CONCERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE NATIONAL DE TOULOUSE

Par

MILAZZO Christopher

MEMOIRE I1 - EXPERIENCE PROFESSIONNELLE XLM AERO

05/10/2020

AVANT PROPOS

Dans ce court mémoire, je parle de mon expérience professionnelle vécue au sein de la société de service XLM Aéro.

Mes principales missions étaient au nombre de 3 et sont synthétisées dans le chapitre **2.1 Mes missions**. Puis j'entre plus en détails dans les chapitres qui suivent (**2.2, 2.3 et 2.4**).

Table des matières

1 - XLM Aero

XLM Aéro a été fondé en 2010 suite à l'acquisition du logiciel **KEOPS**. Ce dernier a été abandonné par **IFR France**, la société qui l'a développé car il devenait déficitaire. Puis il fut repris par XLM Aéro et est maintenant vendu en tant que service à Air France-KLM, Corsaire, Aircalin et quelques autres petites compagnies aériennes.

2 - Mon expérience

2.1 - Mes missions

Au sein **d'XLM Aéro** je me suis vu confié 3 missions :

- Maintien et amélioration de **KEOPS**
- Mise à niveau et correction de failles de sécurité du portail web **Crew Web Access**
- Conception et développement d'un nouveau module : **Crew Mobile Access**

2.2 - KEOPS

2.2.1 - Contexte

KEOPS est une application de gestion des flottes et des coûts pour les compagnies aériennes. C'est tout un écosystème de modules qui offrent une multitude de services. Ces modules sont pilotés par le biais d'une application bureau développée en Java avec EMF (Eclipse Modeling Framework).

2.2.2 - Démarche de travail

Le projet KEOPS suit la méthodologie classique (cycle en V) de la manière suivante :

