

GESTIONNAIRE DE LICENCES À L'UTILISATION

Rapport

Praz Vincent

A series of five parallel red lines of varying lengths, slanted diagonally from the bottom-left towards the top-right, positioned on the right side of the page.

Table des matières

1. Contacts	10
2. Abréviations.....	10
3. Introduction.....	11
3.1. Informations générales	11
3.2. Cahier des charges	11
3.2.1. Description générale :	11
3.2.2. Infrastructure :	12
3.2.3. Détails sur la réalisation :	12
3.2.4. Travaux à réaliser par le candidat :	12
3.2.5. Documents à remettre	13
3.3. Contexte.....	13
3.4. Description du projet	13
3.5. Objectifs	13
3.5.1. Standards	13
3.5.2. Compatibilité.....	13
3.5.3. Service	13
3.5.4. Composant de test	14
3.5.5. Upload	14
3.5.6. Création d'écrans.....	14
3.6. Fonctionnement général	14
3.6.1. Schéma.....	14
3.6.2. Explications	14
3.6.2.1. Plugin	14
3.6.2.2. Application	15
4. Planification	16
4.1. Planification Initiale	16
4.1.1. Version 1	16
4.1.2. Version 2	17
4.2. Planification Réelle	18
4.3. Justifications	19
4.3.1. Général.....	19
4.3.2. Développement de l'interface Java	19
4.3.3. Création des méthodes pour les écrans.....	19
4.3.4. Développement des écrans de tests	19

4.3.5. Tests unitaires	19
5. Journaux de bord.....	20
5.1. Jeudi 16 février	20
5.2. Jeudi 23 février	21
5.3. Vendredi 24 février	22
5.4. Jeudi 2 mars.....	23
5.5. Vendredi 3 mars	25
5.6. Jeudi 9 mars.....	26
5.7. Vendredi 10 mars	27
5.8. Jeudi 16 mars.....	28
5.9. Vendredi 17 mars	29
5.10. Jeudi 23 mars.....	30
5.11. Vendredi 24 mars.....	31
5.12. Jeudi 30 mars.....	31
5.13. Vendredi 31 mars	32
5.14. Jeudi 6 avril	33
5.15. Vendredi 7 avril	34
6. Documentation.....	35
6.1. Blowfish.....	35
6.2. L'architecture MVC	35
6.2.1. Les modèles	35
6.2.2. Les vues	35
6.2.3. Les contrôleurs	35
6.2.4. Chemin de traitement d'une requête	35
6.3. IntelliJ IDEA	36
6.3.1. Description	36
6.4. Spring	36
6.4.1. Définition.....	36
6.4.2. Inversion de contrôle (IoC).....	36
6.4.3. Composition.....	36
6.4.4. Spring Beans.....	36
6.5. Grails, qu'est-ce que c'est ?.....	37
6.5.1. Étymologie	37
6.5.2. Philosophie du Framework	37
6.5.2.1. Éviter les redondances de code	37

6.5.2.2. Établir des conventions	37
6.5.2.3. Architecture orientée modèle	37
6.5.2.4. Prototypage	37
6.5.2.5. Exploiter la puissance de la JVM	37
6.6. Groovy	37
6.6.1. Différences avec Java	37
6.6.1.1. Import	37
6.6.1.2. Initialisation de tableau	38
6.6.1.3. Each	38
6.7. Grails plugin	38
6.7.1. Définition	38
6.7.2. Utilisation	38
6.8. Subversion (SVN)	39
6.8.1. Définition	39
6.8.2. Utilisation	39
6.9. Base de données	39
6.9.1. Base de données embarquée H2	39
6.9.1.1. Définition	39
6.10. SQL-Maintenance	39
6.10.1. Description	39
6.10.2. Versionning	40
6.10.3. Composant	40
6.11. Pap-Crypto	40
7. Déroulement du projet	40
7.1. Mise en place de l'architecture	40
7.1.1. Mise en place de l'architecture via SVN	40
7.1.2. Variables d'environnement	41
7.1.3. Création du plugin en ligne de commande	43
7.1.4. Importation dans IntelliJ	43
7.1.5. Ajouts/Exclusions SVN	46
7.1.6. Récupération du fichier de publication	48
7.1.7. Création du fichier Bootstrap	49
7.1.8. Adaptation de la configuration BuildConfig	50
7.1.9. Adaptation de PapLicenceManagerGrailsPlugin	52
7.1.10. Test de la configuration	52
7.2. Domaines	54

7.2.1.	Création des classes de domaines	54
7.2.2.	Domaine LicenceFile.....	55
7.2.2.1.	Aperçu	55
7.2.2.2.	Description	55
7.2.2.3.	Champs.....	56
7.2.2.4.	Déclaration.....	56
7.2.2.5.	Contraintes.....	56
7.2.3.	Domaine Licence.....	56
7.2.3.1.	Aperçu	56
7.2.3.2.	Description	56
7.2.3.3.	Champs.....	57
7.2.3.4.	Déclaration.....	57
7.2.3.5.	Contraintes.....	57
7.2.4.	Domaine User	58
7.2.4.1.	Aperçu	58
7.2.4.2.	Description	58
7.2.4.3.	Champs.....	58
7.2.4.4.	Déclaration.....	58
7.2.4.5.	Contraintes.....	58
7.2.5.	Insertion de données tests.....	58
7.2.5.1.	Import.....	58
7.2.5.2.	Données tests	59
7.2.5.3.	Affichage des données tests	59
7.3.	Développement de l'interface Java	60
7.3.1.	Description	60
7.3.2.	Création de l'interface.....	60
7.3.3.	Méthodes.....	61
7.4.	LicenceInfo.....	61
7.4.1.	Description	61
7.4.2.	Création de la classe	62
7.4.3.	Code	62
7.5.	Services	62
7.5.1.	LicenceService.....	62
7.5.1.1.	Description	62
7.5.1.2.	Création du service	62

7.5.1.3. Méthodes	63
7.5.2. UserLicenceService	65
7.5.2.1. Description	65
7.5.2.2. Création du service	65
7.5.2.3. Méthodes	65
7.5.2.4. Injection	66
7.6. Écrans.....	66
7.6.1. Contrôleur	66
7.6.1.1. Création du contrôleur	66
7.6.1.2. Exécution automatique du contrôleur	66
7.6.1.3. Méthodes	67
7.6.2. Vue.....	67
7.6.2.1. Création de la vue.....	67
7.6.2.2. Index	68
7.6.3. CSS	69
7.6.3.1. Création et liaison du CSS.....	69
7.7. Fichier de licence.....	69
7.7.1. XML.....	69
7.7.2. XML encodé en Blow Fish.....	69
7.8. Tests Unitaires	69
7.8.1. Type de test.....	70
7.8.2. Structure	70
7.8.3. Méthode de test	70
7.8.3.1. Écriture.....	70
7.8.3.2. Vérification.....	70
7.8.4. Exemple complet	70
7.9. Scripts.....	71
7.9.1. Configuration	71
7.9.2. Création des scripts	72
7.9.2.1. Exemple MySQL	72
7.9.3. Mappings.....	72
7.9.3.1. LicenceFile.....	72
7.9.3.2. Licence.....	73
7.10. Déploiement	73
7.10.1. Exclusions	73

7.10.2.	Publication	74
7.10.3.	Intégration dans une application	75
8.	Test.....	75
8.1.	Sélection de la licence	75
8.2.	Édition d'une licence	76
8.2.1.	Licence avec champs corrects.....	76
8.2.2.	Licence avec champs incorrects	76
8.3.	Utilisateurs.....	77
8.3.1.	Ajout d'utilisateurs	77
8.3.2.	Activation/désactivation utilisateurs	77
8.3.2.1.	Désactivation	77
8.3.2.2.	Activation	77
8.4.	Information sur la licence en cours	77
8.4.1.	Modifications	77
8.4.2.	Valeurs calculées	77
8.5.	Création de fichier de licence encodé.....	78
8.6.	Upload de fichier de licence	78
8.6.1.	Upload de fichier de licence valide	78
8.6.2.	Upload de fichier invalide.....	79
9.	Problèmes rencontrés.....	80
9.1.	Génération d'un fichier XML.....	80
9.1.1.	Contexte	80
9.1.2.	Problème	80
9.1.3.	Résolution	80
9.2.	Lancement des tests unitaires.....	80
9.2.1.	Problème	80
9.2.2.	Tentative de résolution	80
9.2.3.	Résolution	81
10.	Améliorations possibles	82
11.	Conclusion	82
12.	Remerciements	82
13.	Date et Signature	82
14.	Références.....	83
14.1.	Grails	83
14.2.	MVC.....	83

14.3. H2	83
14.4. IntelliJ IDEA	83
14.5. Spring	83
14.5.1. Spring.....	83
14.5.2. IoC.....	83
15. Annexes	83
15.1. Procès-Verbaux	83
15.1.1. Séance du 16.02.2017	83
15.1.2. Séance du 23.02.2017 #1	84
15.1.3. Séance du 23.02.2017 #2	85
15.1.4. Séance du 16.03.2017	86

1. Contacts

Responsable :

M. Patrick Lathion
Responsable des développements
patrick.lathion@proactive.swiss

Apprenti :

M. Praz Vincent
Stagiaire
vincent.praz@proactive.swiss, praz.vinc@gmail.com

2. Abréviations

Abréviation	Description
MVC	Modèle Vue Contrôleur
DRY	Don't Repeat Yourself
JVM	Java Virtual Machine
PV	Procès-Verbal
JBordJourX	Journal de Bord Jour X (X étant le numéro du jour de travail sur le projet)
Resp	Responsable
SVN	Subversion
IDE	Environnement de développement
VCS	Version Control System (logiciel de contrôle des versions)
IoC	Inversion of Control

3. Introduction

3.1. Informations générales

Filière	: Informatique
Année scolaire	: 2016-2017
Responsable	: Lathion Patrick
Apprenti	: Praz Vincent
Experts	: Dubuis Bernard et David Joël
Lieu d'exécution	: Centre de développement Proactive Partners SA Rue de la Drague 18 1950 Sion
Nom du projet	: Gestionnaire de licences à l'utilisation
Nom du plugin	: PapLicenceManagerPlugin

3.2. Cahier des charges

3.2.1. Description générale :

Le centre de développement de Proactive Partners produit des logiciels orientés web, spécialisés dans les domaines de l'administration cantonale, des banques et de la santé.

Afin de s'adapter aux nouvelles tendances, Proactive revoit actuellement sa politique de facturation des logiciels. Jusqu'ici basée sur le modèle standard licence + maintenance, nous nous orientons vers le "pay as you use", à savoir un système de location à prix variable en fonction de l'utilisation (nombre d'utilisateurs, nombre de dossiers sous gestion, taille de la population, ...).

Pour mettre en place ce nouveau modèle, nous voulons intégrer dans nos applications un système de vérification et validation des licences. Pour respecter les principes du DRY, nous voulons un système de base unique, intégrable dans chacune de nos applications, sur lequel des composants spécifiques à l'application peuvent être greffés.

Le framework que nous utilisons dans nos développements (Grails) est très bien adapté à ce système. Il permet la création de plugins réutilisables dans des applications Grails. De plus, comme il s'appuie sur Spring, il est alors aisé d'injecter des composants spécifiques dans des interfaces.

3.2.2. Infrastructure :

Pour son TPI, Vincent disposera de la même infrastructure qu'il a déjà actuellement, à savoir :

- Un poste de travail pour développeur
- L'IDE IntelliJ avec une licence entreprise active
- Java JDK 1.7 et JDK 1.8
- Grails 2.3.11 et Grails 2.4.2
- Des comptes utilisateur actifs sur nos outils internes (SVN, Confluence, Bamboo, ...)
- Accès à toute la documentation interne et externe nécessaire à la réalisation du travail

3.2.3. Détails sur la réalisation :

Un nouveau plugin Grails 2.3.11 sera créé selon les standards de Proactive. Ce plugin, une fois suffisamment avancé pourra être déployé dans notre repository Artifactory, le rendant ainsi utilisable dans nos autres applications. Le code source du plugin sera publié dans notre serveur SVN, au côté des autres plugins existants.

Comme le plugin sera intégré dans les différentes applications Proactive, il devra être compatible avec les bases de données Oracle, SQL Server et MySQL. Pour rendre cela possible, il devra lui-même intégrer le plugin "SQL-Maintenance" développé par Proactive.

De base, le plugin devra proposer un service Grails de vérification des licences. Ce service utilisera une interface (au sens Java du terme) générique dans laquelle un composant spécifique à l'application finale sera injecté par Spring.

Pour simplifier le développement du plugin et aider aux tests, un composant spécifique au plugin lui-même sera créé. Ce composant permettra la vérification de la licence en fonction du nombre d'utilisateur actif enregistré dans l'application. Un écran de test sera mis en place afin de tester/valider le comportement de ce composant spécifique. Toutefois, il devra être marqué comme "exclu" pour les applications utilisant ce plugin.

Afin d'étendre ou réduire les droits de nos clients, le plugin permettra également l'upload de fichiers d'activation de licence. Le contenu de ces fichiers sera encodé en blowfish pour que les clients Proactive ne puissent pas "tricher" en créant leur propre fichier de licence.

Finalement, le plugin mettra également à disposition des fonctions permettant de créer des écrans dans nos applications pour que nos clients puissent consulter l'état de leurs licences :

- Nombre de licences acquises
- Nombre de licences utilisées
- Nombre de licences à disposition

Ce sont les applications finales qui seront responsables du rendu des écrans, en fonction de leur layout et de leur technologie (Grails, AngularJS, ...). Ce point ne fait donc pas partie du TPI de Vincent.

3.2.4. Travaux à réaliser par le candidat :

- A) Planifier le projet.
- B) Réaliser le projet.
- C) Réaliser un journal de bord du projet jour après jour.
- D) Remettre les documents complets à la date prévue.
- E) Exposer le projet au collège d'experts (non inclus dans la durée du projet).

3.2.5. Documents à remettre

Le dernier jour de la période allouée au projet, le candidat remettra en trois exemplaires :

- Toute la documentation papier nécessaire à la compréhension et à la reproductibilité du travail final.
- La planification du projet.
- Le journal de bord du projet.

Un dossier doit parvenir le jour même (le cachet de la poste faisant foi) aux deux experts évaluant le projet.

Un autre dossier doit parvenir au chef responsable du projet dans l'entreprise. Celui-ci doit l'analyser et le remettre ensuite (avec les corrections) à l'expert responsable.

3.3. Contexte

Le centre de développement de Proactive Partners produit des logiciels orientés web spécialisé dans les domaines de l'administration cantonale, des banques et de la santé.

Afin de s'adapter aux tendances actuelles du marché, Proactive revoit sa politique de facturation des logiciels jusqu'ici basée sur un modèle standard comprenant la licence ainsi que la maintenance du produit.

Cette société veut à présent mettre en place un système unique à chacune de leurs applications tout en respectant les principes du DRY qui doit être intégrable à toutes leurs applications sur lequel les composant spécifiques à l'application se grefferont.

3.4. Description du projet

Le projet consiste à la création de ce système. Pour ce faire, un nouveau plugin Grails 2.3.11 sera créé selon les standards de Proactive.

Le plugin final devra proposer un service Grails permettant de vérifier les licences et remplira ses objectifs définis au point suivant.

3.5. Objectifs

3.5.1. Standards

Le code du plugin ainsi que sa création respecteront les standards érigés par l'entreprise.

3.5.2. Compatibilité

Étant donné que le plugin sera intégré dans différentes applications Proactive à l'avenir, il doit être compatible avec les bases de données Oracle, SQL Server ainsi que MySQL.

Pour ce faire, le plugin développé par Proactive nommé « SQL-Maintenance » sera intégré au projet.

3.5.3. Service

Le plugin devra proposer un service Grails de vérification de licences qui utilisera une interface Java générique où le composant spécifique à l'application sera injecté par Spring.

3.5.4. Composant de test

Un composant de test sera créé afin de permettre la vérification d'une licence en fonction du nombre d'utilisateurs actifs enregistrés dans l'application. Un écran de test sera également mis en place afin de tester/valider le comportement de ce composant.

3.5.5. Upload

Afin d'étendre ou de réduire les droits des clients, le plugin proposera également un upload de fichiers d'activation de licence. Ces fichiers seront encodés en blowfish pour éviter que les clients puissent recréer eux-mêmes ces fichiers.

