CONCOURS
TECHNICIEN PRINCIPAL TERRITORIAL DE 2ème CLASSE

EXTERNE, INTERNE & 3ème VOIE

SESSION 2016

EPREUVE DE RAPPORT AVEC PROPOSITIONS

ÉPREUVE D'ADMISSIBILITE :

Rédaction d'un rapport technique portant sur la spécialité au titre de laquelle le candidat concourt. Ce rapport est assorti de propositions opérationnelles.

Durée : 3 heures
Coefficient : 1

SPECIALITE : INGENIERIE, INFORMATIQUE ET SYSTEMES D'INFORMATION

À LIRE ATTENTIVEMENT AVANT DE TRAITER LE SUJET :

- Vous ne devez faire apparaître aucun signe distinctif dans votre copie, ni votre nom ou un nom fictif, ni votre numéro de convocation, ni signature ou paraphe.
- Aucune référence (nom de collectivité, nom de personne, …) **autre que celles figurant le cas échéant sur le sujet ou dans le dossier** ne doit apparaître dans votre copie.
- Seul l'usage d'un stylo à encre soit noire, soit bleue est autorisé (bille non effaçable, plume ou feutre). L'utilisation d'une autre couleur, pour écrire ou pour souligner, sera considérée comme un signe distinctif, de même que l'utilisation d'un surligneur.
- L'utilisation d'une calculatrice de fonctionnement autonome et sans imprimante est autorisée.
- Le non-respect des règles ci-dessus peut entraîner l'annulation de la copie par le Jury.
- Les feuilles de brouillon ne seront en aucun cas prises en compte.

Ce sujet comprend 23 pages

Il appartient au candidat de vérifier que le document comprend le nombre de pages indiqué
S'il est incomplet, en avertir le surveillant

Vous êtes technicien principal territorial de 2ème classe dans la commune de Techniville (3 000 agents), en qualité de chef de projet, en charge de l'intégration des applications NTIC.

La Direction des Systèmes d'Information se compose de 30 agents, répartis au sein de 5 services.

Ses moyens sont les suivants :

- 1 450 PC et 150 serveurs ;
- 250 PC et 10 serveurs du C.C.A.S. ;
- 90 applications métier ;
- 7 sites WEB Internet et 3 sites WEB Intranet.

Ses missions concernent les fonctions classiques d'une Direction des Systèmes d'Information :

- Gestion des systèmes informatiques et téléphoniques et leur maintien en conditions opérationnelles ;
- Accompagnement des métiers dans la mise en œuvre des systèmes d'information ;
- Garantie de la sécurité et du bon usage du S.I. ;
- Anticipation et planification des évolutions.

Dans un premier temps, le Directeur des Systèmes d'Information vous demande de rédiger à son attention, exclusivement à l'aide des documents joints, un rapport technique sur les enjeux des Interfaces de Programmation aux Applications (A.P.I.).

(**10 points**)

Dans un deuxième temps, il vous demande de formuler un ensemble de propositions opérationnelles, sur la mise en place des Interfaces de Programmation aux Applications (A.P.I.).

(**10 points**)

Pour traiter cette seconde partie, vous mobiliserez également vos connaissances.

Liste des documents joints :

DOCUMENT N°01 « Comprendre les interfaces de programmation ».
www.internetactu.net - 24/06/11 | Rédigé par *Hubert Guillaud* (7 pages).

DOCUMENT N°02 « A.P.I. définition et liste de celles qu'il faut connaître ».
www.scriptol.fr / 2015 / (2 pages).

DOCUMENT N°03 « **Les A.P.I., véritables outils d'aide à la décision** ».
blog.atinternet.com – 19/03/2012 | *Rédigé par Benjamin Diolez (5 pages).*

DOCUMENT N°04 « Conception d'A.P.I., construire correctement une interface de programmation d'application. ».
www.lemagit.frl - mars 2015 | Rédigé par *Tom Nolle - CIMI Corporation* (3 pages).

DOCUMENT N°05 « Les nouveaux enjeux des API pour les organisations ». (Extrait)
Cabinet Voirin - juin 2015 | Rédigé par Colin LESPRIT & Sinan SERDAROGLU (4 pages).

DOCUMENT N°01

Comprendre les interfaces de programmation

www.internetactu.net le 24/06/11 | Rédigé par *Hubert Guillaud.*

Les interfaces de programmation permettent à des services de s'échanger des données entre eux. Elles peuvent permettre à un site web d'utiliser le moteur de reconnaissance d'image d'une autre société pour l'intégrer à son service de stockage d'image par exemple ou à une librairie en ligne de publier sur votre profil Facebook ou Twitter le livre que vous venez de lui acheter. Stade suprême de l'intégration des services ou porte ouverte à la publicisation sans contrôle de soi, les API ont un rôle de plus en plus important dans le web d'aujourd'hui.

Pour mieux comprendre le rôle et le fonctionnement des interfaces de programmation (API pour *Application Programming Interface*), le mieux est de le demander à des gens qui les utilisent. Karl Dubost responsable des relations avec les développeurs chez Opera, Johann Daigremont à la tête du departement des communications sociales aux Bell Labs d'Alcatel-Lucent, et Alexandre Assouad, concepteur de projets chez FaberNovel, avec leurs expériences différentes, reviennent sur le rôle, le fonctionnement et les enjeux des API, qui structurent déjà l'internet de demain.

InternetActu.net : Qu'est-ce qu'une API, concrètement ?

Karl Dubost : Une API est une interface. Un protocole de communication pour accéder à un service. De la même façon que dans un logiciel de base de données tu as un vocabulaire pour accéder aux données et faire une requête, l'API permet de construire des interrogations par une interface normalisée.

Selon les services Web, les API offrent un certain nombre de fonctionnalités. Celles-ci également évoluent au cours du temps. Dans le cas de Facebook, le lieu pour découvrir ses APIs est le site des développeurs. Facebook a plus d'une API. Ils en ont pour explorer le graphe, pour s'authentifier, etc. Comme c'est un service, les développeurs doivent identifier leurs éventuelles applications afin que Facebook puisse d'abord contrôler la façon dont l'API est utilisée et ensuite éviter les dépassements de resources (requête trop fréquente par exemple).

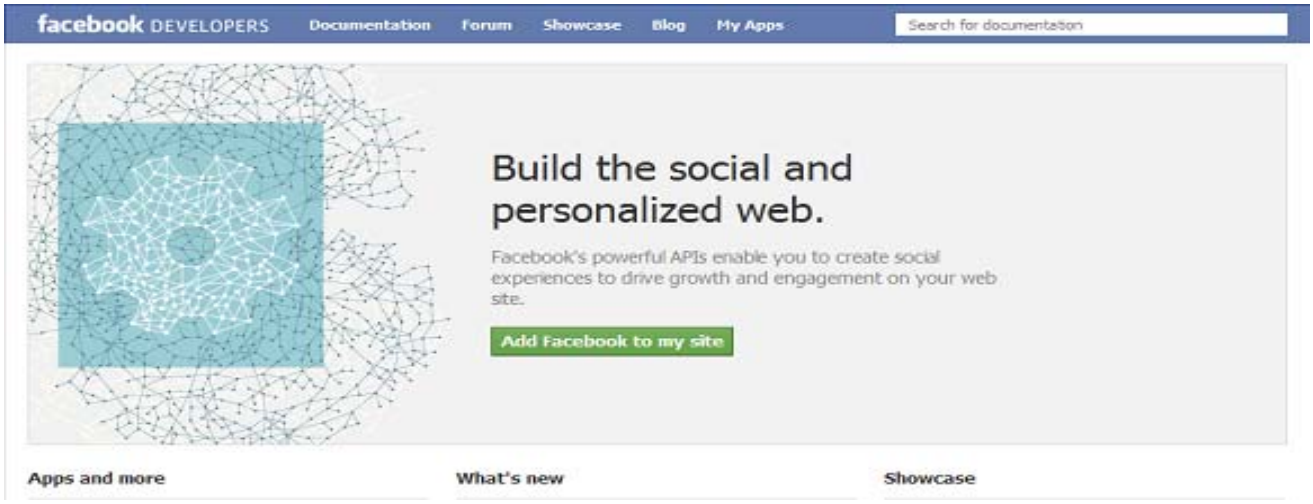


Image : le site des développeurs de Facebook.