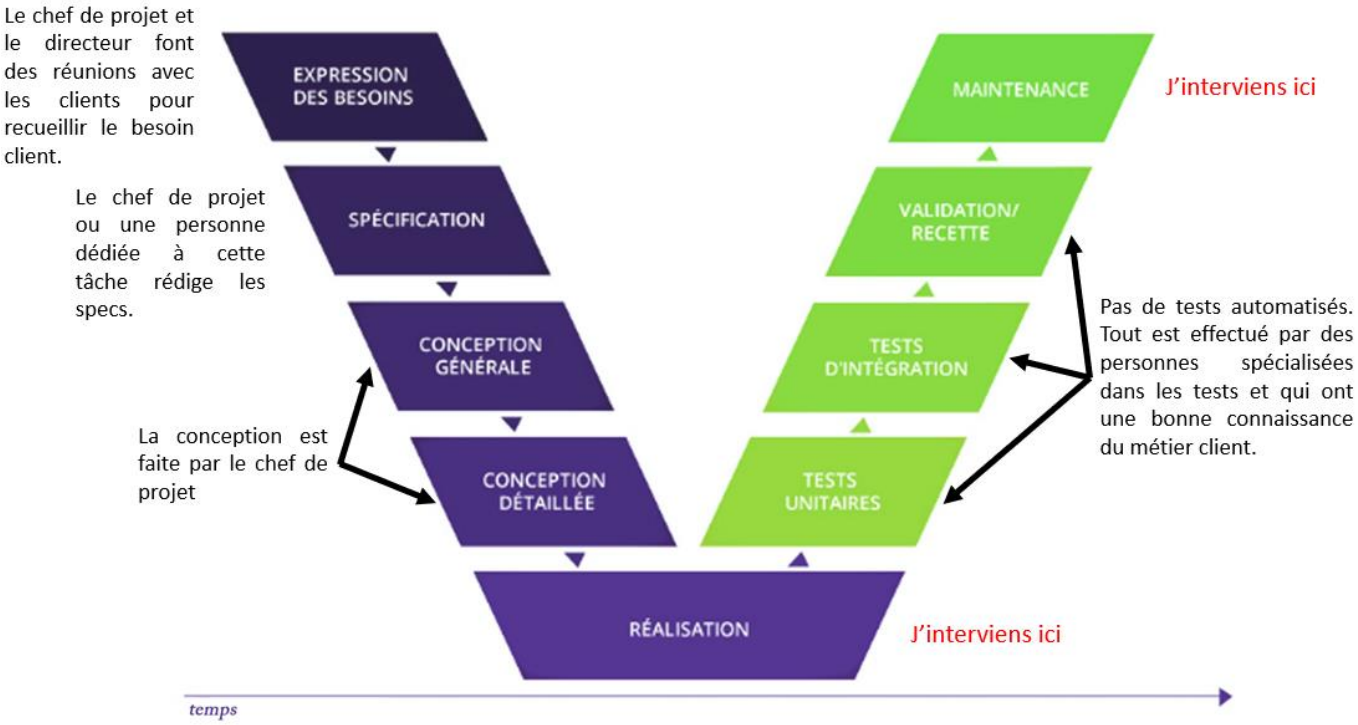

Figure 1 – Méthodologie de travail sur le projet Keops

Pour la partie **issue tracking**, **Mantis** est utilisé. Cela permet au chef de projet d'avoir une vision globale sur la roadmap et de pouvoir affecter les développements/corrections aux développeurs :

