dessine-moi un bouton pour l'IdO

boutonnière n° 1 : Architecture de l'IdO

Veikko Krypczyk (Allemagne)

Sur l'internet des objets (IdO), c'est le printemps toute l'année, et tout pousse vite. Témoin l'éclosion de ce bouton IdO, programmable à volonté pour les applications à votre guise. Il lui suffit d'une connexion au réseau, le reste se passe dans le(s) nuage(s). La première partie de cet article traite de l'architecture de l'IdO basée sur le *cloud computing* sans serveur et de l'interaction entre matériel et logiciel.

Les idées de nouvelles applications pour l'internet des objets (IdO) ne cessent de jaillir. Beaucoup restent expérimentales et ne prendront probablement pas racine dans le monde réel de l'IdO de si tôt. La vision du réfrigérateur à remplissage automatique est difficilement compatible avec le plaisir que nous éprouvons à choisir nos aliments en fonction du goût du jour et du marché. Le *frigIdO* attendra. À la frontière entre application sérieuse et activité ludique, on trouve les boutons IdO qui pourraient par exemple :

- allumer ou éteindre des appareils,
- commander des fonctions,
- passer des coups de fil,
- compter quelque chose.

Les applications possibles pour ce *bouton-internet* sont multiples, aussi bien dans le secteur privé qu'ailleurs. L'idée est d'utiliser du matériel sous la forme d'un bouton pour envoyer un signal sur la toile. Peu importe, pour commencer, où le signal est envoyé, ni quelles seront les actions associées ni les systèmes connectés, ni encore le nombre de boutons. Abordons ce sujet comme une occasion de découvrir et comprendre l'architecture moderne de l'IdO. Nous commencerons par l'interaction des composants et explorerons les fonctions et les possibilités des services de traitement de données dématérialisés dans le nuage. Sur l'IdO, les expériences il est facile de se livrer à des expérience, ce qui s'applique également à l'utilisation des services de *cloud computing*. Avec un matériel (minimal), quelques logiciels et quelques configurations, nous ferons nos premiers pas dans la deuxième partie de l'article et créerons ainsi les conditions pour tester nos propres idées autour de cette variété particulière d'IdO.

Architecture de l'IdO

L'architecture d'une solution IdO moderne peut être imaginée comme le montre la **figure 1,** indépendamment de l'application spécifique et du service choisi. Les appareils de la partie gauche peuvent être des capteurs ou des actionneurs, ou une combinaisons des deux. Les capteurs mesurent par exemple les conditions environnementales telles que la température, surveillent l'état de systèmes ou envoient un signal lorsque