Johann Daigremont : Une API permet à deux programmes de s'échanger des données. Le premier utilise l'API offerte par le deuxième pour bénéficier de ses services et données. L'API définit un langage commun entre les deux programmes. L'ensemble des grands acteurs du web propose désormais leurs services via leurs API. Au lieu de rester fermés, ces acteurs ont en effet décidé de s'ouvrir pour être capable d'offrir des modalités de développement accessibles et bénéficier des millions de développeurs de la toile (on appelle cela le *crowdsourcing*) : comme permettre de développer des quizz ou des jeux en utilisant les interfaces sociales de Facebook par exemple.

L'API décrit des fonctions et des méthodes pour accéder à certaines propriétés de certains sites comme Facebook, Twitter, MySpace… Ces interfaces de programmation permettent à un développeur d'interagir avec le système. Il y a différents types d'interfaces. Certaines ne permettent que de faire des interrogations (on peut chercher de l'information), d'autres permettent d'écrire de l'information (on peut par exemple "écrire" un statut pour une personne). La description des API, est basée sur des requêtes HTTP et du XML, permettant d'utiliser un langage très simple pour les lire et les interroger.

Le site web GoogleApps/APi permet de voir la liste des interfaces de programmation de Google disponibles. Celles de Facebook sont documentées dans le répertoire et dans le site dédié aux développeurs.

Alexandre Assouad : Facebook est un exemple un peu complexe et particulier. On peut distinguer deux grands types d'API : celles qui permettent de créer des services dans Facebook (comme d'y créer Farmville, le célèbre jeu) et les API d'authentification qui permettent de ramener le graphe social d'un utilisateur dans un autre service (notamment via l'API Facebook Connect).

L'API Facebook Connect a créé un cercle vertueux pour améliorer l'expérience utilisateur en permettant d'intégrer une expérience sociale à n'importe quel site. L'API génère du trafic qui transite par Facebook et enrichit le flux des utilisateurs.

InternetActu.net : Qu'est-ce qu'on fait avec les API Facebook ? Quelles sont les API les plus utilisées ? A quoi servent-elles ?

Karl Dubost : Je pense que l'API Facebook la plus utilisée est celle d'authentification. Car elle permet aux gens de pouvoir commenter et voir des amis ou des infos relatives aux pages Web sur tous les sites.

Elle n'est pas innocente non plus. Elle permet à Facebook de tracer toutes les navigations d'un utilisateur Facebook sur tous les sites Web avec des fonctions Facebook. C'est un véritable cheval de troie à l'échelle du Web. C'est bien pour cela que Google sort son +1.

Alexandre Assouad : L'API la plus utilisée de l'internet demeure Google Maps, qui fournit un service de cartographie gratuit. Il faut dire que dans le monde des interfaces de programmation, Google a un modèle spécial : il offre un plein accès à ses API. Ainsi, faire du géocodage d'adresse sur Google Maps est totalement illimité. Mais si on veut faire du géocoding d'adresse en dehors de Google Maps on est limité en terme de requête. Google favorise l'utilisation de son propre service, mais s'offre la possibilité de récupérer le contenu d'une Google Maps pour l'indexer.

Le souci c'est qu'il y a désormais une API pour tout. Le problème est de déterminer où s'arrête la définition de l'API. Celle-ci a tendance à devenir de plus en plus un *widget* qui est intégré sur une page : les petits widgets de Facebook (code en javascript que n'importe qui dépose sur son site web) peuvent être appelés des API, car ils permettent d'intégrer Facebook facilement, mais ce ne sont pas des API au sens le plus technique. Car, si en terme de fonctionnement c'est assez proche, en terme d'intégration, ce n'est pas la même chose. En tant que développeur, on a sa disposition du code pour manipuler les éléments que l'on veut obtenir et faire s'afficher.

InternetActu.net : Peut-on accéder à tout via les API ?

Johann Daigremont : Tout n'est pas ouvert. Les développeurs n'ont pas accès à tout. Facebook par exemple n'ouvre pas tout. D'abord, il faut montrer patte blanche. Il faut s'enregistrer comme développeur auprès de la plupart de ces structures avant qu'ils vous acceptent. Il existe par exemple un *Service Level Agreement* (SLA ou accord de niveau de service) qui définit un contrat d'utilisation entre une application tierce et le site web auquel l'application accède. Ces contrats listent les méthodes autorisées, la périodicité ou la quantité de données qui peuvent transiter entre les deux services. Tous les services ont ainsi des limites par exemple, Twitter limite à 150 le nombre de requêtes par heure par une application externe, Facebook a mis en place des limitations dynamiques sur le nombre de requêtes par jour par une application en fonction de "l'affinité" montrée par les utilisateurs pour votre application.

Les formes de SLA sont variées : avec Apple ou la plupart des opérateurs de télécommunication, il faut faire parvenir un courrier, mais Facebook se contente d'une déclaration en ligne. Apple vérifie que les sources de l'application soient en conformité avec l'API utilisée.

Mais pour tous ces acteurs, l'approche est la même : attirer les développeurs en proposant des interfaces de programmation ouvertes pour que ceux-ci puissent construire des millions d'applications.

Ensuite, les utilisateurs ont également un rôle : l'application doit demander à la personne une autorisation à chercher les données dont elle besoin. Beaucoup de systèmes ont des API spécifiques. Mais il existe quelques protocoles communs comme OpenSocial, un consortium qui décrit les spécifications des interfaces et permettant d'utiliser la même interface de programmation pour interroger de nombreux services. Sur Facebook on dénombre plus de 500 000 applications. Mais par application, on parle de tout et de rien. L'essentiel est très simple et consiste seulement à permettre l'identification, comme c'est le cas avec OpenSocial.

Image : OpenSocial.

InternetActu.net : L'internet des API est-il un internet payant ou gratuit ?

Johann Daigremont : En fait, les deux ! Les acteurs du web ouvrent souvent leurs services de manières modulaires : une première partie des méthodes offertes via l'API est accessible gratuitement, ce qui permet d'attirer un maximum de développeurs, une seconde partie est payante pour des méthodes plus avancées. La partie gratuite est souvent limitée avec des seuils de requête. La plupart des acteurs du web proposent une partie de leur API gratuitement ; les opérateurs de télécommunication, qui ont longtemps proposé des API payantes, évoluent et commencent à offrir des API pour ouvrir les capacités de leurs réseaux à des communautés de développeurs.

Alexandre Assouad : Il existe des API où il y a des limitations en terme de requêtes, car les fournir coûte cher. Il y a un coût des requêtes à la seconde, notamment sur les API de Twitter, Viadeo ou Linked-In, avec des taux maximaux de requête par heure. Sur Google également il y a des API avec des taux limités, et si vous souhaitez les dépasser (c'est-à-dire dépasser les 10 000 ou 20 000 requêtes jours), il faut payer. Il existe des API totalement payantes, mais le modèle est le plus souvent freemium, permettant aux développeurs de tester les services tout en étant limités en terme de volume.