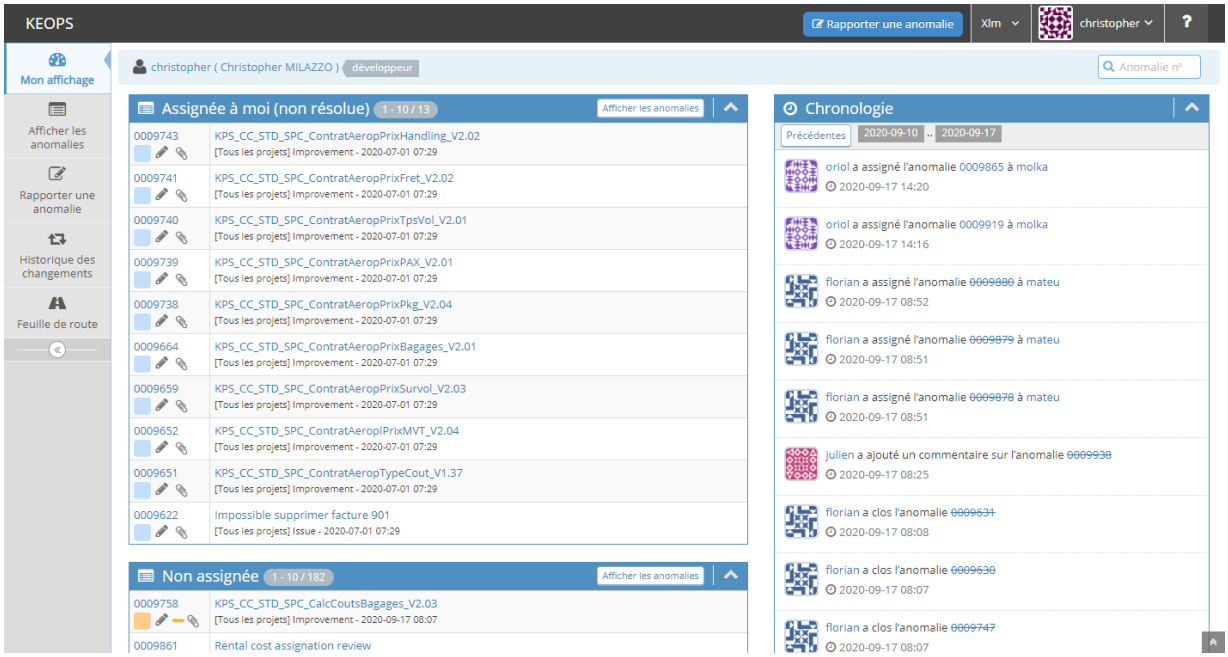

Figure 2 – Ecran principal de Mantis

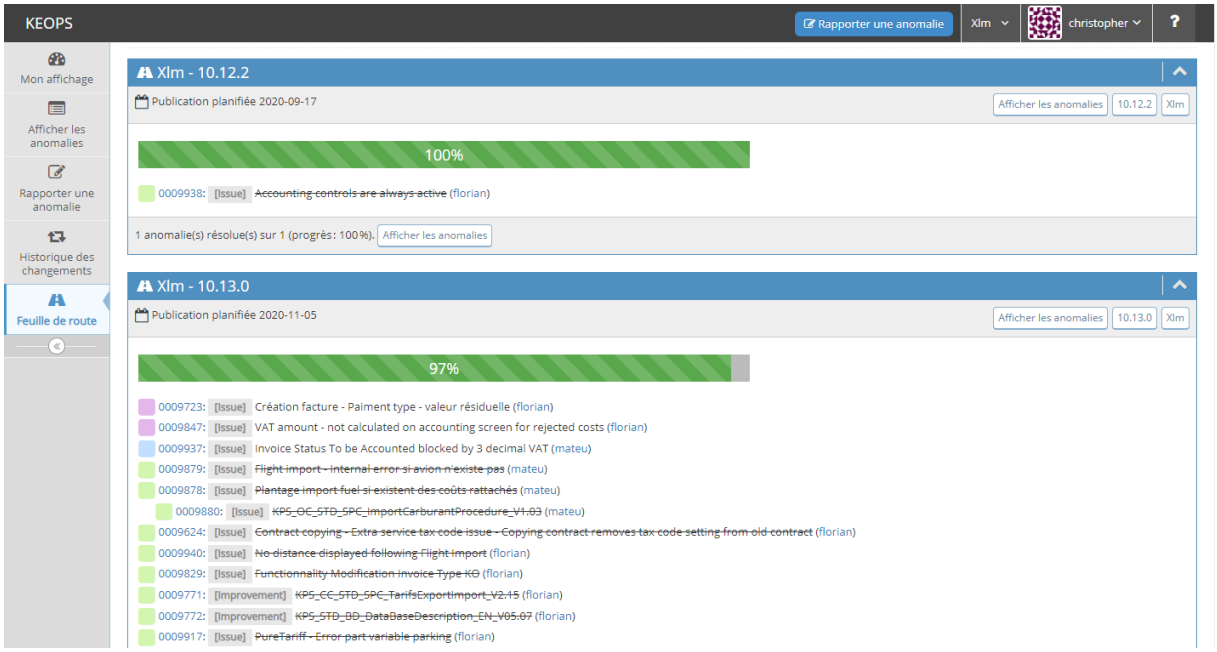

Figure 3 – Roadmaps définis dans Mantis

Le projet ne compte pas de pipeline (au sens de 'devOps'), il n'y a pas de contrôle qualité du code ou de tests unitaires lancés automatiquement. En revanche les builds sont tous automatisés par le biais de Jenkins :

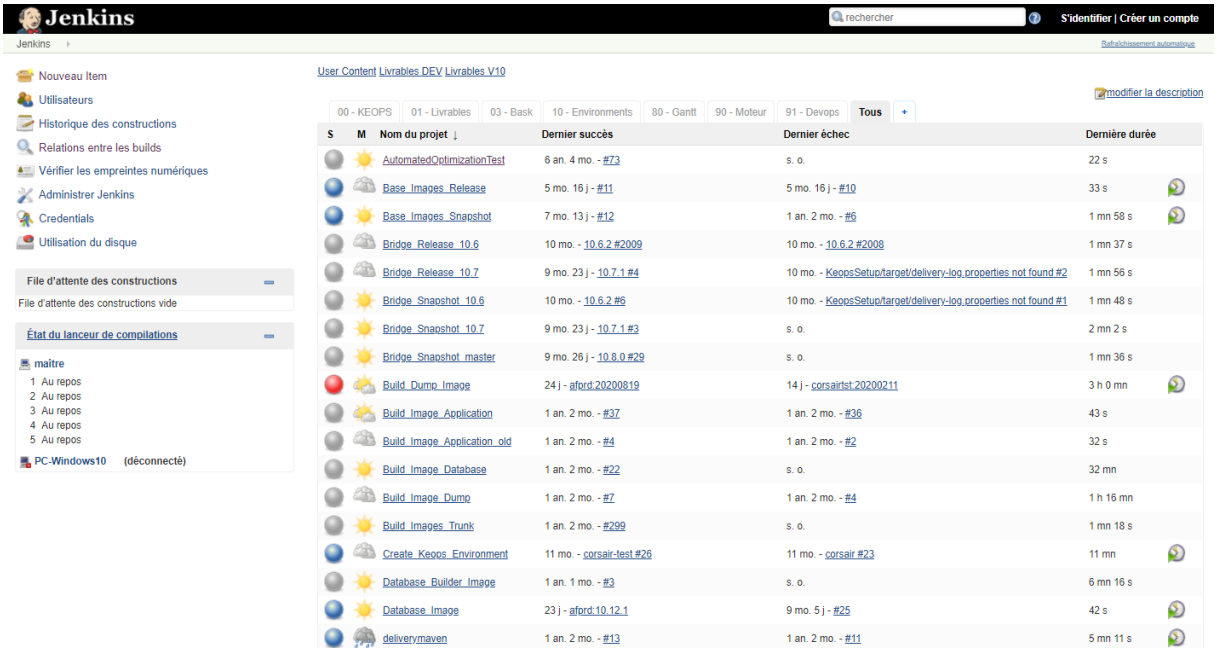

Figure 4 – Ecran principal de Jenkins

2.2.3 - Architecture

Le projet KEOPS date de 1987, à l'époque il fonctionnait sur **AS400**.