3.5.6. Création d'écrans

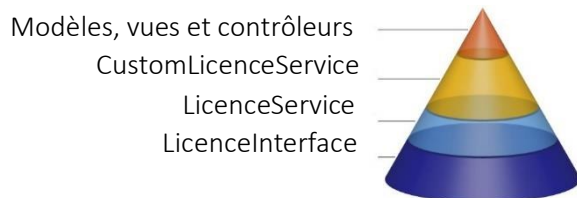
Le plugin mettra à disposition des fonctions qui permettront de créer des écrans dans les applications afin que les clients puissent consulter l'état de leurs licences.

Les écrans permettront de visualiser les nombres de licences acquises, utilisées et à disposition.

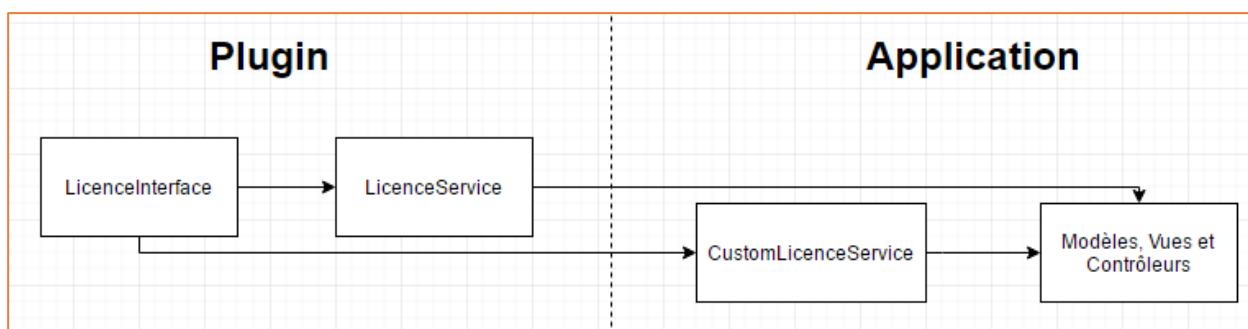
Les applications finales seront responsables du rendu de ces écrans.

3.6. Fonctionnement général

3.6.1. Schéma



Les parties bleutées représentent le contenu du plugin tandis que celles qui sont jaunes-orangées représentent la partie qui doit être fournie par l'application.



3.6.2. Explications

3.6.2.1. Plugin

Le plugin contient un service 'LicenceService' ainsi qu'une interface 'LicenceInterface'. Ce service utilise l'interface afin de pouvoir fournir à l'application les données concernant les licences.

3.6.2.2. Application

L'application quant à elle contiendra un service '*CustomLicenceService*' gérant les 'objets' qui consommeront la licence, par exemple : le nombre de places de parcs, les utilisateurs, cas médicaux... Ce service devra implémenter notre '*LicenceInterface*' et retournera différentes méthodes et permettra le calcul du nombre de licences utilisées.

La partie applicative contiendra également les écrans nécessaires à la gestion des licences

Dans les configurations de l'application, une référence Spring injectera le service '*CustomLicenceService*' dans le '*LicenceService*' du plugin.

4. Planification

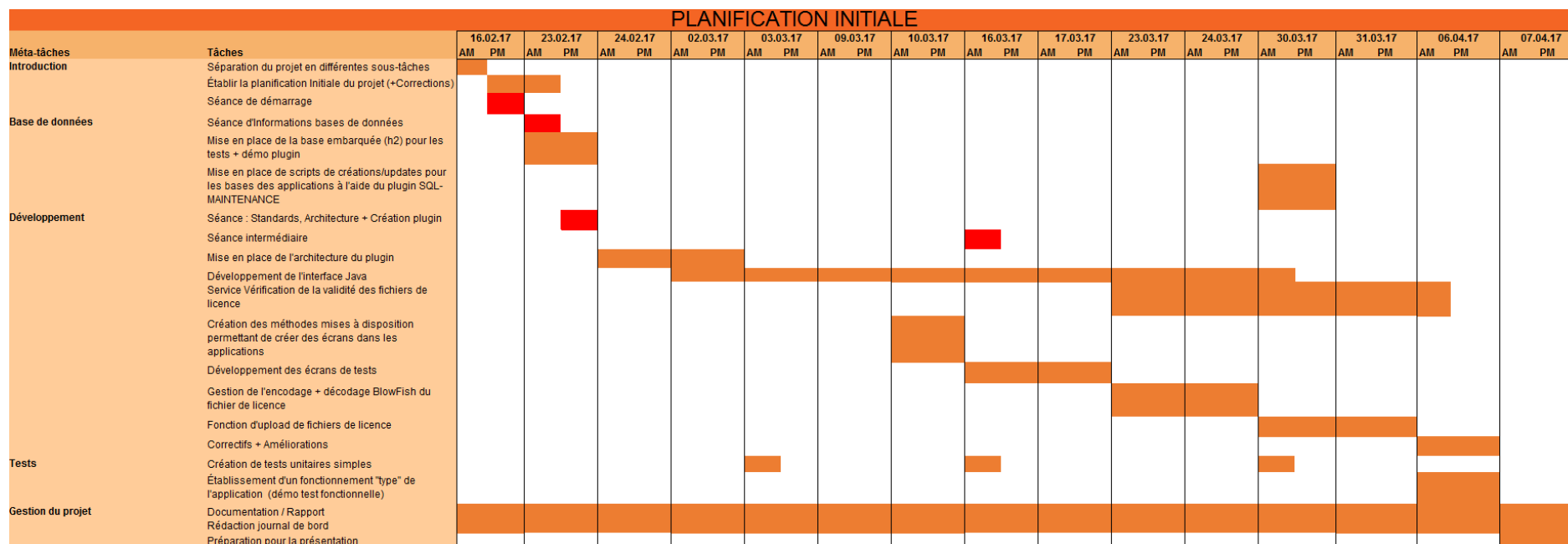
4.1. Planification Initiale

4.1.1. Version 1

		PLANIFICATION INITIALE																														
Méta-tâches	Tâches	16.02.17		23.02.17		24.02.17		02.03.17		03.03.17		09.03.17		10.03.17		16.03.17		17.03.17		23.03.17		24.03.17		30.03.17		31.03.17		06.04.17		07.04.17		
		AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	
Introduction	Séparation du projet en différentes sous-tâches																															
	Établir la planification Initiale du projet (+Corrections)																															
	Séance de démarrage																															
Base de données	Séance d'Informations bases de données																															
	Mise en place de la base embarquée (h2) pour les tests + démo plugin																															
	Mise en place de scripts de créations/updates pour les bases des applications à l'aide du plugin SQL-MAINTENANCE																															
Développement	Séance : Standards, Architecture + Création plugin																															
	Mise en place de l'architecture du plugin																															
	Développement de l'interface Java																															
	Gestion de l'encodage + décodage BlowFish du fichier de license																															
	Service Vérification de la validité des fichiers de license																															
	Création des méthodes mises à disposition permettant de créer des écrans dans les applications																															
	Développement des écrans de tests																															
	Fonction d'upload de fichiers de license																															
	Correctifs + Améliorations																															
	Création de tests unitaires simples																															
Tests	Établissement d'un fonctionnement "type" de l'application (démo test fonctionnelle)																															
Gestion du projet	Documentation / Rapport																															
	Rédaction journal de bord																															
	Préparation pour la présentation																															

Cette version présentant beaucoup de risques clairement visibles, il a été décidé de la modifier afin d'éviter des retards et imprévus.

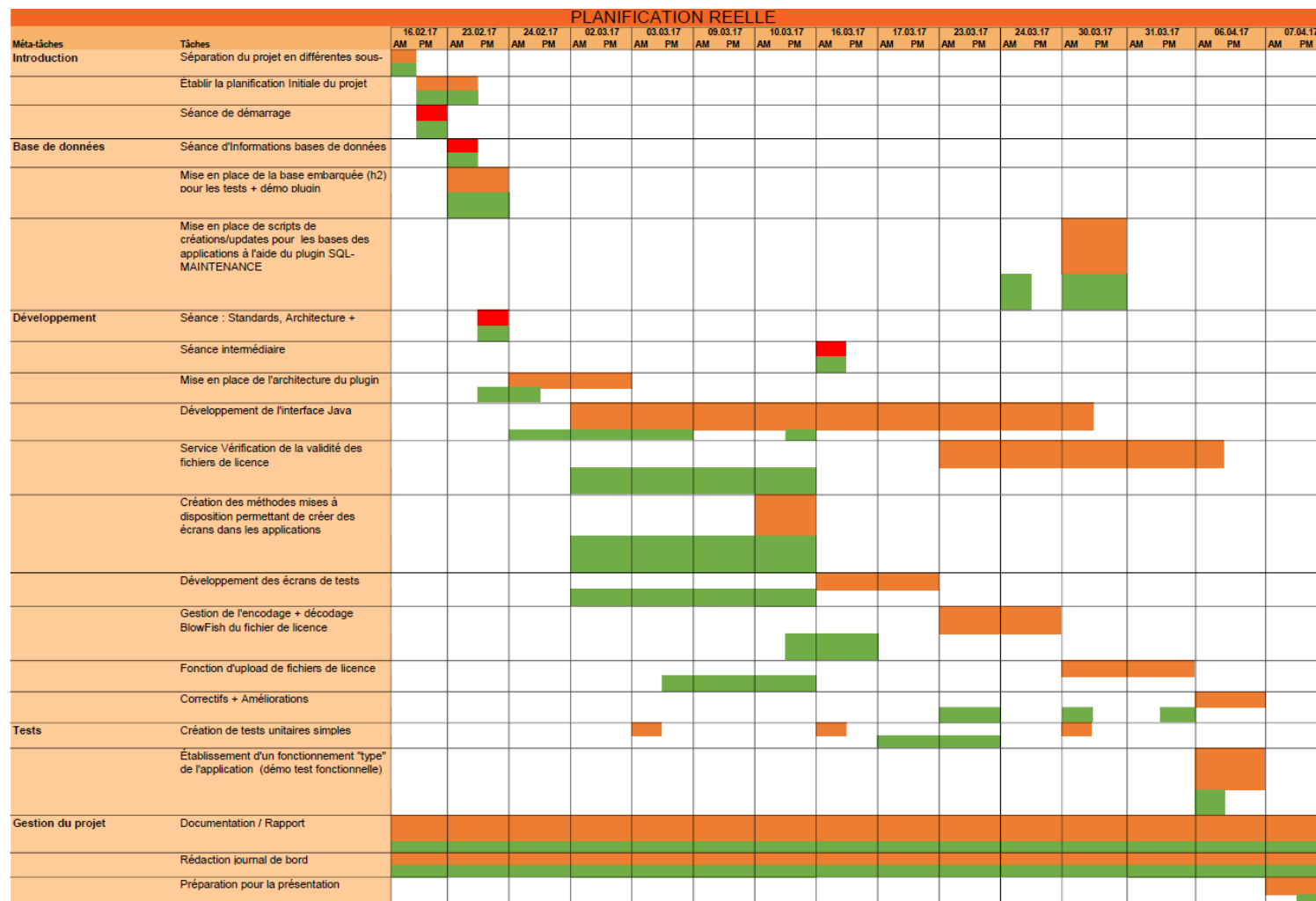
4.1.2. Version 2



Dans cette version, Les tâches concernant les scripts ont été déplacées vers la fin du projet, des tests unitaires ont été ajoutés tout au long du projet, les tâches de gestion des fichiers de licences ont été regroupées et raccourcies, la création des méthodes mises à disposition permettant la création des écrans a été rapprochée, la mise en place de l'architecture a été rapprochée, et finalement, la création de écrans tests a été décalée après la tâche précédente et est passée à 2 jours.

La séance intermédiaire a également été ajoutée à la planification.

4.2. Planification Réelle



4.3. Justifications

4.3.1. Général

Le service de vérification, la création des méthodes pour les écrans, le développement des écrans tests et la fonction d'upload ont été réalisés simultanément car j'avais besoin d'avancer dans chacune de ces tâches pour qu'elles avancent toutes.

4.3.2. Développement de l'interface Java

Le développement de l'interface m'a pris beaucoup moins de temps que prévu. Je pensais qu'il me serait beaucoup plus difficile de me lancer dans cette architecture interface/service.

4.3.3. Création des méthodes pour les écrans

La création des méthodes mises à disposition permettant de créer des écrans dans les applications m'a pris plus de temps que prévu car j'en avait besoin pour développer mes écrans tests.

4.3.4. Développement des écrans de tests

Le développement des écrans de tests a été débuté plus tôt afin de pouvoir tester l'avancement de mon code.

4.3.5. Tests unitaires

J'ai préféré attendre d'avoir avancé dans le code afin de pouvoir coder les tests unitaires. J'ai également créer ces tests d'un seul bloc pour ne pas me déconcentrer et savoir où j'en était. Lors de la création de ces tests, j'ai également eu un problème qui sera documenté plus bas et qui a renforcé mon envie de créer ces tests d'une seule traite.

5. Journaux de bord

5.1. Jeudi 16 février

Journal de bord				
Date		jeudi, 16. février 2017		
Projet				
Nom du projet			Date de remise	
Gestionnaire de licences à l'utilisation			vendredi, 07. avril 2017	
Membres du projet				
Nom		Praz		
Prénom		Vincent		
Profession		Informaticien		
Classe		EMVs IN 4		
H. Début	Période	Déroulement		
Périodes		Tâches effectuées	Remarques sur les tâches effectuées	
07:30	08:30	* Création et modification du journal de bord * Préparation de la planification initiale	* Modification de l'aspect graphique, ajout du bon nombre de jours, établissement d'une plage horaire convenable au nombre d'heures/jours + pauses * Découpage du projet en différentes sous-tâches + préparation du fichier Excel	
08:30	09:00	* Début de la rédaction du rapport * Préparation pour les séances à venir	* Préparation, mise en forme du document, début de la documentation 'générale' du projet * Création d'un fichier template de 'PV'	
09:00	10:00	* Infos Plugins * Rédaction rapport	* Explications sur le fonctionnement général d'un plugin et son utilisation dans l'entreprise + mise à dispositions de codes * Mise en place du 'squelette' du rapport	
10:30	12:00	* Rédaction rapport	* Documentation + Recherche et essais de mise en forme du code à l'intérieur du document Word	

13:00	15:00	* Préparation à la séance * Rapport	* Préparation de questions à poser et impressions des prototypes (planification + mise en forme code) * Documentation
15:20	16:00	* Rapport	* Documentation
16:00	17h20	* Séance de démarrage	* Séance en présence des experts pour démarrer le TPI
Problèmes rencontrés et résolution			
-			
Remarques sur la planification du projet			
-			
Objectifs pour le prochain jour			
* Corriger la planification initiale et rédiger puis envoyer le PV de la séance de démarrage			

5.2. Jeudi 23 février

Journal de bord			
Date		jeudi, 23. février 2017	
Projet			
Nom du projet		Date de remise	
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017	
Membres du projet			
Nom	Praz		
Prénom	Vincent		
Profession	Informaticien		
Classe	EMVs IN 4		
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Rédaction du PV * Rapport	* Rédaction du PV en fonction de ma prise de notes puis ajout de celui-ci au rapport * Ajout du JBordJour1 au rapport
08:30	09:00	* Rapport	* Documentation
09:00	10:00	* Corrections de la planification * Modification du PV * Rapport	* Analyse de l'ancienne planification, Discussion avec Patrick Lathion * Envoi du mail * Ajout des 2 versions de planifications ainsi qu'une explication

10:30	12:00	* Documentation * Séance explicative #1	* Documentation base de données * Explication sur les bases h2, SQL Server, ... ainsi que sur le plugin SQL-MAINTENANCE
13:00	15:00	* Documentation * Rédaction du PV de la séance #1	* Documentation SQL-Maintenance et base H2 + autres * Rédaction + ajout du PV au rapport
15:20	16:00	* Séance explicative #2 * Rédaction du PV séance #2	* Séance sur les standards, l'architecture et la création d'un plugin * Rédaction et ajout au rapport
16:00	17h20	* Mise en place de l'architecture * Documentation	* Création du plugin et début de config * Documentation mise en place Architecture
Problèmes rencontrés et résolution			
-			
Remarques sur la planification du projet			
* La planification initiale du projet à été modifié * Légère avance sur la planification			
Objectifs pour le prochain jour			
* Poursuivre la documentation et la mise en place de l'architecture du plugin			