InternetActu.net : On a l'impression que les API sont hors économie. Que les échanges B2B sont devenus un échange basé sur la confiance et la réciprocité ? Est-ce si juste ? Ou au contraire, est-ce la base de lourdes négociations de l'ombre comme l'ont montré les tensions entre Facebook et Google ?

Karl Dubost : Pas du tout. :)

Les API sont soient en accès libre, soient identifiées et même payantes, comme par exemple le service Amazon S3. Il n'y a aucune confiance, ni réciprocité. Lorsqu'une société te laisse mettre l'authentification Facebook "gratuitement" sur ton site, c'est pour mieux pomper les données des utilisateurs. À chaque fois que Google donne la possibilité aux gens de mettre une carte Google Map, c'est l'opportunité de tracer les gens et leurs intérêts avec la combinaison de recherche géographique et Doubleclick, la régie publicitaire de Google. Ce qui est en jeu, c'est la construction fine de profils marketing pour mieux vendre de la publicité.

Alexandre Assouad : Dès le début il y a eu les 2 modèles. Les modèles d'API existent depuis longtemps. La réservation de billets dématérialisés pour le train par exemple utilisait des API d'un prestataire payantes.

Plusieurs modèles existent entre le modèle gratuit et le modèle payant freemium (c'est-à-dire un modèle qui permet d'accéder à un service de base, gratuit, mais qui devient payant quand on veut augmenter l'accès au service que ce soit en terme de capacités ou de fonctionnalités). Dans le cas du modèle payant freemium, il s'agit d'une prestation avec des conventions en terme de qualité de service ou de réactivité par exemple.

C'est le modèle gratuit qui est plus récent et qui a pour but de créer un cercle vertueux entre les acteurs qui l'utilisent, afin que chacun y trouve son compte. Ce modèle gratuit est assez lié au *crowdsourcing*. Par exemple, l'API de Foursquare a été auto-entretenue par les utilisateurs. Les gens renseignaient des lieux

qu'ils n'y trouvaient pas. Tant et si bien que l'API de géolocalisation de Foursquare est devenue très intéressante, car elle ne me propose plus une longitude et une latitude, mais des lieux et leurs noms…

Parfois, le *crowdsourcing* permet de fournir un service plus intéressant, de le développer parce que les gens créent le contenu et l'apportent sur la plateforme… On a donc des modèles économiques differents dans les API, mais aussi des types de services différents, qui expliquent que l'un est souvent payant et l'autre souvent gratuit.

Johann Daigremont : Ouvrir ses services à des tiers via une API n'est pas anodin. C'est au contraire une réelle stratégie économique choisie par celui qui fournit ce service. Cela permet d'attirer des communautés de développeurs et de bénéficier d'une masse critique que vous n'avez pas en interne dans votre entreprise pour apporter de nouvelles fonctionnalités, de nouvelles applications, auxquelles vous n'auriez pas pensées ou que vous n'auriez pu développer. Cela permet également de suivre vos utilisateurs dans leurs usages d'autres applications, ce qui permet de construire des profils utilisateurs plus complets, profils pouvant ensuite être revendus pour du marketing ciblé.

Il me semble qu'on constate une réelle volonté des acteurs du web pour ouvrir les informations, même si la donnée est ce qui fait la richesse. Car c'est une ouverture intelligente. Il s'agit de développer des applications qui tirent partie de ces données, sans permettre pour autant de copier ou de pouvoir reconstruire la base de données à laquelle on accède.

InternetActu.net : Quelles sont les limites des API ? En les utilisant, les développeurs dépendent de règles qu'ils ne maîtrisent pas et qui peuvent changer au cours du temps, comme l'a récemment montré Google.

Alexandre Assouad : Ce que les développeurs attendent d'une API, c'est qu'elle soit normalisée, stable dans le temps. Pendant longtemps, il y a eu trop de changement dans les API de Facebook pour que les développeurs puissent suivre (et aussi sur leurs orientations en matière de vie privée) sans compter que Facebook communiquait assez peu en amont sur les modifications qu'ils apportaient. C'est effectivement le risque des API : en les utilisant, elles peuvent ne pas être toujours valides demain.

Google prévient assez en amont et versionne ses API. Sur Google Maps par exemple, il y a plusieurs versions d'API qui sont utilisables et maintenues. Ils communiquent sur leurs nouvelles API, ils avertissent des changements et ont une politique d'accompagnement dans le changement.

La différence entre Google et Facebook c'est qu'ils préviennent un peu plus tôt.

On constate également que Google change actuellement son fusil d'épaule : la stratégie est en train de se modifier et on trouve beaucoup plus d'API payantes qu'avant. Bien évidemment, quand cela devient payant, les développeurs cherchent à utiliser d'autres services, ce qui encourage d'autres alternatives. On trouve en Open Source des briques d'alternatives qui peuvent remplacer certaines API. L'open source a l'avantage de permettre une meilleure intégration : on peut les déployer sur ses propres serveurs, ce qui permet de s'affranchir des limitations en terme de requêtes.

Karl Dubost : De la même façon que dans le monde physique nous allons dans un restaurant parce qu'il propose un menu végétarien, il est fort probable que le service disparaisse ou soit modifié de manière conséquente si le propriétaire ou le cuisinier change. On doit alors se trouver un nouveau restaurant…

L'enjeu en ligne est de deux ordres. Les services sont souvent uniques ou peu nombreux (Facebook, Twitter, Google Maps…) et les APIs utilisées sont rarement normalisées. Si l'un des services ferme et que tout un développement s'appuyait sur les options de l'API, les développeurs sont marron. D'où le besoin de maîtriser l'indépendance de ses données.

InternetActu.net : Peut-on encore développer un site web sans se brancher sur des API ?

Alexandre Assouad : Bien sûr, un site peu exister sans API. Mais les API servent à améliorer les services ou ajouter des fonctionnalités. L'API on peut la voir de deux manières : soit elle enrichit et améliore l'expérience de l'utilisateur sur mon site, soit elle répond à un besoin technologique que je n'ai pas envie de réinventer. Il y a des API qui permettent d'intégrer de la reconnaissance d'image par exemple, ce qui permet au développeur d'un service de ne pas avoir à tout redévelopper alors que d'autres le font très bien. On peut profiter d'un écosystème existant. D'un autre côté, intégrer un Facebook Connect ou Twitter commence à entrer dans la norme…

Utiliser des API permet de se focaliser sur son corps de métiers, sans s'occuper des briques techniques qu'on utilise, un peu comme quand on sous-traite dans une entreprise. On peut clairement monter des services en développant très peu et en se basant sur plein d'API externes et en proposant un mixe de services très intéressants. Foursquare par exemple est assez proche d'un mashup d'API : une API de géolocalisation et une de partage avec ses relations… Ils ont recréé une base technique bien sûr, mais leur service aurait pu tout entier reposer sur des API existantes… En tout cas, on pourrait facilement créer un nouveau Foursquare avec des API existantes. C'est du Lego. Il faut juste que les briques tiennent et aient du sens.

Karl Dubost : Oui on peut développer un site Web sans forcément utiliser l'API d'une autre société. La question est plutôt qu'elles sont les API que je peux utiliser tout en minimisant les risques pour mon business d'en être victime.

InternetActu.net : On a l'impression que désormais, la plupart des nouveaux services naissent du croisement des API, comme le montre ProgrammableWeb, une base de données d'API rachetée par Alcatel…

Johann Daigremont : Les *Mashups* permettent à un développeur d'utiliser plusieurs API et des les "composer" pour créer un service. Pages jaunes par exemple utilise les API de Google Maps avec une API de données sémantiques pour afficher sur une carte la liste des docteurs. C'est la une composition très classique. Le plus souvent les développeurs se contentent de combiner leurs propres données avec l'interface de Facebook.

