Avec les compliments d'IBM

Édition limitée IBM

Les API

POUR

LES NULS®

Apprenez à :

- Mettre la puissance des API au service de vos objectifs métiers
- Définir la nature Produit des API modernes
- Identifier les API à fournir ou à consommer
- Créer une plate-forme d'API efficace

Claus T. Jensen

Les API

POUR

LES NULS®

Édition limitée IBM

Claus T. Jensen

WILEY

Les API pour les Nuls®, Édition limitée IBM

Publié par
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN: 978-1-119-19332-6 (pbk); 978-1-119-19334-0 (ebk)

Produit aux États-Unis

10 9 8 7 6 5 4 3 2 1

Remerciements

Parmi ceux qui ont participé à la création et à la commercialisation de cet ouvrage, figurent les personnes suivantes :

Éditrice de projet : Carrie A. Burchfield

Éditrice en charge du développement : Kathy Simpson

Responsable éditorial : Rev. Mengle

Représentante du développement commercial : Sue Blessing

Coordinatrice de production : Melissa Cossell

Table des matières

Introduction

Les API sont un sujet en vogue, qui fait l'objet de vifs débats entre commerciaux, responsables informatiques et développeurs. Cette agitation au sein de l'espace public porte essentiellement sur les API publiques ouvertes. Pour oser une comparaison, ne pas avoir une API publique aujourd'hui, c'est comme ne pas avoir de site Web à la fin des années 1990. Cependant, bon nombre d'entreprises considèrent les API publiques comme le cadet de leurs soucis. Créer des solutions omnicanal, innover plus vite que la concurrence, devenir une entreprise mobile ou mettre en place un environnement de cloud hybride constituent des priorités.

Pourtant, les API jouent un rôle central dans tous ces projets et bien d'autres, d'où cet intérêt de la part d'acteurs divers et variés. Mais qu'est-ce qu'une API ? En quoi se différencie-t-elle d'une interface de programmation d'applications ancienne génération ? Et pourquoi faut-il s'en soucier ? L'acronyme API (pour *Application Programming Interface*) signifie Interface de programmation d'applications. Cette notion a considérablement évolué avec le temps. Les API d'aujourd'hui sont très différentes des anciennes. Lisez ce livre pour découvrir ce que cette évolution recouvre.

À propos de ce livre

Les écosystèmes d'entreprise modernes doivent repenser leur approche de l'innovation et de l'intégration. Ce livre vous explique comment utiliser les API pour aider les entreprises à relever des défis tels que la modification des modèles économiques ou l'intégration d'équipements et de capteurs différents. La puissance des API va bien au-delà de la simple monétisation des données. Que vous soyez fournisseur ou consommateur d'API, vous devez prendre des décisions métiers et informatiques intelligentes. Ce livre définit la nature des API modernes et vous guide dans vos activités de prise de décision, depuis l'identification des API à fournir ou à consommer jusqu'à la création d'une plate-forme d'API efficace.

Quelques devises importantes reviennent à plusieurs reprises dans cet ouvrage. Il s'agit des devises suivantes :

- **Considérez une API comme un produit.** Elle représente quelque chose que vous avez choisi de partager avec un public particulier, dans des conditions bien définies.
- **Tester tôt, apprendre vite, adapter facilement.** Une approche expérimentale des API est efficace dans la plupart des cas, et de nombreuses API seront de nature opportuniste.
- **Utilisez toujours des API pour délimiter votre domaine.** Ainsi, vous disposez d'un contrôle et d'une visibilité sur les trafics entrant et sortant.
- **Les plates-formes d'API sont adaptées à la tâche.** Vous voulez simplifier et sécuriser au maximum la création, l'utilisation et le partage des API.
- **La monétisation des API publiques n'est pas le seul point d'entrée.** Beaucoup d'entreprises utilisent des API pour favoriser la collaboration et l'innovation au sein de leurs écosystèmes métiers et informatiques.

Icônes utilisées dans ce livre

Comme tous les autres livres de la série *Pour les Nuls*, celui-ci affiche quelques icônes dans les marges. Voici leur signification.

Cette icône signale des informations utiles.

Gardez à l'esprit tout ce qui apparaît en regard de cette icône.

Si les détails techniques vous rebutent, ignorez les endroits repérés par cette icône.

Accordez une attention particulière aux paragraphes signalés par cette icône. Ils traitent de sujets qui pourraient avoir un impact négatif sur votre entreprise.

Pour approfondir le sujet

Ce livre n'a pas pour vocation d'être exhaustif sur le sujet. Si vous souhaitez en savoir plus, consultez les liens suivants :

- `developer.ibm.com/api/blog` : blog d'IBM sur les API
- `www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg248188.html?Open` : ressource complète sur le middleware d'intégration
- `ibm.com/soa` : téléchargez le livre électronique *Les principes de la conception SOA pour les Nuls*.

Chapitre 1

Anatomie d'une API

Dans ce chapitre

- Définir ce qu'une API n'est pas
- Savoir comment développeurs et consommateurs veulent utiliser les API
- Catégoriser les API
- Intégrer les API dans le modèle économique

Les API modernes sont des outils souples qui mettent en avant vos activités, auprès d'un public extérieur à votre équipe. Bien conçues, les API permettent aux entreprises d'innover plus rapidement et de toucher de nouveaux publics. D'où leur intérêt. Mais en quoi les API consistent-elles et quelles questions faut-il se poser avant de se lancer dans la création d'API ? Ce chapitre tente de répondre à ces questions.

Définir ce qu'une API n'est pas

Parfois, il vaut mieux définir quelque chose en commençant par expliquer ce que cette chose n'est pas. Donc, voici ce qu'une API n'est pas :

- **Un logiciel :** un logiciel n'est pas une API (même s'il peut se présenter sous la forme d'une API pour faciliter l'utilisation de ses fonctionnalités).
- **Une interface utilisateur :** une interface utilisateur n'est pas une API (mais elle peut s'exécuter sur une interface utilisateur).
- **Un serveur :** un serveur n'est pas une API (mais il peut héberger une ou plusieurs API qui fournissent les données et fonctions mises à disposition par le serveur).

Les API : des produits

Considérez une API comme un produit. Vous la concevez pour qu'elle séduise le consommateur cible, et donc qu'elle se vende. Peu importe que ce consommateur achète ou pas l'API, qu'il soit membre de l'une de vos équipes ou qu'il ne fasse pas partie de votre entreprise. Vous voulez qu'il utilise votre API, parce qu'elle crée de la valeur pour vous deux.

La nature « produit » des API est une caractéristique essentielle de leur puissance. Elle les différencie d'une interface de programmation d'application ancienne génération, qui n'est rien de plus qu'un logiciel que vous avez créé et déployé. Une API moderne offre de multiples fonctionnalités recherchées par un public cible et qui fonctionnent indépendamment des logiciels en cours d'exécution dans votre système principal. Bien que dotées d'une interface de programmation définie, les API modernes sont délibérément conçues pour le consommateur visé.

Comme une API est un produit, avant d'en développer une, posez-vous ces questions clés :

- Quels consommateurs visez-vous ?
- Comment allez-vous toucher ces consommateurs ?
- Dans quelles conditions ces consommateurs peuvent-ils utiliser cette API ?

Des API à l'économie des API

L'économie des API apparaît lorsque les API deviennent parties intégrantes du modèle économique. Les API publiques et partenaires sont des outils stratégiques pour plusieurs modèles économiques, comme ceux de Twitter et d'Amazon.

Par exemple, les API de Twitter enregistrent dix fois plus de trafic que le site Web de Twitter. Clairement fondé sur l'interaction par les tweets, le modèle économique de cette société permet à tous ceux qui le souhaitent de proposer une expérience utilisateur.

Dès le début, Amazon a décidé de ne pas se positionner comme un simple revendeur sur Internet, mais comme un portail marchand à l'échelle mondiale. La plate-forme marchande d'Amazon s'appuie résolument sur des API qui facilitent l'intégration de nouveaux marchands.

En tant qu'outils de réseau métiers, les API ne constituent pas une nouveauté. Depuis plusieurs décennies, les banques créent des infrastructures de paiement et des chambres de compensation basées sur des API spécifiques. À ceci près que les API modernes sont explicitement conçues pour un écosystème ouvert (interne ou externe) et non pour des réseaux privés fermés. De plus, les modèles de consommation des API sont standardisés, privilégiant la simplicité de consommation à la facilité de création.

Certaines personnes emploient le terme API métiers pour désigner toutes les API modernes. Un terme bien choisi, dans la mesure où les API, en tant que produits, doivent faire partie intégrante de votre stratégie métier. Sachez simplement que lancer une API publique ou partenaire n'est pas la seule manière d'intégrer des API dans votre modèle économique. Les modes d'utilisation des API consommées en interne sont nombreux, le plus connu étant la nécessité de proposer une expérience client omnicanal différente.

Que votre entreprise soit « née sur le Web » ou qu'elle existe depuis un siècle, vous vivez à l'âge du cloud, de l'analyse des données, de l'informatique mobile et des réseaux sociaux, où l'omnicanal est devenu l'enjeu principal. Pour vous démarquer de vos concurrents, vous devez proposer aux clients une expérience immédiatement agréable. Et pour ce faire, vous devez être libre d'expérimenter et d'innover. Saisissez l'opportunité, testez votre API tôt, apprenez vite et adaptez-la facilement.

Comprendre ce que les développeurs veulent

Les développeurs veulent utiliser des API afin d'innover et d'expérimenter. Pour eux, la réutilisation raccourcit les délais de livraison, le partage est synonyme de pragmatisme, et l'encapsulation réduit le travail d'apprentissage. Ils se fichent de savoir comment les API ont été créées ou si elles sont faciles à consommer.

La facilité de consommation n'est pas seulement liée à l'apparence de l'API. Pour le développeur averti, la facilité de consommation signifie également qu'une API doit être facile à trouver et à enregistrer. De plus, le niveau de fiabilité de l'API dans des solutions stratégiques doit être connu.