En 2006, la couche service (business layer) est modernisée et devient un service backend en **JAVA 1.6** et communique via HTTP en Spring Remoting.

En 2010, avec l'évolution de Java, il a été entrepris une refonte de l'interface utilisateur en technologie Java avec **l'Eclipse Modeling Framework**.

L'AS400 est maintenu pour certains clients. Toutefois, afin de faire évoluer **KEOPS** et de proposer une interface plus récente (**KEOPS CLIENT**), il a fallu créer un connecteur (**KEOPS BRIDGE**) capable de faire du **REST** et du **SOAP** tout en maintenant la communication avec le noyau (**KEOPS WEB**).

L'architecture physique de **KEOPS** peut se résumer ainsi :

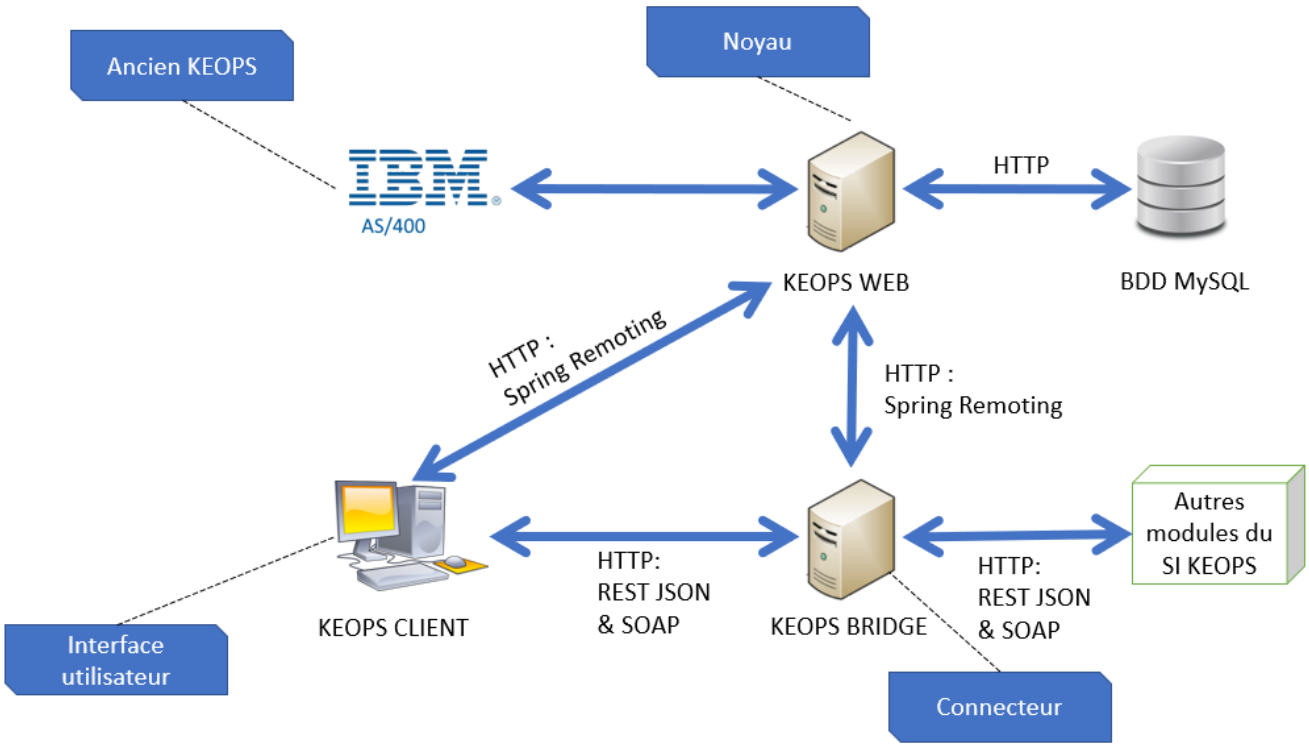

Figure 5 – Architecture physique du système Keops

2.2.4 - Mise en œuvre

Comme vu sur la *Figure 5* juste au dessus, Keops est constitué en majeur partie de :

- **Keops Client**, une application développée avec le FrameWork EMF (Eclipse Modeling Framework),
- Du noyau **Keops WEB**, un **web service** communiquant en **Spring Remoting** développé avec **J2EE** en **Java 1.6**
- Et d'une base de données **MySQL**.