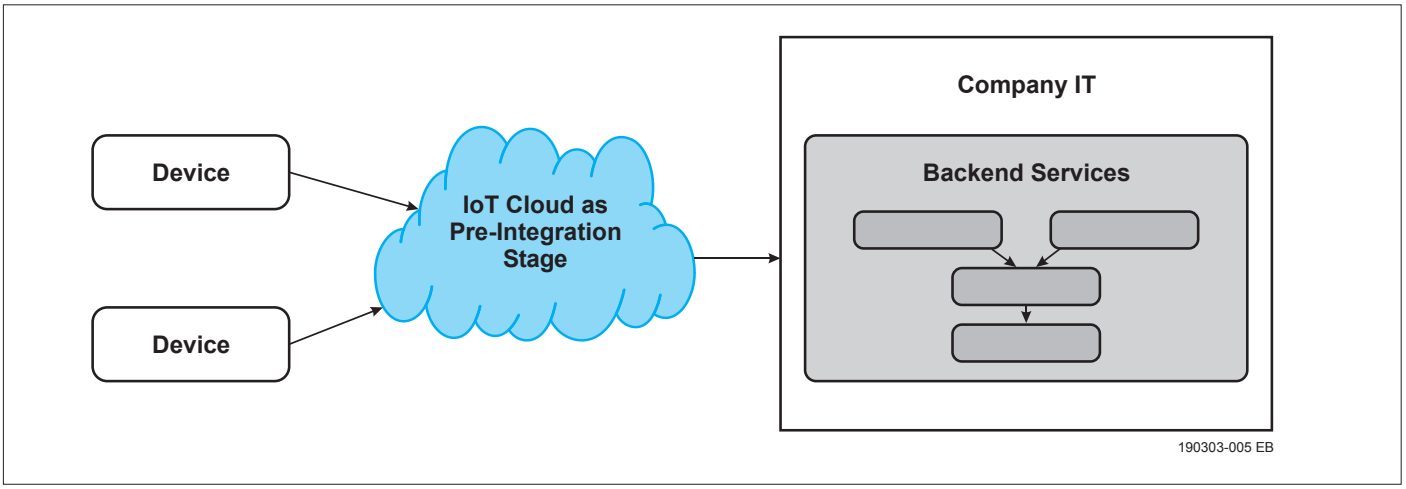


Figure 1 : Architecture de base d'une application IdO avec services d'infrastructure (back-end) dans le nuage.

l'état du système surveillé change, par exemple un chauffage central. Les actionneurs ouvrent ou ferment la vanne d'un radiateur par exemple. Dans certaines applications de l'IdO, on utilise conjointement capteurs et actionneurs pour former une boucle de contrôle action-réaction – gérée par un système d'application à distance. Dans d'autres scénarios de l'IdO, il s'agit seulement d'enregistrer ou de surveiller une condition. Avec notre simple bouton IdO, seul l'actionnement d'un bouton est surveillé.

L'information de changement d'état du bouton est transmise sur un réseau (généralement sur l'internet). Le destinataire des signaux est un service installé sur un serveur. Les services spécifiques adaptés aux exigences des applications de l'IdO sont regroupés dans ce que l'on appelle le nuage de l'IdO ou *IoT Cloud*. Le rôle de ce nuage IdO est central ; il assume le rôle de serveur et est donc le partenaire de communication des appareils IdO. Comme ici seules des fonctions spécifiques sont utilisées et non un serveur complet, on parle de *calcul sans serveur*.
Ces services fonctionnent sur le serveur d'un fournisseur, et vous n'avez donc pas à vous soucier ni de leur configuration, ni de leur administration, ni des accessoires. En cas de sollicitation accrue, p. ex. lors de l'augmentation du nombre des appels au service, des ressources supplémentaires sont fournies automatiquement pour satisfaire la demande (on parle de mise à l'échelle horizontale). En tant qu'utilisateur, vous n'avez pas à vous en soucier. Les fonctions centrales du nuage IdO sont :

- **authentification des périphériques :** permet une variété de connexion et d'authentification des périphériques, à base p. ex. de jetons ou certificats.
- **communication sécurisée :** elle fournit un canal de communication sécurisé par lequel les données peuvent être échangées avec les dispositifs IdO.
- **transmission des données :** Certaines données peuvent être évaluées et traitées directement dans le nuage de l'IdO, d'autres doivent être transmises aux services pour un traitement ultérieur.
  Dans l'illustration, ces services sont conçus comme des services d'infrastructure (*backend*) d'une entreprise IT existante. Dans ce cas, il s'agirait d'une solution industrielle de l'IdO. Les applications plus simples se contentent de transmettre les données à un logiciel, pour évaluation et visualisation.

Boutons et boutonnières

Dans le cas du bouton IdO, il faut décider de la réaction à déclencher quand le bouton est pressé. Ça peut consister à enclencher ou déclencher un autre dispositif IdO, p. ex. un actionneur.
En plus de ces tâches, on attend du nuage IdO qu'il soit capable de communiquer une grande variété d'appareils, et suffisamment flexible pour fournir des interfaces de programmation pour les différents systèmes. Idéalement, ces interfaces sont mises à disposition dans différents langages cibles (par ex. C, Python, Java) via des SDK (bibliothèques) faciles à intégrer. Une communication générique via l'interface REST (voir **l'encadré**) devrait également être possible. Le nuage IdO doit également prendre en charge différents protocoles, p. ex. HTTPS, AMQP ou MQTT pour la communication avec les appareils.

Services dématérialisés de l'IdO

Les grands fournisseurs de traitement dans le nuage (*cloud computing*) proposent également des fonctions adaptées aux besoins spécifiques de l'IdO. Les acteurs principaux du marché des solutions IdO dans le nuage sont *Windows Azure*, *Amazon Web Services*, *Q-loud* et *Oracle Cloud*. Le choix de la bonne plateforme n'est pas facile et dépendra des objectifs [1]. Les critères suivants doivent être pris en compte pour faire ce choix :

- **plate-forme** : quelles sont exactement les fonctions proposées, notamment la gestion des périphériques, l'authentification et l'autorisation, et le transfert de données ?
- **caractéristiques** : dont notamment les aspects de la protection des données, du cryptage et de l'interface utilisateur (tableau de bord) ;
- **connexion** : langages de programmation, SDK et interfaces pris en charge ;
- **autres critères** : tels que la communauté et la documentation, le soutien, les coûts et les modèles de tarification.