Idéalement, un écosystème d'API doit être axé sur une communauté. Une communauté efficace montre aux développeurs les API qu'elle utilise pour effectuer ses tâches courantes. L'enregistrement en

libre-service et les approbations préalables sont déjà en place pour les API accessibles à la communauté. Les fonctionnalités sociales de la communauté permettent aux membres de donner une appréciation sur une API particulière, tandis que l'analyse des données sur les consommateurs révèle le comportement opérationnel prévu de l'API en question. Historiquement, ces fonctionnalités ne font pas partie de la gouvernance informatique, mais des solutions de gestion des bonnes API. Ces solutions de gestion des bonnes API valorisent le travail des fournisseurs d'API, en facilitant la création des API et en améliorant le contrôle de leur comportement d'exécution (expliqué en détail dans le chapitre 2).

Quatre catégories d'API

Lorsque vous décidez de créer des API, choisir celles à créer en premier peut être compliqué. Une bonne API doit se démarquer de ce qui existe déjà.

Cet encadré décrit une approche qu'IBM a trouvée très utile en pratique. Se demander « Quelles situations métiers pourrais-je améliorer, et comment m'y prendre ? » est un bon point de départ. Trouvez la réponse à ces deux questions et vous saurez quelles API créer en premier parmi les quatre catégories proposées ci-après :

- API de détection : ces API vous aident à identifier les opportunités qui impliquent des clients, des employés, des partenaires et des équipements. Elles intègrent des fonctionnalités, telles que la géolocalisation mobile, la surveillance par capteurs, l'analyse prédictive et l'observation humaine.
- API d'enrichissement : ces API améliorent la compréhension de la situation grâce à des données historiques émanant de systèmes de gestion de la relation client (CRM), de fichiers de comptabilité, d'analyses démographiques, de dossiers médicaux, etc.
- API de perception : ces API fournissent un contexte dynamique de la situation actuelle et vous permettent de savoir ce que pensent les personnes que vous ciblez. Il peut s'agir d'API sociales (des personnes partageant des projets futurs ou des intérêts actuels) ou de solutions d'analyse de capteurs (pour l'état global du système, comme la consommation des ressources ou la congestion du trafic).
- API d'action : ces API vous permettent d'agir en quasi-temps réel. Elles peuvent inclure des notifications push, des équipements instrumentés ou des systèmes de gestion de tâches humaines.

Chapitre 2

Gestion des API (ou pas)

Dans ce chapitre

- Examiner les rôles et tâches de gestion des API
- Définir la gouvernance des API
- Savoir quand utiliser les API non gérées

Les discussions sur les API tournent invariablement autour de la notion de gestion des API. Même si ce ne sont pas des éléments logiciels au sens traditionnel du terme (voir le chapitre 1), elles occupent une place importante dans les environnements d'exploitation (métier et informatique). À ce titre, leur gestion doit être appropriée. Dans ce chapitre, vous allez découvrir comment.

Savoir en quoi consiste la gestion d'une API

Pour un consommateur d'API, un bon portail de développeurs est essentiel. Pour un fournisseur d'API, la gestion des processus d'externalisation et de partage des API ne constitue que la partie émergée de l'iceberg (voir la figure 2-1). Les aspects métiers et informatiques qui facilitent la création, le déploiement et l'exploitation des API restent cachés. Ces aspects incluent le mapping des données, la sécurité, la régulation du débit, le suivi et la gestion des versions.

Non seulement une API gérée dispose d'une interface bien définie et s'adresse à un public cible, mais son fonctionnement est régi par des contrôles métiers et informatiques correctement mis en œuvre. Chaque groupe joue un rôle particulier dans la gestion des API, comme vous allez le découvrir dans cette partie.

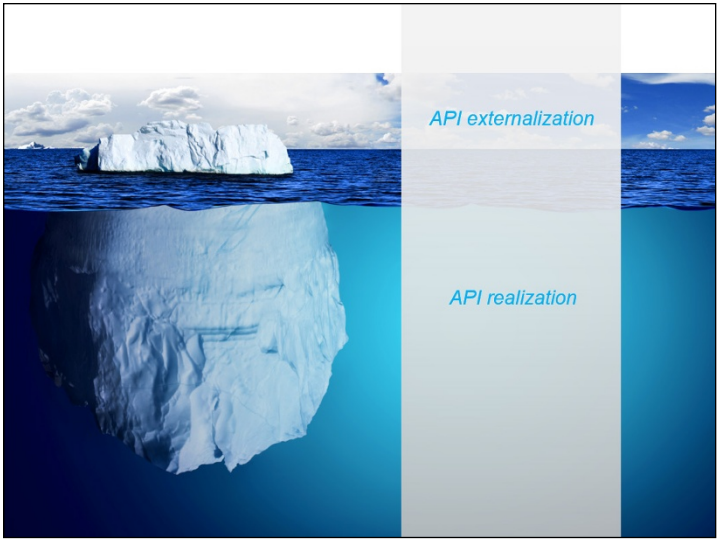

Figure 2-1 : La gestion des API ne se limite pas à leur conception et à leur externalisation.

Pour une efficacité optimale, les trois rôles présentés dans ce chapitre – propriétaire d'API, responsable des opérations informatiques et concepteur d'API – doivent avoir leur propre expérience utilisateur. Leurs besoins sont tellement différents que les outils de l'un sont inefficaces pour les autres.

Propriétaire d'API

Le propriétaire d'API prend les décisions suivantes :

- Les conditions dans lesquelles l'API peut être consommée
- Les communautés avec lesquelles l'API sera partagée
- Si l'API remplit ses objectifs (en cas d'échec, le modèle économique doit être ajusté)

Tout ceci peut être fait sans modifier ni la définition ni la mise en œuvre de l'API. Pour en savoir plus sur le rôle du propriétaire d'API, consultez la section « Nécessité d'une gouvernance des API » plus avant dans ce chapitre.

Responsable des opérations informatiques

Les opérations informatiques doivent garantir certaines caractéristiques opérationnelles, ce qui ne nécessite aucune modification de la définition ou de la mise en œuvre de l'API. Ces caractéristiques sont les suivantes :

- Le runtime (moteur d'exécution) qui héberge l'API fonctionne de manière fiable et sécurisée.
- L'API est correctement authentifiée et l'autorisation est en place pour qui l'utilise.
- Le trafic de l'API est optimisé et hiérarchisé en fonction des besoins métier.

Il existe une différence fondamentale entre ce que le propriétaire d'API vend en terme de conditions d'utilisation de l'API et ce que l'infrastructure informatique sous-jacente peut apporter. Mettre en place des capacités permettant de gérer le maximum de trafic correspondant aux conditions d'utilisation de l'API peut se révéler très onéreux.

Afin d'éviter des coûts d'exécution prohibitifs, vérifiez que votre runtime d'API est hautement évolutif (pour réduire l'importance des charges de travail) ou régulez le trafic lorsque la charge courante dépasse les capacités disponibles (pour lisser les pics d'activité). Ces fonctionnalités doivent être disponibles dans le runtime d'API que vous avez choisi.

Pour en savoir plus sur les opérations informatiques, consultez la section « Nécessité d'une gouvernance des API » plus avant dans ce chapitre.

Concepteur d'API

Cette personne est chargée de créer et de déployer l'API. Elle doit effectuer les opérations suivantes :

- Définir l'interface de l'API
- Identifier les points de terminaison principaux capables de fournir les données ou les fonctions nécessaires pour mettre en œuvre l'API
- Configurer le mapping entre l'interface de l'API et les sources principales de données ou de fonctionnalités

Le concepteur d'API doit être capable d'effectuer ces tâches sans générer un volume de code important. Lorsque la création d'une API repose davantage sur la génération de code que sur la configuration dynamique, le rythme d'innovation chute inévitablement, même chez les équipes de développement les plus agiles. La distinction entre la configuration d'une API et le développement de données ou de fonctionnalités est essentielle dans la réflexion sur les API. Comme nous l'avons déjà vu dans le chapitre 1, les API modernes ne sont pas des logiciels, mais un moyen souple de présenter des fonctionnalités à des publics extérieurs à votre équipe.

Nécessité d'une gouvernance des API

Dans le milieu des API, l'une des idées reçues est que la gouvernance paralyse tout. Or une gouvernance permet de prendre les bonnes décisions, c'est-à-dire de s'assurer que les bonnes personnes prennent les bonnes décisions au bon moment, pour les bonnes raisons et sur la base d'informations fiables. Donc, si une API est importante pour votre entreprise, il vaut mieux prendre des décisions pertinentes. En tant que fournisseur ou consommateur d'API, plusieurs décisions vous incombent.

Cette gouvernance est différente de celle qui est fréquemment appliquée au cycle de mise sur le marché d'un logiciel, mais elle reste importante.

Décisions du fournisseur d'API

La tâche du propriétaire d'API consiste à décider qui peut utiliser l'API et dans quelles conditions commerciales. Cette décision opérationnelle métier s'applique à toutes les API : celles utilisées par votre équipe de développement pour créer des applications mobiles, celles que vous utilisez pour intégrer des systèmes dans différentes activités et celles utilisées par des consommateurs externes. Il est donc essentiel de prendre différentes décisions pour déterminer quel public peut utiliser quelle API et dans quelles conditions.

Le service en charge des opérations informatiques doit également prendre des décisions appropriées – en général, sous la forme de stratégies de sécurité et de trafic – pour protéger l'infrastructure contre toute utilisation abusive ou surcharge.

Le régime de gouvernance doit être très léger et les décisions doivent être de nature opérationnelle contrairement aux décisions classiques

prises pendant le cycle de mise d'un logiciel sur le marché. S'il est impossible de prendre ou d'appliquer les bonnes décisions facilement, la nature ouverte et dynamique des API est compromise (n'oubliez pas, les bonnes API reposent sur la configuration et non sur le code). Les décisions métiers et informatiques sont des rouages essentiels d'une bonne gestion des API et doivent être prises en charge par la plate-forme d'API choisie. Pour en savoir plus sur le middleware d'API, consultez sur le chapitre 5.

Décisions du consommateur d'API

Les consommateurs d'API doivent également prendre de bonnes décisions. Ils doivent notamment choisir les API qu'ils vont utiliser et dans quels buts. Ensuite, ils vont se poser les questions suivantes sur chaque API :

- Quel est le modèle de paiement de l'API et est-il acceptable pour votre objectif ?
- Aurez-vous besoin d'un proxy d'entreprise en amont de l'API pour gérer les licences, le paiement, etc., ou chaque développeur devra-t-il s'enregistrer indépendamment ?
- L'API est-elle sécurisée et fiable pour des missions stratégiques ? Les données historiques sur le comportement de l'API peuvent renforcer la confiance des consommateurs qui l'utilisent.

