**Explication présentation mémoire**

**Introduction** :

Lorsque nous parlons du monde du développement des logiciels, des sites web, des APIs, une spécification est un ensemble d’exigence qu’un système, logiciel ou API doit satisfaire.

Elle décrit les fonctionnalités et les attentes généralement exprimées par les utilisateurs.

Pour les APIs ou Application Programming Interface qui permettent à deux logiciels indépendants de communiquer l’un avec l’autre. Pour les utilisateurs, l’interface de programmation d’application rend les données ou les fonctionnalités d’une application disponible. Par exemple le propriétaire d’une base de données peut décider de partager les informations de sa base de données à travers une API. Une spécification est une interface permettant aux utilisateurs ou développeurs d’applications de comprendre les fonctionnalités de l’API. Ce qui leurs permet d’intégrer des services à distance offerts par l’API dans leurs applications ou sites web.

Actuellement la construction des logiciels ne nécessite plus une équipe d’ingénieurs ou de coûteux serveurs. Une clé d’API et sa documentation sont suffisantes pour intégrer aisément une fonctionnalité externe dans notre système.

Cependant, il existe une spécification pour les APIs d’un modèle particulier qui peut être différent du modèle publication/abonnement.

**C’est quoi le modèle publication/abonnement** :

Le modèle publication/abonnement ou (publish/subscribe ou pub/sub en anglais) est un modèle qui permet à une en entité de publier ses informations et à une autre de s’aboner et consommer ces informations publier. Quand un producteur publie un message, les consommateurs abonnés sont notifiés.

L’unité de communication ou l’information envoyé et reçu est appelé message.

Ce modèle est constitué d’un ou plusieurs producteurs, d’un intermédiaire appelé broker et d’un ou plusieurs consommateurs.

* Les producteurs : produisent les messages et les publient
* Le Broker (courtier en français) : permet la livraison des messages jusqu’aux consommateurs
* Les consommateurs : s’abonnent aux messages sur une ou plusieurs catégories ou topic et les consomment.

**Le fonctionnement** :

Après les producteurs aient produits les messages, ils les publient c’est-à-dire ils les envoient au broker (l’élément d’intermédiaire) car sa fonction est de stocké et transférer. Le broker est constitué des topics et des files d’attentes. Les consommateurs s’abonnent aux files d’attentes et les producteurs publient les messages sur les topics. La manières d’envoie des messages aux files d’attentes est appelé Exchange. Il existe quatre types d’exchanges :

Et les messages peuvent être envoyés à une ou toutes les files d’attentes et tout dépends du type d’Exchange qui peut être direct (un seul), topic (n consommateur), header (Header au lieu de routing key) et fan out (broadcast).

La publication/abonnement est un modèle connu et parmi ces implémentations nous pouvons cités quelques protocoles populaires qui sont :

**AMQP** (Advanced Message Queuing Protocol) est un protocole open source, interopérable (ce qui permet aux programmes de différentes organisations de se comprendre sans difficulté.), binaire 🡺 il est en contact direct avec les différents programmes. Il a été confié à un groupe de travail OASIS (une organisation à but non lucratif). Ce protocole garantit d’une part une transmission des données fiable (à l’aide d’un message broker). D’autre part, l’AMQP permet de stocker des messages dans des files d’attente, permettant ainsi une communication asynchrone.

**MQTT** (Message Queuing Telemetry Transport) : c’est un protocole open source, léger car les messages ont tous une faible empreinte logicielle et assure une communication non permanente. Il peut être implémenté en plusieurs langages. MQTT est un protocole standardisé reposant sur TCP/IP. Il est développé par l’OASIS pour IoT. Microsoft utilise MQTT pour remonter les données sur leur plateforme cloud.

**Apache Kafka** : est un protocole développé par LinkedIn qui devenu maintenant un projet open source de Apache. Il est rapide et base sur la communication publication/abonnement. Kafka est utilisé pour la réalisation streaming temps-réel généralement dans les applications streaming. Il est aussi utilisé comme broker médiateur de communication entre deux applications. Il utilise les queue pour la distribution des messages a plusieurs consommateur ce qui permet une haute evolutivité.

Les avantage de Pub/Sub sont qu’il :

Découplé

Pas de collision : utilisation de la fille d’attente : en séparant différents composants avec les files d'attente de messages, nous créons davantage de flexibilité. Si une partie du système est inaccessible, l'autre peut tout de même continuer à interagir avec la file d'attente. Elles suppriment les dépendances entre les composants. plusieurs instances de votre application peuvent ajouter des requêtes à la file d'attente sans risque de collision.

Anonymats des entités

Adapté aux IoTs :

Pub/Sub est un modèle de communication efficace qui n’a encore de spécification pour les APIs qui l’implémente car

**La spécification existence pour les API : OpenAPI**

OpenAPI est spécification open source, qui décrit une interface standard pour les APIs REST, compréhensible par les machines ainsi que par humains, indépendante des langages de programmations (quel qu’en soit le langage du développeur il faire appel aux services de l’API). Cette spécification permet de découvrir, de consulter et de comprendre les capacités du service web sans connaissance de l'implémentation du serveur ou sans avoir accès au code source.