La plupart des services dématérialisés de l'IdO peuvent être testés gratuitement jusqu'à un certain niveau. Sur les sites des fournisseurs, il y a généralement des exemples documentés sous forme de code source et pour la configuration (*setup*).

Appareil IdO et connexion

La connexion entre appareils IdO et services dématérialisés se fait par l'internet public. Peu importe dans un premier temps comment ce lien est établi. En plus d'une connexion directe câblée (LAN) d'appareils fixes, la connexion peut également être sans fil via un réseau local (WLAN) ou par radio (LTE) ; tout dépend de l'application, de l'infrastructure existante, du matériel choisi et de la consommation d'énergie. Avec notre bouton IdO, nous supposons qu'il s'agit d'une intégration la plus simple possible dans un réseau local câblé local. Comme le bouton IdO doit être flexible dans son utilisation et que le boîtier doit être compact (p. ex. sous la forme d'un interrupteur pour un luminaire), les spécifications pour le matériel sont claires. Seules entrent en ligne de compte des cartes peu encombrantes, dotées d'une connexion de réseau sans fil commode, et à faible consommation d'énergie, en particulier en mode veille. Nous reviendrons sur ces aspects dans la deuxième partie de l'article, quand nous expérimenterons avec nos premiers prototypes.
Si vous n'avez pas à vous soucier de restrictions de taille, d'alimentation électrique et de connexion sans fil au réseau, le choix de la plate-forme pour l'appareil IdO est beaucoup plus étendu et donc plus facile. Si p. ex. le bouton IdO est sédentaire, il peut être alimenté par la tension du réseau, de sorte que la consommation en veille est moins critique et une connexion LAN par câble est envisageable.

Fournisseur de services dans le nuage de l'internet des objets

Il faut maintenant sélectionner un fournisseur de services IoD dématérialisés. Le service *Azure IoT Hub* [2] est conçu par *Microsoft* pour une communication bidirectionnelle entre les dispositifs IdO et le portail *Azure*. Il s'agit d'une infrastructure hébergée dans le nuage auquel une variété de dispositifs peuvent être liés. Les appareils sont intégrés via des SDK, disponibles pour différentes plateformes et différents langages de programmation. La passerelle IoD offre également des fonctions pour la gestion des appareils connectés. Chaque appareil a sa propre identité avec des données d'accès ou des certificats, chaque appareil peut être activé ou désactivé indépendamment des autres sur le tableau de bord, directement dans le navigateur.

Cette fonction est également offerte par le biais d'API pour permettre aux développeurs de gérer les dispositifs IdO dans leurs propres applications. La documentation en ligne fournit des instructions détaillées, étape par étape : Dans [3], par exemple, on verra comment connecter un Raspberry Pi à la passerelle IdO et recevoir des données du périphérique. L'utilisation de la passerelle IdO *Azure* est gratuite pour le prototypage ainsi que pour un usage commercial ou privé tant que l'on accepte une limitation du nombre d'appels. Le tarif *Standard n'*entraîne

Oh, aurions-nous oublié un événement ?

Vous organisez une conférence, un salon… ou bien vous participez à un séminaire ou tout autre événement qui aurait sa place ici, partagez cette information avec tous les lecteurs.

Envoyez-nous tous les détails à **redaction@elektor.fr**.

mars 2020

◇ **JEC World** (Salon intern. des composites)
du 03 au 05 mars à Paris Villepinte
www.jec-world.events/fr/

◇ **Salon du livre**
du 05 au 08 mars à Bruxelles
https://flb.be

◇ **EXPO ELETTRONICA**
le 07 mars 2020 à Faenza (Italie)
www.expoelettronica.it/

◇ **DATE** Salon de la conception et du test des circuits électroniques
du 9 au 13 mars à Grenoble
www.date-conference.com/

◇ **Salon du livre**
du 10 au 12 mars à Londres
https://www.londonbookfair.co.uk/