Lorsque les API consommées sont les vôtres, ces décisions sont relativement simples puisqu'elles concernent principalement la conception métier. Lorsque les API sont celles de tiers, les décisions deviennent plus compliquées. L'expérience utilisateur et la responsabilité du maintien de l'intégrité de l'entreprise ne peuvent pas être déléguées. Vous avez besoin qu'un membre de votre personnel soit responsable de l'expérience utilisateur et prenne les bonnes décisions sur les API qu'il convient de consommer dans le cadre de votre modèle de livraison.

Défendre les API non gérées

Chacun sait que les API modernes requièrent une solution de gestion, n'est-ce pas ? Pas si sûr. Il n'est pas nécessaire de gérer toutes les API. Dans ce chapitre, j'ai défini ce qu'est une bonne gestion des API, mais que se passe-t-il lorsqu'une API n'est pas gérée ?

Voici les principales différences entre les API gérées et non gérées :

- Une API non gérée peut avoir un public cible, mais celui-ci n'est que rarement défini avec précision. Si un utilisateur a accès à l'API via le réseau, il peut en général l'utiliser.
- Une API non gérée ne met pas en œuvre de contrôles métiers et informatiques indépendamment. Un contrôle est fourni par la logique lors de la mise en œuvre de l'API, généralement sous forme de code.

En d'autres termes, les API non gérées ont toujours une interface bien définie, mais il est impossible d'appliquer des contrôles sur leur comportement d'exécution ou sur leurs utilisateurs. Alors, pourquoi vouloir une API non gérée ?

Si une API fait partie intégrante de votre modèle économique, vous opterez probablement pour la gestion. Ceci dit, voici quelques exemples de situations dans lesquelles les API non gérées sont appropriées voire incontournables :

- Un équipement ou un capteur, comme un thermostat domestique programmable à distance ou un Fitbit (dispositif portable qui surveille l'activité physique) est doté d'une API qui synchronise les données avec un ordinateur via une interface définie.
- Un logiciel existant – un système standard de type SAP ou un gros système équipé d'une interface REST native – affiche une micro-API.
- Comme l'API réside dans votre domaine, tout ce dont vous avez besoin, c'est d'une connexion pour y accéder.

Les API non gérées peuvent être des ressources importantes dans beaucoup d'écosystèmes, en fournissant des données et des fonctionnalités clés de manière uniforme. C'est grâce à ce modèle de consommation uniforme que vous voudrez encore considérer ces interfaces comme des API. Souvent, il est même souhaitable de cataloguer toutes les API non gérées auxquelles vous avez accès, pour pouvoir les retrouver le plus facilement possible et les utiliser dans un modèle de programmation particulier.

La méthode la plus simple et la plus efficace pour innover et collaborer dans un environnement hybride consiste à tout transformer en une API destinée au consommateur. Cela signifie que la réflexion sur les API ne se limite pas à la gestion des API et aux API gérées. Elle doit s'articuler dans une stratégie d'intégration plus globale pour transformer votre entreprise en moteur d'innovation.

Chapitre 3

La véritable nature des bonnes API

Dans ce chapitre

- Utiliser des API opportunistes
- Combiner des API à une architecture SOA
- Comprendre la conception d'une bonne API

La rapidité d'innovation est directement liée à la qualité de conception, ce qui comprend l'interface et les caractéristiques techniques d'une API. Surtout, une conception est de qualité lorsque l'on sait quelles API fournir et quand. Il existe de multiples types d'API et d'utilisations. Donc, le but de ce chapitre est de vous apprendre à reconnaître une bonne API.

Comparer les API aux courses automobiles

On pourrait faire une analogie entre les API et la façon dont les équipes de Formule 1 construisent et font évoluer les voitures de course (voir la figure 3-1). En Formule 1, chaque voiture est un prototype. Aucune équipe n'utilise la même voiture pour deux courses consécutives. Une voiture de course est constituée de composants rapidement remplaçables aux interfaces bien définies, et intègre des commandes et des outils d'analyse. Si certaines pièces restent stables tout au long de la saison, d'autres composants sont optimisés en permanence en fonction des enseignements tirés de la course précédente.

À beaucoup d'égards, les entreprises modernes ressemblent aux équipes de Formule 1 en ce qu'elles cherchent en permanence à optimiser le modèle économique et à trouver le compromis idéal entre évolution et stabilité. Les API permettent de maîtriser l'expérimentation pour le bénéfice de l'entreprise. « Tester tôt, apprendre vite et adapter facilement » est un bon principe à appliquer au monde des API.

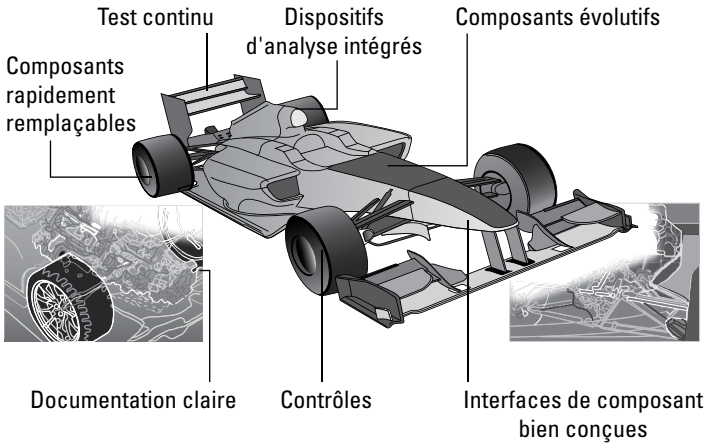

Figure 3-1 : Les voitures de course et les API devraient suivre les mêmes principes de conception.

Défendre les API opportunistes

Les API doivent-elles toujours être conçues dans une optique de réutilisation ? Le fait de réutiliser une API implique une stabilité sur une période relativement longue. La stabilité est importante si une API doit être réutilisée pour intégrer des partenaires ou montrée à l'extérieur comme composante de votre modèle économique. Mais si l'API ne vise qu'à améliorer la collaboration entre, par exemple, une équipe de développement mobile et l'équipe chargée de la maintenance d'un système principal, la réutilisation peut n'être ni souhaitable, ni appropriée.

Une API doit être agréable à utiliser. Pour un développeur, utiliser cette API doit être plus rapide et plus commode que de coder une autre solution. Si les besoins d'un développeur mobile changent rapidement, les API doivent évoluer au même rythme pour garder leur intérêt. Cette situation plaide en faveur des *API opportunistes*, des API créées rapidement et adaptables tout aussi rapidement aux attentes des consommateurs.

La réutilisabilité et la stabilité ne sont absolument pas des impératifs pour une API. L'importance de ces deux aspects dépend exclusivement de votre objectif métier.

Pour fournir des API opportunistes, il faut que la création et la maintenance des API soient à la fois simples et peu onéreuses. Sinon, le coût des modifications opportunistes s'envole.

Les bonnes solutions de gestion d'API créent des API par configuration et non par codage. En général, créer ou modifier une API ne prend que quelques minutes. Le coût de création et de maintenance mis à part, la gestion correcte des API opportunistes est importante.

La gestion des API opportunistes offre les avantages suivants :

- Définition et mise en œuvre des contrôles métiers et informatiques
- Analyse globale des performances d'une API d'un point de vue métier
- Flexibilité opérationnelle permettant de déplacer et de dimensionner dynamiquement les charges de travail des API

Ces avantages constituent des facettes importantes de la devise « Tester tôt, apprendre vite et adapter facilement ». Ils sont essentiels pour optimiser le changement dans un monde d'opportunités et d'innovation.

Penser les API et les services

L'architecture orientée services ou SOA (Service-Oriented Architecture) est sur le devant de la scène depuis une décennie environ. Les API modernes sont plus récentes. Ces deux approches de l'intégration ont leurs tenants et répondent aux préoccupations métiers et informatiques. Mais qu'est-ce qui les différencie et faut-il choisir entre les deux ?

API ou services

La notion de service est au cœur du concept de SOA. Par exemple, l'Open Group définit un service comme une « représentation logique d'une activité répétable dont le résultat est spécifique ». Les services sont autonomes et opaques pour leurs consommateurs, et leurs contrats d'interaction sont bien définis. Du point de vue technique, ces caractéristiques s'appliquent également à toute API bien conçue, donc techniquement parlant, une API est aussi un service.

Dans ce cas, les API sont-elles également des services ? En fait, il existe une différence importante entre services et API : l'objectif qui sous-tend leur conception (voir la figure 3-2). Les API sont conçues pour séduire le consommateur cible et évoluer en fonction des besoins de celui-ci. En revanche, les services sont généralement conçus pour répondre à deux préoccupations principales : le coût global et la stabilité.

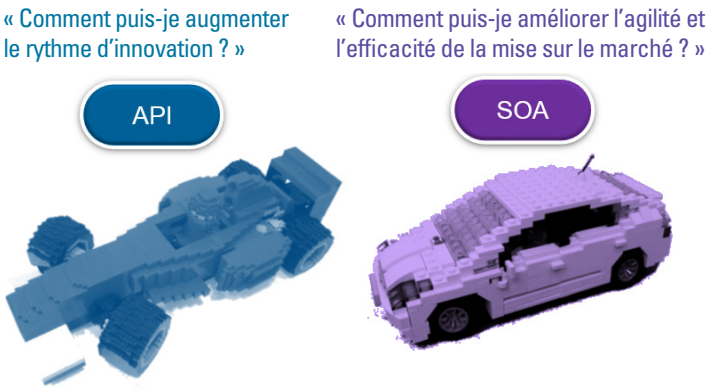

Figure 3-2 : Les API et les services répondent à différents besoins.

