extraction dans cette base de code.
- Introduire une gouvernance pour éviter les risques de prolifération technologique résultant de l'autorisation de l'informatique fantôme. • Envisager si la fédération est approprié modèle d'exploitation et fournir un modèle de locataire partagé (instances de passerelle gérées centralisées), réduisant les frais généraux d'intégration et de gestion de l'infrastructure fédérée

- Envisagez d'utiliser les passerelles cloud dédiées Kong pour réduire les frais d'intégration et de gestion des passerelles Kong au sein des équipes fédérées

- Assurer une solution bien documentée et facile à utiliser Le pipeline APIOps est disponible pour réduire le temps et les efforts nécessaires à l'intégration des API sur la plateforme

6. Permettre différentes manières de travailler sur différentes technologies de mise en œuvre

Une plateforme d'API doit être totalement agnostique et prendre en charge un environnement polyglotte, où chaque développeur d'API est libre d'utiliser les meilleurs outils et langages de programmation pour résoudre ses problèmes. Une gestion des API différente selon les technologies entraîne un verrouillage fournisseur et favorise des méthodes de travail différentes, ce qui rend difficile la portabilité des implémentations d'API entre les technologies.

Une complexité supplémentaire est également introduite, ce qui augmente la difficulté de suivre les normes organisationnelles et les meilleures pratiques avec les API et de développer des ensembles d'outils pour prendre en charge chaque technologie de mise en œuvre.

Si une plateforme de gestion d'API est totalement indépendante des API qu'elle gère et qu'APIOps est suivi dans l'organisation, les développeurs peuvent continuer à utiliser les outils et techniques de programmation qu'ils souhaitent, mais l'organisation continue de bénéficier d'API de haute qualité détectables.

Impact:

- Augmentation du verrouillage des fournisseurs en raison de l'impossibilité de replateformer les implémentations d'API en raison des approches propriétaires utilisées, ce qui augmente la complexité
- Augmentation du coût et de la complexité du développement des outils pour aider à fournir les meilleures pratiques API et le respect des normes dans toute l'organisation
- Des connaissances spécialisées sont nécessaires pour maintenir Des API avec des compétences différentes dans chaque outil différent, ce qui signifie que si une réorganisation a lieu, il y a un risque accru avec le fonctionnement de chaque API

Remède:

- Concevoir une manière agnostique de gérer les ressources API (comme les spécifications, la documentation du catalogue et les tests contractuels) qui sont découplés de la technologie de mise en œuvre
- Mettre en œuvre APIOps comme moyen de facto de concevoir, de construire, d'expédier et d'exécuter des API
- Démocratiser les meilleures pratiques et normes API pour permettre une flexibilité dans l'approche afin que chaque La communauté de développement d'API peut intégrer ses exigences dans la liste centrale des normes

7. Les développeurs ne comprennent pas toute l'étendue des fonctionnalités qui leur sont offertes

Il s'agit de l'un des défis les plus courants observés dans les initiatives API lorsqu'une plateforme API a été créée.
Une équipe de plateforme construit une plateforme moderne qui prend en charge de nombreuses fonctionnalités. (Kong dispose de plus de 100 plugins prêts à l'emploi et de milliers de plugins communautaires sont disponibles).

La plateforme étant conçue pour permettre aux développeurs d'API de se concentrer sur leur développement sans se soucier des contraintes informatiques liées à une passerelle, ils ne découvrent pas l'intégralité des fonctionnalités à leur disposition. Ils se contentent d'utiliser les plugins globaux configurés par le processus d'intégration automatisé, qui les adapte aux normes et bonnes pratiques centrales. Par conséquent, ces fonctionnalités sont

Les développeurs passent à côté de fonctionnalités qui pourraient les aider dans de nombreux aspects du développement d'applications modernes, comme l'amélioration de la résilience des services, des outils d'aide à la simulation et bien d'autres intégrations pré-intégrées.
Conséquence négative : les API ne sont pas conçues de manière optimale en raison de fonctionnalités manquantes, ce qui peut impacter leur résilience et leur fiabilité.