5.3. Vendredi 24 février

Journal de bord			
Date		vendredi, 24. février 2017	
Projet			
Nom du projet		Date de remise	
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017	
Membres du projet			
Nom		Praz	
Prénom		Vincent	
Profession		Informaticien	
Classe		EMVs IN 4	
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Mise en place de l'architecture	* Mise en place + documentation, importation projet, configurations
08:30	09:00	* Mise en place de l'architecture	* Mise en place + documentation, configurations

09:00	10:00	* Mise en place de l'architecture	* Mise en place + documentation, configurations, test de la configuration
10:30	12:00	* Mise en place de l'architecture * Documentation * Développement interface Java	* Mise en place + documentation, configurations, test de la configuration * Ajout documentation IDEA * Réflexion sur les différents domaines et fonctions utiles + création diagramme pour plus de visuel
13:00	15:00	* Développement interface Java	* Création et documentation des domaines, déclaration + commentaires, discussions sur les contraintes sur les champs + création contraintes
15:20	16:00	* Développement interface Java	* Données tests
16:00	17h20	* Développement interface Java	* Données tests + documentation
Problèmes rencontrés et résolution			
-			
Remarques sur la planification du projet			
La mise en place de l'architecture m'a pris moins de temps que prévu j'ai pu donc prendre de l'avance sur le développement de l'interface java			
Objectifs pour le prochain jour			
Poursuivre le développement de l'interface Java			

5.4. Jeudi 2 mars

Journal de bord		
Date	jeudi, 02. mars 2017	
Projet		
Nom du projet	Date de remise	
Gestionnaire de licences à l'utilisation	vendredi, 07. avril 2017	
Membres du projet		
Nom	Praz	
Prénom	Vincent	
Profession	Informaticien	
Classe	EMVs IN 4	
H. Début	Période	Déroulement
Périodes	Tâches effectuées	Remarques sur les tâches effectuées

07:30	08:30	<ul style="list-style-type: none"> * Développement de l'interface Java * Service 	<ul style="list-style-type: none"> * Création, documentation de l'interface, réflexion sur les méthodes à créer + création * Création du service + documentation
08:30	09:00	<ul style="list-style-type: none"> * Développement de l'interface Java * Service * Écrans 	<ul style="list-style-type: none"> * Modification de l'interface et documentation * Discussion sur les méthodes + questions * Création et documentation des contrôleurs, méthodes et vues pour les écrans + réflexion
09:00	10:00	<ul style="list-style-type: none"> * Écrans * Création des écrans de test 	<ul style="list-style-type: none"> * Création et documentation des contrôleurs, méthodes et vues pour les écrans + discussion * Discussion, création et personnalisation des écrans de tests, documentation
10:30	12:00	<ul style="list-style-type: none"> * Création des écrans de tests 	<ul style="list-style-type: none"> * Créations des écrans et documentation (adaptation page index) => mise en forme affichage des données
13:00	15:00	<ul style="list-style-type: none"> * Écrans 	<ul style="list-style-type: none"> * Modification mise en forme et affichage des données+ modification du contrôleur Redirection, sélection et actualisation page en changeant de licence
15:20	16:00	<ul style="list-style-type: none"> * Domaines 	<ul style="list-style-type: none"> * Ajout d'un domaine + documentation à modifier et à compléter + discussions
16:00	17h20	<ul style="list-style-type: none"> * Domaines * Écrans 	<ul style="list-style-type: none"> * Ajout d'un domaine + documentation à modifier et à compléter * Suite des écrans
Problèmes rencontrés et résolution			
-			
Remarques sur la planification du projet			
* Exécution simultanée du développement de l'interface, du service, et des écrans tests afin de pouvoir coder + tester			
Objectifs pour le prochain jour			
* Continuer la page de démo + créer le service UserService (côté application)			

5.5. Vendredi 3 mars

Journal de bord			
Date		vendredi, 03. mars 2017	
Projet			
Nom du projet		Date de remise	
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017	
Membres du projet			
Nom		Praz	
Prénom		Vincent	
Profession		Informaticien	
Classe		EMVs IN 4	
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Écrans	* Modification du contrôleur ajout de fonctionnalité enable/disable user, début de l'ajout d'user
08:30	09:00	* Services + Interfaces	* Début service User : Création + Injection + test
09:00	10:00	* Service + Interfaces	* Modification des services, écritures des méthodes
10:30	12:00	* Écrans * Service	* Suite de la modification du contrôleur et de la vue pour la page de démo * Création d'un objet LicenceInfo + méthode ShowLicenceInfo()
13:00	15:00	* Écran * Service	* Gestion 'INSERT- DELETE - UPDATE' * Ajout d'une méthode pour l'update
15:20	16:00	* Service + Controller	* UPDATE (validation)
16:00	17h20	* Écrans * Service	* INSERT des users depuis l'écran * UPLOAD (Controller + Service, ...)
Problèmes rencontrés et résolution			

-
Remarques sur la planification du projet
<p>* Je n'ai pas pu commencer à coder les tests unitaires car cela nécessite que le reste du code soit plus complet.</p> <p>* Pour le reste, j'exécute toujours simultanément les tâches de service, écrans, méthodes, interface.</p>
Objectifs pour le prochain jour
<p>* Continuer le code (UPLOAD), améliorer la mise en forme de l'écran de démonstration et éventuellement commencer la création de tests unitaires</p>

5.6. Jeudi 9 mars

Journal de bord			
Date		jeudi, 09. mars 2017	
Projet			
Nom du projet		Date de remise	
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017	
Membres du projet			
Nom		Praz	
Prénom		Vincent	
Profession		Informaticien	
Classe		EMVs IN 4	
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Service + Controller	* Code upload + proposition d'un fichier XML
08:30	09:00	* Service + Controller	* Code upload + code lecture XML
09:00	10:00	* Service + Controller	* Code upload + code lecture XML
10:30	12:00	* Service * Écran	* Code upload + code lecture XML + Code création Fichier de licence et licence + Gestion de la validation et de l'historisation de la licence * Modification formulaire édition
13:00	15:00	* Service * Documentation	* Upload + création licence et validation + historisation des licences * Réalisation d'un schéma démontrant le principe de l'architecture : Recherche outil de schématique + documentation et réalisation

15:20	16:00	<ul style="list-style-type: none"> * Documentation * Code 	<ul style="list-style-type: none"> * Réalisation d'un schéma démontrant le principe de l'architecture : Recherche outil de schématique + documentation et réalisation * Nettoyage et commentaires du code
16:00	17h20	<ul style="list-style-type: none"> * Gestion d'erreur * Écrans * Documentation 	<ul style="list-style-type: none"> * Discussion sur la façon de gérer les erreurs et logs + début de gestion et tests * Légère modification graphique des écrans * Corrections + validation schéma
Problèmes rencontrés et résolution			
-			
Remarques sur la planification du projet			
* J'ai pu continuer à affiner les méthodes du service mais je préfère attendre d'avoir pu faire quelques tests pratiques et réaliser des méthodes de gestion d'erreurs avant de commencer les tests unitaires.			
Objectifs pour le prochain jour			
<ul style="list-style-type: none"> * Discuter, commencer et documenter la partie BlowFish * Gestion d'erreurs, logs, Tests + tests unitaires * Documenter Spring + Injections 			

5.7. Vendredi 10 mars

Journal de bord			
Date		vendredi, 10. mars 2017	
Projet			
Nom du projet		Date de remise	
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017	
Membres du projet			
Nom		Praz	
Prénom		Vincent	
Profession		Informaticien	
Classe		EMVs IN 4	
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Documentation	* Modifications du rapport
		* Gestion d'erreur	* Amélioration de la gestion d'erreur
08:30	09:00	* Gestion d'erreur	* Amélioration de la gestion d'erreur

09:00	10:00	* Documentation	* Amélioration de la documentation et ajout de la documentation concernant Spring
10:30	12:00	* Documentation * Code	* Amélioration de la documentation et ajout de la documentation concernant Spring + Recherche sur le BlowFish * Commentaires
13:00	15:00	* BlowFish * Code	* Recherche d'exemples de code BlowFish dans les autres applications de Proactive, lecture et compréhension du code + Installation plugin pap-crypto * Ajout de Documentation dans le code
15:20	16:00	* BlowFish * Code	* Recherche d'exemples de code BlowFish dans les autres applications de Proactive, lecture et compréhension du code + Installation plugin pap-crypto * Ajout de Documentation dans le code
16:00	17h20	* BlowFish * Code	* Recherche d'exemples de code BlowFish dans les autres applications de Proactive, lecture et compréhension du code + Installation plugin pap-crypto * Ajout de Documentation dans le code
Problèmes rencontrés et résolution			
-			
Remarques sur la planification du projet			
* Toujours de l'avance dans la réalisation			
Objectifs pour le prochain jour			
* Commencer le code de cryptage / décryptage du fichier de licence			

5.8. Jeudi 16 mars

Journal de bord		
Date	jeudi, 16. mars 2017	
Projet		
Nom du projet		Date de remise
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017
Membres du projet		
Nom	Praz	
Prénom	Vincent	
Profession	Informaticien	
Classe	EMVs IN 4	

H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* BlowFish	Début du code pour encrypter / décrypter un fichier de licence
08:30	09:00	* Séance Intermédiaire	
09:00	10:00	* Rédaction du PV * Documentation	
10:30	12:00	* BlowFish	* Suite encodage : travail sur l'écran afin de pouvoir créer un fichier de licence au format XML et encodé
13:00	15:00	* BlowFish	* Suite encodage : travail sur l'écran afin de pouvoir créer un fichier de licence au format XML et encodé
15:20	16:00	* BlowFish	* Découverte de la lenteur et de la non optimisation de la création du fichier, discussion collègues
16:00	17h20	* BlowFish	* Modification du code et finalisation de la création
Problèmes rencontrés et résolution			
* Problème lors de la génération fichier de licence. Résolution : Discussion avec collègues et utilisation d'une méthode fournie par Grails			
Remarques sur la planification du projet			
* Avance dans la réalisation			
Objectifs pour le prochain jour			
* Réalisation des tests unitaires et documentation			

5.9. Vendredi 17 mars

Journal de bord			
Date		vendredi, 17. mars 2017	
Projet			
Nom du projet		Date de remise	
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017	
Membres du projet			
Nom		Praz	
Prénom		Vincent	
Profession		Informaticien	
Classe		EMVs IN 4	
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Documentation	* Documentation diverse et documentation du test de l'application
08:30	09:00	* Tests Unitaires	* Recherche et essais tests unitaires
09:00	10:00	* Tests Unitaires	* Recherche et essais tests unitaires

10:30	12:00	* Tests Unitaires	* Code des tests
13:00	15:00	* Tests Unitaires	* Code des tests + Lecture doc
15:20	16:00	* Tests Unitaires	* Code des tests + Lecture doc
16:00	17h20	* Tests Unitaires	* Code des tests + Lecture doc
Problèmes rencontrés et résolution			
-			
Remarques sur la planification du projet			
* La création des tests unitaires va être effectuée à la suite			
Objectifs pour le prochain jour			
* Terminer la création des tests unitaires et les perfectionner			

5.10. Jeudi 23 mars

Journal de bord			
Date		jeudi, 23. mars 2017	
Projet			
Nom du projet		Date de remise	
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017	
Membres du projet			
Nom		Praz	
Prénom		Vincent	
Profession		Informaticien	
Classe		EMVs IN 4	
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Tests Unitaires Simples	* Écritures des tests + recherche erreurs
08:30	09:00	* Tests Unitaires Simples	* Écritures des tests + résolution erreur
09:00	10:00	* Documentation	* Documentation du problème
10:30	12:00	* Documentation	* Documentation tests unitaires
13:00	15:00	* Correctifs et améliorations du code	* Optimisation du code
15:20	16:00	* Correctifs et améliorations du code	* Optimisation du code
16:00	17h20	* Correctifs et améliorations du code	* Optimisation du code
Problèmes rencontrés et résolution			
* Problème impossible de lancer des tests unitaires => erreurs 'transactionnelles'. Résolution : Changement import pour la classe transactionnelle dans les tests et le service			
Remarques sur la planification du projet			
* Correctif en même temps que les tests unitaires pour optimiser son fonctionnement			
Objectifs pour le prochain jour			
* Réaliser les scripts de bases et documenter les méthodes du code dans le rapport + amélioration rapport			

5.11. Vendredi 24 mars

Journal de bord			
Date		vendredi, 24. mars 2017	
Projet			
Nom du projet		Date de remise	
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017	
Membres du projet			
Nom		Praz	
Prénom		Vincent	
Profession		Informaticien	
Classe		EMVs IN 4	
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Documentation	* Documentation des méthodes + amélioration de la documentation
08:30	09:00	* Documentation	* Documentation des méthodes + amélioration de la documentation
09:00	10:00	* Documentation	* Documentation des méthodes + amélioration de la documentation
10:30	12:00	* Création des scripts	* Lecture des conventions de nommage + Création et documentation
13:00	15:00	* Documentation	* Corrections et améliorations du rapport
15:20	16:00	* Documentation	* Corrections et améliorations du rapport
16:00	17h20	* Documentation	* Corrections et améliorations du rapport
Problèmes rencontrés et résolution			
-			
Remarques sur la planification du projet			
* Toujours de l'avance. J'en ai donc profité pour améliorer mon rapport			
Objectifs pour le prochain jour			
* Terminer la création des scripts			

5.12. Jeudi 30 mars

Journal de bord	
Date	jeudi, 30. mars 2017
Projet	
Nom du projet	Date de remise
Gestionnaire de licences à l'utilisation	vendredi, 07. avril 2017

Membres du projet			
Nom	Praz		
Prénom	Vincent		
Profession	Informaticien		
Classe	EMVs IN 4		
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Documentation	* Relecture et corrections
08:30	09:00	* Documentation	* Relecture et corrections
09:00	10:00	* Correction du code	* Optimisation
10:30	12:00	* Réalisation des scripts	* Création des scripts
13:00	15:00	* Réalisation des scripts	* Création des scripts
15:20	16:00	* Réalisation des scripts	* Test des scripts
16:00	17h20	* Réalisation des scripts	* Test des scripts + Documentation
Problèmes rencontrés et résolution			
-			
Remarques sur la planification du projet			
-			
Objectifs pour le prochain jour			
* Effectuer et documenter le déploiement sur l'artifactory			

5.13. Vendredi 31 mars

Journal de bord			
Date		vendredi, 31. mars 2017	
Projet			
Nom du projet		Date de remise	
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017	
Membres du projet			
Nom		Praz	
Prénom		Vincent	
Profession		Informaticien	
Classe		EMVs IN 4	
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Déploiement sur l'artifactory	* Informations
08:30	09:00	* Déploiement sur l'artifactory	* Déploiement
09:00	10:00	* Déploiement sur l'artifactory	* Documentation
10:30	12:00	* Documentation	* Relecture et amélioration documentation
13:00	15:00	* Correctifs + améliorations	* Découverte d'un bug dans le service + résolution
15:20	16:00	* Documentation	* Relecture et amélioration documentation

16:00	17h20	* Documentation	* Relecture et amélioration documentation
Problèmes rencontrés et résolution			
* Problème le service n'annulait pas l'édition d'une licence incorrecte => modification du mode de transaction			
Remarques sur la planification du projet			
-			
Objectifs pour le prochain jour			
* Se préparer pour le rendu du TPI			

5.14. Jeudi 6 avril

Journal de bord			
Date		jeudi, 06. avril 2017	
Projet			
Nom du projet		Date de remise	
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017	
Membres du projet			
Nom		Praz	
Prénom		Vincent	
Profession		Informaticien	
Classe		EMVs IN 4	
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Test application	* Lancement et test du bon fonctionnement de l'application
08:30	09:00	* Documentation	* Relecture de la documentation et corrections
09:00	10:00	* Documentation	* Relecture de la documentation et corrections
10:30	12:00	* Documentation	* Relecture de la documentation et corrections
13:00	15:00	* Préparation pour le rendu	* Fichiers sous clés USB, Renseignements sur la reliure d'un rapport
15:20	16:00	* Préparation pour le rendu	* Achat de matériel pour le rapport
16:00	17h20	* Préparation pour le rendu	* Achat de matériel pour le rapport
Problèmes rencontrés et résolution			
-			
Remarques sur la planification du projet			
-			
Objectifs pour le prochain jour			
* Faire la reliure et poster le rapport			

5.15. Vendredi 7 avril

Journal de bord			
Date		vendredi, 07. avril 2017	
Projet			
Nom du projet		Date de remise	
Gestionnaire de licences à l'utilisation		vendredi, 07. avril 2017	
Membres du projet			
Nom		Praz	
Prénom		Vincent	
Profession		Informaticien	
Classe		EMVs IN 4	
H. Début	Période	Déroulement	
Périodes		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	* Rendu	* Impression du rapport
08:30	09:00	* Rendu	* Relecture
09:00	10:00	* Rendu	* Reliure + mise en forme
10:30	12:00	* Rendu	* Reliure + mise en forme
13:00	15:00	* Rendu	* Postage
15:20	16:00	* Rendu	* Préparation de la présentation
16:00	17h20	* Rendu	* Préparation de la présentation
Problèmes rencontrés et résolution			
-			
Remarques sur la planification du projet			
-			
Objectifs pour le prochain jour			
-			

6. Documentation

6.1. Blowfish

Blowfish est un algorithme de chiffrement symétrique par blocs. Il utilise une taille de bloc de 64 bits et la longueur de la clé peut varier de 32 à 448 bits.