Pour un consommateur, utiliser une API est synonyme de gain de temps, de commodité et de faible apprentissage. Ces critères de conception constituent des différences fondamentales entre les API et la notion classique de service :

- Pour un fournisseur de services, la réutilisation permet de minimiser le travail nécessaire pour créer une API. Pour les consommateurs d'API, elle réduit les délais de mise sur le marché de leur logiciel, quel que soit le coût de mise à disposition des API consommées dans le cadre de ce logiciel.
- Pour un fournisseur de services, le partage apporte l'efficacité. Pour un consommateur, il est synonyme de commodité (si elle n'est pas pratique, l'API ne sera pas utilisée).
- Pour un fournisseur de services, l'encapsulation permet de minimiser le *changement*. Pour un consommateur, elle réduit l'*apprentissage* (si l'interface est complexe, l'API ne sera pas utilisée).

D'un côté, un développeur mobile veut simplement faciliter l'apprentissage de son application. De l'autre côté, l'équipe en charge du système principal veut que tout le monde utilise le même modèle standardisé de service et de données. Au lieu d'imposer une solution, est-il possible de concilier les deux points de vue sans générer un coût prohibitif ?

Il existe une analogie historique dans l'évolution des bases de données. Les premières générations de bases de données se focalisaient exclusivement sur les structures internes, les tableaux et les schémas. Mais très vite, il a fallu afficher des sous-ensembles contrôlés de données sous une forme particulière adaptée à un groupe de consommateurs de données. La notion de vue de données est alors apparue

comme une fonctionnalité essentielle dans la plupart des bases de données modernes, avec un proxy léger au-dessus du domaine de données représenté par les structures internes de la base de données.

Les API sont des vues (proxy) contrôlées des données et fonctionnalités d'un domaine, qui sont adaptées aux besoins des consommateurs d'API. Tant que la création et la maintenance d'API proxy restent bon marché, vous pouvez les utiliser pour restituer un domaine sous différentes formes optimisées pour chaque groupe de consommateurs d'API. (Après tout, vous voulez sans doute donner à vos partenaires extérieurs une vue de vos capacités qui soit différente de celle proposée à vos développeurs internes.)

API et services

La SOA est devenue un outil pour protéger les consommateurs de services contre les changements dans le système principal. Mais qui protège les fournisseurs de services contre l'évolution rapide des besoins dans les solutions d'interface omnicanal ? L'application simultanée d'API et de services vous permet de créer un périmètre de stabilité au cœur d'un ouragan de changement. Les fournisseurs exploitent les services pour codifier les fonctionnalités de base de leurs domaines. Les API servent à reconditionner ces fonctionnalités (services), à les transformer en produits et les partager sous une forme simple d'utilisation. Les API et les services sont complémentaires. Ensemble, ils améliorent considérablement l'efficacité globale de l'entreprise en matière d'innovation.

Reconnaître une API bien conçue

Bien sûr, la conception technique des API est importante, mais elle varie énormément selon les technologies choisies pour la conception et la mise en œuvre d'une API. Des livres entiers traitent de la conception d'interfaces avec un niveau de détail bien supérieur à celui de ce modeste ouvrage. Il suffit de dire que la conception de l'interface des API est généralement bien comprise, comme le montrent les exemples suivants :

- Les interfaces REST s'appuient sur des ressources. L'aspect le plus important de la conception d'interfaces tient à la structure d'URI qui permet à un consommateur de naviguer dans le graphique représenté par l'API.
- Les interfaces SOAP s'appuient sur des méthodes. Les aspects les plus importants de la conception d'interfaces sont les méthodes prises en charge et les structures de données de chacune d'entre elles.

- Les interfaces MQTT s'appuient sur des événements. Les aspects les plus importants de la conception d'interfaces sont les événements (émis ou reçus) ainsi que les messages d'événement associés.

Toutes les API ne sont pas de type REST. En général, les interfaces REST sont excellentes pour des utilisateurs humains et ont la préférence des développeurs mobiles. Mais ces interfaces ont tendance à être prolixes et, malgré leur évolutivité, elles n'embarquent pas de structures de données complexes fortement typées. Les interfaces SOAP sont idéales pour une intégration de système à système. Les équipes en charge des opérations informatiques les préfèrent pour leur prolixité moindre et leurs structures de données plus précises. Les interfaces MQTT sont surtout utilisées pour les communications avec l'Internet des objets, où les priorités sont la bande passante et la durée de vie des batteries, et où la garantie de mise à disposition peut faire la différence entre prévenir les accidents et les laisser survenir par inadvertance.

La prétendue chaîne de valeur des API ne représente pas toute la valeur des API

En grande partie, l'effervescence autour des API est centrée sur la chaîne de valeur des API illustrée par la figure dans cet encadré : la manière dont les API transforment un modèle économique en écosystème ouvert.

Malheureusement, la plupart des exemples utilisés dans l'industrie au sens large concernent exclusivement des API publiques, alors que ce n'est pas le seul mode d'utilisation des API, et de loin. Surtout, cette image générique de la chaîne de valeur ne permet pas de déterminer quelles API fournir et pourquoi. En effet, elle ne tient aucun compte des différents types d'objectifs métiers qui peuvent vous avoir amené à mettre en place des API. Le chapitre 4 traite de cet écart conceptuel en définissant les points d'entrée types pour API, avec des critères de décision associés.

Chapitre 4

Points d'entrée pour les API

Dans ce chapitre

- Monétiser vos données
- Tirer parti de l'innovation
- Accélérer la transition vers le mobile
- Passer à l'hybride
- Programmer tout

Que signifie vraiment l'expression « penser API » ? Certaines personnes font simplement référence à la prétendue chaîne de valeur des API (voir le chapitre 3), mais est-ce vraiment aussi simple ? Cela justifie-t-il un tel enthousiasme ? Je suis convaincu que la compréhension des API va bien au-delà. Comme nous l'avons vu dans le chapitre 2, il est nécessaire de prendre des décisions concernant l'équilibre entre les API d'entreprise et les API opportunistes. Certaines de ces décisions portent sur les termes et conditions de partage des API, d'autres sur la marche à suivre pour mapper les API requises aux ressources existantes pendant la mise en œuvre des API. Toutes ces décisions, et bien d'autre encore, dépendent de l'objectif métier fixé.

Le mot-clé dans « Penser API » est le premier : penser. Déterminez l'objectif métier que vous souhaitez atteindre, votre public cible, les types d'API requis pour toucher ce public et le traitement à appliquer à vos données et à vos applications (services) pour prendre en charge les API que vous devez fournir. Tout au long de ce processus, n'oubliez pas d'inclure dans vos réflexions les API que vous allez consommer vous-même et les fournisseurs de celles-ci. Ne vous limitez pas à votre rôle de fournisseur d'API : de nombreuses organisations consomment beaucoup plus d'API qu'elles n'en fournissent. Ces éléments constituent le cœur d'une stratégie efficace en matière d'API.

Ce chapitre définit cinq points d'entrée qui, d'après mon expérience, sont représentatifs des objectifs métier et informatiques en lien avec

la réflexion sur les API. Une entreprise peut avoir plusieurs objectifs en même temps, mais chacun a ses propres critères de décision concernant l'adoption d'API. Vous pouvez lire ce chapitre en entier ou uniquement la partie correspondant au point d'entrée le plus adapté à vos besoins.

Monétiser vos données

La monétisation de vos données s'appuie sur l'externalisation des connaissances ou des fonctions sous une forme qui donne envie à des tierces parties d'utiliser ces données ou fonctions. Cette monétisation peut prendre plusieurs formes. L'exemple le plus courant est celui où la tierce partie paie pour utiliser votre API. Dans d'autres cas de figure, vous pouvez payer le consommateur d'API en échange d'une extension de vos capacités commerciales et d'un renforcement de votre écosystème. Vous pouvez même intégrer des partenaires via des API sans qu'aucun paiement direct n'intervienne. Les principaux objectifs de la monétisation sont la consolidation de la position de votre entreprise, la modification de votre chaîne de valeur et l'accroissement de votre influence.

Avec ce point d'entrée, la réussite tient à la qualité de la planification. Si vous pouvez effectuer – je vous le conseille – des tests de type « opportuniste », le produit final doit être un ensemble d'API d'entreprise stables, sur lesquelles une tierce partie peut s'appuyer pendant une période prolongée.

Pour mener votre réflexion sur les API dans un contexte de monétisation des données, voici quelques conseils :

- **Objectif :** le résultat souhaité est monétaire ou basé sur une valeur non monétaire, comme l'extension de votre influence.
- **Public cible :** le public cible est nécessairement une tierce partie. En général, il s'agit de partenaires et de développeurs externes (qui ne font pas partie de votre entreprise ou qui sont recrutés par celle-ci). Traiter une autre activité de votre entreprise comme un partenaire ou une tierce partie n'est pas rare, surtout lorsque les différentes activités sont gérées comme des entités économiquement indépendantes.
- **API à fournir :** les API requises pour mobiliser ce groupe doivent apporter la valeur que vous souhaitez vendre. Ce choix mérite une réflexion approfondie, pas seulement en terme de valeur fournie, mais également au niveau de la forme qui stimule la consommation.

- **Termes et conditions d'utilisation des API :** n'oubliez pas d'inclure dans votre réflexion les termes et conditions qui régissent la consommation des API (« freemium », paiement à l'utilisation ou contrat prépayé).
- **Mise en œuvre des API :** la préparation des données et des fonctions en vue de la mise en œuvre des API obéit à deux impératifs : la qualité et la fiabilité. Certaines personnes pensent que le coût de mise en œuvre est le critère le plus important, mais son rôle dans une stratégie de monétisation des données n'est pas central. Une telle stratégie n'est viable à long terme que si les consommateurs des API en retirent de la valeur et un sentiment de confiance.
- **Consommation d'API tierces :** parfois, vous devez également sélectionner des API à consommer. Même si vous générez de la valeur essentiellement en fournissant des API à des tiers qui les consomment, la mise en œuvre de ces API peut nécessiter la création de versions plus complètes, le plus souvent en fusionnant les API existantes avec des éléments qui font votre spécificité.

La monétisation des données est peut-être le point d'entrée dont vous avez le plus entendu parler, mais ce n'est pas le plus répandu. Actuellement, les projets d'API sont en grande majorité destinés à des utilisations internes, même lorsque les échanges d'API publiques atteignent leur pleine maturité.