◇ **EMV** – Salon et congrès sur la compatibilité électromagnétique
du 17 au 19 mars à Cologne (Allemagne)
www.mesago.de/de/EMV/home.htm

◇ **eLearning expo | I-EXPO**
Salon des professionnels de l'information numérique
du 17 au 19 mars à Paris - Porte de Versailles
www.i-expo.net/
www.e-learning-expo.com

◇ **IT MEETINGS**
Salon B2B des réseaux, des télécoms, de la mobilité, du Cloud Computing, des datacenters et de la sécurité
du 17 au 19 mars à Cannes
www.it-meetings.fr/

◇ **Rencontres de l'électronique imprimée**
du 17 au 18 mars à Paris
www.rencontreselectroniqueimprimee.com

◇ **Salon des Solutions MtoM et des objets communicants**
Salon des systèmes embarqués & logiciels temps-réel
du 18 au 19 mars 2020 à Paris – Paris - Porte de Versailles
http://www.embedded-mtom.com/

◇ **Salon du livre de Paris**
du 20 au 23 mars 2020 à Paris
https://www.livreparis.com/

◇ **Semaine de la presse et des médias dans l'école**
du 23 au 28 mars
https://www.clemi.fr/fr/semaine-presse-medias.html

… suite à la page 65

aucun frais tant que vous n'échangez pas plus de 8 000 messages par jour avec l'appareil et que la taille d'un message ne dépasse pas 0,5 Ko.

L'offre de *Google* avec le service *Cloud IoT Core* va dans le même sens [4]. Il s'agit d'un service entièrement géré qui vous permet de gérer des appareils et d'échanger des données sur l'internet. *Cloud IoT Core* peut interagir avec d'autres services de *Google* pour le traitement dans le nuage, tels que *Google Big Data Analytics* et les services ML comme *Dataflow, BigQuery, BigTable, ML, Data Studio* ou les outils de BI des partenaires. Cela permet une évaluation, un traitement et une visualisation efficaces des données de l'IdO en temps réel. Le service supporte les protocoles courants MQTT et HTTP, ce qui assure un haut niveau de compatibilité. L'une des fonctions essentielles de *Cloud IoT* est le gestionnaire de périphériques intégré. Les appareils gérés peuvent être paramétrés par l'intermédiaire d'un terminal, ou bien commandés par du code. Le Gestionnaire de périphériques détermine l'identité d'un périphérique et assure une authentification unique. Pour la transmission sécurisée des données, nous nous appuyons sur un cryptage dit de bout en bout. Sur le plan de la technique, le *Cloud IoT Core* est aussi un service sans serveur qui évolue horizontalement, sans latence notable, au fur et à mesure que les besoins augmentent en quantité.

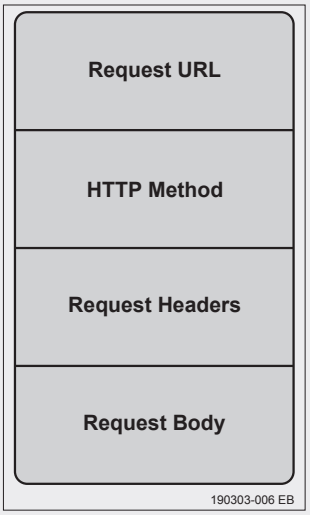
Techniquement, une interface REST est utilisée, ce qui permet d'obtenir un haut degré d'indépendance par rapport aux systèmes des appareils. Le coût du service est basé sur la consommation et est échelonné. Jusqu'à un volume de données mensuel de 250 Mo, le *Cloud IoT Core* de *Google* est gratuit. Ce service convient donc bien pour des expériences, le prototypage, les projets privés ou de petits projets commerciaux. Des offres similaires – mais presque exclusivement destinées aux clients industriels – sont proposées par *Oracle Cloud Services for IoT* [5], *Amazon AWS IoT Services* [6] et la plate-forme *Q-loud IoT* [7].

N'oublions pas les plateformes de *cloud computing* à code source ouvert proposées pour les besoins de l'IdO par *ThingsBoard* [8]. L'utilisation est gratuite si c'est vous qui installez le logiciel sur votre ordinateur. D'une certaine manière, vous créez votre propre infrastructure IdO, qui s'appelle *On Premise* : c'est

Et c'est quoi le REST ?