Mais il existe beaucoup plus d'API que n'en recense Programmable Web.

Karl Dubost : Ce n'est pas forcément le cas de tous les services. Certains vont créer des protocoles originaux, d'autres vont réutiliser entièrement un autre protocole. Par exemple, le service de suivi de conférence Lanyrd ne propose un *login* que et exclusivement par Twitter (ce qui est cela dit un peu limité).

L'enjeu de la normalisation est bien plus intéressant. Il nous évite d'avoir à ouvrir des comptes sur X services pour pouvoir les utiliser, sans être non plus dépendant d'un service tiers spécifique…

InternetActu.net : On a un peu l'impression que le paysage qui se dessine ressemble à un internet des API, ouvert, dans les nuages, mais réservé aux développeurs et un web des applications, un monde clos, limité, réservé aux usagers ?

Karl Dubost : Une API est un point de communication pour une application. Ce n'est pas une opposition. Il y a des API fermées, d'autres ouvertes, etc. Il y a des applications libres et des applications fermées.

Alexandre Assouad : Sauf que les API ne sont pas si ouvertes que cela. Ce qui est vraiment ouvert, c'est le code en open source qu'on peut répliquer. Sur Google, l'API de Geocoding (qui permet d'associer des coordonnées géographiques à des adresses) me limite à 20 000 requêtes par jour : si je les atteints, pour l'instant, je ne peux plus rien faire. Par contre si j'intègre une brique open source pour intégrer ce géocoding sur mes propres serveurs, je ne suis pas limité. L'API permet une certaine ouverture, mais il n'est pas si ouvert… Si je ferme l'API, si je la modifie, je mets dans une situation difficile beaucoup de gens.

Cela dit, l'API est un monde fermé pour l'utilisateur final. C'est un monde réservé aux développeurs.

Sur Twitter par exemple, l'API permet de faire plus de choses que ne le propose le site. Il y a une dissymétrie entre le service disponible en ligne et ce que l'on peut faire via les API. L'écosystème lié aux API s'est beaucoup développé ce qui a permis de résoudre les problèmes de charge que connaissait le service qui tombait fréquemment. Ils ont développé des API plus solides et construit des services depuis celles-ci. Et l'API Twitter permet de faire plus que ne le propose le site : comme de rechercher en profondeur dans l'historique des comptes par exemple. Elle est devenue prépondérante par rapport au site. Elle permet à de très bons services d'exister, comme Tweetdeck.

DOCUMENT N°02

A.P.I. définition et liste de celles qu'il faut connaître.

www.scriptol.fr / *2015* / (2 pages).

L'API est avec le langage de programmation, le principal outil du programmeur, et il convient avant de démarrer tout projet de rechercher les APIs disponibles sur le Web qui permettront d'éviter de réécrire un code déjà existant.

Une API (Application Programming Interface) est un ensemble
- d'en-têtes de fonctions,
- de classes et leurs membres (pour un langage orienté objet),

qui sont fournis par une bibliothèque logicielle, par un service web, par un système d'exploitation, de sorte qu'ils puissent être utilisés pour programmer un logiciel qui les emploie.

L'implémentation des fonctions et classes, qui peuvent être propres à un matériel, se distingue de l'API qui en est indépendante. Toutefois l'API n'existe que si on dispose d'au moins une implémentation.

Accéder au hardware... un SSD de Seagate

On doit faire la différence entre l'API, et la bibliothèque. Une même API peut avoir différentes implémentation, comme nous le rappelle le procès opposant Oracle à Google, le second ayant réalisé une implémentation propre de l'API Java standard, ce qui a incité Oracle, pour tenter de récupérer des royalties sur Android, de demander que l'on statue en faveur d'un copyright sur les API aussi, ce qui serait fort dommageable. L'Europe pour sa part a statué contre cela.

Définir une telle interface ne se limite pas aux bibliothèques, une API peut être aussi être définie pour une application finale de sorte qu'elle puisse communiquer avec d'autres applications.

Caractéristiques des APIs

Dépendance au langage

Une API peut être utilisable dans un unique langage de programmation ou être indépendante des langages. Dans le second cas un langage intermédiaire comme XML peut être utilisé comme format de données pour les requêtes aux fonctions et méthodes.

Licence d'utilisation
Elle est sous licence libre et utilisable sans frais par tout programmeur, ou sous licence propriétaire et accessible uniquement à une communauté restreinte, ce qui le cas par exemple des API de consoles de jeux.

Niveau de langage

On distingue d'une part l'API de haut niveau, en matière de langage de programmation, comme les API graphiques et d'autre part l'ABI (Application Binary Interface), proche du système, comme la Linux Standard Base ou les interfaces de pilotes de matériels.
Liste d'APIs et bibliothèques d'usage général

Il va de soi que le choix d'une API dépend du langage de programmation (Java a une standard API complète) et de la nature du projet envisagé. Mais certaines API sont communes à tous les projets pour un type d'application donné...
Cette liste contient des bibliothèques ayant une API standard et d'autres qui ne le sont pas. Dans le second cas la bibliothèque doit être incluse avec l'application pour éviter les incompatibilités futures.

OpenGL.
Bibliothèque graphique portable en 2D et 3D.

WebGL.
C'est une interface au code d'OpenGL sur les navigateurs.

SDL.
Bibliothèque graphique pour réaliser des jeux sur ordinateur.

Windows API.
Programmation Windows.

Chromium.
Le code du navigateur Chrome est devenu un substitut au runtime C ou à l'API Windows avec l'avantage sur cette derniere d'être portable. On trouve en effet dans le code l'essentiel des ressources pour réaliser le backend d'une application. Cet article (en anglais) en donne la démonstration : Chromium is the new C runtime. Mais c'est un ensemble très lourd, et il sans doute plus simple d'utiliser Qt ou une liste de bibliothèques spécifiques.

Portable Runtime Project
Comme Chromium mais conçu spécifiquement pour cela.

WebAPI de Mozilla.
Un ensemble d'interfaces pour les mobiles et leurs parties, en cours de développement mais utilisés déjà sur Firefox OS.

Google Map API.
Utilisation de cartes géographiques sur un site web.

Youtube API.
Pour utiliser les vidéos de Youtube.

Les APIs :
Véritables outils d'aide à la décision.

blog.atinternet.com – 19/03/2012 | Rédigé par Benjamin Diolez

La collecte d'informations, leur traitement, leur analyse et leur affichage, le tout depuis une seule interface, c'est génial. La possibilité de réaliser l'ensemble de ces opérations depuis plusieurs interfaces, c'est encore mieux !

Les données deviennent transversales et peuvent ainsi s'adapter en fonction des besoins de chacune des interfaces concernées, tout en restant à jour. Voici ce que j'appellerai l'idée de base d'une **Application Programming Interface** (ou **interface de programmation** en Français).

Plus concrètement, une **API** permet l'interconnexion entre plateformes pour la lecture et/ou l'écriture de données. Selon la plateforme utilisée, les actions possibles peuvent être très variées.

Voyons ensemble quelques exemples d'utilisations nous permettant d'illustrer les possibilités offertes par les APIs.

Exemple de widget réalisé à partir des informations de l'API Yahoo! Weather

Yahoo! Weather

Le moteur de recherche « Yahoo! » propose une API permettant de récupérer les informations météorologiques de n'importe quel lieu au monde en renseignant simplement les coordonnées GPS dans la requête. Les informations récupérées (exemple : températures minimales et maximales) peuvent ensuite être intégrées dans différents modules.