Il permet de créer de très grandes clés aléatoires. Cet algorithme possède une bonne rapidité d'exécution.

Blowfish est entièrement libre d'utilisation.

6.2. L'architecture MVC

Une application conforme au motif MVC comporte trois types de modules :

- Les modèles
- Les vues
- Les contrôleurs

6.2.1. Les modèles

Un modèle est un élément qui contient les données ainsi que la logique qui y est liée : la validation, la lecture et l'enregistrement.

Il peut contenir toute forme d'information. Prenons pour exemple une application bancaire, le modèle pourrait représenter les clients, les comptes et ainsi que des opérations telles que des crédits et débits.

6.2.2. Les vues

Les vues forment la partie visible de l'interface graphique. Elles se servent du modèle, de formulaires, boutons, etc...

Une vue contient des éléments visuels ainsi que la logique nécessaire pour afficher les données contenues dans le modèle.

Dans le cas d'une application web, une vue est composée de balises HTML.

6.2.3. Les contrôleurs

Les contrôleurs sont des modules qui traitent les actions commises par les utilisateurs et modifient les données du modèle ainsi que la vue.

6.2.4. Chemin de traitement d'une requête

Lorsqu'un utilisateur envoie une requête, celle-ci va effectuer un parcours spécifique :

- Elle est envoyée par l'utilisateur depuis la vue et est analysée par le contrôleur
- Celui-ci demande au modèle correspondant d'effectuer les traitements appropriés et informe la vue que la requête est en cours de traitement
- La vue ayant été notifiée envoie une requête au modèle pour qu'il se mette à jour.

6.3. IntelliJ IDEA

6.3.1. Description

IntelliJ est un IDE Java performant et rapide. Il possède de grands avantages tels que la complétion intelligente du code qui propose uniquement les types attendus dans le contexte actuel. Il comprend et fournit également une assistance de codage intelligent pour une grande variété de langages tels que SQL, JPQL, HTML, JavaScript, etc., même lorsque ce langage est injecté à l'intérieur d'un code Java. Il est également très ergonomique et permet de gagner facilement du temps avec des menus rapidement accessibles et intuitifs.

6.4. Spring

6.4.1. Définition

Spring est un framework libre permettant de construire et définir l'infrastructure d'une application java tout en facilitant son développement et ses tests.

Il prend en charge la création et mise en relation d'objets via un fichier de configuration. Le principal avantage est qu'avec Spring, les classes n'ont pas besoin d'implémenter une quelconque interface pour être utilisées par le framework. Cela est possible grâce à l'un des concepts clés du framework : l'inversion de contrôle.

6.4.2. Inversion de contrôle (IoC)

Avec l'inversion de contrôle, le flot d'exécution du logiciel n'est plus sous le contrôle direct de l'application mais du framework ou de la couche logicielle sous-jacente. En d'autres termes, c'est le framework qui gère les appels à l'application. Avec Spring, l'IoC est assurée de deux manières différentes : la recherche de dépendances et l'injection de dépendances. Cette dernière, consiste à créer dynamiquement ou '*injecter*' les dépendances entre les différentes classes et objets en s'appuyant sur un fichier de configuration. De cette façon, les dépendances entre composants logiciels ne sont plus écrites statiquement mais déterminées dynamiquement lors de l'exécution de l'application.

6.4.3. Composition

Le '*Noyau de base*' de Spring est basé sur une fabrique de composants informatiques nommés beans ainsi qu'un conteneur capable de stocker ces beans. Les beans peuvent être définies par le biais de fichiers de configurations ou XML.

6.4.4. Spring Beans

Les « Spring Beans » sont les objets qui constituent l'épine dorsale d'une application et sont gérées par le conteneur Spring IoC. Une Bean est un objet instancié, assemblé et géré par ce conteneur. Ces beans sont créées avec les métadonnées de configuration qui sont fournies au conteneur sous la forme de définition xml `<bean />`

6.5. Grails, qu'est-ce que c'est ?

Grails est un Framework open source qui permet de développer, à l'aide de la méthode agile, des applications web basées sur le langage Groovy en s'appuyant sur l'architecture MVC (Modèle-Vue-Contrôleur). Il permet d'écrire et exécuter des classes Groovy mais également des classes Java.

6.5.1. Étymologie

Le terme **GRAILS** est la contraction de deux mots : *Groovy* qui est un langage de programmation sur lequel il se base ainsi que *On Rails* pour donner l'image d'un Framework de développement rapide.

6.5.2. Philosophie du Framework

Comme chaque Framework, Grails a sa propre idéologie et ses principes fondamentaux.

6.5.2.1. Éviter les redondances de code

Grails permet d'éviter les répétitions de codes au travers de l'architecture MVC et de Groovy.

6.5.2.2. Établir des conventions

Il permet d'éviter d'avoir à préciser des détails inutiles lorsqu'ils respectent les conventions mises en place. Grails exploite cette idée et propose des comportements par défaut pour une grande partie de ses fonctionnalités.

6.5.2.3. Architecture orientée modèle

Le principe de base de cette architecture est l'élaboration de modèles.

6.5.2.4. Prototypage

Les mécanismes fournis par le Framework tels que le Scaffolding permettent de générer automatiquement un prototype d'application dès la création des classes de domaine.

6.5.2.5. Exploiter la puissance de la JVM

Grails exploite la richesse et la puissance de la machine virtuelle Java en compilant les scripts Groovy en bytecode Java.

6.6. Groovy

Groovy est le langage qui est utilisé par Grails. Il constitue une alternative au langage Java et s'inspire de Python, Ruby, Perl ainsi que Smalltalk.

Groovy est très proche de Java et peut être compilé à la volée dynamiquement ou à l'aide d'un compilateur en bytecode.

Ce langage étant compatible avec la JVM, il peut donc utiliser des bibliothèques Java et s'intégrer parfaitement dans des classes Java.

6.6.1. Différences avec Java

Groovy comporte des différences notables par rapport à Java.

6.6.1.1. Import

Les packages suivants sont automatiquement importés. Il n'est donc pas nécessaire de les importer explicitement.

- java.io.*
- java.lang.*
- java.math.BigDecimal
- java.math.BigInteger
- java.net.*
- java.util.*
- groovy.lang.*
- groovy.util.*

6.6.1.2. Initialisation de tableau

Les tableaux doivent être initialisés avec des crochets contrairement à Java qui le fait avec des accolades :

Java :

```
int[] array = { 1, 2, 3 }
```

Groovy :

```
int[] array = [1,2,3]
```

6.6.1.3. Each

Pour faire une boucle each en Groovy, il suffit d'entrer sa liste.each{ }.

```
list.each { println it }
```

6.7. Grails plugin

6.7.1. Définition

Un plugin Grails est un ensemble autonome de fonctionnalités pouvant être installé dans une application Grails.

6.7.2. Utilisation

Lorsqu'il est installé, il permet d'effectuer plusieurs opérations telles que :

- La définition de nouvelles « Spring Beans »
- L'ajout de nouvelles méthodes aux différents artefacts de l'application (Contrôleurs, domaines, classes, service, etc.)
- Fournir de nouveaux « tags » de bibliothèques
- Créer de nouvelles ressources et classes disponibles dans l'application ainsi que de nouvelles commandes Grails

6.8. Subversion (SVN)

6.8.1. Définition

Subversion est un logiciel de gestion des sources, des versions, et des conflits.

6.8.2. Utilisation

Pour ce projet, SVN a été mis en place par les administrateurs de l'architecture Proactive afin que je puisse y déposer mon plugin.

6.9. Base de données

6.9.1. Base de données embarquée H2

6.9.1.1. Définition

H2 est un système de base de données écrit en Java. Il peut être intégré à une application ou fonctionner en mode Client-serveur.

Il supporte un sous-ensemble du standard SQL, est Open Source et propose des interfaces de programmation SQL.

Les tables peuvent être créées en mémoire vive ou sur un disque/fichier externe.

La base se protège des injections SQL avec une fonctionnalité appelée *disabling literals* (désactivation des valeurs littérales).

La base de données H2 qui vient avec son outil Hibernate est intégrée dans Grails. La base de données à laquelle va s'attacher la base doit être définie dans le datasource.

Hibernate va créer automatiquement toutes les tables et liaisons correspondantes à la base liée.

6.10. SQL-Maintenance

6.10.1. Description

SQL-Maintenance est un plugin Grails développé par Proactive-Partners. Son but est de gérer les mises à jour des différentes bases de données utilisées dans les applications de l'entreprise. Les types de ces bases de données sont les suivants :

- H2
- SQLServer
- Oracle
- MySQL

Il permet d'écrire des scripts de modifications dans l'application afin de pouvoir modifier ces bases. Cette gestion manuelle certes plus contraignante permet une meilleure gestion des différentes bases.

6.10.2. Versionning

SQL-Maintenance est équipé d'un système de Versionning. Les scripts doivent être nommés avec un numéro de version. SQL-Maintenance va exécuter tous les scripts depuis la version actuelle vers la version ciblée.

S'il existe plusieurs scripts à passer pour une seule version, le nommage des scripts doit contenir un '#x' (x étant le numéro du script) pour les ordonner et ainsi les exécuter correctement.

Il permet aussi d'exécuter certains scripts uniquement chez un certain client. Pour ce faire, il faut ajouter un '@client' (client étant le nom du client dans l'application).

6.10.3. Composant

SQL-Maintenance gère différents composants qui seront répartis dans différents répertoires qui seront créés automatiquement lors de l'ajout du plugin à l'application.

- Groovy
- Oracle
- SQLServer

Pour chacun de ces composants, on pourra créer des scripts correspondants aux différentes bases de l'application. Le composant Groovy permet d'écrire des scripts plus complexes et de gérer des procédures stockées, ...

6.11. Pap-Crypto

Pap-crypto est un plugin développé par Proactive permettant de faire de la cryptographie BlowFish. Il sera également utilisé dans mon projet.

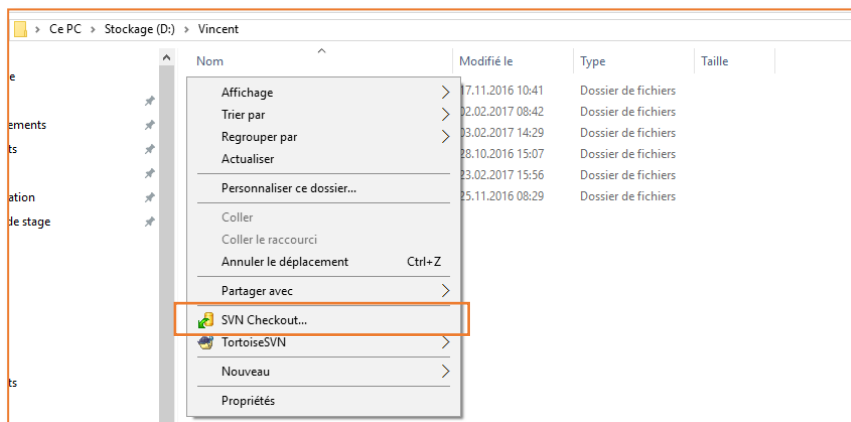
Il met à disposition une méthode d'encryptage et de décryptage.

7. Déroulement du projet

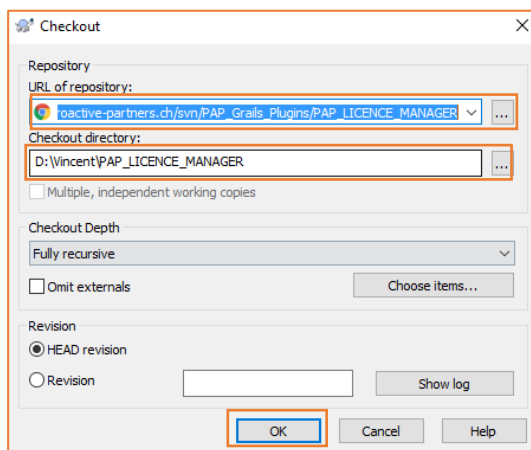
7.1. Mise en place de l'architecture

7.1.1. Mise en place de l'architecture via SVN

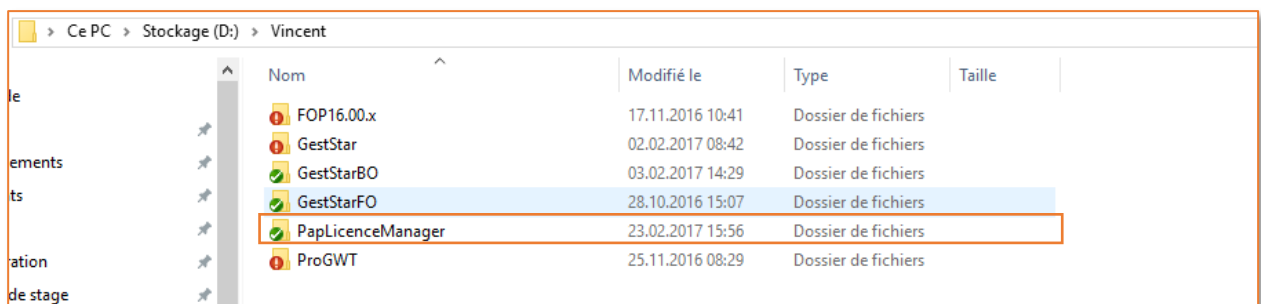
Le SVN a été précédemment configuré par les administrateurs architecture proactive. Une fois le SVN en place, il va falloir procéder à un « Checkout » à l'emplacement voulu à l'aide du programme Tortoise SVN afin de récupérer l'architecture mise en place.



Un pop-up apparaît. Il faut à présent entrer les informations du SVN.



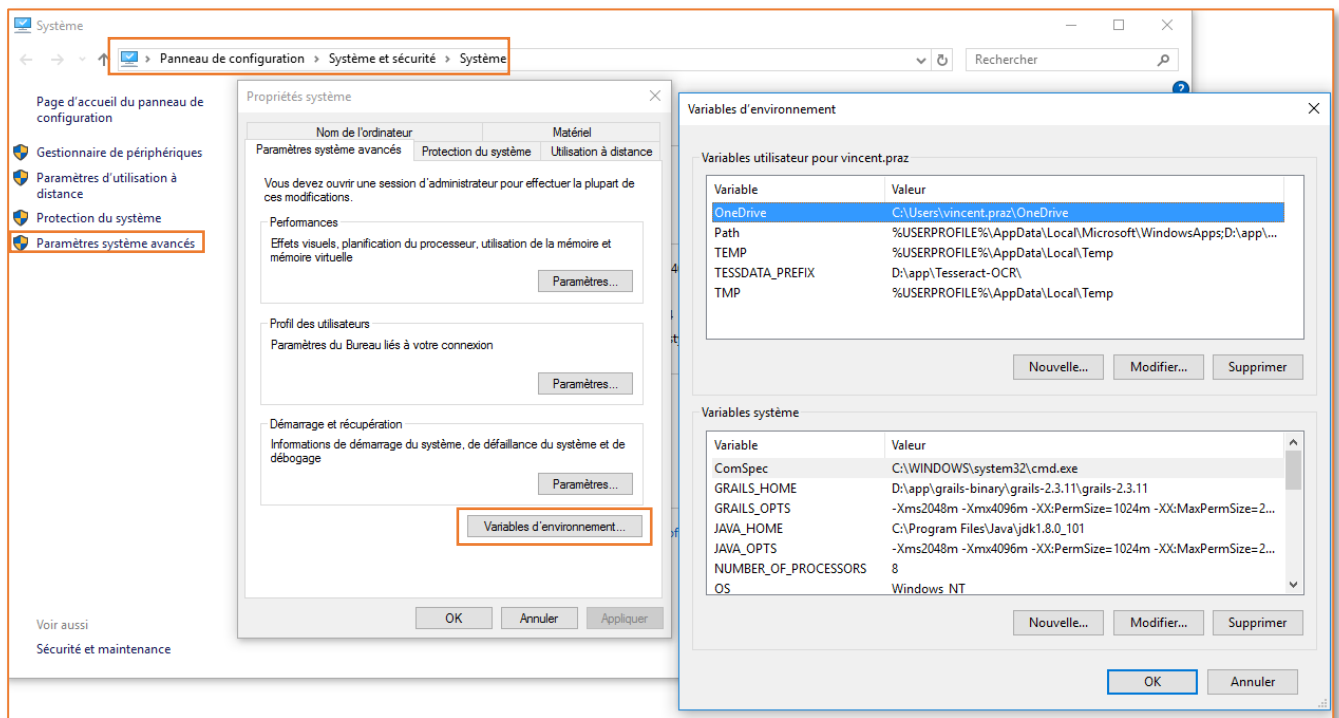
Après avoir entré les informations, l'architecture mise en place a été créée à cet emplacement.