Quand client et serveur sont connectés via le réseau internet, l'API qui s'en charge est généralement basée sur REST [9][10]. L'acronyme vient de *REpresentational State Transfer*. Les API RESTful sont basées sur des procédures standardisées telles que HTTP/S, URI, JSON ou XML. Les principes suivants s'appliquent :

- **modèle client-serveur** : communication est basée sur un modèle client-serveur. L'objectif est une utilisation flexible et générique des services au-delà des frontières de la plate-forme.
- **apatridie :** communication toujours apatride. Chaque requête du client vers le serveur doit être complète. Le serveur ne peut accéder aux données des requêtes précédentes.
- **mise en cache :** les clients peuvent stocker les réponses du serveur (possibilité de mise en cache). Cela permet de surmonter les pannes de courant ou d'activer temporairement un fonctionnement hors ligne. Lorsqu'une nouvelle requête intervient, les données mises en cache peuvent servir, sans attendre de nouvelle réponse du serveur.
- **uniformité :** Les services utilisent une interface uniforme, découplée du service fourni.



Structure d'une requête basée sur REST.

Une API REST est généralement mise en œuvre via http ou https. Les services sont utilisés via l'URL et les méthodes http telles que GET, POST, PUT, où le client envoie une *requête au* serveur et reçoit une *réponse.* Une requête comporte quatre volets : *endpoint*, *method*, *header* et *data*. Ici *endpoint* se compose de l'*endpoint* racine, du chemin et des paramètres éventuels. Par exemple, *https://api.github.com* est l'*endpoint* racine *de* GitHub et */users/veikkoef/repos* est le *chemin* vers les dépôts de l'auteur. Ce qui donne *https://api.github.com/users/veikkoef/repos*. Les paramètres peuvent suivre, p. ex. *?query1=valeur1&query2=valeur2.* Pour les méthodes, on a le choix entre *GET, POST*, *PUT/PATCH* et *DELETE, c'est-à-dire AFFECTER*, *EFFECTUER*, *RECHANGER* ou SUPPRIMER.

GET porte sur des opérations de lecture de données. Comme les *requêtes GET* n'ont qu'un accès en lecture au serveur, elles peuvent être envoyées aussi souvent que nécessaire. *GET* est la méthode par défaut. *POST* est utilisé pour créer des opérations, c'est-à-dire pour l'enregistrement de données. *POST* n'est pas exempt d'effets secondaires. Un *appel POST* peut être utilisé pour modifier des champs dans une base de données ou pour lancer des processus sur le serveur. *PUT/PATCH* est utilisé pour les opérations de mise à jour et *DELETE* pour la suppression de données.
Le troisième élément d'une requête est l'*en-tête*. L'en-tête est utilisée pour fournir des informations au client et au serveur. Elle peut être utilisée à de nombreuses fins, par exemple pour authentifier et fournir des informations sur le contenu ultérieur.

Enfin, et comme dernier élément d'une requête, suit le corps des données (*Body*) que vous voulez transmettre au serveur. Ce corps n'est utilisé que pour les opérations *POST*, *PUT*, *PATCH* ou *DELETE*.