Le Framework de type ORM (Object Resource Mapping) **Hibernate** à été utilisé pour la génération des DAOs pour **Keops WEB**.

Une fois les DAOs générés, on doit faire en sorte d'aligner les modèles de données de **Keops Client** et de **Keops WEB** avec le **MLD** de la **base de données**. Pour cela on s'appuie sur les outils de génération de code de EMF. En effet, EMF fonctionne sur le principe de la dématérialisation d'une problématique en un modèle de données pouvant proposer un ensemble de solutions orientées métier, c'est le fondement même du paradigme objet. Pour cela, on passe par le fichier **keops.ecore** qui décrit la structure de données, puis par le biais du fichier **keops.gen model**, on génère les classes java façons objets EMF.

Une fois le modèle de données EMF généré, nous passons par un outil JAVA appelé **JET** (Java Emitter Templates) pour transformer les objets EMF en objets **JAVA (POJO)** qui seront utilisés pour la couche domain de **Keops WEB**. De manière générale, **JET** permet de générer du code JAVA en fonction d'un template contenant divers directives pour le générateur. Dans le cas de Keops, le template est un fichier XML appelé **ContextStandard.xml** qui va décrire de quelle manière les **fichiers EMF** doivent être transformés en **POJO**.

La séquence de génération citée au-dessus peut être schématisée de la manière suivante :

