2. Réponse aux besoins

2.1 Fonctionnalités

La génération de paquets dans le cadre du projet Paquito se fait en plusieurs étapes, en voici donc le principe de fonctionnement :

- Un développeur effectue un commit dans le **dépôt GitHub**. Il a pu par exemple modifier le code source de son projet. L'interface GitHub prend en compte les modifications qui ont eu lieu.

  Remarque : Parmi le code source que le développeur a officialisé se trouve un fichier de configuration. Ce fichier, primordial pour l'infrastructure Paquito, contient un ensemble de **méta-données** telles que le nom du paquet ou sa version, mais les plus importantes sont les dépendances.

- **Le service Jenkins CI**, détecte un changement et procède donc au téléchargement du projet (il récupère par la même occasion un maximum d'informations qui serviront lors de la phase de compilation). Il lancera ensuite une série de containers Docker adaptés à chaque distribution, ses versions et les architectures.

  Remarque : Normalement utilisé pour réaliser de l'intégration continue, Jenkins CI est utilisé dans le cadre du projet uniquement pour ses fonctionnalités d'écoute de dépôts et de démarrage de containers Docker.

- Chaque **container Docker** (qui a une configuration vierge) se chargera de compiler le code source (notamment à partir des informations récoltées par Jenkins CI) en considérant les dépendances (autre logiciel, librairie...). En cas d'echec, le container concerné renvoi les messages d'erreurs au développeur puis s'arrête (les autres containers ne sont pas impactés).

  Remarque : Les éventuelles erreurs renvoyées par le container seront sauvegardées sous forme d'un fichier.

- Une fois la compilation réussie et **le package créé**, celui-ci est testé sur un nouveau container Docker créé pour cela (comme pour la compilation, ce container est vierge). L'idée est de s'assurer que l'installation du package aboutisse avec succès et que le programme associé fonctionne correctement. Ce dernier point est vérifié grâce aux tests arbitraires que le développeur fournit sous formes de scripts . En cas d'échec d'un test, le container en question est arrêté et le développeur est notifié.

- Une fois que le package est réputé conforme et fonctionnel, celui-ci est inséré dans un dépôt de logiciel lié à sa distribution, sa version et

son architecture. Les utilisateurs n'auront qu'à télécharger le logiciel par l'intermédiaire de leur gestionnaire de paquets (après avoir bien entendu ajouté l'adresse du miroir où se trouve le package dans le fichier **source.list**).

Nous allons dans un premier temps tester ce qui fonctionne déjà c'est-à-dire la création de paquets pour les distributions Debian , Redhat et ArchLinux (en prenant en compte les différentes architectures, c'est-à-dire 32bits et 64bits).

- Identifier les points communs et les différences pour la construction des paquets (selon les systèmes de gestions de paquets de chaque distribution).
- Modifier le fichier de configuration de l'ancien projet Paquito pour prendre en compte les nouveaux types de paquets (en se basant notamment sur les recherches sur les points communs et les différences entre types de paquets) ainsi que pour y inclure toutes les dépendances nécessaires pour la génération de ces paquets.
- Modifier et adapter si nécessaire les scripts de l'ancien projet paquito pour ajouter le support des paquets pour Mac en plus des paquets Redhat, Archlinux et Debian.
- Création d'un service web connecté à GitHub permettant de lancer la construction des paquets à chaque 'push' ou 'tag'. (réalisées indépendemment de l'outil paquito).
- Création d'une interface web autour de ce service.

La condition de validation finale de ce scénario est donc d'obtenir un ensemble de paquets installables et fonctionnels (les logiciels doivent fonctionner).