Récupération d'une cartographie via l'API Google Maps

Google Maps

Google propose de nombreuses APIs permettant de répondre aux différents besoins d'une application. L'une des plus utilisées est l'API « Maps ». Cette interface permet une grande variété d'interactions avec des cartographies du globe.

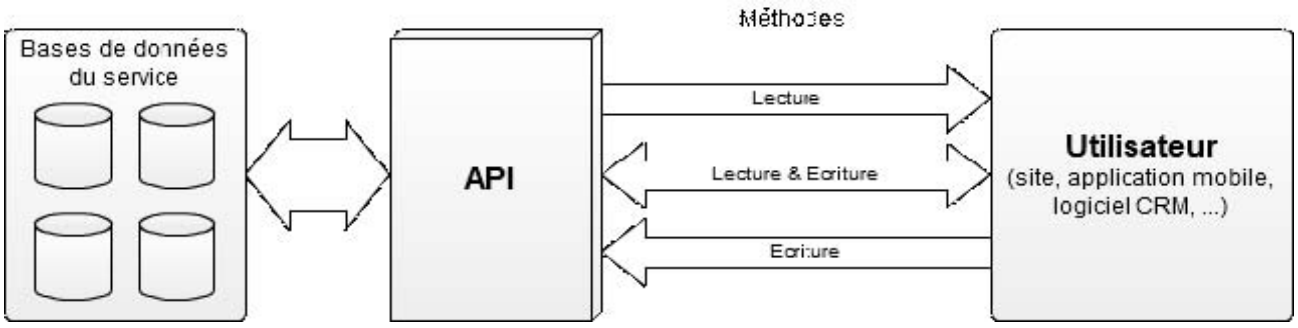
Le système de localisation est similaire à celui utilisé par « Yahoo! Weather » puisqu'il s'appuie sur les coordonnées GPS afin de centrer correctement la carte qui sera récupérée. Le fonctionnement de base consiste à récupérer une image fixe de la carte. Il est ensuite possible d'y ajouter des marqueurs et de la rendre interactive *via* différents paramètres.

Concrètement, comment fonctionnent les APIs ?

Chaque opération, ou interaction, proposée par une API est appelée «méthode». Celles-ci peuvent être de plusieurs natures :

- **Lecture seule**: il s'agit là de la récupération d'informations, afin de les traiter côté utilisateur;
- **Lecture et écriture**: ce type de méthode permet la récupération de données afin de les traiter, ainsi que la modification/ajout/suppression d'informations ;
- **Ecriture seule**: une méthode de ce type permet l'ajout d'informations, sans possibilité d'obtenir celles déjà présentes.

Voici un schéma du fonctionnement d'une API:



Afin de classer ces méthodes de façon logique et d'en simplifier l'utilisation et la gestion, elles sont organisées en groupes de méthodes appelés «services». Un utilisateur peut donc avoir ou non accès à un service, c'est-à-dire à l'ensemble des méthodes qu'il contient.

Comment accède-t-on aux informations fournies par les APIs ?

Il existe plusieurs façons d'accéder à ces informations. Chaque API est libre de proposer la méthode qui lui convient le mieux, en fonction des informations à transmettre.

Voici les principaux formats utilisés:

Requête

- **REST**: Il s'agit d'appeler une URL, avec différents paramètres afin d'affiner la demande. Cette méthode est la plus utilisée, avec près de 75%(*) des APIs utilisant ce mode de requête. Sa mise en place est très simple et permet un développement rapide de l'application.

- **SOAP**: Ce mode de requête demande un format bien précis, et nécessite un envoi plus conséquent que la méthode REST. Ce système tend à disparaître, je ne rentrerai donc pas dans les détails techniques de son utilisation.

Réponse

- **XML**: Le format XML est le plus utilisé pour les réponses d'APIs, puisqu'il permet de véhiculer un grand nombre d'informations, en respectant un format strict. 65% des APIs utilisent ce format de données.

- **JSON**: Le JSON est un format en pleine expansion, qui permet le transfert d'informations dans un format facilement utilisable dans la majorité des langages de programmation, en particulier le Javascript. Cette facilité d'utilisation permet la mise en place rapide d'applications.

Ces quelques formats ne sont qu'une sélection parmi tant d'autres. Il n'existe donc pas encore de standards concernant ces formats de requête et de réponse. Cependant, les formats REST pour les demandes et JSON pour les réponses semblent en bonne voie de le devenir.

(*) Source: ProgrammableWeb: http://www.programmableweb.com/apis

A quoi sert l'API AT Internet ?

AT Internet met à disposition de ses clients une API leur permettant de nombreuses actions, telles que la gestion de leur compte, ou la **création d'analyses**. Celle-ci permet à certains clients de manipuler les données de leur côté, et ainsi générer des analyses pertinentes pour leur métier. Ils peuvent ainsi redistribuer les données de plusieurs façons, soit en interne, afin d'orienter les décisions stratégiques de l'entreprise et des différentes équipes, à leurs partenaires afin de les renseigner sur les performances de leurs campagnes, ou encore au grand public pour miser sur la transparence.

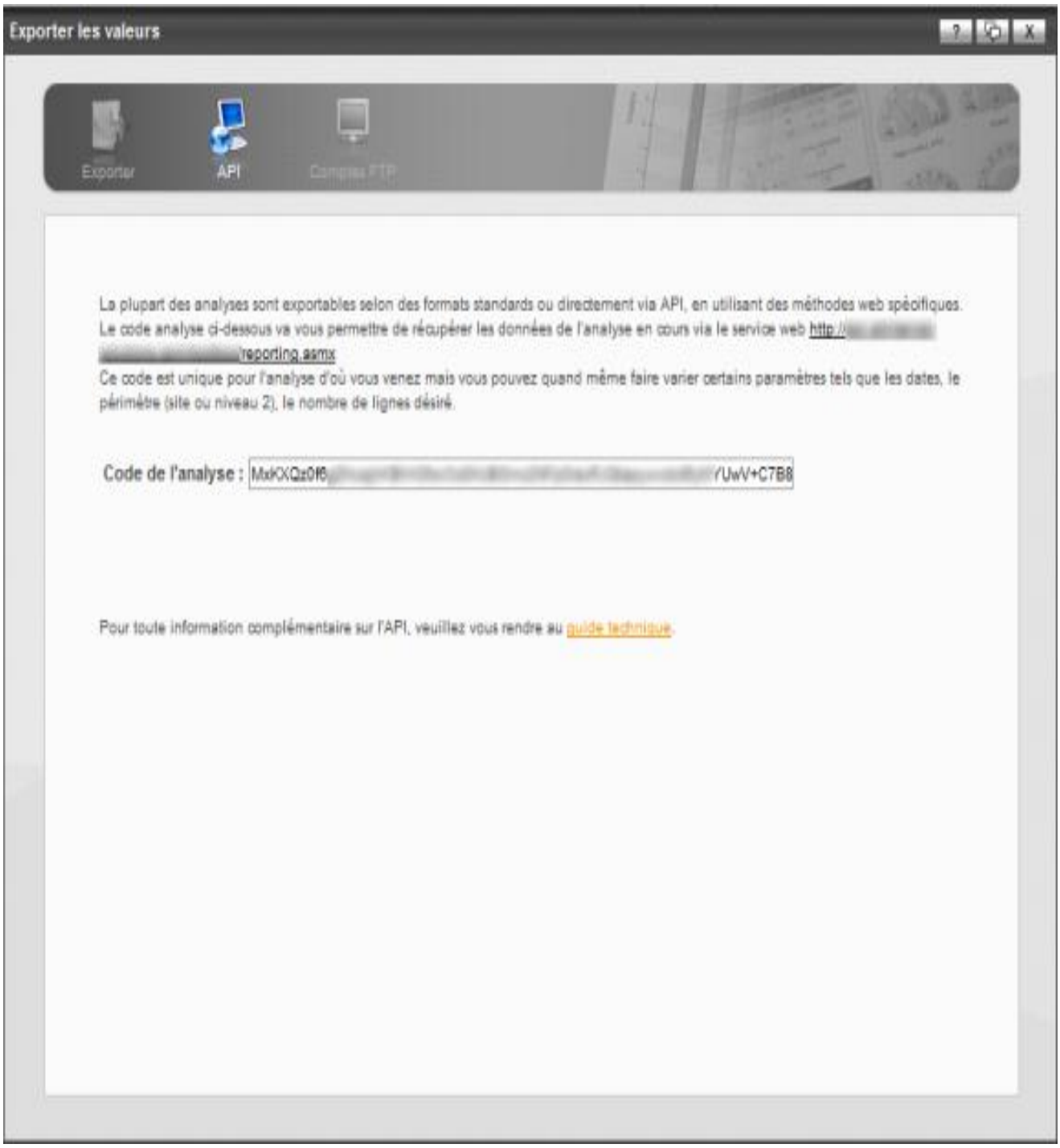
Quels sont les services proposés par l'API AT Internet ?

