**Conclusion générale** :

Au cours de ce travail, nous avons mis en évidence la compatibilité de Pub/Sub (détaillé dans le chapitre 2) avec les files d’attentes qui peuvent avoir un grand impact sur la communication distribué. Et nous avons aussi comparés une spécification d’architecture client/serveur (*OpenAPI*) à une spécification d’architecture sans serveur (*OpenPS*) utilisant le broker. La spécification *OpenAPI* étant considéré comme légère pour le développement d’API client/serveur, elle n’exige pas la réécriture d’une API existante, elle n’exige pas non plus la liaison d’un logiciel à un service. Car le service décrit peut ne pas appartenir au descripteur et tous les services ne peuvent pas être décrits par *OpenAPI* [1]. Cette spécification n'est pas destinée à couvrir tous les styles d'API web vu l’existence d’autre style de communication comme Pub/Sub et d’autre spécification pour les API [2]. Cependant, la spécification *OpenPS* peut aussi décrire les capacités des services web pour les protocoles de messagerie asynchrone. *OpenPS* utilise un broker gestionnaire des files d’attentes. Ces filles d’attentes permettent une communication asynchrone, ce qui signifie que les points de terminaison qui produisent et consomment les messages interagissent avec la file d'attente et non les uns avec les autres. En séparant les deux entités avec les files d’attentes on aura davantage de flexibilité et une réduction de collision [3].

Bien que *OpenAPI* et son générateur de code soient un outil presque complet et bien qu’il soit le meilleur choix pour les fournisseurs d’API, ses exigences de sécurité comme *api\_key* dans les requêtes et les modèles de chemin URL (Uniform Ressource Locator) ne sont pas compatible avec le Cloud Endpoints. Pour les Cloud Endpoints la précision de port dans la documentation *OpenAPI* ou *OpenPS* et l’écriture de « *parameter* » de type tableau n’est pas non plus accepté [4]. Ces deux spécifications n’acceptent pas des référencements récursifs par exemple un *schema* qui fait référence à lui-même. En considérant que le générateur *OpenAPI* n’est pas tout à fait sécuriser, il ne doit pas être un espoir de générateur de code pour *OpenPS*. Car lors de la génération du code, les fichiers créés (par la fonction File.createTempFile("codegen-", "-tmp")) ne sont pas contrôlés ce qui veut dire que des ajouts ou des suppressions peuvent être faites par un instru [5]. Donc avec le manque d’outil de génération de code pour *OpenPS*, cette spécification n’est pas encore au point mais la description des APIs Pub/Sub avec celle de OpenAPI ne sera pas aussi complet qu’avec *OpenPS*.

[1] <https://nordicapis.com/what-should-you-consider-before-openapi-adoption/>

[2] <https://stackshare.io/openapi-specification#integrations>

[3] <https://aws.amazon.com/fr/message-queue/benefits/>

[4] <https://cloud.google.com/endpoints/docs/openapi/openapi-limitations>

[5] <https://github.com/OpenAPITools/openapi-generator/security/advisories/GHSA-23x4-m842-fmwf>