Une bonne pratique consiste pour l'équipe de la plateforme à organiser des sessions d'activation avec ces équipes d'application pour découvrir les points faibles techniques des équipes et à faire des suggestions sur la manière dont elles peuvent résoudre ces points faibles avec les fonctionnalités fournies par la plateforme.
Cela peut être exécuté sous forme de session planifiée dans le cadre de l'intégration, des heures de bureau ou même de hackathons axés sur la résolution de problèmes particuliers tels que la résilience.

Impact:

- Les API présentent des lacunes dans leur résilience qui peuvent être automatiquement comblées
- Les API présentent des lacunes dans leur sécurité ou leur contrôle d'accès posture •

Les API présentent des lacunes dans leurs performances en raison de l'absence de mise en cache au niveau d'une couche de passerelle (le cas échéant)

- Les développeurs continuent d'écrire du code à intégrer dans leurs implémentations, qui pourrait être remplacé par une configuration de politique, ce qui entraînerait un coût de possession plus élevé et un délai de mise sur le marché plus lent.

Remède:

- Mettre en place une pratique d'habilitation axée sur comprendre les difficultés techniques des développeurs d'applications et suggérer des solutions que la plateforme peut fournir prêtes à l'emploi
- Organiser des communautés ou des guildes qui hébergent régulièrement des forums ou des heures de bureau pour répondre aux questions des autres développeurs
- Organiser des hackathons à fort impact axés sur des points sensibles spécifiques, tels que la stabilité et la résilience des services

8. Ne pas avoir une observabilité correcte sur l'ensemble du parc API

Les plates-formes API doivent gérer toutes les API d'une organisation, à la fois les API externes et internes, en garantissant qu'il n'y a pas d'intégrations point à point dans une architecture distribuée.

Non seulement cela présente des avantages en termes de sécurité et de gouvernance, mais cela facilite également l'intégration avec les outils d'observabilité et la collecte d'analyses produit API à des fins de visualisation et d'utilisation. En gérant activement et en observant toutes les API, le temps moyen de résolution (MTTR) en cas de panne dans l'architecture distribuée peut être considérablement réduit grâce à l'identification rapide et efficace des composants défectueux.

Une autre erreur commise avec les programmes API est de ne pas suivre les KPI d'une API. Par exemple, qu'est-ce qui fournit des informations commerciales plus riches : définir le KPI « cette API doit être réutilisée » ou définir les KPI « applications consommatrices uniques », « utilisation moyenne par consommateur » et « temps de traitement total pour le consommateur » ? Ces derniers sont de meilleurs KPI pour comprendre véritablement la valeur d'une API individuelle pour l'organisation, mais ne peuvent être efficacement surveillés et analysés que lorsque l'ensemble du trafic est capturé et envoyé vers une interface unique pour observation.

Impact:

- Dans une architecture distribuée, les défauts peuvent être difficiles à isoler et à corriger
- Risque accru de coût élevé de panne en raison de l'impossibilité d'isoler les problèmes
- Manque d'informations exploitables sur les API en raison de la définition de mauvais indicateurs de performance clés

Remède:

- Concevoir une plateforme API pour répondre aux cas d'utilisation internes et externes afin que tous les services

soient gérés • Mettre en œuvre une stratégie d'observabilité couvrant la surveillance, la journalisation, l'analyse et les traces

- Se concentrer sur la conception des KPI cela permettra des informations exploitables sur la valeur d'un produit spécifique API plutôt que des objectifs génériques