La spécification OpenAPI peut être décrite en deux langage dont le premier est :

JSON (JavaScript Object Notation) : C’est un langage léger qui est devenu le format principal d’échange d’information. Il permet de représenter les données avec une structure arborescence dont l’affichage respect la règle clé-valeur

YAML (Yet Another Markup Language) YAML est un langage de sérialisation de données conçu pour être fonctionnel et lisible pour les humains [8]. Ce n’est pas un langage de balisage, il est utilisé dans les applications où les données sont stockées ou transmises.

OpenAPI est constituée des objets JSON qui sont :

Openapi : version de la spécification OpenAPI utilisé

Into : décrit l’entête de l’api

Serveur : description et url du serveur qui contient les services auxquels l’API est supposée partager

Paths : les différentes méthodes ou opérations d’accès aux services par le protocole http. Il décrit les manières d’accès aux services autorisées par l’API

C’est la sécurité de l’API décrit par l’objet security. Ces manières d’accès sont :

Query : pour la sécurité des requêtes ou sécurité API\_KEY

Header : dans l’entête des paquets

Components : il décrit les parties réutilisables de l’API pour éviter la duplication. Une fois ces parties définit dans component on la fait référence avec l’objet $ref fourni par OpenAPI.

Security : il décrit la sécurité utilisée par l’API pour avoir accès aux donnés. OpenAPI utilise quatre types de sécurité qui sont : API\_key, Aouth2 :

Tags : c’est array tableau de description d’un ensemble d’opération

ExternalsDocs : pour faire référence a un document externe qui peut guider l’utilisateurs de l’API.

**La Problématique** : le problème avec OpenAPI est qu’elle faite pour uniformiser les APIs REST qui sont limités

Un seul protocole qui est le protocole http alors que pub/sub est multi protocolaire.

Une communication synchrone alors que pub/sub est asynchrone et est sans état ce qui veut dire qu’après l’envoie d’une première requête et la réception de la réponse, l’API ne se souviendra plus de rien.

Et utilisation des URI (uniform ressource identifier) alors que pub/sub utilise des topics et files d’attentes.

**La solution proposée** : la spécification OpenAPI-PS

Elle a des similarités avec OpenAPI, elle décrit une interface standard pour les APIs Pub/Sub, compréhensible par les machines ainsi que par consommateurs, indépendante des langages de programmations (quel qu’en soit le langage du développeur il faire appel aux services de l’API). Cette spécification permet de découvrir, de consulter et de comprendre les capacités du service web sans connaissance de l'implémentation du serveur ou sans avoir accès au code source.

font partie des composants des différents programmes. Utile pour permettre à un produit ou un service d’être en interaction avec d’autres éléments, l’API sert à simplifier le processus de développement d’une application.

. D’une manière plus concrète, une Application Programming Interface joue le rôle d’intermédiaire entre deux systèmes informatiques indépendants pour pouvoir interagir de façon automatique, en minimisant ou sans demander l’intervention d’un utilisateur.

Désormais, la construction de logiciels ne nécessite plus une équipe d’ingénieurs ou de coûteux serveurs. Une clé API et sa documentation sont suffisantes pour intégrer aisément une fonctionnalité externe.

Les Avantages de OpenAPI-PS sont :

Elle a des points communs avec OpenAPI (car OpenAPI est une spécification universelle pour un type d’API qui est sous gouvernance de la fondation linux et est soutenu les grandes entreprises tels que Microsoft, IBM etc.

Elle supporte JSON ainsi que YAML pour une compréhension facile car Sur le plan technique, YAML et JSON présentent très peu de différences, ce qui permet d’automatiser la conversion d’une définition API existante d’un langage à l’autre

Elle est multi protocole c’est quel qu’en soit le protocole il suffit qu’il soit du modèle pub/sub.

Comme OpenAPI, OpenAPI-PS est aussi constituée des objet JSON qui sont :

Openapips

Info

Broker

Protocol

Security

Topics

ExternalsDocs

Tags

Components

**Le générateur de code OpenAPI-PS**

Le générateur de code OpenAPI-PS est un générateur de code basé sur celui de OpenAPI. Il est constitué des fichiers Codegen, des fichiers de configuration OpenAPI et des fichiers templates.

Il prend comme entrée un fichier spécification OpenAPI et ce fichier est passé au Codegen OpenAPI qui la de la structure. Après la validation ce fichier est passé au Codegen d’un langage particulier, le langage dont nous voulons implémentés l’API. Pour générer ce code d’implémentation, ce Codegen a besoin d’un modèle qui décrit les codes d’implémentation ou la manière de communication. Après nous avons un code de sortie qui représente le code d’implémentation de l’API décrit dans la spécification.

Désavantage : broker duré d’essais de délivrer le message

Dans une usine une peut détecter les pannes et les publier et un abonné (afficheur) peut récupérer et l’afficher et c’est un problème au ou l’afficheur tombera en panne.