Afin de proposer le maximum de possibilités aux clients, **AT Internet** propose différents services, permettant chacun des actions particulières, que ce soit en lecture et/ou écriture.

Voici une brève description des deux principaux webservices proposés par AT Internet pour ses clients.

Reporting

Le service **Reporting** est le principal webservice proposé par AT Internet, et il est aussi le plus utilisé. Celui-ci permet la récupération de n'importe quelle analyse présente dans l'interface, grâce à l'utilisation du «code analyse» présent dans chacune des analyses.



Ce service propose deux types de méthodes:

- Méthodes **synchrones**: les données sont retournées directement dans la réponse à l'appel.
- Méthodes **asynchrones**: l'analyse est générée parallèlement aux appels effectués par le client, et chaque appel reçoit un «numéro». Ce numéro permet la récupération des données via un second appel.

Le choix de l'utilisation de l'un ou l'autre type d'appel se fait en fonction de la quantité de données à récupérer. La méthode synchrone sera utilisée pour de petites volumétries, suffisamment faibles pour que le temps de transfert soit transparent. La méthode asynchrone permet de récupérer des jeux de données plus importants et plus longs à calculer sans que l'application qui utilise l'API ne soit bloquée en attendant les données.

CampaignSources

La solution proposée par AT Internet permet de connaître la source d'un visiteur pour tout type de campagne marketing, telles que : liens sponsorisés, affiliation, emailing, RSS, Publicité.

La déclaration et la gestion de ces campagnes se fait généralement via l'interface. Le service **CampaignSources** permet de délocaliser la gestion de l'ensemble des campagnes marketing liées à un site. Il est ainsi possible :

- D'ajouter une campagne ;
- De fermer une campagne ;
- De récupérer les informations d'une campagne (ou de l'ensemble des campagnes);
- De mettre à jour une campagne (en modifier le nom et la description).

Cela permet, par exemple, d'offrir la possibilité à un service Marketing en charge de ces campagnes, et qui n'aurait pas accès à l'interface, de déclarer de façon simple (voire automatique) ces campagnes lors de leur lancement fonctionnel.

Comme nous venons de le voir, les APIs sont des éléments important dans la gestion de l'information, et permettent de constituer de véritables applications, en retraitant uniquement les données fournies par différentes APIs.

Cela permet donc de bénéficier de tous les avantages d'une solution SaaS en les intégrant à part entières dans des outils internes.

Ce type d'interface peut s'avérer décisive dans la gestion stratégique et opérationnelle d'une campagne, d'un site, voire d'une entreprise. Il est donc intéressant de prendre le temps d'étudier les possibilités offertes par différentes APIs relatives au marché dans lequel une entreprise évolue, afin d'obtenir des données qu'il sera possible d'intégrer dans le système d'information interne, et qui aideront à la prise de décision.

Conception d'API : construire correctement une interface de programmation d'applications

www.lemagit.frl - mars 2015 **| Rédigé par** ***Tom Nolle - CIMI Corporation***

Ne faites pas capoter tout votre projet d'API. Bien connaître les applications permet notamment de fixer la conception et d'encourager la fidélité des développeurs.

Dans un contexte de logiciels interactifs et fractionnés en de nombreux composants, rien n'est plus important que les interfaces de programmation d'applications(API) utilisées pour relier ces composants entre eux, ainsi qu'aux appareils mobiles et aux navigateurs . Bien conçue, une API permet de garantir l'intégration fonctionnelle et la fidélité des développeurs ; mal conçue, elle peut faire échouer tout un projet.
Trois facteurs permettent de rester du bon côté de la force en matière de création d'API :

- connaître les applications et leurs contraintes d'utilisation ;
- s'intéresser à l'architecture de composants et à l'infrastructure de liaison ; et enfin
- s'assurer d'une bonne gestion des changements.

Les API présentent les fonctionnalités et les services aux développeurs. Le mode d'utilisation d'une API et la gamme des services représentés constituent les principaux moteurs de son développement. L'une des erreurs les plus importantes commises par les développeurs et les architectes lors de la création d'API consiste à ignorer un élément déterminant. Il est vital que la conception d'une API s'intègre parfaitement dans l'écosystème des développeurs, des langages et des autres API.

Problèmes courants de conception d'API

Le débat REST contre SOAP constitue un exemple de cadre contraignant pour les API. Lorsque des applications dépendent déjà des unes ou des autres, toute nouvelle API doit évidemment se fondre dans le moule. Ce qui est moins évident, c'est que la plupart des API font partie d'une tendance allant vers le fractionnement en composants et la divulgation des fonctionnalités. Ce mouvement peut, au fil du temps, orienter un ensemble d'API vers REST ou SOAP ; une migration qu'il faut donc absolument anticiper.

Les problèmes de constitution des API peuvent détruire une application plus vite et plus radicalement que tout autre type d'erreur d'architecture.

Les architectes risquent d'être facilement échaudés s'ils se conforment à l'architecture objet et à l'infrastructure de liaison. Il est important de choisir le bon modèle d'API car les développeurs ont du mal à utiliser une interface non conforme à l'architecture des applications qu'ils mettent au point. Remarquez que les API RESTful représentent généralement des ressources, tandis que les API SOAP représentent des processus ou procédures distant(e)s.

Un protocole peut servir à lier les API à leurs utilisateurs et aux applications Web ; il s'agit en principe du tandem HTTP (Hypertext Transfer Protocol)/HTTPS. L'utilisation du protocole HTTP avec un format de données HML (Hypertext Markup Language) ou XML (eXtensible Markup Language), ou encore JSON (JavaScript Object Notation) et JavaScript sur les appareils clients, facilite la création d'interfaces utilisateur graphiques à partir d'une API.

En revanche, il peut ne pas convenir lorsque l'accès au navigateur n'est pas l'objectif de l'application. Certaines applications et API peuvent utiliser un port TCP (Transmission Control Protocol) ou UDP (User Datagram Protocol) particulier au lieu du port Web 80. Ce dispositif contribue certes à séparer le trafic des API de l'activité Web, mais les conséquences au niveau du pare-feu ou de la sécurité risquent d'exiger une configuration système spéciale entraînant une exposition des API ou leur utilisation à distance.

