**Introduction Générale**

Sur le web d’aujourd’hui l’architecture avec API est une l’une des architectures le plus réputée suite aux outils de description d’API comme Swagger, Postman (une plate-forme pour la création, le test et l'utilisation d'API), etc. Alors qu’il y a 20 Ans de cela que Swagger n’existait pas, chaque entreprise avait leur propre compréhension, description et documentation d’API. Au fil des années certains experts utilisateurs et développeurs ont exigé la normalisation des API pour simplifier la compréhension et l’implémentation. Car le défi était de mettre en place une manière d’uniformisation pour une consommation simple. De ce fait, les APIs synchrones basée sur le protocole http ont été standardisés et outillés avec l’écosystème Swagger et *OpenAPI* que nous utilisons dans nos micro services et applications pour générer une interface utilisateur, pour la documentation et la collaboration [1]. Swagger étant un ensemble d’outils permettant de travailler avec *OpenAPI*. *OpenAPI* est la spécification open source indépendante du langage de programmation, lisible par les humains ainsi que les machine. Cette spécification sert d’uniformiser les API http tout en offrant une implémentation simplifiée pour les développeurs. Quant aux APIs asynchrones, ils ont remarqué une variété de protocole (AMQP, MQTT, Kafka etc.) différent qui complexifie une unification globale. Ce qui a conduit à voir différentes implémentations architecturales. Malgré cette différence d’architecture des APIs asynchrone aux synchrones, elles ont des objectifs en commun tels que : l’indépendance en langage de programmation, simple à implémenté etc. Vu que les APIs asynchrones sont en traines de gagner en exposition au sein de la communauté API suite à leur partenariat avec Postman. Il est donc crucial d’établir un standard de spécification compatible aux différents protocoles pour décrire uniformément ces interfaces asynchrones. Pour mettre en place une spécification efficace asynchrone, nous devons s’intéresser à un modèle asynchrone. Le modèle publication/abonnement (publish/subscribe ou Pub/Sub) est un modèle asynchrones efficace. Depuis qu’il est connu pour la diffusion d'information, la contribution dans la recherche et dans la commercialisation des produits, il est utilisé dans plusieurs contextes d'applications comme IoT (internet of Things). Les systèmes de communication dotés de ce paradigme (pub/sub) ont connu un gain de popularité significatif et ont montré son efficacité dans un grand nombre de domaine. La conception de base de pub/sub dérive de la manière dont les notifications circulent des expéditeurs aux destinataires et prend son efficacité de la manière dont l'information est communiquée. C'est-à-dire que, les destinateurs expriment leur intérêt en émettant des abonnements pour des notifications spécifiques, indépendamment des éditeurs (Publisher) qui les produisent, puis ils sont notifiés de manière asynchrone pour toutes les notifications concernant son abonnement [2]. Pub/Sub est modèle dont nous nous sommes intéressés pour établir une spécification pour les API asynchrones.

Toutes les APIs suivant cette architecture doivent offrir aux moins une fonction de publication (publish) et une fonction d’abonnement (subscribe). Donc notre but n’est pas seulement de mettre au point une spécification indépendante des langages de programmation ce qui est commun avec *OpenAPI*, mais aussi utilisable avec les différents protocoles asynchrones quel qu’il soit. Ce qui veut dire, qu’elle prend en charge le concept distribution des messages et aussi les langages de description API qui sont JSON et YAML. Mais supporter les variétés des protocoles (ce qui est généralement rencontré dans les applications IoT et similaire) reste important. Cette nouvelle spécification s’appelle *OpenPS*, elle est open source n’a pas d’outil comme Swagger pour documenter et générer les code développeurs. Elle permettra une uniformisation des APIs asynchrones au lieu que chacun se bat de sont coté pour créer une API IoT.

Pour une meilleure compréhension nous avons divisés ce travail en quatre (4) chapitre comme suit :

Dans le chapitre 1 de ce document, nous décrirons la spécification OpenAPI et l’outil de description, de documentation et génération de code de cette spécification qui est Swagger.

Dans le chapitre 2, nous présentons le modèle publication/abonnement et certain protocole de messagerie asynchrone de ce modèle.

Dans le chapitre 3, nous détaillerons une spécification (*OpenPS*) utilisant l’architecture de publication/abonnement pour la description et unification des APIs asynchrones.

Dans le chapitre 4, nous décrirons une spécification d’API asynchrone avec la spécification *OpenAPI* et générerons le code d’implémentation avec le générateur automatique locale de *OpenAPI* personnalisé appelé *OpenAPI-PS*.