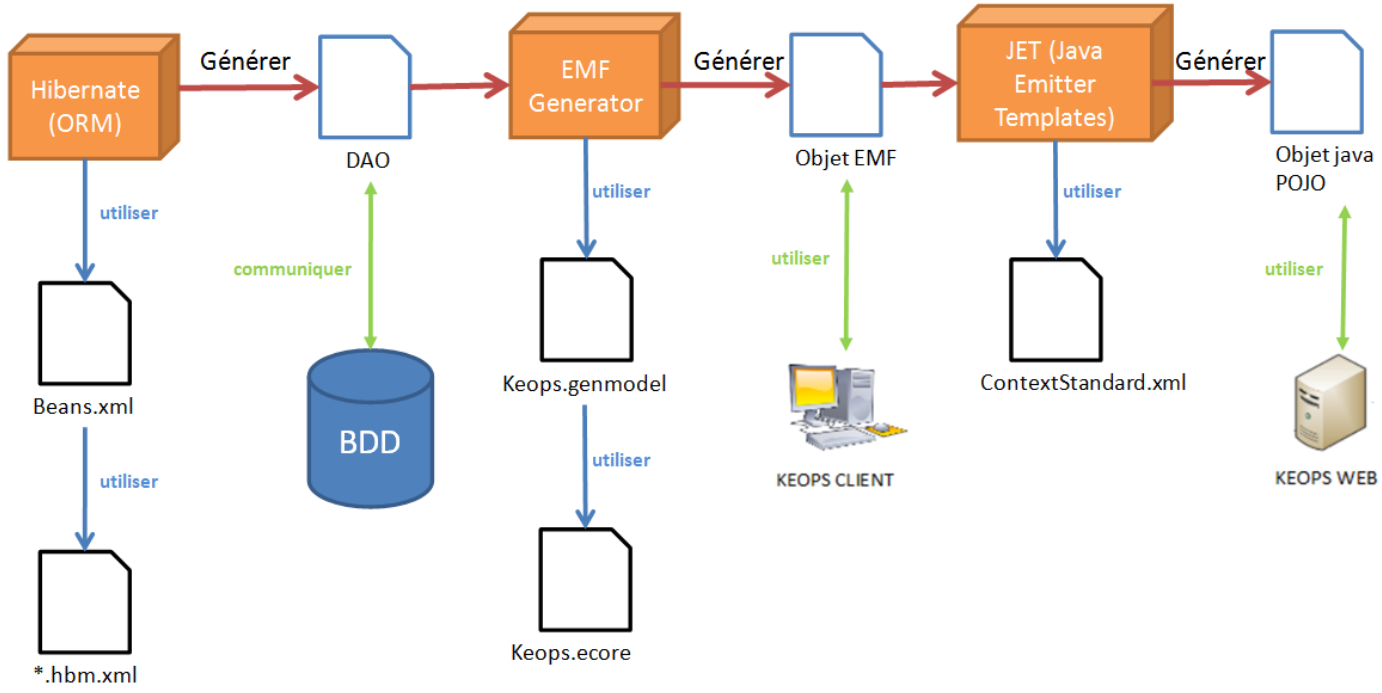

Figure 6 – Les couches applicatives de Keops

2.3 - Crew Web Access

2.3.1 - Contexte

Crew Web Access est un portail WEB qui vient se greffer au système d'information de **KEOPS** par le biais de **l'AS400** (voir *Figure 5*).

C'est un portail permettant aux **PN** (Personnel Naviguant) de **consulter leur planning**, de recevoir des **notifications** (des messages) de la part de la compagnie et de communiquer des **désidératas** en matière de congés et d'activités sur le planning.

2.3.2 - Mise en œuvre

Lors d'une campagne de **tests d'intrusion** sur le portail « Crew Web Access » menée fin 2019, il a été découvert **4 failles majeures**, listées ci-dessous, que j'ai eu à corriger :

- R13 – Mettre à jour la solution Crew. L'application doit contrôler la session et les droits de l'utilisateur avec la valeur du paramètre txtActId utilisé par plusieurs fonctions de l'application.
- R5 - Implémenter une politique de mot de passe robuste.
- R14 – Retourner un message d'erreur générique en cas d'échec d'authentification. Par exemple, le message suivant peut être retourné par l'application après un échec d'authentification : « L'identifiant ou le mot de passe sont incorrects. »
- R17 – HTTPS. Rediriger tous les flux HTTP vers le service Web correspondant en https en modifiant la configuration des serveurs applicatifs.

De plus, il a fallu **migrer** le code source en **PHP4** vers **PHP7** afin de le faire tourner sur des serveurs à jour pour limiter les failles de sécurité.

2.4 - Crew Mobile Access

2.4.1 - Contexte

Crew Mobile Access (CMA) est une refonte du **Crew Web Access**. Il permet de fournir les mêmes services que le portail Crew Web Access mais sous forme **d'application mobile**. Cela permettra aux **PN** (Personnel Naviguant) d'accéder au planning directement à partir du téléphone de manière plus ergonomique.

Le Crew Mobile Access est un service qui vient se greffer au système d'information de KEOPS par le biais du **KEOPS BRIDGE** (voir *Figure 5*).

2.4.2 - Démarche de travail

La méthodologie de travail est sensiblement la même que pour l'ensemble du projet Keops, puisque Crew Mobile Access s'inscrit comme étant un module complémentaire venant s'ajouter au système d'information de Keops.

2.4.3 - Conception

Angular permet de développer des application WEB (site web) de manière rapide et efficace.

NativeScript permet de créer des applications natives pour mobile (Android et IOS) en utilisant des technologies WEB.

Principe de fonctionnement :