Liberté d'innover

La liberté d'innover est aujourd'hui le premier impératif pour de nombreuses entreprises. Tester tôt, apprendre vite et adapter facilement sont des caractéristiques essentielles d'une entreprise dynamique. Le but de ce point d'entrée consiste à identifier les opportunités commerciales de manière agressive et à faire de l'innovation un processus d'apprentissage grâce au modèle suivant :

- Tout est prototype tant que l'efficacité dans la pratique n'est pas démontrée.
- Découvrir que quelque chose n'a pas fonctionné pas n'est pas un échec, mais un apprentissage.
- L'immobilisme est un tremplin vers le déclin.

Comme nous l'avons vu dans le chapitre 3, le rôle des API est de créer un périmètre de calme dans une tempête de changements. Ce rôle revêt deux aspects :

- Fournir rapidement au consommateur d'API ce dont il a besoin et le retirer dès que ce besoin n'existe plus
- Protéger le fournisseur contre l'attrition (consultez le chapitre 3 pour un rappel des raisons pour lesquelles la définition et le déploiement de nouvelles « vues » d'API sur les ressources existantes doivent être faciles et bon marché)

La liberté d'innover est peut-être un point d'entrée moins « glamour » que la monétisation des données (voir la section précédente), mais c'est de loin le cas d'utilisation des API que nous rencontrons le plus chez nos clients. La capacité à créer des fonctionnalités innovantes (internes ou externes) sans grever le budget est un objectif que tout le monde cherche à atteindre. Pour accélérer l'innovation, il faut associer une planification minutieuse à une réaction opportuniste.

Pour mener votre réflexion sur les API dans un contexte de liberté d'innovation, voici quelques conseils :

- **Objectif :** le résultat souhaité consiste à identifier rapidement ce qui fonctionne et ce qui permet de se démarquer, puis à capitaliser sur les réussites.

  Même lorsque vous pensez connaître les besoins du marché, une attitude d'apprentissage offre un retour salutaire au contact de la réalité. Il est facile de mélanger symptôme et cause lorsqu'on étudie des dynamiques de marché complexes.
- **Public cible :** le public cible est essentiellement composé de développeurs internes (salariés ou externalisés). Parfois, des agences externes sont recrutées pour contribuer au travail d'innovation. Dans ces cas, il convient de préparer des contrats plus formels déterminant les API à fournir et les délais. Pourtant, même alors, il vaut mieux garder un comportement opportuniste pour maintenir le rythme d'apprentissage. Les négociations contractuelles limitent la rapidité d'innovation.
- **API à fournir :** les API requises pour mobiliser le public cible mélangent des API d'entreprise prédéfinies et des API opportunistes (voir le chapitre 3) qui répondent aux besoins d'une application innovante. Intégrer des API d'entreprise pour fournir des données essentielles dans un format facilement accessible permet de dynamiser à la fois le développement et la production d'idées nouvelles.

  Les API prédéfinies doivent rester petites et simples. Afficher la structure complète des données telle qu'elle apparaît dans le système d'information principal du client ne faciliterait pas leur consommation par un développeur de canaux innovants.

- **Conditions des API :** les termes et conditions régissant la consommation d'API restent importants, pas en terme de paiement, mais de protection de la sécurité et de la stabilité des systèmes principaux. Après tout, l'innovation est imprévisible.
- **Mise en œuvre des API :** votre traitement des données et fonctions requises pour mettre en œuvre les API est différent pour les API d'entreprise prédéfinies et les API opportunistes fonctionnant à la demande :
  - *Pour les API prédéfinies*, vous devez décider des segments de données à rendre accessibles à l'échelle de l'entreprise. De plus, votre mise en œuvre (en général par proxyfication des services d'entreprise) doit tenir compte du coût d'exécution. Les API prédéfinies étant utilisées de manière imprévisible, le coût d'exécution ne doit pas être un frein à cette utilisation.
  - *Pour les API opportunistes*, les critères primordiaux sont la vitesse et le coût de développement. Si vous devez donner un coup de rein afin de finaliser l'API aujourd'hui au lieu de la semaine prochaine, faites-le tant que vous disposez d'un plan viable de correction de la mise en œuvre de l'API au cas où celle-ci rencontrerait le succès.

    De nombreuses API opportunistes ne vivent pas longtemps. Si vous réalisez qu'il vous faut quelque chose de différent, supprimez l'API opportuniste et reprenez le processus d'apprentissage au début.

- **Consommation d'API tierces :** avec ce point d'entrée, l'identification des API tierces à consommer est un élément important. Voici un exemple simple : créer des applications mobiles est difficile si l'on n'a pas accès aux API des réseaux sociaux bien connus, comme Twitter, Facebook ou LinkedIn. Ces API tierces doivent figurer dans le catalogue que vous mettez à la disposition de vos développeurs internes. Ces derniers ne devraient pas consacrer du temps à les rechercher eux-mêmes. Vous pouvez même convertir ces API tierces en une version personnelle simplifiée, car certaines d'entre elles (celles qui sont publiques) sont relativement complexes dans leur format natif.

Innover n'est jamais simple, mais il existe plusieurs leviers pour faciliter le processus. Une utilisation appropriée des API permet d'intégrer le système principal de l'entreprise dans votre moteur d'innovation, au lieu d'en faire un boulet. Les entreprises

existant depuis longtemps ont l'avantage d'avoir de nombreuses ressources à montrer sous forme d'API. Ceci dit, même pour les start-ups, une approche mettant en œuvre des solutions innovantes à l'aide d'API offre bien plus de flexibilité pour l'apport de données et de fonctions. Grâce aux API, les développeurs peuvent se concentrer sur l'expérience utilisateur et non sur l'intégration. L'utilisation d'API favorise également une expérience omnicanal, car les données et fonctions situées derrière l'API sont par définition distantes et accessibles depuis n'importe quel canal ou application dans toute l'entreprise.

Mobile en 10 minutes

Ce point d'entrée présente une forte corrélation avec le point d'entrée Liberté d'innover (voir la section précédente) et peut donc être considéré comme une variante À ceci près que, dans ce cas, le but consiste à apporter à votre équipe de développement mobile tout ce dont elle a besoin ici et maintenant, selon une démarche opportuniste.

Aujourd'hui, les entreprises doivent créer de nombreuses applications à la fois petites et autonomes, au lieu de portails traditionnels complets. Je suis intimement persuadé que les consommateurs mobiles préfèrent orchestrer leurs propres expériences, plutôt que laisser quelqu'un d'autre prédéfinir le processus. Je le constate dans mon propre comportement. Je consacre rarement plus d'une minute à une application mobile avant de passer à autre chose.

Le point d'entrée Mobile en 10 minutes vise à améliorer la collaboration entre les propriétaires de systèmes principaux et les équipes de développement mobile. Ces équipes peuvent être internes ou externes, mais dans tous les cas, elles doivent pouvoir réutiliser les données et fonctionnalités existantes pour proposer une expérience attrayante. Par exemple, pouvoir consulter les cours de la Bourse est intéressant, mais effectuer des opérations dans le cadre de votre portefeuille d'investissement à partir de cette consultation est un véritable plus.

Ce point d'entrée mise sur la simplicité contrôlée. Masquez la complexité, simplifiez ce que le développeur mobile voit et consomme, et en même temps fournissez les contrôles opérationnels, métier et informatiques appropriés. Notez que le processus doit être rapide pour ne pas ralentir l'innovation mobile.

Juste pour le plaisir, des développeurs IBM ont fait des tests pour voir s'il était possible de prendre un bloc de données sur un système mainframe et de le placer sur un périphérique mobile en 10 à 15 minutes, en utilisant des API. En fait, ils ont réussi ! L'API n'avait rien de séduisant, mais l'expérience a montré que la complexité de la logique d'intégration peut, dans une large mesure, être retirée de l'équation grâce à une utilisation appropriée des API et des technologies d'intégration.

Le point d'entrée Mobile en 10 minutes est entièrement tourné vers l'innovation opportuniste. Pour mener votre réflexion sur les API dans ce contexte, voici quelques conseils :

- **Objectif :** le résultat souhaité est une prise en charge immédiate des besoins de vos équipes de développement mobile. L'équipe mobile doit identifier les données requises par les utilisateurs. Le propriétaire du système principal crée les API qui fournissent ces données. Le mappage aux données et fonctions (services) existantes n'incombe pas au développeur mobile. Cette tâche intervient lors de la mise en œuvre de l'API.
- **Public cible :** votre équipe de développement mobile est le groupe intéressé par ces API.
- **API à fournir :** les API requises pour mobiliser ce groupe sont presque exclusivement opportunistes. Si la chance vous sourit, vous aurez la possibilité d'appliquer des API existantes à une nouvelle application mobile, mais il est préférable de ne pas multiplier les dépendances entre les applications sur certains contrats d'API.
- **Termes et conditions d'utilisation des API :** les termes et conditions autorisant la consommation des API concernent la protection de la sécurité et de la stabilité des systèmes principaux. Cela vaut également pour le point d'entrée Liberté d'innover (voir la section précédente).
- **Mise en œuvre des API :** comme les API produites sont très opportunistes et vivent rarement longtemps, les facteurs les plus importants à prendre en compte sont la vitesse et le coût de développement. Si vous devez donner un coup de rein afin de finaliser l'API aujourd'hui au lieu de la semaine prochaine, faites-le tant que vous disposez d'un plan viable de correction de la mise en œuvre de l'API au cas où celle-ci rencontrerait le succès.
- Consommation d'API tierces : en général, les applications mobiles attractives requièrent une dimension sociale. Il est donc crucial d'identifier les API sociales publiques (comme Twitter, Facebook ou LinkedIn) à consommer et les moyens pour contrôler cette consommation. Si vous le souhaitez, convertissez ces

API en une version personnelle simplifiée (voir la section « Liberté d'innover » plus haut dans ce chapitre). Au moins, vous devez tenir compte de la face cachée des API (voir le chapitre 6) et tenter de minimiser l'impact sur la réputation de votre entreprise si votre utilisation des API publiques soulève des problèmes de stabilité ou éthiques.