9. Traiter l'API-first comme un défi purement technique

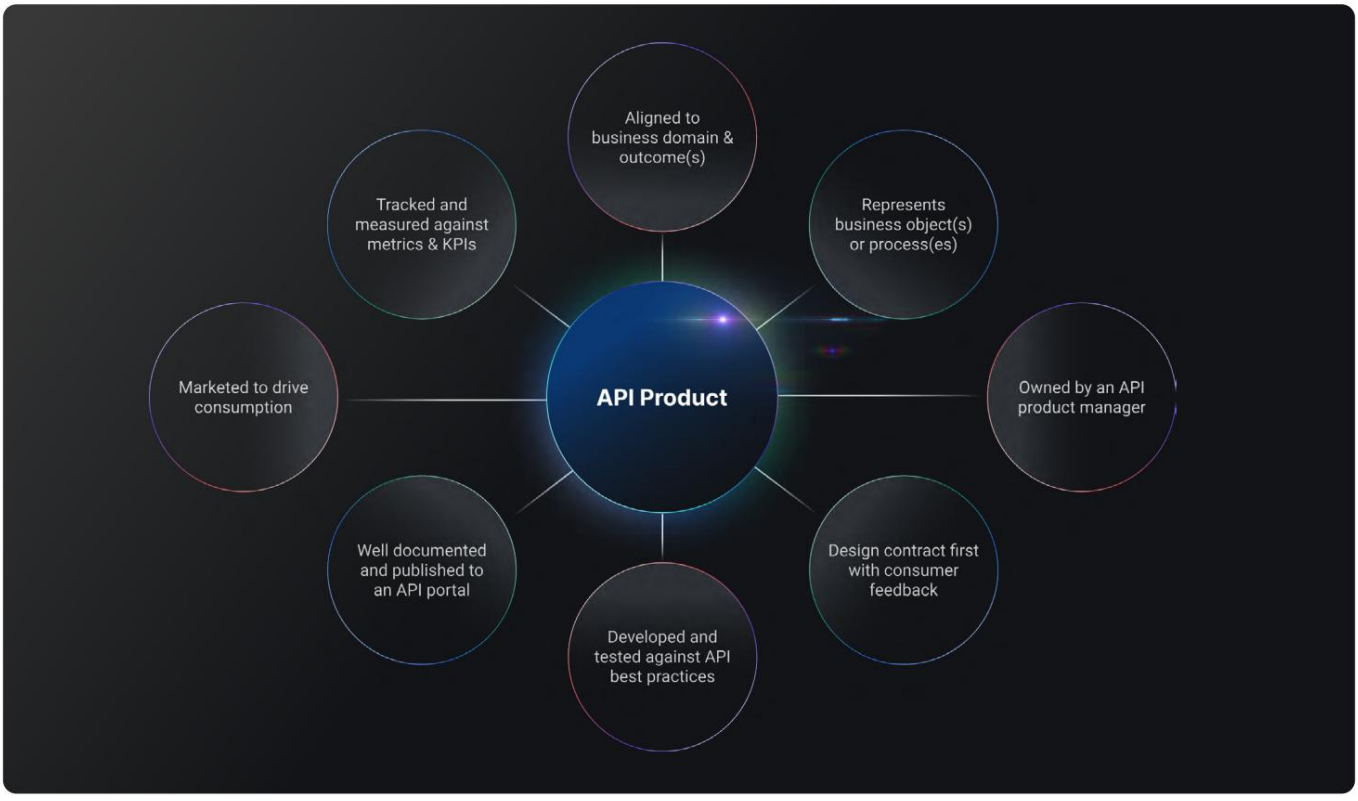
Une organisation API-first positionne les API au cœur de son activité, en décomposant ses domaines d'activité en un ensemble de blocs de construction modulaires qui améliorent l'agilité, l'expérience client, l'interopérabilité et l'innovation, favorisant ainsi l'optimisation et la croissance des revenus.

Cependant, de nombreuses organisations peinent à exploiter pleinement le potentiel des API, car elles ne les considèrent pas comme des produits essentiels à leur activité, exigeant la même attention stratégique que d'autres offres commerciales aux résultats mesurables. De plus, les objectifs financiers à court terme et les ressources ponctuelles sont insuffisants pour réaliser un changement significatif. Si une plateforme de pointe et les pratiques d'accompagnement présentées dans ce document constituent un enjeu majeur, une approche axée sur les API ne peut être véritablement efficace que si elle s'appuie sur une transformation organisationnelle adaptée, notamment axée sur la culture, les compétences et les processus opérationnels.