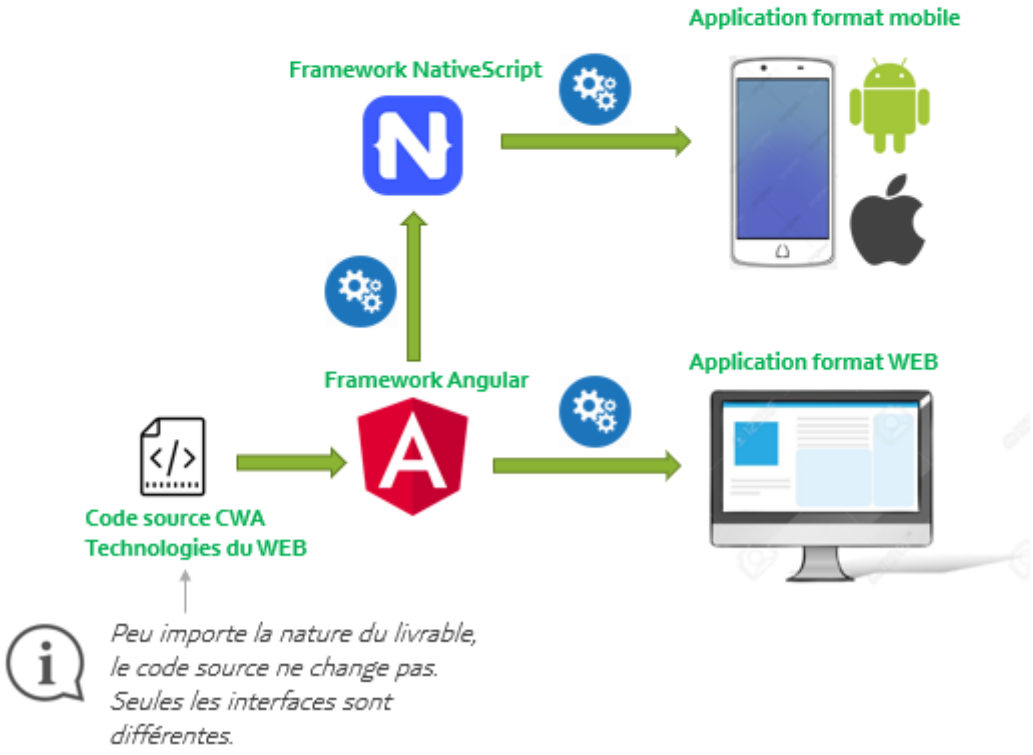

Figure 7 – Flow de construction de Crew Mobile Access

2.4.4 - Architecture

Pour des raisons de compatibilité, le **Crew Mobile Access** ne communique pas directement au Web Service de **KEOPS**. En effet, le **KEOPS BRIDGE** est là pour jouer un rôle de **connecteur** entre les services nouvelle génération et le noyau de KEOPS : **KEOPS WEB.**

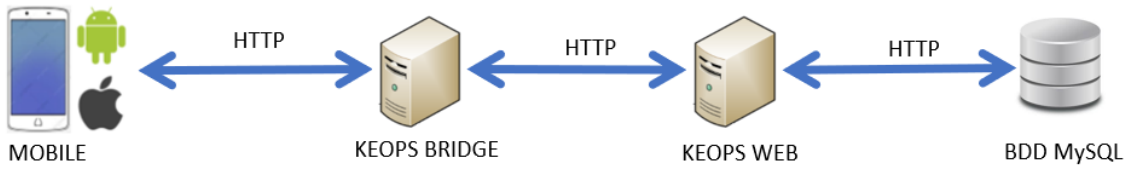

Figure 8 – Intégration de Crew Mobile Access dans le SI Keops

Ci-dessous, vous trouverez un diagramme résumant l'architecture logiciel mise en œuvre pour Crew Mobile Access :