L'approche opportuniste de ce point d'entrée peut sembler ingérable en raison du nombre d'API impliquées et de leur réutilisation nécessairement limitée. D'ailleurs, avec la technologie disponible il y a trois ou quatre ans, l'opération aurait été pratiquement impossible pour bon nombre de grandes entreprises. Ce n'est plus le cas ! Les solutions de gestion des API modifient la donne en permettant de définir facilement une API légère et centrée sur le consommateur, à partir du portefeuille de données et de services d'entreprise. Si la création d'une API est un jeu d'enfant, si la gestion et le partage d'un grand nombre d'API avec plusieurs communautés sont simples, la réutilisation et la rigueur architecturale des interfaces d'API deviennent secondaires. La priorité, en revanche, c'est ce qui va permettre au consommateur d'API de réussir.

Vivre dans un monde hybride

Aucun ouvrage sur les API ne serait complet sans évoquer les liens entre les API et le cloud. Ce quatrième point d'entrée traite de l'utilisation des API comme modèle de consommation uniforme dans un écosystème hybride mêlant systèmes locaux et environnements de cloud privé et public. La règle d'or du commerce, c'est la « liberté de choix », la liberté de choisir comment obtenir des fonctions ou des données, et la liberté de déployer des solutions sous la forme souhaitée.

Une approche de l'intégration, fondée sur les API, est régie par les éléments suivants :

- Une entreprise déconnectée n'est pas compétitive. Le cloud permet de mettre en place des solutions hybrides avec des parties dans le cloud et des parties en local. Quant à l'intégration, elle doit être gérée globalement.
- Le cloud s'attache aux fonctionnalités (métiers et informatiques), pas à l'emplacement physique. Un modèle de consommation universel ne doit donc pas être tributaire du lieu.
- Dans un « système de systèmes », il n'y a aucun périmètre de réseau traditionnel à instaurer. Les interactions doivent être contrôlées au niveau applicatif.

Si tout est API (accessible à distance), le consommateur n'a pas besoin de savoir où et comment l'API est hébergée. Les catalogues d'API publiés peuvent et doivent fournir une visibilité sur les limites du domaine et du fournisseur. Dans ce point d'entrée, le catalogue d'API immédiatement consommables est l'un des outils les plus importants pour un développeur. N'incluez pas toutes les API (elles sont trop nombreuses). N'ajoutez que celles qui intéressent le développeur concerné. Celui-ci n'a pas à se demander comment ou pourquoi cette API est à sa disposition. Il doit se focaliser exclusivement sur les possibilités offertes par l'API dès lors qu'elle est disponible.

Dans un monde hybride, la structure du domaine est obligatoirement complexe. Les marchés publics, les catalogues d'API privés, les portails de partenaires, etc. font partie de l'environnement des API. Par conséquent, une structure de communauté bien définie est plus importante que jamais, avec une corrélation directe entre les décisions relatives à la conception de la communauté et celles concernant la conception des API.

La figure 4-1 montre comment un développeur peut utiliser des API pour accéder en toute sécurité à n'importe quelle partie d'un environnement hybride. Lors du développement, le marché des API fournit des informations sur les API mises à la disposition de la communauté de ce développeur. Lors de l'exécution, les passerelles du cloud sécurisent la communication entre l'environnement du consommateur d'API et le terminal de l'API, indépendamment du lieu.

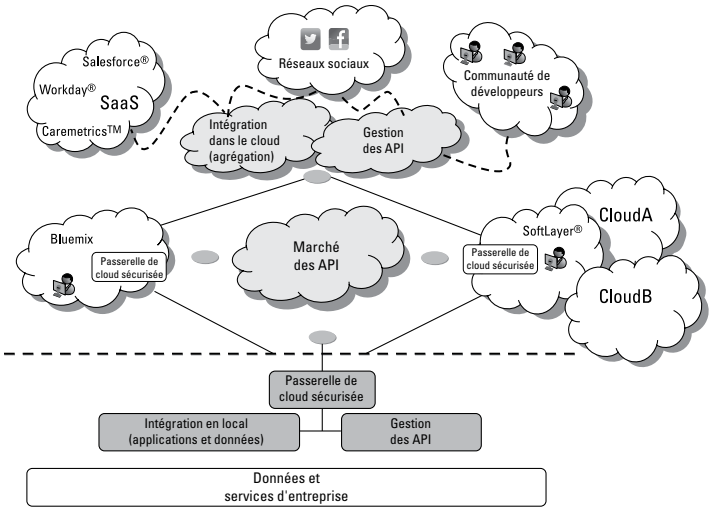

Figure 4-1 : Utilisation des API comme lingua franca d'un environnement hybride.

Pour mener votre réflexion sur les API dans le contexte d'un monde hybride, voici quelques conseils :

- **Objectif :** le résultat souhaité est de rendre autonomes les développeurs et les opérations informatiques. Il convient de placer des contrôles sur les API mises à disposition, et toutes les communications sur un réseau ouvert doivent être sécurisées et gérées de manière appropriée.

  

  La première raison évoquée pour refuser la mise en place d'un cloud hybride est la sécurité. La deuxième est la perte du contrôle des opérations.

- **Public cible :** le public cible regroupe des développeurs internes et externes qui font partie de l'écosystème hybride. D'où l'importance de la conception appropriée des structures de la communauté. Prendre des décisions pour chaque personne est tout sauf pratique. Vous avez besoin d'une structure qui permette de prendre et d'appliquer des décisions concernant le partage d'API au niveau de la communauté, en traitant tous les développeurs d'une communauté de manière équitable.
- **API à fournir :** les API requises pour mobiliser ce groupe dépendent de la structure de la communauté de ce public cible. Pour les publics externes, vous aurez probablement besoin d'un jeu complet d'API d'entreprise prédéfinies. Pour des publics internes, il vous faudra des API opportunistes qui facilitent la création rapide d'applications innovantes. Il n'existe aucune règle absolue pour l'ensemble des cas de figure hybrides. Chaque cas est différent.
- **Termes et conditions d'utilisation des API :** les termes et conditions autorisant la consommation des API peuvent être compliquées dans un monde hybride, tant pour l'aspect métier que pour l'aspect informatique. Peu importe que les API soient gérées ou non (voir le chapitre 2). Pourtant, les mécanismes de mise en œuvre sont très différents dans les deux cas :
  - *Pour les API gérées*, vous appliquez les stratégies métiers et informatiques normalement sur la plate-forme d'API – c'est l'un des avantages d'utiliser des API dans le modèle de consommation uniforme. Assurez-vous simplement que votre plate-forme d'API est axée sur la communauté, pour pouvoir prendre des décisions à ce niveau.
  - *Pour les API non gérées*, le seul moyen d'appliquer des conditions commerciales réside dans la visibilité sur la communauté et dans les contrats formels que vous signez via des canaux secondaires. Lorsque les

API ne sont pas gérées, les opérations informatiques ne peuvent pas mettre en place des contrôles de sécurité et de trafic au niveau des API. Les API non gérées doivent être appelées via des tunnels sécurisés créés au niveau du réseau.

- **Mise en œuvre des API :** votre traitement des données et fonctions requises pour mettre en œuvre les API est différent pour les API d'entreprise prédéfinies et les API opportunistes fonctionnant à la demande. Comme nous l'avons déjà vu dans la section « Liberté d'innover » de ce chapitre, les principales différences portent sur la robustesse et le coût d'exécution, et non sur le temps et le coût de développement.
- **Consommation d'API tierces :** l'identification des API tierces à consommer est plus compliquée pour ce point d'entrée que pour les autres, car, avec le temps, le nombre et la diversité des API disponibles augmentent considérablement.

Le meilleur conseil est de simplifier la consommation d'API au début. Choisissez un petit nombre d'API importantes que vous souhaitez consommer : API sociales, API analytiques, API mobiles d'échange avec les systèmes back-end ou autres, selon vos besoins métiers immédiats. Vous pouvez également convertir les API tierces en une version plus simple ou plus contrôlée. Gardez cette possibilité à l'esprit dans votre processus de prise de décision.

Les environnements hybrides sont intrinsèquement complexes. L'utilisation d'API comme lingua franca permet de faciliter la gestion de cette complexité pour un développeur. Du point de vue des opérations informatiques, la complexité peut devenir ingérable en raison de l'évolution des plates-formes de cloud hybride.

Programmer votre monde

Ce dernier point d'entrée concerne le monde des équipements et des machines. Il repose sur deux idées reçues : la révolution mobile axée sur le consommateur ne concerne pas que les téléphones. La fabrication et la logistique vont contribuer à l'émergence d'un Internet des objets.

Établissements de soins, services collectifs, villes, usines de fabrication, pratiquement partout, une approche plus intelligente unifiant des personnes, des logiciels et des machines s'impose. Aujourd'hui acceptée, l'idée de pouvoir tout programmer avec un même processus intelligent reste nouvelle. Mais, tous les jours, on enregistre

des progrès vers cet objectif dans des domaines comme les voitures intelligentes, les expériences d'achat interactives et les bâtiments écologiques.

Une caractéristique spécifique distingue ce point d'entrée des autres : en général, on ne peut pas modifier un équipement lorsqu'il existe sous sa forme physique. Ce qui fait que les API d'équipement restent en l'état. La priorité consiste donc à proposer une expérience enrichie avec une exécution optimisée :

- Ajoutez la dimension logicielle à la dimension physique.
- Contrôlez les composants, logiciels ou équipements avec des API programmables.
- Utilisez les retours d'information et les connaissances pour optimiser le comportement du système complet, pas seulement d'un seul composant.
- Le cas échéant, monétisez vos capacités de contrôle globales sous la forme d'un jeu personnel d'API (voir la section « Monétiser vos données » dans ce chapitre).

En général, le point d'entrée Programmer le monde porte davantage sur la consommation d'API que sur la fourniture de nouvelles API.

Pour mener votre réflexion sur les API dans le contexte de programmation du monde, voici quelques conseils :

- Objectif : le résultat souhaité est un environnement 100 % programmable. Recherchez une infrastructure physique programmable et créez votre propre logiciel également contrôlable à l'aide d'API.
- Public cible : le public cible est principalement interne. Vos propres développeurs sont ceux qui vont créer les systèmes intelligents équipant les API d'équipement et les API logicielles. Si vous êtes fabricant, vous devez vous assurer que vos propres équipements sont équipés d'API consommables par des tiers, conformément à votre modèle économique.
- API à fournir : les API dont vous avez besoin pour mobiliser ce groupe sont en grande partie définies par les interfaces intégrées dans vos équipements et machines physiques. Le choix n'est pas large, car la plupart des API que vous utiliserez sont prédéfinies.
- Termes et conditions d'utilisation des API : si vous avez accès à un équipement via le réseau, vous pouvez communiquer avec

lui car, par définition, ses API ne sont pas gérées. Les termes et conditions ne sont pas aussi importants pour ce point d'entrée que pour les autres. L'important, c'est d'avoir une bonne bibliothèque d'API d'équipement disponibles.