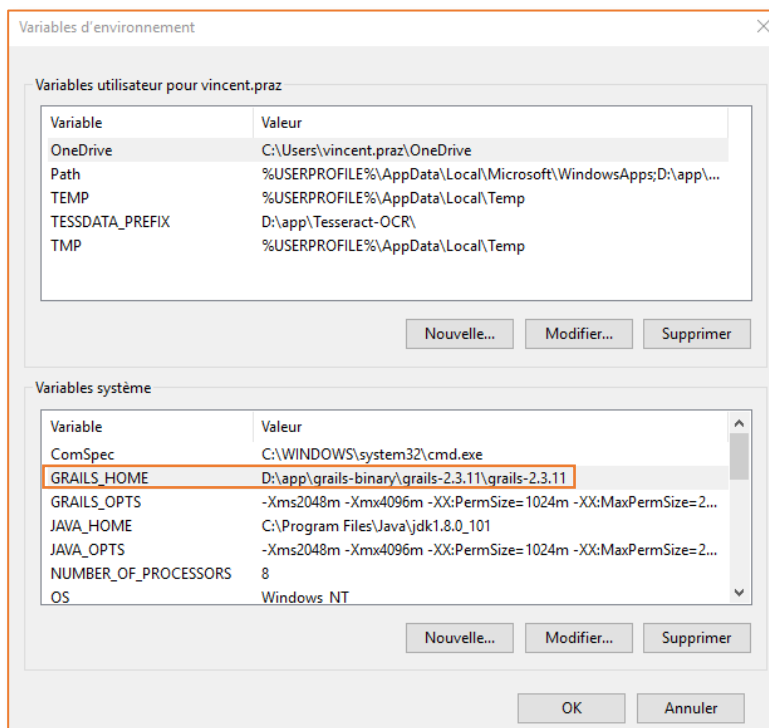
7.1.2. Variables d'environnement

Afin de créer le projet manuellement en ligne de commande, il va falloir configurer certaines variables d'environnement.

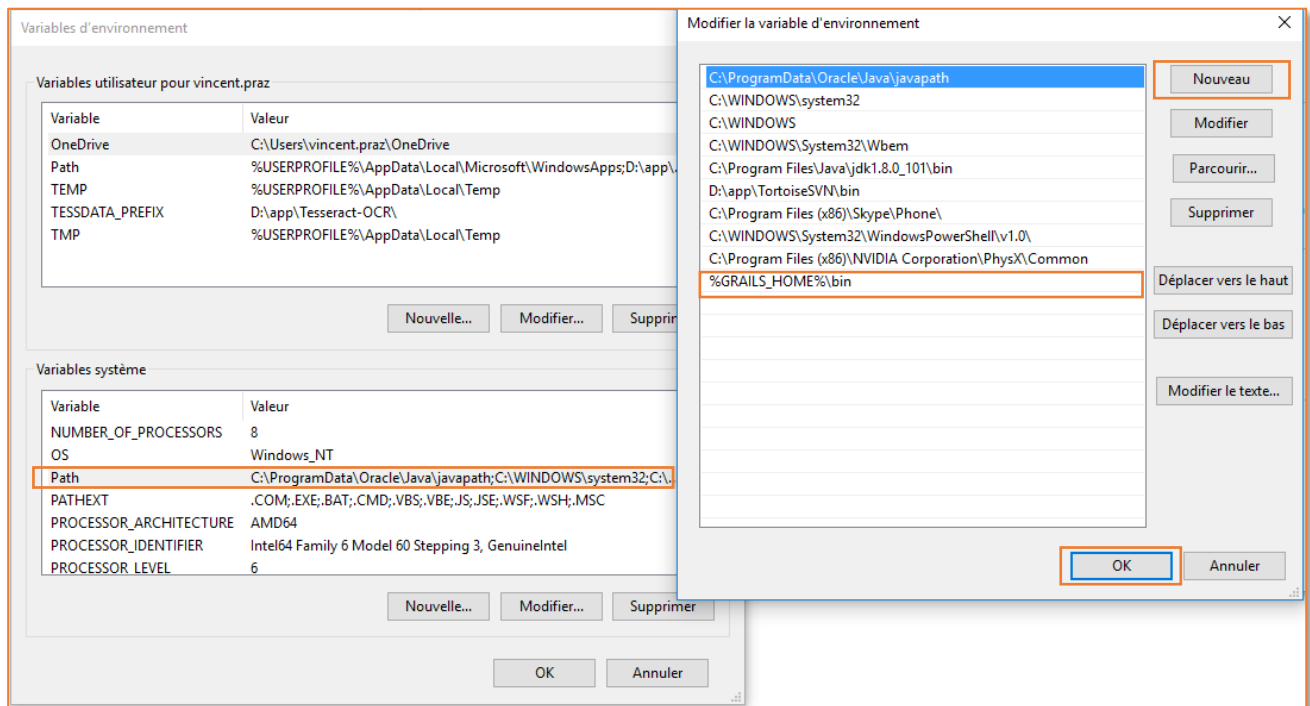
Pour ce faire, rendez-vous dans le panneau de configuration, Système et sécurité > Système > Paramètres Système avancés > Variables d'environnement...



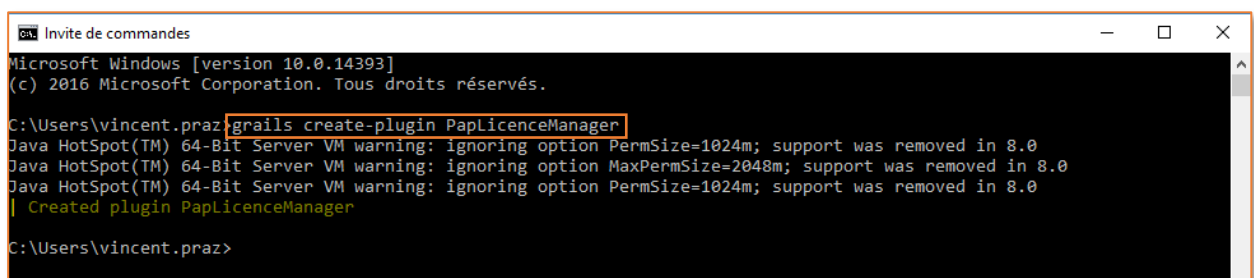
Une fois sur cette fenêtre, il faut compléter la valeur de la variable `GRAILS_HOME` qui sera le chemin vers le répertoire contenant la version de Grails avec laquelle sera lancée la commande.



Ensuite, il faudra ajouter une valeur à la variable Path qui définira l'emplacement des binaires de grails 2.3.11. Pour éviter d'avoir à la modifier à chaque changement du GRAILS_HOME, on passe cette variable en paramètre du chemin.



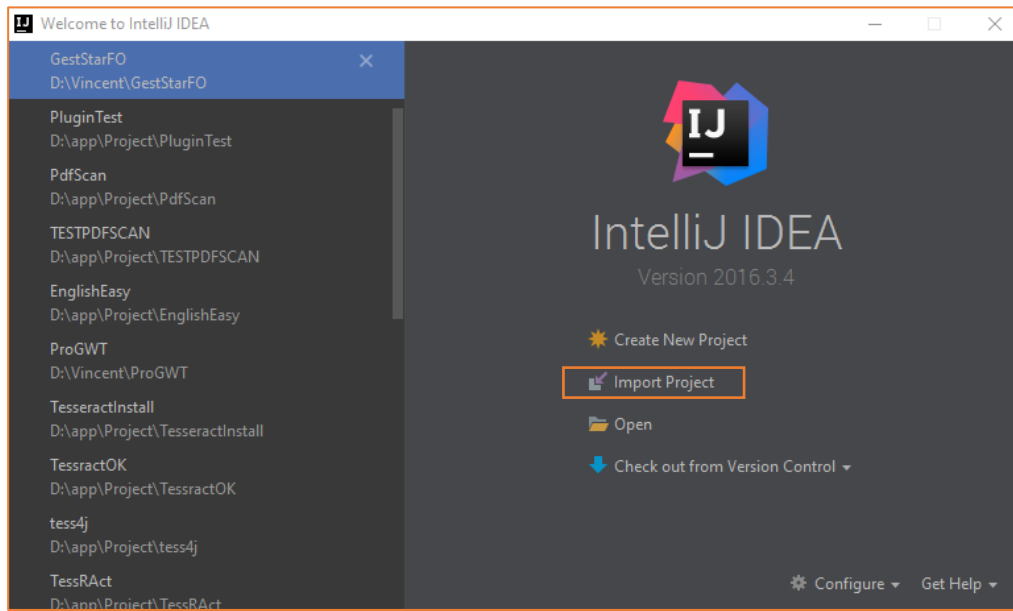
7.1.3. Création du plugin en ligne de commande



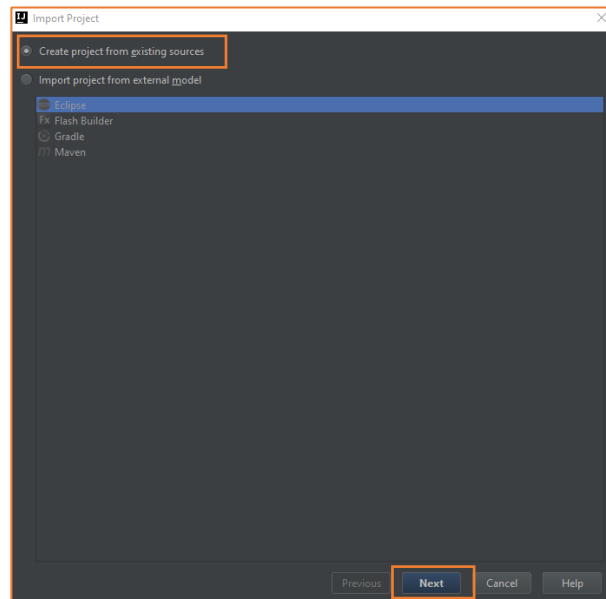
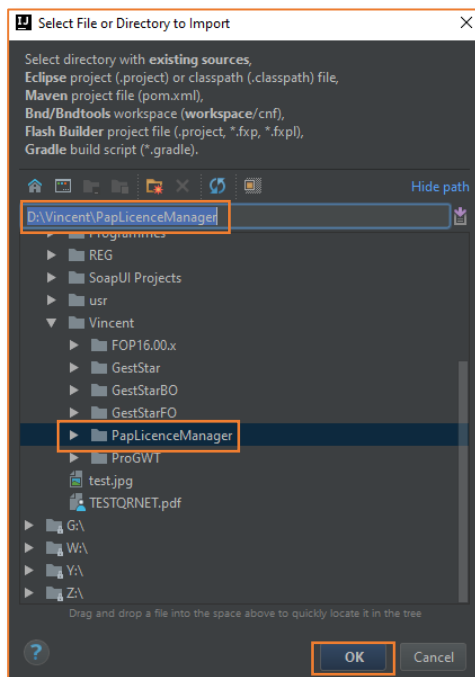
Créer le plugin à un emplacement sur le disque puis le déplacer à l'intérieur du répertoire reçu via SVN afin qu'il soit reconnu par celui-ci comme étant des nouveaux fichiers.

7.1.4. Importation dans IntelliJ

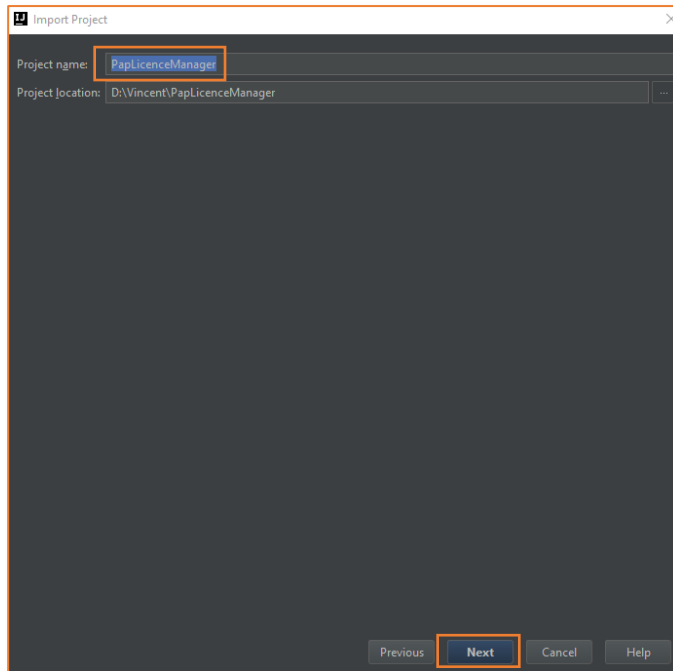
À présent, il va falloir importer notre plugin dans IntelliJ. Pour ce faire, lancez IntelliJ, fermez tous les projet en cours et cliquez sur import Project sur la page d'accueil.



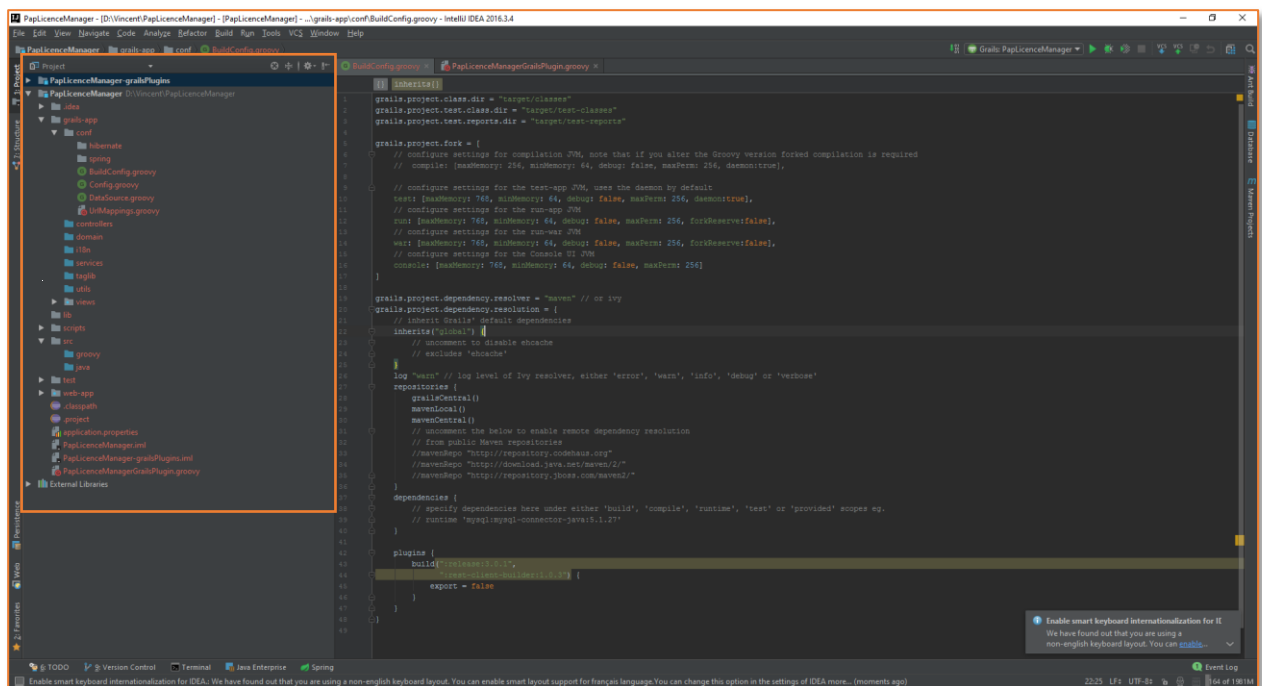
Sélectionnez le plugin précédemment créé puis 'create project from existing sources'.