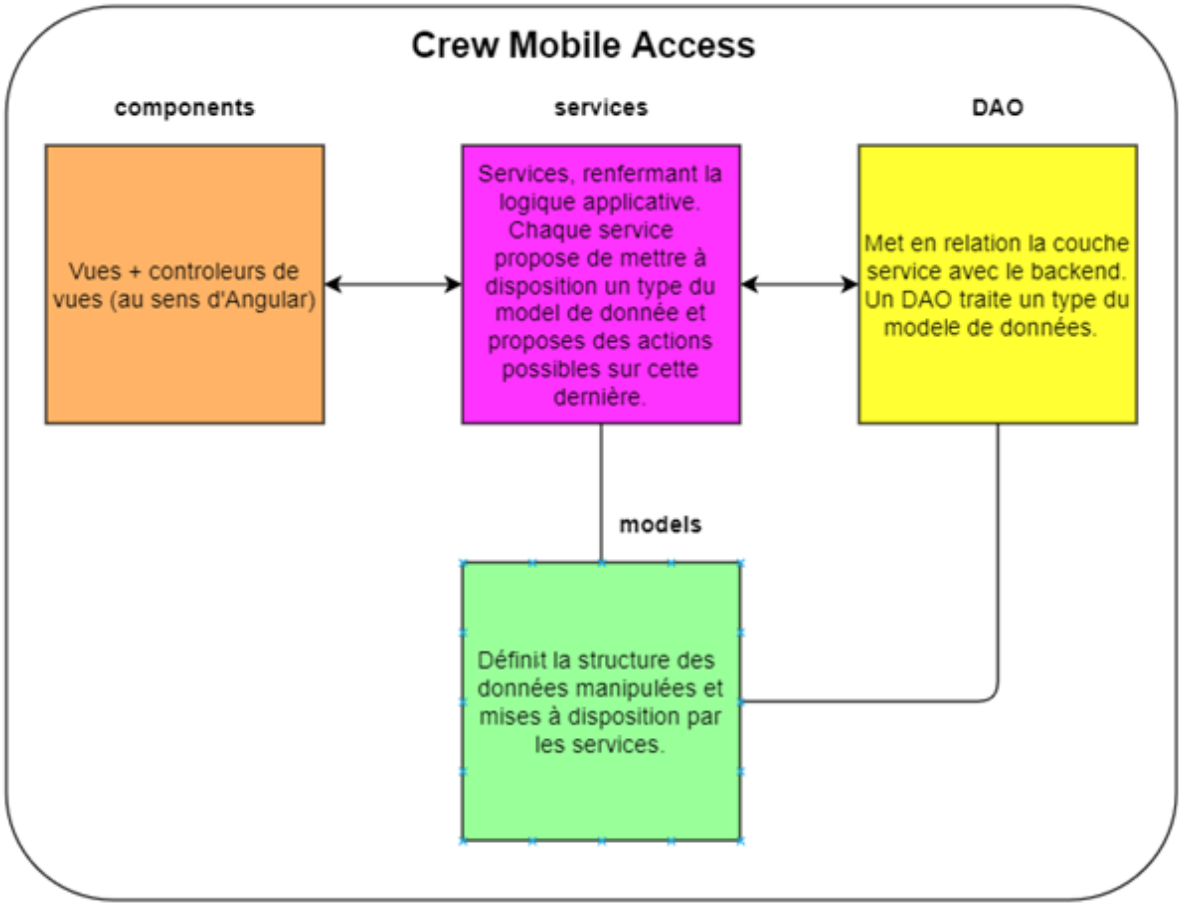

Figure 9 – Architecture logiciel de Crew Mobile Access

2.4.5 - Mise en œuvre

Comme indiqué sur la *Figure 9*, l'application se compose de **4 types de modules** :

Les **composants** (components) qui sont constitués de templates pour décrire les vues (version WEB et MOBILE), de feuilles de style et un contrôleur de vue. Une page de l'application peut être composée de un à plusieurs composants.

Les **contrôleurs** de vue définissent le flow d'exécution de la vue. Lorsque les contrôleurs veulent travailler sur une donnée en particulier, ils font appel aux services.

Les **services** mettent à disposition des mutateurs permettant de travailler sur les données, et un sujet sur lequel les utilisateurs du service peuvent souscrire afin d'être informé et de recevoir la données associées à chaque fois que celle-ci est modifiée. En effet, le patron de conception des observables est utilisé afin de garantir une haute réactivité de l'ensemble des composants formant les vues de l'application.

Les **modèles** (models) sont des objets TpyeScript représentant les états que peut prendre le système, autrement dit, c'est le modèle de données.

Pour finir, les **DAO** (Data Access Object) servent à assurer la communication avec les services du backend, **Keops bridge**. Keops bridge est une **API REST** utilisant **JSON** en guise de **DTO** et **JSON Web Token** en guise d'authentification.

3 - Conclusion

En travaillant sur le **maintien** et **l'évolution** de **Keops** j'ai appris à travailler sur des projets de grande envergure utilisant le gestionnaire de dépendance **Maven** et **Jenkins** en guise de constructeur d'artefact automatisé.

En maintenant et en migrant l'ancien portail WEB développé en **PHP,** j'ai pu voir ce qui se faisait en terme de développement au début du paradigme objet avec **PHP4,** et j'ai appris ce que cela engendrait de **migrer** une application entière d'une version dépréciée vers une version plus récente.

Pour finir, le développement en quasi-autonomie de **Crew Mobile Access**, m'a permis de pouvoir explorer des technologies intéressantes et moderne : **Angular** et **NativeScript**.

Fort de mes expériences précédentes au sein d'entreprise pour le service du numérique sur des projets récents, j'ai eu l'occasion chez **XLM Aéro** de travailler sur des projets moins récents mettant en œuvre des techniques de communication interprocessus et des pratiques de développement non étudiées en cours. Ces dernières ne sont certes plus très modernes, mais ne sont pas obsolètes pour autant car elles sont encore beaucoup rencontrées dans le monde professionnel. Il est donc intéressant et même conseillé de les avoir pratiquées au moins une fois dans un contexte professionnel pour prétendre devenir un développeur confirmé.