• Les gens. Devenir une entreprise axée sur les API dépend du développement des compétences et des talents adéquats pour concevoir, construire et exploiter des API, y compris des rôles tels qu'architectes d'API, responsables de portefeuille, et les chefs de produit. Les équipes d'habilitation sont essentielles pour faire connaître et comprendre les API au sein de l'organisation, en collaborant avec les équipes métier pour identifier et développer les fonctionnalités adéquates. La propriété métier des API doit être établie de la même manière que celle des autres produits numériques. Plus important encore, la propriété métier doit être établie pour les API de la même manière que pour les autres produits numériques.

• Processus. Aux premiers stades de maturité des API, Les organisations considèrent les API comme des outils techniques plutôt que comme des produits (actifs métier). Ces API naissent souvent de demandes de projets réactives, dont la portée et la valeur sont limitées.

Au fil du temps, cela conduit à une prolifération des API, où les API sont développées sans stratégie cohérente.



Impact:

- Manque de sensibilisation et de compréhension parmi les dirigeants et les parties prenantes de l'entreprise sur la façon dont les API peuvent générer de la valeur commerciale relèguent souvent les API au rang d'outils techniques avec un investissement limité et un potentiel de transformation.
- Les API développées sans stratégie définie sont souvent non standardisées et incohérentes, ce qui les rend difficiles à utiliser, ce qui conduit à la prolifération des API, où les API prolifèrent de manière incontrôlable au sein d'une organisation, augmentant les coûts et augmentant la complexité.
- Les API créées pour répondre à un projet spécifique Les exigences offrent généralement une valeur limitée au-delà de leur cas d'utilisation initial ; sans propriété et gouvernance appropriées, ces API peuvent devenir des « API zombies » — des actifs inutilisés et non gérés qui présentent un risque de sécurité
- La disponibilité des bonnes API au sein d'une entreprise peut réduire les opportunités d'innovation et de revenus, ce qui peut avoir un impact sur les initiatives numériques et d'IA et souvent ralentir l'adaptation des organisations aux changements marketing ou à la conformité du secteur.

Remède:

- Commencez par définir une stratégie API d'entreprise qui fournit une analyse de rentabilisation et un cadre clairs pour la création de produits API ; la stratégie doit se référer à la stratégie commerciale et aux objectifs de l'organisation
- Sensibilisez et soulignez l'importance de la stratégie auprès de votre direction exécutive — la stratégie API doit être parrainée et mandatée par les principales parties prenantes pour être couronnée de succès

- API-first est un engagement de toute l'organisation, pas seulement technique, donc assurez-vous que toutes les parties de l'entreprise sont sensibilisées à l'importance stratégique des API pour stimuler l'innovation, l'expérience client et la croissance des revenus.
- Établir un nouveau modèle opérationnel basé sur une approche axée sur le produit plutôt que sur une approche axée sur les projets, avec les parties prenantes informatiques et commerciales collaborant pour identifier les produits API qui sous-tendent leurs capacités commerciales et permettent d'obtenir des résultats commerciaux.

- Évitez les engagements à court terme et investissez dans les bonnes personnes pour diriger votre nouveau modèle opérationnel en établissant une pratique d'activation des API pour superviser et guider la stratégie API ; en nommant Chefs de produits API et architectes API au sein d'équipes commerciales fédérées pour piloter la livraison de produits API au sein des domaines d'activité
- Définir des métriques et des indicateurs clés de performance pour mesurer la performance de la stratégie et des API au sein de l'entreprise : devenir plus qualitatif et quantitatif permet de plaider en faveur des API auprès de la direction et des propriétaires d'entreprise

Conclusion

> Le chemin vers des initiatives API réussies peut être semé d'embûches, mais avec une approche stratégique, le respect des meilleures pratiques et la bonne plateforme, les organisations peuvent augmenter la productivité des développeurs et réduire les délais de mise sur le marché.

Kong vous accompagne à chaque étape de votre projet, en vous proposant les solutions et l'expertise nécessaires à la réussite de vos initiatives API. Visitez www.konghq.com. pour en savoir plus.

Powering the API world

Konghq.com

Kong Inc.
contact@konghq.com

77, rue Geary, bureau 630
San Francisco, Californie 94108
USA