Choisir le nom du projet.



Pour la suite, gardez les options sélectionnées de base et continuez l'importation avec 'Next'. À la fin de l'installation, IntelliJ ouvre le projet importé.



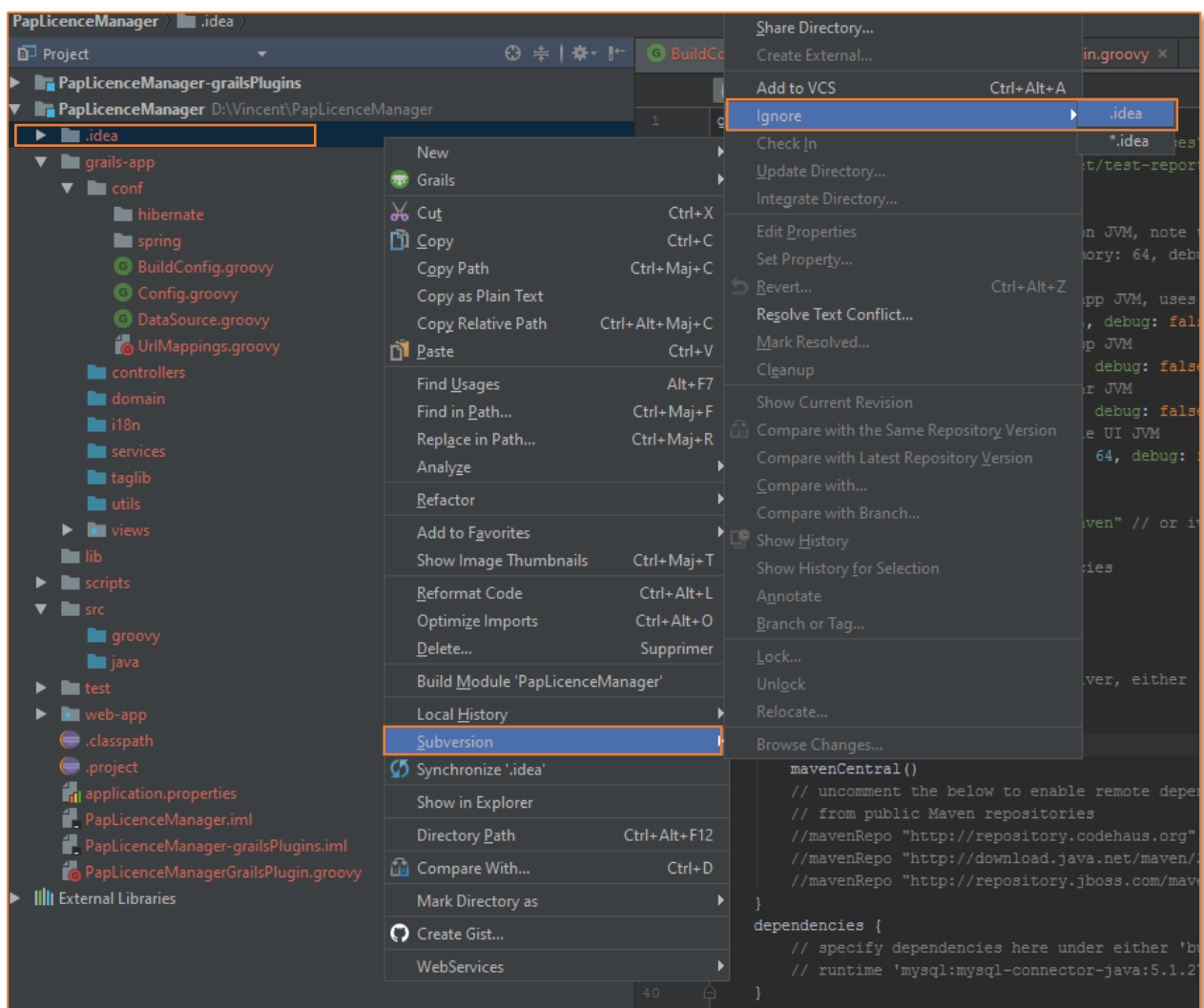
Les fichiers et répertoires sont écrits en rouge car ils n'ont pas été ajoutés à subversion.

7.1.5. Ajouts/Exclusions SVN

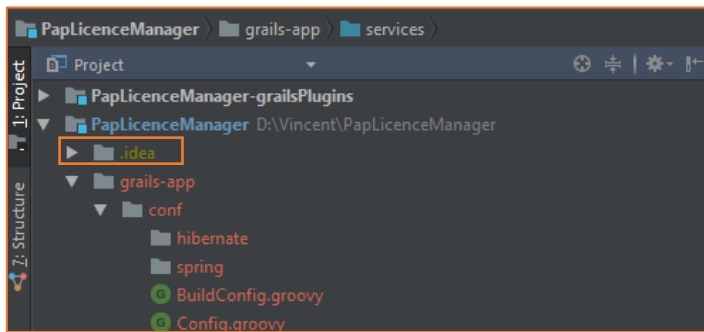
Pour pouvoir bien travailler, il va falloir ajouter ces fichiers à subversion. Cependant, certains fichiers comme les configurations personnelles ne devront être exclus. Voici les fichiers que nous allons exclure :

- Tous les fichiers avec l'extension '*.iml'
- Le répertoire '.idea'
- Le répertoire '.classpath'
- Le répertoire '.target' si existant
- Le répertoire '.project'
- Le répertoire '.slcache' si existant

Pour exclure un répertoire, clic gauche sur le dossier pour le sélectionner puis clic droit > subversion > ignore > choix de l'extension / répertoire

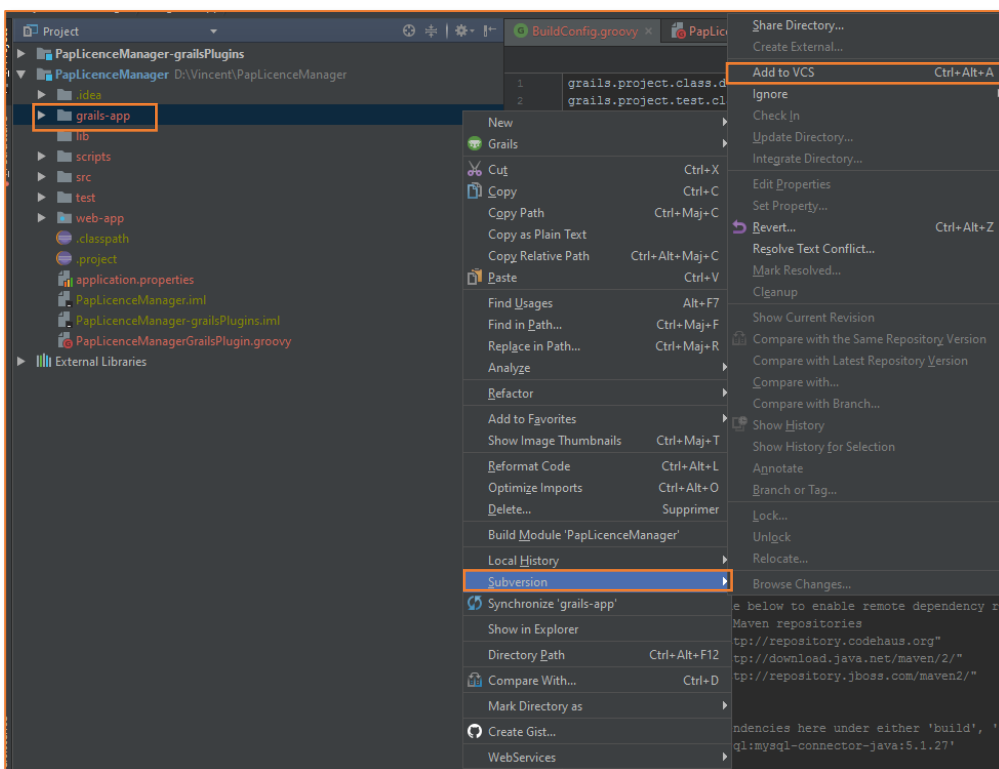


Une fois ajouté à la liste des répertoire ignorés, le répertoire prend une couleur verte.

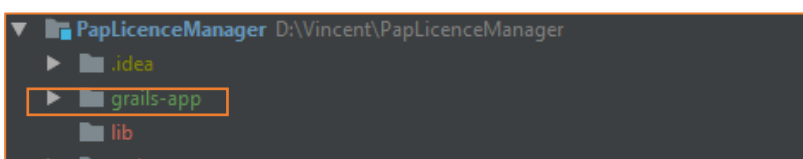


Appliquer ce procédé de façon identique pour les autres répertoires exclu en faisant attention de bien ignorer tous les fichier iml (*.iml').

À présent il faut ajouter tous les autres répertoires à subversion. Clic gauche pour sélectionner > clic droit > subversion > Add to VCS.

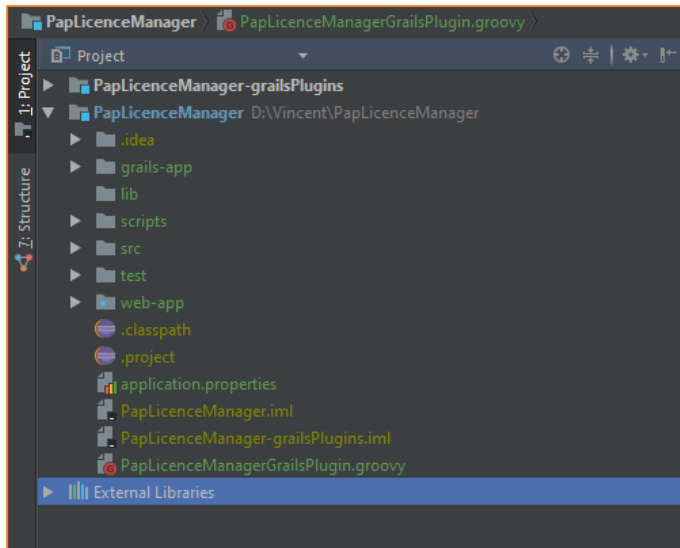


Un fichier ajouté à subversion apparait dans un couleur verte différente de ceux ignorés.



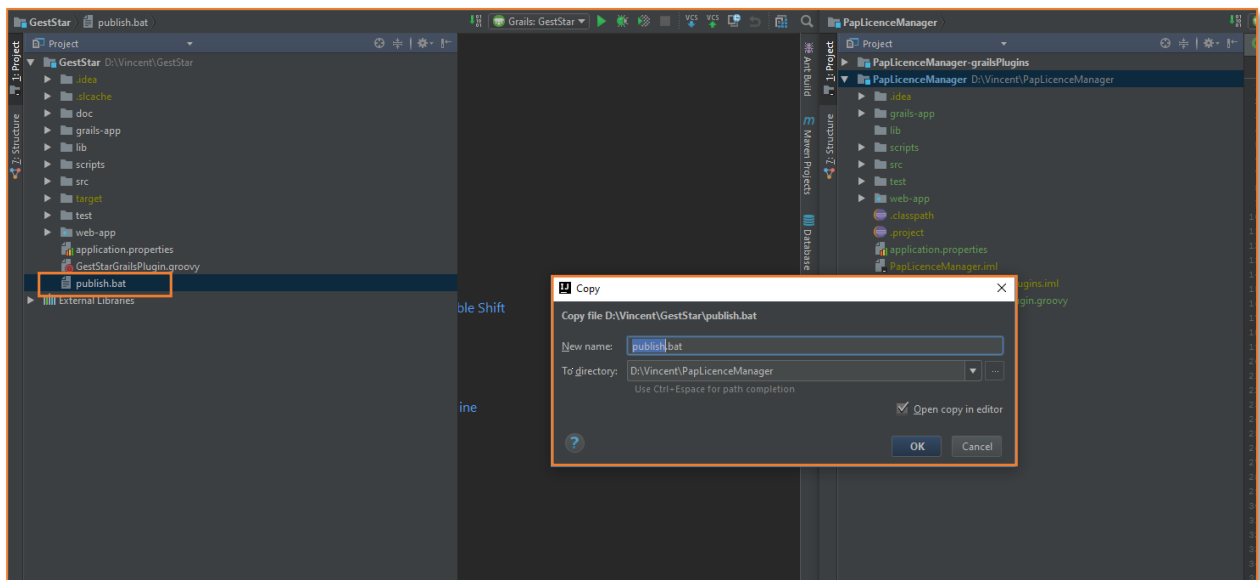
Répétez cette action pour tous les éléments du projet.

Résultat :

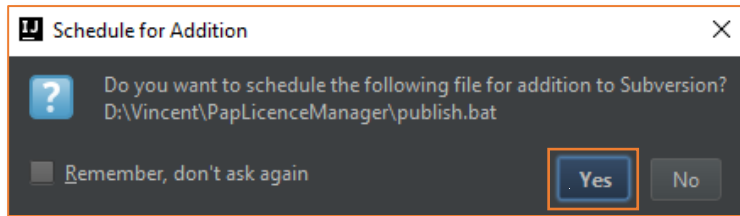


7.1.6. Récupération du fichier de publication

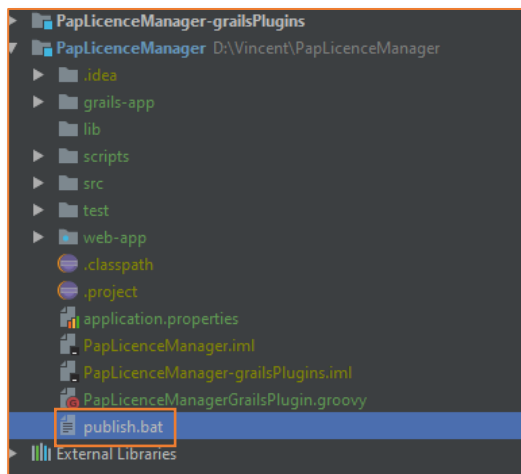
Proactive intègre un fichier de publication à chacun de ces projets. Il va servir à publier l'application. Il va donc falloir que j'aille chercher ce fichier dans une de leurs applications et que je l'intègre à mon projet.



Une fois le fichier copié, une fenêtre va demander si le fichier doit être ajouté à subversion 'Yes'.



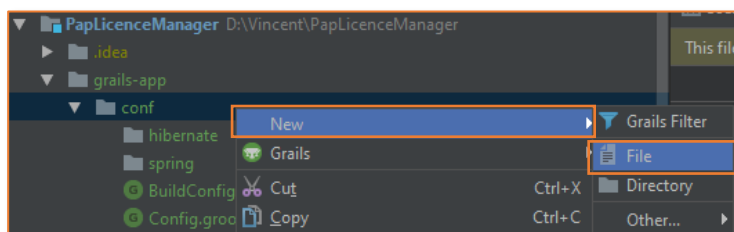
Le fichier se trouve à présent à la racine du projet.



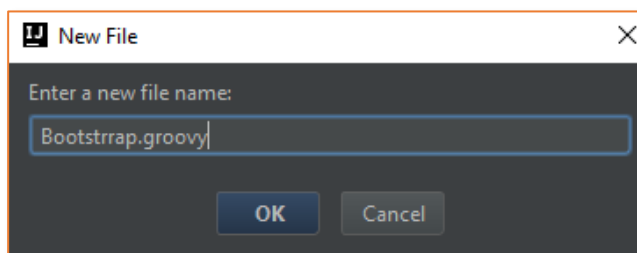
7.1.7. Création du fichier Bootstrap

Avec Grails, un fichier bootstrap est automatiquement lancé à chaque démarrage de l'application. Nous allons en avoir besoin pour vider/remplir la base de données ou pour effectuer des configurations au démarrage de l'application. Cependant il n'est pas créé par défaut, nous allons donc devoir le créer.

Aller dans `grails-app > conf` et créer un fichier



Il faudra le nommer tel quel puis l'ajouter à subversion.



Puis y ajouter le code suivant :

```
class Bootstrap {
    def init = { servletContext ->
        // code d'initialisation
    }
}
```

7.1.8. Adaptation de la configuration BuildConfig

Il va falloir à présent adapter la configuration de notre plugin pour qu'il puisse se lancer en mode applicatif. Autrement dit, il va falloir lui apporter ce dont il a besoin pour se lancer sans application liée.