Si vous souhaitez contrôler l'accès aux API d'équipement, il vous suffit d'ajouter, en amont de celles-ci, une couche d'API proxy gérées avec des contrôles de sécurité intégrés.

- Mise en œuvre des API : les méthodes utilisées pour convertir les données et fonctions permettant de mettre en œuvre les API d'équipement n'entrent pas dans vos préoccupations (sauf bien sûr si vous fabriquez des équipements physiques).
- Consommation d'API tierces : pour ce point d'entrée, l'identification des API tierces à consommer est extrêmement importante. Avoir les bons contrats avec des fournisseurs tiers d'équipements et de machines est essentiel. Si la documentation sur les API n'est pas à jour, aucune communication efficace n'est possible avec l'équipement.

Le monde devenant plus intelligent et plus instrumenté, les fonctionnalités de programmation vont susciter un intérêt croissant. Comme cet environnement est relativement différent d'un environnement exclusivement logiciel classique, il vaut mieux commencer tôt. Cette initiative oblige les développeurs et les acteurs économiques à déterminer ce qu'il faut pour programmer de manière transparente les interactions entre les équipements, les logiciels et les personnes.

Chapitre 5

API et middleware d'intégration

Dans ce chapitre

- Apprendre à connaître les API d'intégration middleware
- Choisir une topologie de domaine

Comme nous l'avons vu dans les chapitres précédents, les API gérées sont différentes des API non gérées. Une API gérée dispose des contrôles métiers et informatiques appropriés, ce qui pose la question de savoir où et comment ces contrôles sont mis en œuvre lors de l'exécution.

La mise en œuvre des contrôles dans le code est généralement une mauvaise idée. Tout d'abord, selon la qualité du code de chaque mise en œuvre, la fiabilité est rarement au rendez-vous. Ensuite, une entreprise a besoin de pouvoir modifier les contrôles sans toucher ni redéployer l'API. Autrement dit, les contrôles doivent être régis par des stratégies gérées indépendamment et définissant à la fois l'objectif opérationnel (métier et informatique) et le runtime d'API mettant en œuvre cet objectif. Par exemple, ces stratégies déterminent le niveau de sécurité requis et le volume de trafic autorisé.

Souvent, un middleware qui héberge les API et met en œuvre des stratégies d'API est appelé *passerelle d'API*. N'oubliez pas que les plates-formes d'API ne gèrent pas seulement l'exécution, mais elles requièrent également des fonctionnalités de temps de développement.

Le middleware d'API n'est pas qu'« un autre ESB »

N'avez-vous jamais entendu dire « J'ai déjà un ESB (bus de service d'entreprise), donc j'ai tout ce qu'il me faut pour les API » ? Faut-il en déduire que le middleware d'API n'est « qu'un autre ESB » ? En fait, pas vraiment :

- Idéalement, les API sont définies par configuration et non par codage. Définir l'API par configuration préserve la légèreté du proxy d'API et permet un remplacement rapide des API nouvelles ou modifiées. En revanche, les outils ESB génériques fonctionnent par flux ou par code, et la mise en œuvre des services obéit au cycle de déploiement des logiciels de l'entreprise.
- Les runtimes d'API doivent être extrêmement rapides, 100 % sûrs, robustes et hautement évolutifs. Idéalement, les caractéristiques réseau d'une passerelle d'API doivent être proches de celles d'un routeur. Donc, l'ajout d'un proxy d'API dans le cadre d'une interaction de bout en bout ne crée jamais de problèmes de temps de réponse ou de latence. Les passerelles d'API les plus rapides et les plus évolutives s'appuient sur des langages de configuration légers, propres au domaine, avec une exécution sans état. Si les runtimes ESB génériques doivent également être rapides, robustes et évolutifs, les moteurs d'exécution doivent prendre en charge la composition complète et certaines interactions avec état. Au final, les runtimes ESB génériques ne peuvent pas être optimisés pour le débit.
- Régis par des stratégies, les contrôles métiers et informatiques des API vont de l'authentification aux contrôles du trafic, en passant par les conditions encadrant la consommation de l'API (plan d'API). Bien que certains ESB complets incluent des stratégies de gestion du trafic, rares sont ceux qui disposent de fonctionnalités de stratégie de sécurité et d'une certification de passerelle d'API complète. Et aucun ne fournit les contrôles métiers intégrés dans un plan d'API.
- Les API doivent être accessibles en libre-service aux développeurs d'application. Tout processus d'approbation après déploiement ralentit le rythme d'adoption et, au final, génère des coûts organisationnels importants.

Le mécanisme de partage privilégié – qui s'est révélé très efficace – consiste à publier les API sur des portails pour développeurs. Surtout dans le cas d'une utilisation interne des API, le partage doit être géré au niveau de la communauté, en s'assurant que les développeurs concernés n'ont accès qu'aux API que leur communauté est censée utiliser.

Les ESB génériques ne proposent aucune de ces fonctionnalités. Même les solutions de gestion de services visent davantage les contrôles du temps de développement et les contrôles opérationnels que l'optimisation du processus de partage entre développeurs.

- Enfin, les propriétaires d'API ont besoin de statistiques métiers pour savoir qui utilise leurs API et comment. Ces statistiques permettent d'évaluer la réussite par rapport aux objectifs métiers du portefeuille d'API au lieu de se concentrer sur les aspects informatiques, comme les tableaux de bord opérationnels généralement fournis par les plates-formes ESB.

Certains estiment qu'il suffit d'un bus reconfigurable pour prendre en charge des besoins d'intégration classiques, une architecture orientée services (SOA) et la gestion des API. Pourtant, en ne proposant pas une expérience spécifique au développeur d'API, au propriétaire d'API et au consommateur, vous risquez d'aligner le middleware sur le plus petit dénominateur commun. De plus, si vous n'avez besoin que des fonctionnalités de gestion des API ou d'intégration génériques, pourquoi payer pour les deux ?

Les entreprises modernes ont besoin à la fois de fonctionnalités de gestion des API et d'intégration, mais sous la forme de deux plates-formes distinctes pour différents publics cibles. Cependant, gardez à l'esprit qu'avec la mise en œuvre d'une architecture SOA, vous disposez d'un bon panel de ressources à découvrir rapidement, à assembler en API (proxy) et à montrer via une plate-forme de gestion d'API.

Topologie d'intégration (répétable)

Traditionnellement, on considère les différents types de passerelles – web, sécurité, messagerie ou API – comme des éléments séparés dans une topologie de déploiement. Cependant, dans un monde hybride d'équipements, de personnes et de logiciels, les frontières entre ces éléments s'estompent rapidement. Face à l'intensification et à la diversification des interactions, il devient avantageux de mettre en place une stratégie de

passerelle unifiée avec un centre de contrôle et de commande pour le trafic franchissant les frontières internes ou externes.

En règle générale, le trafic qui entre ou sort d'un domaine doit toujours transiter par une API. Cette approche de l'intégration offre un niveau souhaitable de visibilité et de contrôle. De plus, elle apporte une flexibilité optimale pour équilibrer la mise en œuvre des API d'entreprise et des API opportunistes, en toute transparence vis-à-vis du consommateur d'API.

Le choix de la granularité de la topologie de domaine incombe à chaque entreprise. Typiquement, les frontières de domaine sont régies par les contraintes suivantes :

- Organisation (par exemple, développeur mobile ou équipe principale)
- Appartenance (frontières sectorielles)
- Sécurité (zone délimitarisée ou réseau interne)
- Qualité de service (système mainframe ou distribué)
- Écosystème (public ou partenaire)

Il est important que ce modèle de topologie d'intégration soit répétable et composable. Pour chaque domaine d'un écosystème, il convient de spécifier la même structure topologique, avec des interactions interdomaines pendant l'exécution qui transitent systématiquement par une passerelle (API), comme le montre la figure 5-1. Dans cette figure, chaque domaine affiche la même structure en cercles concentriques, avec différents types de fonctionnalités d'intégration. La topologie d'intégration standard de chaque domaine peut se présenter sous la forme d'une topologie de « système de systèmes » en répétant la même structure. Cette structure correspond idéalement à la nature « métasystémique » des environnements d'intégration hybride et convient bien à tous les points d'entrée définis dans le chapitre 4.

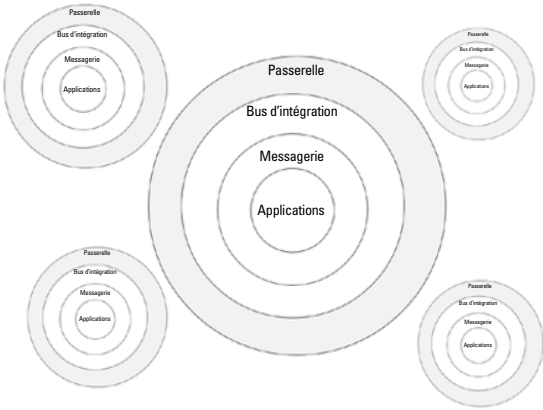

Figure 5-1 : La topologie d'intégration est un processus répétable.

Certaines personnes estiment que la latence ou le temps de réponse est un problème dans une structure où les interactions inter-domaines transitent systématiquement par une API. Les bonnes passerelles d'API relèguent la latence et le temps de réponse au second plan, comme nous l'avons vu dans la section « Le middleware d'API n'est pas qu'un autre ESB » dans ce chapitre. Pour faire une analogie, selon quelle fréquence pondérez-vous le nombre de routeurs entre un navigateur et un serveur Web ? Presque jamais. Vous supposez que l'infrastructure des routeurs est suffisamment rapide et évolutive pour que le nombre de routeurs ne soit pas un problème. À cet égard, les bonnes passerelles d'API sont comme des routeurs. L'ajout d'une de ces passerelles entre le consommateur d'API et le fournisseur d'API ne fait pratiquement aucune différence.