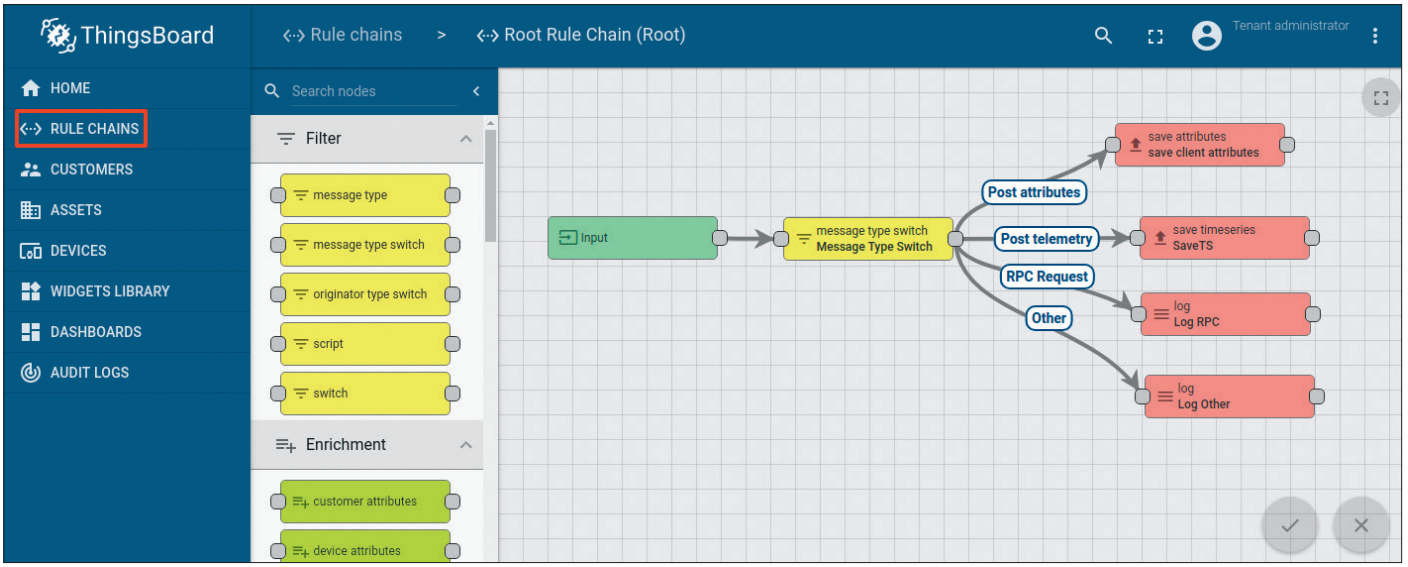


Figure 2 : Le traitement des événements peut être défini graphiquement (source : ThingsBoard).

vous qui êtes responsable de l'installation, de la maintenance et de l'exploitation. On appréciera la variété des systèmes supportés ; tout fonctionne sur différentes distributions Linux (*Ubuntu*, *CentOS*, *Red Hat*), *Windows* ou sur un Raspberry Pi 3. Ce dernier pourrait être intéressant pour les projets communautaires (Maker) et pour le prototypage.

Le logiciel de *ThingsBoard* est également proposé en tant que solution de nuage géré, avec des coûts échelonnés en fonction du volume d'utilisation, à partir de 10 $/mois. Les fonctions de ce serveur ou de ce service de traitement dans le nuage comprennent les tâches typiques d'une infrastructure (*back-end*) de l'IdO, c'est-à-dire la gestion des appareils, la collecte et la visualisation des données, le traitement des événements et les appels de procédures à distance pour commander les appareils. Les fonctions sont également proposées via une API REST et peuvent donc être utilisées indépendamment du système et de l'appareil. Grâce au moteur de règles, il est possible de définir de façon pratique, au niveau graphique, la façon dont seront traités les événements qui se produisent (**fig. 2**).

Conclusion et perspectives

Vous devriez avoir à présent une vue d'ensemble de l'architecture d'une solution IdO, dont la composante centrale est un service dématérialisé (dans le nuage) pour la gestion des appareils IdO et pour la réception et l'envoi des données. Bientôt nous mettrons ces connaissances en pratique. Il s'agira de concevoir un prototype de dispositif IdO sous la forme d'un bouton IdO. De telles solutions sont disponibles toutes faites, mais cela vaut la peine d'expérimenter ; c'est le meilleur moyen de comprendre les connexions, ce qui vous facilitera la conception de vos propres solutions.

(190303-02 VF)

Liens

[1] Comparaison de plates-formes : www.informatik-aktuell.de/betrieb/virtualisierung/iot-in-der-cloud-erkenntnisse-und-erfahrungen-eines-plattformvergleichs.html
[2] Microsoft Azure : https://azure.microsoft.com/de-de/services/iot-hub/
[3] Raspberry Pi sur Azure : https://docs.microsoft.com/de-de/azure/iot-hub/iot-hub-raspberry-pi-kit-c-get-started
[4] Google Cloud : https://cloud.google.com/iot-core/
[5] Oracle IoT Cloud : www.oracle.com/internet-of-things/
[6] Amazon IoT Cloud : https://aws.amazon.com/de/iot/
[7] Q-loud-Cloud : www.q-loud.de/
[8] Thingsboard : https://thingsboard.io/
[9] REST : www.codecademy.com/articles/what-is-rest
[10] REST API : https://restfulapi.net/