Ouvrir et afficher le fichier BuildConfig.groovy. Dans la méthode repositories, ajouter 3 'mavenrepo' entre Grailscentral() et MavenLocal()

```
repositories {
    grailsCentral()

    // our and 3rd party plugins
    mavenRepo "http://svn.sierre.proactive-partners.ch:8081/artifactory/plugins-
release-local"
    // our code releases only
    mavenRepo "http://svn.sierre.proactive-partners.ch:8081/artifactory/libs-
release-local"
    // 3rd party releases only
    mavenRepo "http://svn.sierre.proactive-partners.ch:8081/artifactory/ext-
release-local"

    mavenLocal()
    mavenCentral()
    // uncomment the below to enable remote dependency resolution
    // from public Maven repositories
    //mavenRepo "http://repository.codehaus.org"
    //mavenRepo "http://download.java.net/maven/2/"
    //mavenRepo "http://repository.jboss.com/maven2/"
}
```

Pour la partie plugin, il va falloir ajouter les plugins

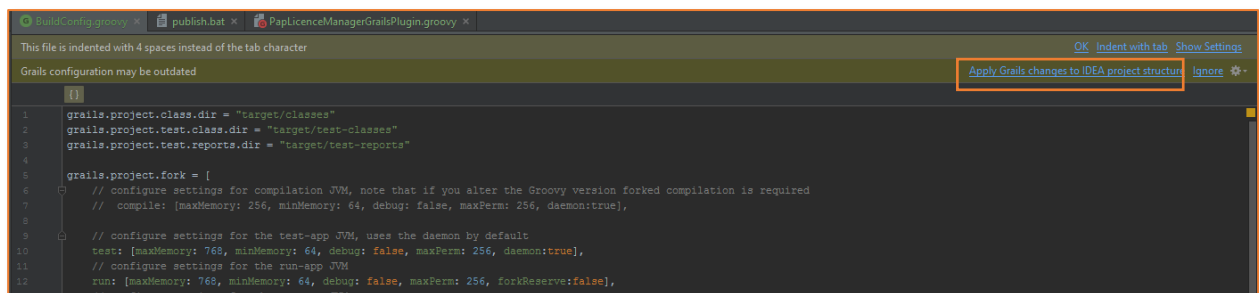
- Tomcat
- Hibernate
- Scaffolding
- Cache
- Cache-headers
- Cache-ehcache
- Scoped-proxy
- Pap-Crypto
- Sql-maintenance

```
plugins {  
    build(":release:3.0.1",  
        ":rest-client-builder:1.0.3") {  
        export = false  
    }  
  
    // plugins for the build system only  
    build ":tomcat:7.0.54"  
  
    // plugins needed at runtime but not for compilation  
    runtime (":hibernate:3.6.10.17") {  
        excludes "ehcache-core"  
    }  
  
    // plugins for the compile step  
    compile ":scaffolding:2.0.2"  
    compile ':cache:1.1.1'  
  
    // required plugin for cached-resources  
    runtime ":cache-headers:1.1.6"  
  
    // plugin for caching data  
    compile ":cache-ehcache:1.0.1"  
    compile ":scoped-proxy:0.2"  
  
    //Plugins Proactive  
    //compile ":sql-maintenance:16.6.0" Laisser commenté jusqu'à la création des  
    bases de données  
    compile ":pap-crypto:15.11.1"  
}
```

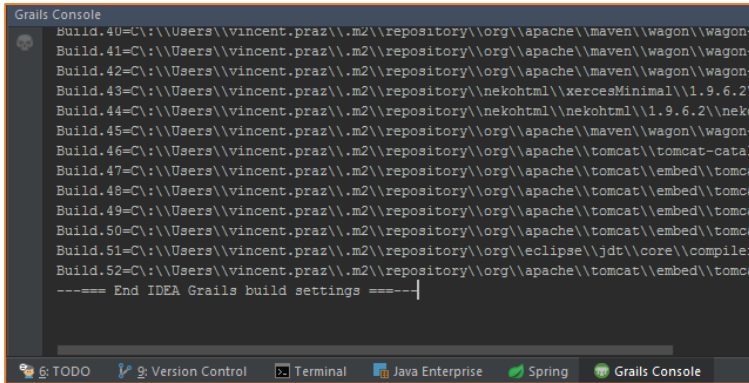
À la fin du fichier, il va falloir ajouter le repository proactive comme ceci :

```
grails.project.dependency.distribution = {  
    remoteRepository(id: "pap-plugins", url: "http://svn.sierre.proactive-  
partners.ch:8081/artifactory/plugins-release-local") {  
        authentication username: "grails", password: 'Pr0Clv$'  
    }  
}
```

Après avoir effectué ces modifications, un message indique que la configuration est 'outdated' il va falloir appliquer les changements à la structure du projet.



La console grails va s'ouvrir et installer les plugins et configurer les repositories.



7.1.9. Adaptation de PapLicenceManagerGrailsPlugin

Il va falloir changer les paramètres généraux du projet comme le nom de l'auteur, le titre, la version, etc. Pour changer ces paramètres, ouvrir le fichier PapLicenceManagerGrailsPlugin.groovy.

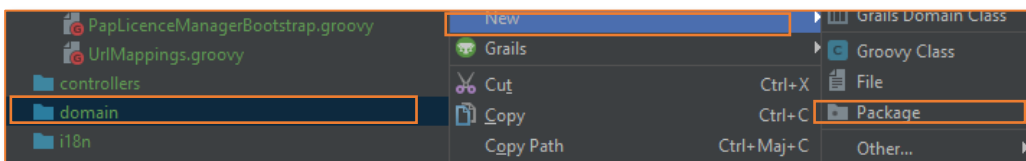
Modifiez les paramètres ci-dessous :

```
def title = "Pap Licence Manager Plugin" // Headline display name of the plugin
def author = "Vincent Praz"
def authorEmail = "vincent.praz@proactive.swiss"
def description = '''
Ce plugin fournit les méthodes nécessaires à la vérifications des licences
dans les applications proactives qui devront elles-mêmes utiliser ces méthodes
'''
```

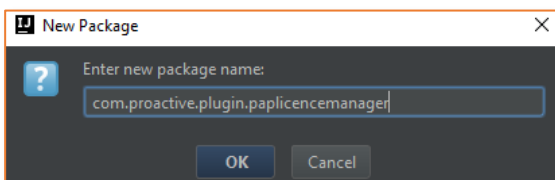
7.1.10. Test de la configuration

Pour tester la configuration, nous allons créer un domaine simple gérant les utilisateurs.

Création du package.

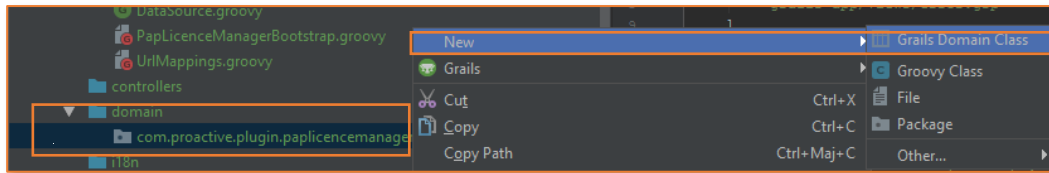


Voici la façon de nommer le package principal selon les standards de l'entreprise :

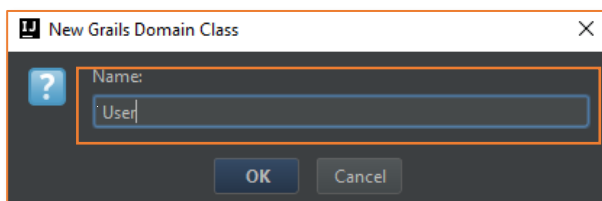


> ajouter à subversion

Création du domaine :



Choisir un nom pour le domaine : User



Ajouter 2 champs à l'utilisateur : login et password

```
package com.proactive.plugin.paplicencemanager

class User {

    String login
    String password

    static constraints = {

    }

}
```

Maintenant il faut créer un utilisateur au lancement de l'application pour avoir la possibilité de tester notre domaine. Pour cela, il va falloir le créer au lancement de l'application. Ouvrir le fichier PapLicenseManagerBootstrap.groovy précédemment créé et y ajouter le code nécessaire.

```
import com.proactive.plugin.paplicencemanager.User

class Bootstrap {
    def init = { servletContext ->
        // code d'initialisation
        def user = new User(login:"prav",password: "pass").save(failOnError: true)
        println(user.login)
    }
}
```

Une fois ce code ajouté, lancez l'application en mode Debug  et consultez les logs

```

Debug Grails: PapLicenceManager
Debugger Console
C:\Java\jdk1.7.0_67\bin\java ...
|Loading Grails 2.3.11
|Configuring classpath
févr. 24, 2017 10:40:58 AM java.util.prefs.WindowsPreferences <init>
AVERTISSEMENT: Could not open/create prefs root node Software\JavaSoft\Prefs at root 0x800000
.
|Environment set to development
.....
|Packaging Grails application
.....
|Compiling 1 source files
.....
|Running Grails application
Listening for transport dt_socket at address: 51773
Connected to the target VM, address: '127.0.0.1:51773', transport: 'socket'
prav
|Server running. Browse to http://localhost:8080/PapLicenceManager
  
```

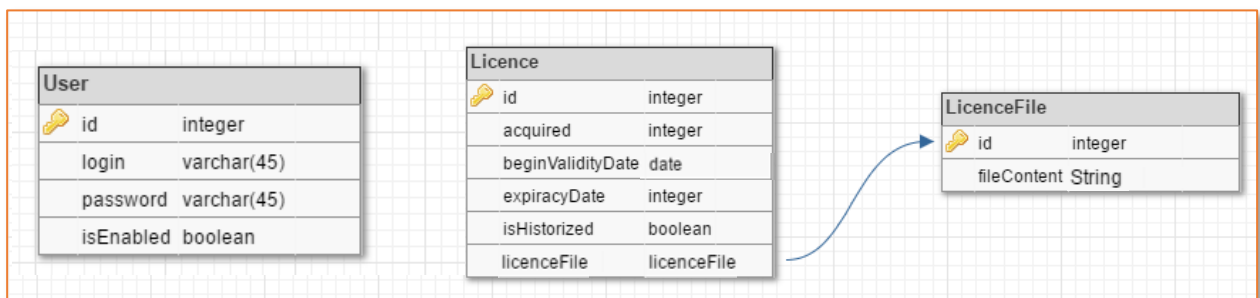
La configuration est donc fonctionnelle, maintenant effacez le domaine User et le code s’y rapportant pour pouvoir commencer le projet.

7.2. Domaines

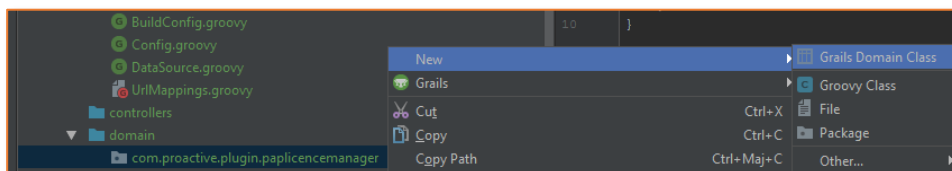
7.2.1. Création des classes de domaines

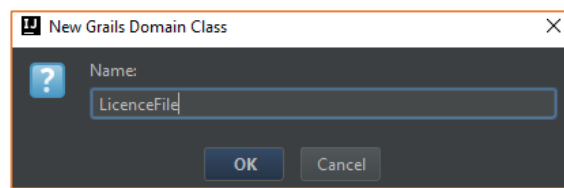
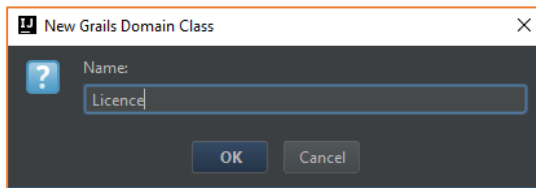
Avant de pouvoir débuter à développer l’interface, il va falloir créer les différents domaines nécessaires au démarrage du projet.

Voici les trois domaines représentés en tables dont nous avons besoin pour commencer le projet :

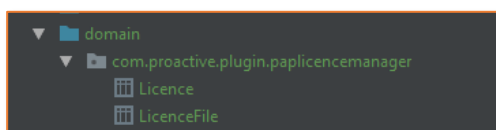
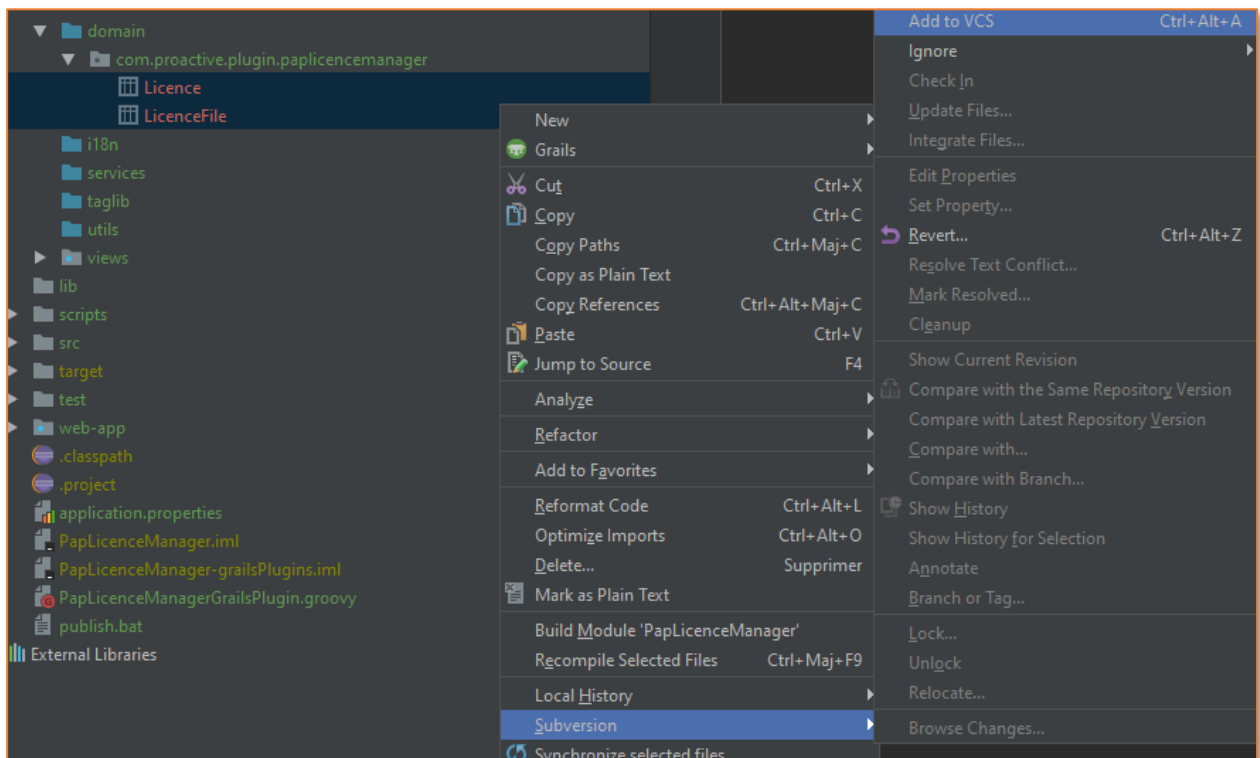


Créer chaque domaine et choisir leur nom (Licence, LicenceFile et User)



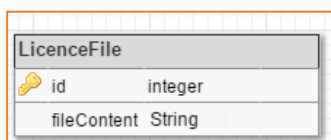


Ajouter ces deux domaines au VCS



7.2.2. Domaine LicenceFile

7.2.2.1. Aperçu



7.2.2.2. Description

Ce domaine doit contenir le contenu entier des fichiers de licence afin de pouvoir le parcourir et vérifier la validité des licences auxquelles il est lié.

7.2.2.3. Champs

LicenceFile		
1	id	integer
2	fileContent	String

Description	
1.	Clé primaire du domaine
2.	Contenu complet du fichier de licence

7.2.2.4. Déclaration

Les champs id sont gérés automatiquement par Grails il n'y a donc pas besoin de les créer dans notre classe. Les transformations entre les types dits plutôt 'Java' et les types 'SQL' sont elles aussi gérées par Grails.