Utilisez des ESB génériques dans le bus d'intégration, en transformant les données et la fonction en services correctement conçus. Ensuite, configurez les API (proxy) pour contrôler l'utilisation de ces services dans l'écosystème au-delà de votre propre domaine. Enfin, dirigez l'intégralité du trafic d'API via une passerelle d'API hautement évolutive, robuste et sécurisée. Ainsi, chaque composante de la plate-forme d'intégration de bout en bout est utilisée là où elle est la plus performante (voir la figure 5-1).

Si vous voulez en savoir plus sur les fonctionnalités de non-chevauchement d'une topologie d'intégration de bout en bout, consultez le redbook IBM *Integration Throughout and Beyond the Enterprise* (en anglais) à l'adresse `www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg248188.html?Open`.

Modèle de référence de l'économie des API et des services

Dans les années 2000, IBM a défini un modèle de référence de l'architecture SOA, qui décrivait les fonctionnalités middleware requises pour créer, déployer et gérer des services efficacement. Dans l'économie des API, ces fonctionnalités restent importantes, car les services constituent la base permettant de définir des API. Mais elles ne suffisent plus.

La figure 5-2 présente le modèle de référence de cette plate-forme middleware étendue. C'est une recommandation pour les fonctionnalités middleware à ajouter dans vos plates-formes d'intégration et d'API.

Dans le modèle de référence SOA initial d'IBM, les éléments relatifs à l'intégration sont restitués sous forme compressée dans la couche inférieure du modèle de référence de l'économie des API et des services. De plus, le nouveau modèle de référence est doté de fonctionnalités permettant de définir et gérer les API destinées aux marchés et portails pour développeurs, ainsi que de fonctionnalités qui accélèrent la consommation d'API.

Il est important de ne pas prendre de décisions concernant les composantes du modèle de référence, prises isolément. Optez plutôt pour une stratégie de middleware intégrée pour transformer vos ressources en avantages concurrentiels.

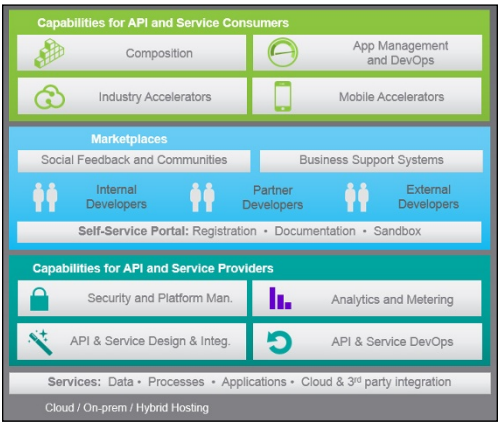

Figure 5-2 : Modèle de référence de l'économie des API et des services.

Chapitre 6

10 choses à savoir sur les API

Dans ce chapitre

- Seeing that APIs are products
- Recognizing that design never stops
- Understanding that every API needs an owner

La manière dont une organisation mène une réflexion efficace sur les API fait partie du processus. Ce chapitre répertorie certains enseignements que j'ai tirés de mon propre parcours dans le monde des API.

L'expérience omnicanal crée le besoin d'API

Le concept de base des API n'est pas nouveau. Aujourd'hui, la différence réside dans le fait que les utilisateurs modernes (consommateurs et entreprises) attendent une expérience omnicanal qui soit à la fois sociale et personnelle. Pour être véritablement personnelle, cette expérience doit être auto-orchestrée, du moins dans une certaine mesure. Aucune entreprise ne peut plus suivre un processus de canal universel.

L'auto-orchestration s'adresse inévitablement aux micro-applications qui, en retour, créent un besoin d'API spécialisées. Une expérience omnicanal implique un écosystème comprenant des personnes, des logiciels et des équipements, ce qui, là encore, rend les API spécialisées nécessaires.

Les API sont des produits métiers

Considérer les API comme des produits métiers permet de faire plus facilement la distinction entre une approche axée sur les API et une approche classique fondée sur la fourniture de logiciels. Pour les produits, vous devez poser plusieurs questions clés :

- Quel est le public cible ?
- Que souhaite acheter ce public cible ?
- Dans quelles conditions suis-je prêt à vendre ?

Les termes *acheter* et *vendre* sont utilisés volontairement même si les modèles économiques sous-jacents des API varient considérablement. Peu importe que le retour se matérialise sous forme d'argent ou d'influence, que ce soit le consommateur ou le fournisseur qui paie, l'API reste un produit.

La conception métier est une initiative de bout en bout

Les API ne relèvent plus que des seules compétences informatiques. Elles doivent faire partie de votre conception métier de bout en bout.

Imaginez quelqu'un qui partage ses expériences de voyage sur les réseaux sociaux. Un jour, il envoie un tweet pour raconter une mauvaise expérience avec la compagnie aérienne A. Dix minutes plus tard, il reçoit un courrier électronique de cette compagnie, disant : « Nous nous excusons pour le désagrément que vous avez subi. Voici ce que nous pouvons faire pour vous. » La semaine suivante, il fait un excellent voyage avec la compagnie aérienne B, et comme à son habitude, il le signale en envoyant un tweet. Cinq minutes plus tard, cette compagnie lui renvoie le tweet suivant : « Nous sommes très heureux que vous soyez satisfait. À très bientôt sur nos lignes. »

Ces deux compagnies aériennes ont prévu à l'avance d'imbriquer les réseaux sociaux dans leurs modèles opérationnels métiers grâce à l'utilisation d'API.

L'instrumentation des API permet d'acquérir des connaissances

Tester tôt, apprendre vite, adapter facilement – une partie de l'équation réside dans la capacité à apprendre vite, et la meilleure manière d'apprendre consiste à exploiter dans les informations disponibles dans le système opérationnel métier. Vous accédez facilement à ces informations grâce à l'instrumentation des API et à l'utilisation de fonctionnalités d'analyse métier associées qui doivent faire partie d'une plate-forme middleware d'API 100 % fonctionnelle.

Toutes les API ne sont pas de type REST

L'affirmation « SOAP est mort ; les API sont toujours de type REST » est une idée reçue. Bien que la plupart des API modernes reposent sur REST/JSON plutôt que sur SOAP, rien ne dit que le formalisme de SOAP soit hors course. Simplement, il n'est pas nécessaire pour la consommation d'API par des utilisateurs (le cœur du débat actuel sur les API). Le protocole SOAP est encore exploitable de différentes manières dans les communications entre machines et les outils de composition formelle. Ce n'est pas sans de bonnes raisons que l'industrie a élaboré plusieurs protocoles de liaison. Les bons concepteurs choisissent celui qui est le plus approprié à leur objectif.

Chaque API a besoin d'un propriétaire

Pour faire simple, une API a besoin d'un propriétaire qui en soit responsable et qui puisse prendre des décisions. Naturellement, l'homme refuse de posséder quelque chose qu'il ne contrôle pas totalement (comme dans la formule « Seule cette partie relève de ma responsabilité »). Pourtant, vous devez attribuer un propriétaire à chaque API. Cette personne devient décisionnaire sur les questions de productisation et de partage.

Les versions des API doivent être gérées

Ne pas gérer les versions des API, c'est comme ne pas changer les couches d'un bébé. Inévitablement, les problèmes arrivent, et alors, c'est à vous de les gérer. Affirmer que la gestion des versions est superflue signifie laisser les consommateurs d'une API s'en occuper eux-mêmes.

Comme les API sont des produits métiers, vous devez gérer leurs versions judicieusement. Limitez la multiplication des versions en ne proposant une nouvelle version officielle qu'en cas de non-rétro-compatibilité d'une mise à jour.

Les stratégies facilitent le contrôle des API

Les stratégies sont les outils traditionnellement utilisés pour codifier les intentions des opérations métiers et informatiques. Elles conditionnent donc la cohérence et l'intégrité. Les conditions dans lesquelles les API sont consommées et gérées doivent être codifiées sous la forme de stratégies mises en œuvre par des plates-formes d'API.

Il est important de s'assurer que ces stratégies sont modifiables indépendamment de la logique de l'API (comme l'interface et le mappage des données). Ceci favorise la modification dynamique du comportement de fonctionnement.

Les API ont une face cachée

Sans intégrité, l'innovation est fragile. Qu'il s'agisse de non-respect de la promesse d'intégrité, de divulgation d'informations sensibles ou de comportement inapproprié, le résultat est en grande partie le même. Il suffit d'une mauvaise expérience pour qu'une personne vous retire sa confiance.

Lorsque vous utilisez une API tierce, veillez à ce que l'intégrité de votre entreprise n'en souffre pas. Le vecteur utilisé – contrats formels assortis de sanctions, mécanismes de compensation ou évaluations pertinentes de la robustesse et de la sécurité des API – importe moins que la mise en place de mesures de précaution appropriées. N'oubliez pas d'inclure les questions éthiques dans votre réflexion.

Mettre la puissance des API au service de vos objectifs métiers

Croyez à la puissance des API ! Votre entreprise a besoin de créer des solutions omni-canal, d'innover plus vite que la concurrence et de devenir une entreprise mobile ou de travailler dans un environnement de cloud hybride. Les API modernes vous permettent d'utiliser des écosystèmes ouverts et de nouveaux modèles économiques. Ce guide pratique vous explique comment utiliser les API pour le bénéfice de votre entreprise.

- ***Considérez une API comme un produit :*** ***elle représente quelque chose que vous avez choisi de partager avec un public cible.***
- ***Tester tôt, apprendre vite et adapter facilement :*** ***découvrez une approche expérimentale des API.***
- ***Utilisez toujours les API pour délimiter votre domaine :*** ***gardez un contrôle et une visibilité sur les trafics entrant et sortant.***
- ***Utilisez des plates-formes d'API spécialisées :*** ***simplifiez et sécurisez à 100 % le partage d'API.***

Commencez votre lecture et découvrez :

- **L'anatomie d'une API**
- **Comment gérer les API**
- **La nature des bonnes API**
- **Les points d'entrée pour API**
- **Le middleware d'API qu'il vous faut**
- **10 choses à savoir sur les API**

WILEY

ISBN: 978-1-119-19332-6
Part #: WSM14025-FRFR

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.