Règles générales en matière de conception d'API

On peut considérer que la plupart des API forment une syntaxe de verbes et de noms. Par exemple, on a une phrase avec un verbe représentant une action demandée (get, put, delete) et des noms désignant les arguments qui s'appliquent à l'action. Une bonne pratique consiste à toujours générer une variable d'état/de résultat qui indique les conditions d'erreur ou la réussite de l'exécution. Les conditions d'erreur doivent être suffisamment précises pour décrire les problèmes sans ambigüité.

La sémantique de l'API, c'est-à-dire la syntaxe des fonctions fournies, est importante car la capacité de l'API à transmettre clairement ses services et paramètres évite aux développeurs de commettre des erreurs. Si l'API représente un service avec état, il est essentiel que la sémantique de la fonction soit axée sur la session (find-record, update-record, delete-record) lorsque la nature « avec état » du service est clairement établie.

Il en résulte que si, comme dans cet exemple, les fonctions update et delete portent sur l'élément de données précédemment localisé, elles ne fournissent pas leurs propres clés d'élément de données ; en effet, celles-ci seraient redondantes et risqueraient de dérouter les développeurs. En revanche, un service sans état doit systématiquement indiquer toutes les données car la session ne fournit aucun contexte.

Questions et problèmes récurrents

Les problèmes de syntaxe créés par les mises à jour ou les modifications apportées à l'API sont souvent ignorés. Chaque API présente deux faces, que le processus de changement peut désynchroniser. Pour leur API, certains architectes prévoient une variable de version afin de s'assurer que les mêmes formats soient attendus sur les deux faces. Au minimum, les côtés serveur et client d'une API doivent procéder à une validation élémentaire pour se protéger des modifications susceptibles d'induire un conflit de syntaxe, et ce afin d'éviter qu'elles contaminent les informations ou bloquent les applications.

Une autre question récurrente concerne le format des données. Le langage XML est le plus répandu pour définir les paramètres et échanger des informations ; il s'applique aux interfaces REST et SOAP. Mais le traitement XML est lourd et s'avère surtout utile pour exprimer les données sans structure. Pour l'architecture REST, JSON rencontre un succès grandissant, car il est plus facile à mettre en oeuvre et assure un typage spécifique des variables, largement utilisé et attendu dans la création d'API. Dans les cas où les API échangent des éléments de données strictement définis, il est sans doute préférable d'utiliser JSON pour les échanges RESTful.

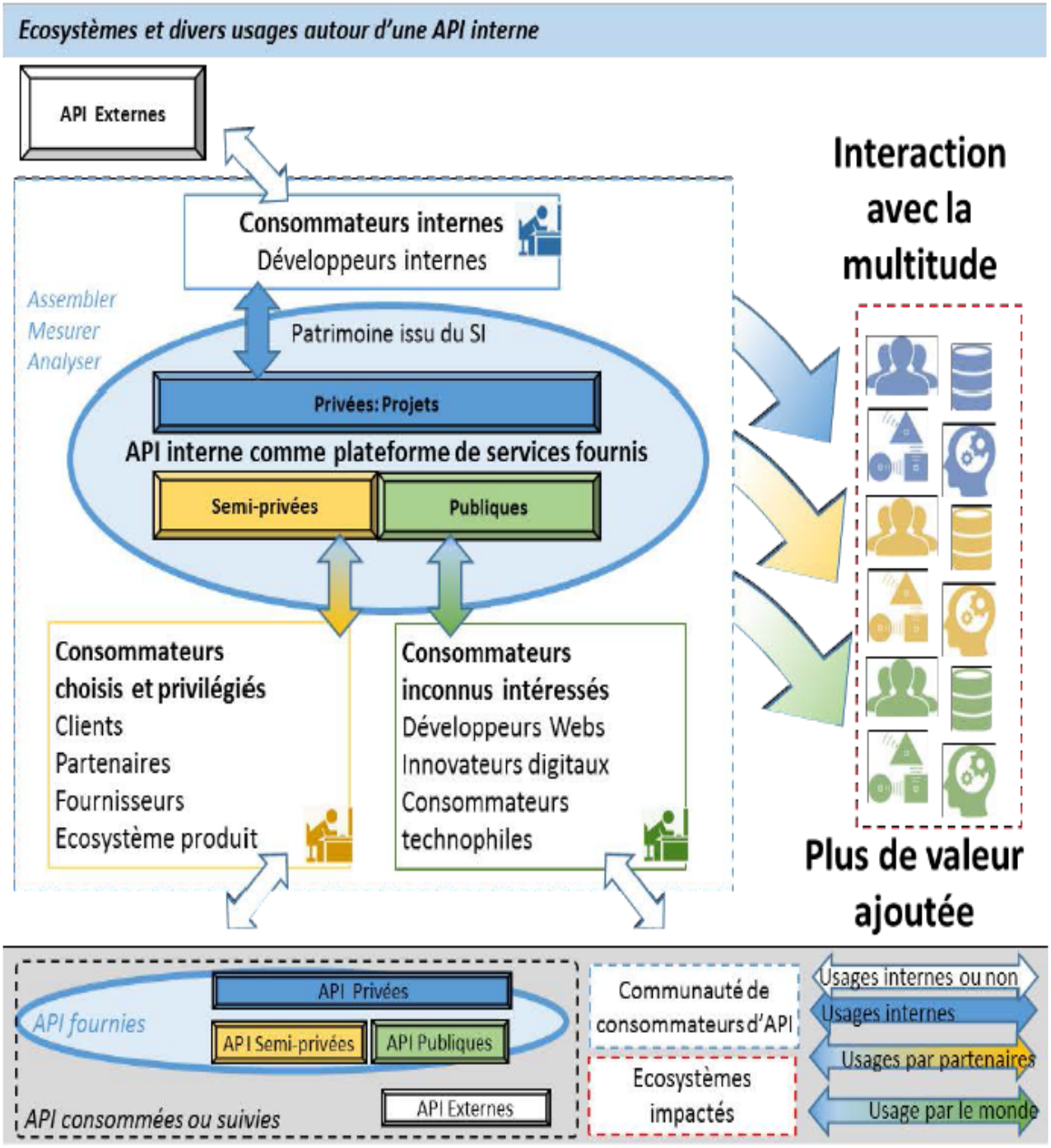
Les tests d'API sont souvent intégrés aux processus de gestion du cycle de vie des applications. C'est justifié pour une partie d'entre eux, mais des tests unitaires spécifiques doivent également être conçus afin de valider les API et de garantir leur exécution correcte même lorsque les données contiennent des erreurs. Moins les liaisons de données et le typage d'une API sont contraignants, plus il est risqué de transmettre des informations qui entraîneront une erreur ou un blocage ultérieur.

C'est pourquoi il est important d'adopter des contraintes strictes sur les variables et de tester chaque API avec un large éventail de données.

Les problèmes de constitution des API peuvent détruire une application plus vite et plus radicalement que tout autre type d'erreur d'architecture. Le temps supplémentaire passé à concevoir des API de façon à prévoir les conditions d'erreur présentes et futures sera du temps bien employé.

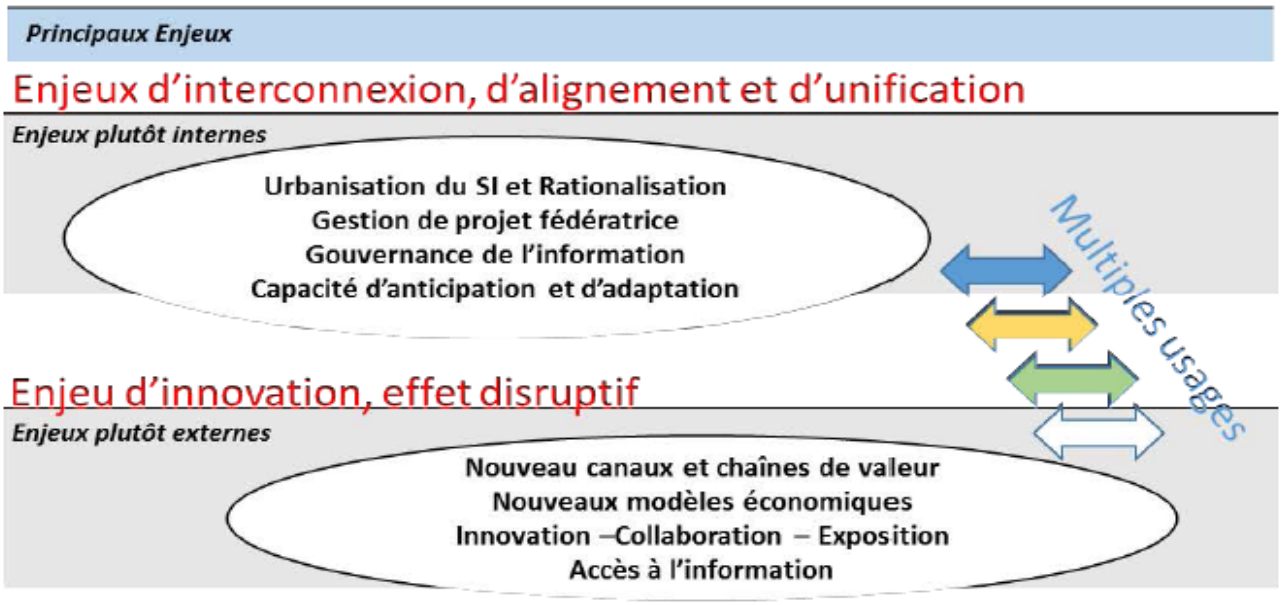
LES NOUVEAUX ENJEUX DES API POUR LES ORGANISATIONS

Cabinet Voirin - juin 2015 | Rédigé par Colin LESPRIT & Sinan SERDAROGLU.



Les bénéfices escomptés pour l'organisation

On sent qu'il y a des possibilités nouvelles et des bouleversements des codes établis. En utilisant les différents types d'API, on peut créer une multitude de nouveaux ponts, entre les départements en interne tout comme avec le monde extérieur. Voici un schéma résumant les enjeux :



En prenant le recul nécessaire pour s'ouvrir et définir les services qu'elle veut proposer, une équipe interne utilise une API comme une vitrine qui valorise son travail au sein de l'organisation et au-delà.
La réutilisation possible du travail de l'équipe créé un climat stimulant et favorise une multitude de partenariat.

Ce faisant ces organisations et équipes autonomes :

- Considèrent ces API flexibles comme un canal majeur de l'organisation, analysant le trafic engendré et pouvant traiter des transactions provenant de sources infinies.
- Conçoivent les divers clients (mobiles, web et autres) comme une surcouche modulable au-dessus de la couche API.
- Permettent aux clients d'interagir avec le coeur des systèmes, créant plus de valeur ajoutée et renforçant la relation client-fournisseur.
- Développent de solides réseaux de partenaires qui permettent de réutiliser et de revendre ses actifs, et d'atteindre des publics.
- Unifient leurs interfaces de telle façon que les équipes internes et les partenaires externes sont libres d'ajouter de nouvelles couches fonctionnelles.

Responsabilités clarifiées et articulations fluidifiées en interne

Urbanisation du SI

La notion de plateforme inhérente aux API permet d'avoir une approche modulaire et de rationaliser les flux entre SI tout en renforçant la cohérence interne.

En ce sens, on peut dire qu'avoir une stratégie orientée API permet de se rapprocher des objectifs du SOA (Service Oriented Architecture) et par conséquent d'urbaniser le Système d'Information :

	Hier Diriger le SI sans API-Web	Demain Diriger le SI avec API-Web
Mobile	Applis mobiles et développements spécifiques à une plate-forme donnée.	Une plateforme back-office API unique, à laquelle se connectent les clients des différentes plateformes du front.

Intégration Clients	Intégration « une-à-une » coûteuse, avec usage de technologies « point-à-point » à l'image du SOA.	Entrepôt de services API, que les clients et partenaires peuvent intégrer de manière autonome.
Canaux de distribution digitaux	Choix limité et coûteux pour impliquer des partenaires.	Entrepôt structuré de contenu et de transactions mis à disposition par l'API, si possible avec des kits de développements de logiciel (SDK) pour accélérer l'intégration du partenaire.
Développement interne	Directement dans la base de données des applications dispersées au travers des départements, générant des interdépendances fragiles et des failles de sécurité. Cela va contre l'agilité et freine les innovations.	Insertion d'une couche API unifiée qui démocratise l'accès aux divers back-office et créée un environnement de développement plus modulaire et flexible.
Accès direct à l'infrastructure	L'infrastructure n'est accessible qu'en interne depuis certaines interfaces restreintes.	Les API sont utilisées pour externaliser et accéder à l'infrastructure à distance.

Remise en question des processus et de la gouvernance des données.

Pour chaque département de l'organisation, les réflexions sur les services à proposer au travers de l'API seront l'occasion de remettre en cause les processus existants : pourquoi a-t-on besoin de cette information à ce moment ? Est-ce la bonne information ? N'y a-t-il pas un moyen de s'en passer ? Une autre solution serait-elle plus sensée ?

- Il est préférable d'avoir des équipes qui soient prêtes à se remettre en question et à accepter le changement. C'est un prérequis pour mettre en oeuvre une stratégie centrée sur les API.

Au-delà de l'analyse des processus, le sujet de la qualité des données utilisées suscitera des débats.
L'objectif sera de pouvoir enfin répondre aux questions ci-dessous:

- Où chercher des données fiables ?
- De quels référentiels disposons-nous ?
- Qui en est responsable ?
- Quelles sont les règles de confidentialité ?

Les premières vertus d'un chantier API sont donc strictement internes, il y a l'opportunité de :

- Définir les responsabilités liées à gouvernance des données numériques ;
- Extraire de l'information de qualité de la myriade de logiciels qui composent leur SI.

Collaborer et innover autrement avec l'externe

De la même manière que les API privées stimuleront l'innovation interne, les API semi-privées et publiques seront les clés pour interagir avec les consommateurs externes.

Les API représentent la capacité à traiter digitalement avec quiconque, de manière programmée : c'est un moyen pour pouvoir interagir avec une multitude d'acteurs potentiels en simultané, ceux-ci

étant autonomes pour se connecter aux services. Il y a un effet de découplage qui convient à la multiplication des acteurs.

De plus, elles sont l'opportunité de se connecter plus en profondeur avec les partenaires, les clients et le monde entier, d'apprendre à mieux les connaitre, cerner leur besoins et décider comment réagir pour les satisfaire au mieux.

L'analyse du trafic généré par les API mises à disposition permettra de suivre de près les réseaux de consommateurs et de rafraîchir les réponses aux questions suivantes :

- Quelles sont nos valeurs ajoutées stratégiques ?
- Quelles interfaces doivent-elles adopter ?
- Comment piloter les transactions jusqu'à elles ?

Ces trois questions illustrent le passage d'une vision processus à une vision plateforme de services : on part de la finalité et du service voulu pour trouver une solution technique qui subviendra au besoin.
Une fois que ces questions auront trouvé réponse, l'organisation peut considérer quelles autres interfaces, applications et canaux doivent être ajoutés à ces interfaces clés.