Pour déclarer le champ voulu, ajoutez le code 'String fileContent'

```
package com.proactive.plugin.paplicencemanager

class LicenceFile {
    String fileContent //Contenu du fichier de licence

    static constraints = {
    }
}
```

7.2.2.5. Contraintes

Le contenu du fichier ne doit pas dépasser 4000 caractères, ne doit pas être nul et ne doit pas être vide.

```
static constraints = {
    fileContent maxSize: 4000, blank: false, nullable: false
}
```

7.2.3. Domaine Licence

7.2.3.1. Aperçu

Licence		
	id	integer
	acquired	integer
	beginValidityDate	date
	expiryDate	date
	isHistorized	boolean
	licenceFile	licenceFile

7.2.3.2. Description

Ce domaine va contenir les informations importantes pour reconnaître une licence active ou historisée.

7.2.3.3. Champs

Licence		
1	id	integer
2	acquired	integer
3	beginValidityDate	date
4	expiryDate	date
5	isHistorized	boolean
6	licenceFile	licenceFile

Description
1. Clé primaire du domaine
2. Nombre de licences acquises
3. Date de début de la validité de la licence
4. Date d'expiration de la licence
5. Valeur indiquant si la licence est historisée (true) ou non (false)
6. Clé étrangère liée au fichier de licence

7.2.3.4. Déclaration

La clé étrangère est de type LicenceFile. Grails gère les clés étrangères automatiquement si elles sont du type du domaine étranger.

Voici le code utile pour déclarer les champs de ce domaine :

```
package com.proactive.plugin.paplicencemanager

class Licence {

    int acquired           //Nombre de licences acquise
    Date beginValidityDate //Date de début de validité
    Date expiryDate        //Date d'expiration
    Boolean isHistorized    //Historisé t/f
    LicenceFile licenceFile //Fichier de licence lié

    static constraints = {
    }
}
```

7.2.3.5. Contraintes


Les contraintes sur les champs de ce domaine sont les suivantes :

- Le nombre de licences acquises ne peut pas être nul
- La date de début de validité de la licence ne peut pas être nulle
- La date d'expiration de la licence peut être nulle
- L'historisation ne peut pas être nulle
- Le fichier de licence peut être nul

```
static constraints = {
    acquired nullable: false
    beginValidityDate nullable: false
    expiryDate nullable: true
    isHistorized nullable: false
    licenceFile nullable: true
}
```

7.2.4. Domaine User


7.2.4.1. Aperçu

User		
	id	integer
	login	varchar(45)
	password	varchar(45)
	isEnabled	boolean

7.2.4.2. Description

Ce domaine va contenir les utilisateurs actifs ou non liés à la licence actuelle.

7.2.4.3. Champs

User		
	id	integer
	login	varchar(45)
	password	varchar(45)
	isEnabled	boolean

Description
1. Clé primaire du domaine
2. Login de l'utilisateur
3. Mot de passe de l'utilisateur
4. Activé ou non

7.2.4.4. Déclaration

```
package com.proactive.plugin.paplicencemanager

class User {

    String login        //login de l'utilisateur
    String password     //Mot de passe de l'utilisateur
    Boolean isEnabled   //variable déterminant si une licence est active sur cet utilisateur

    static constraints = {
    }
}
```

7.2.4.5. Contraintes

```
static constraints = {
    login    maxSize: 45, blank: false, nullable: false //Le login ne peut pas être vide, nul,
    dépasser 45 caractères
    password maxSize: 45, blank: false, nullable: false //Le mot de passe ne peut pas être vide,
    nul, dépasser 45 caractères
    isEnabled nullable: false //L'activation ne peut pas être nulle
}
```

7.2.5. Insertion de données tests

Afin de pouvoir poursuivre le projet, il est préférable d'insérer au préalable des données tests. Pour ce faire, nous allons à nouveau utiliser le fichier Bootstrap.groovy

7.2.5.1. Import

Afin de pouvoir utiliser nos classes dans le fichier Bootstrap.Groovy, nous allons importer le package ainsi que son contenu correspondant.

```
import com.proactive.plugin.paplicencemanager.Licence
import com.proactive.plugin.paplicencemanager.LicenceFile
import com.proactive.plugin.paplicencemanager.User
```

7.2.5.2. Données tests

À présent, il faut ajouter les licences et fichiers de licences à la méthode init de notre fichier BootStrap.

```
def init = { servletContext ->
  // code d'initialisation

  //Fichier de Licences
  def licenceFile = new LicenceFile(fileContent: "Contenu du fichier de
licence").save(failOnError: true)

  //Licences

  //Licence invalide sans fichier lié
  def licenceInvalidWithoutFile = new Licence(acquired: 5, beginValidityDate: new
GregorianCalendar(2013, Calendar.FEBRUARY, 21).time, expiryDate: new GregorianCalendar(2014,
Calendar.MARCH, 14).time, isHistorized: true).save(failOnError: true)

  //Licence valide avec fichier lié
  def licenceValidWithFile = new Licence(acquired: 3, beginValidityDate: new
GregorianCalendar(2014, Calendar.FEBRUARY, 24).time, expiryDate: new GregorianCalendar(2020,
Calendar.DECEMBER, 11).time, isHistorized: false, licenceFile: licenceFile).save(failOnError:
true)

  //Licence invalide avec fichier lié
  def licenceInvalidWithFile = new Licence(acquired: 25, beginValidityDate: new
GregorianCalendar(2015, Calendar.JANUARY, 01).time, expiryDate: new GregorianCalendar(2015,
Calendar.FEBRUARY, 04).time, isHistorized: true, licenceFile: licenceFile).save(failOnError:
true)

  def userWithLicenceEnabled = new User(login: "prav",password: "123",isEnabled:
true).save(failOnError: true)

  def userWithLicenceDisabled = new User(login: "Philippe", password: "strongestPass",isEnabled:
false).save(failOnError: true)
```

7.2.5.3. Affichage des données tests

Pour s'assurer de la qualité des données tests, nous allons ajouter le code suivant pour afficher les licences et fichiers de licences ainsi que la valeur de leurs champs :

```
println("Affichage des données tests")

println('----- LICENCES -----')

Licence.getAll().each {
  println('-----')
  println("id : " + it.id)
  println('-----')
  println("acquired : " + it.acquired)
  println("beginValidityDate : " + it.beginValidityDate)
  println("expiryDate : " + it.expiryDate)
  println("isHistorized : " + it.isHistorized)
}

println('----- FICHIERS DE LICENCES -----')

LicenceFile.getAll().each {
  println('-----')
  println("id : " + it.id)
```

```
println('-----')
println("fileContent : " + it.fileContent)
}

println('----- Users -----')

User.getAll().each {
    println('-----')
    println("id : " + it.id)
    println('-----')
    println("login : " + it.login)
    println("isEnabled : " + it.isEnabled)
}
}
```

Lancer l'application en mode Debug et consulter les logs :

```
|Running Grails application
Listening for transport dt_socket at address: 58015
Connected to the target VM, address: '127.0.0.1:58015', transport: 'socket'
Affichage des données tests
----- LICENCES -----
-----
id : 1
-----
acquired : 5
beginValidityDate : Thu Feb 21 00:00:00 CET 2013
expiryDate : Fri Mar 14 00:00:00 CET 2014
isHistorized : true
-----
id : 2
-----
acquired : 3
beginValidityDate : Mon Feb 24 00:00:00 CET 2014
expiryDate : Fri Dec 11 00:00:00 CET 2020
isHistorized : false
-----
id : 3
-----
acquired : 25
beginValidityDate : Mon Jun 01 00:00:00 CEST 2015
expiryDate : Wed Feb 04 00:00:00 CET 2015
isHistorized : true
----- FICHIERS DE LICENCES -----
-----
id : 1
-----
fileContent : Contenu du fichier de licence
----- Users -----
-----
id : 1
-----
login : praz
isEnabled : true
-----
id : 2
-----
login : Philippe
isEnabled : false
|Server running. Browse to http://localhost:8080/PapLicenceManager
```

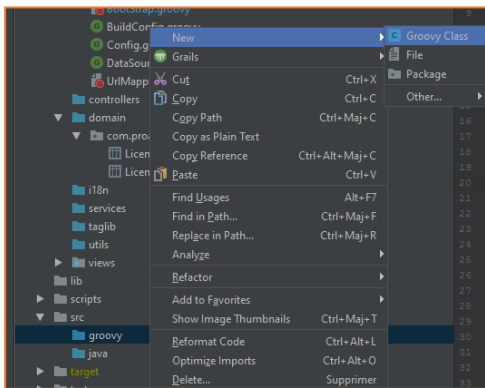
7.3. Développement de l'interface Java

7.3.1. Description

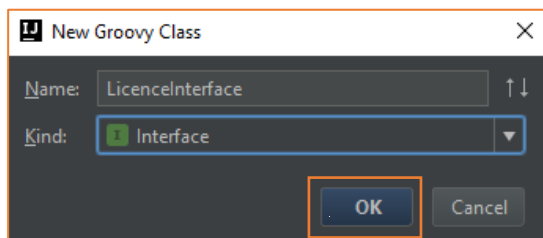
Cette interface sert à fournir les méthodes nécessaires au LicenceService ainsi qu'au Service de l'application gérant les objets qui consommeront la licence.

7.3.2. Création de l'interface

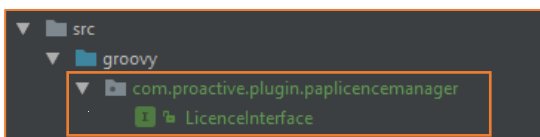
Dans l'architecture de fichiers, l'interface doit être créée dans PapLicenceManager\src\groovy.



Entrez le titre puis choisir Interface dans la liste déroulante.



L'interface a été créée et est prête à être codée. Créez un package et placez-la à l'intérieur de celui-ci.



7.3.3. Méthodes

Dans notre interface, nous aurons besoin de créer la méthode '*usedLicence*' qui sera réutilisée dans les services.

```
/**
 * Méthode retournant le nombre de licences utilisées après
 * calcul de cette valeur
 * @return Retourne le nombre de licences utilisées
 */
int usedLicence() //Affiche le nombre de licences utilisées
```

7.4. LicenceInfo

7.4.1. Description

Pour faciliter l'affichage et la récupération des informations principales d'une licence, il faut créer une classe LicenceInfo permettant de créer des objets du même nom.

7.4.2. Création de la classe

Cette classe se situe dans l'architecture de fichier sous
PapLicenceManager\src\groovy\com\proactive\plugin\paplicencemanager

Clic droit > New > Groovy Class > Nom : LicenceInfo

7.4.3. Code

Voici le code nécessaire à la création de cette classe.

```
package com.proactive.plugin.paplicencemanager

/**
 * Created by vincent.praz on 03.03.2017.
 */
class LicenceInfo {
    Licence activeLicence //Objet Licence contenant la licence active
    boolean isLicenceValid //boolean contenant la validité de la licence (valide/invalid)
    int usedLicence //Le nombre de licences utilisées
    int remainingLicence //Le Nombre de licences restantes
}
```

7.5. Services

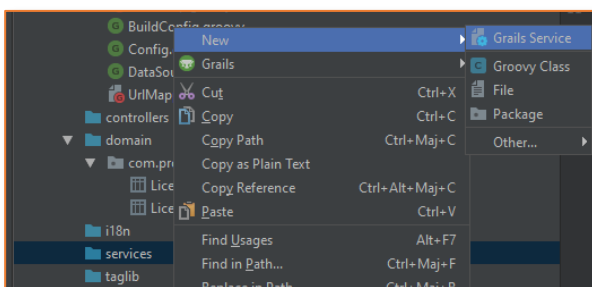
7.5.1. LicenceService

7.5.1.1. Description

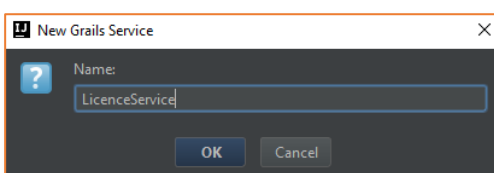
Ce service utilise l'interface LicenceInterface afin de fournir à l'application les données concernant les licences.

7.5.1.2. Création du service

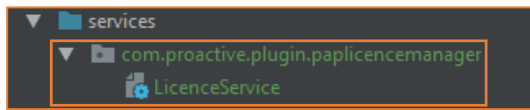
Dans l'architecture de fichiers, le service doit être créé dans PapLicenceManager\grails-app\services.



Nous allons nommer ce service LicenceService.



Créez un package et placez le service dans celui-ci.



7.5.1.3. Méthodes

Dans notre service, nous allons réécrire les méthodes de l'interface en leur donnant une valeur de retour afin de pouvoir les mettre à disposition des applications. Des méthodes supplémentaires sont ajoutées pour la gestion des licences.

Vérification de la validité des licences :

```
/**
 * Permet de vérifier si la licence actuelle (Licence non historisée) est valide à
 * l'heure actuelle
 * selon différent critères
 * @return <code>true</code> si la licence actuelle est valide et <code>false</code> si
 * elle est invalide
 */

boolean isLicenceValid() {
    Licence licence
    try {
        //Récupération de la licence actuelle
        licence = Licence.findWhere(isHistorized: false)
    } catch (Exception e) {
        log.error(e)
        return false
    }

    //S'il n'y a aucune licence
    if (!licence) {
        return false
    }

    // On vérifie si le nombre de licences restantes n'est pas négatif
    // Que le nombre de licences utilisées est plus petit ou égal au nombre de licences
    acquises
    // Que la date de début de validité est antérieure ou égale à la date du jour
    // Que la date d'expiration est postérieur ou égale à la date du jour
    Date now = new Date()
    if (remainingLicence() >= 0
        && usedLicence() <= licence.acquired
        && licence.beginValidityDate <= now
        && licence.expiryDate >= now) {
        //La licence est valide
        return true
    }
    return false
}
```

Création du contenu XML encodé :

```
/**
 * Créer un fichier de licence en xml encodé à partir des champs de créations
 * @param acquired
 * @param beginValidityDate
 * @param expiryDate
```

```

* @return content.encodeAsBlowfish ( ) Licence en XML encodé
*/
String createLicenceXmlEncoded(String acquired, Date beginValidityDate, Date expiryDate) {
    //Création d'une licence
    Licence licence = new Licence(acquired: acquired, beginValidityDate: beginValidityDate,
    expiryDate: expiryDate)
    //Conversion de la licence au format XML
    String content = licence as XML
    log.debug(content)
    //Retourne la licence au format XML encodé
    return content.encodeAsBlowfish()
}

```

Création d'une licence à partir d'un fichier uploadé :

```

/**
 * Permet de créer une licence à partir d'un fichier de licence uploadé
 * Récupère le contenu et les champs du fichier XML et les attribue à
 * une nouvelle licence pour finalement la créer avec la méthode updateLicence()
 * Crée également un LicenceFile avec le contenu du fichier uploadé
 * @param uploadedFile Fichier CommonsMultipart uploadé contenant la licence
 * @see LicenceService#updateLicence
 */
void uploadLicenceFile(CommonsMultipartFile uploadedFile) {
    // Permet de récupérer le contenu du fichier de licence et de créer une Licence et un fichier
    de licence

    //Valeurs récupérées du fichier de licence
    String licenceFileContent //Contenu du fichier de licence
    int acquired //nbrLicence acquise
    Date beginValidityDate //Date de début de validité
    Date expiryDate //Date d'expiration

    //Récupération du contenu du fichier de licence
    if (uploadedFile != null && !uploadedFile.isEmpty()) {
        ByteArrayInputStream stream = new ByteArrayInputStream(uploadedFile.getBytes())
        licenceFileContent = IOUtils.toString(stream, "UTF-8")
        log.debug(licenceFileContent)
    } else {
        log.error("Aucun fichier de licence")
    }

    //Lecture du contenu du fichier de licence
    if (licenceFileContent != null) {
        try {
            //Décodage
            String licenceFileContentDecoded = licenceFileContent.decodeBlowfish()
            log.debug(licenceFileContentDecoded)

            //Lecture du contenu du String
            def rootNode = XML.parse(licenceFileContentDecoded)

            //Attribution du contenu du fichier de licence aux variables
            acquired = Integer.parseInt(rootNode.acquired.text())
            beginValidityDate = new SimpleDateFormat('yyyy-MM-dd').parse(rootNode.beginValidityDate.text())
            expiryDate = new SimpleDateFormat('yyyy-MM-dd').parse(rootNode.expiryDate.text())

            //Affichage du contenu récupéré
            log.debug("Licences acquises : " + acquired.toString())
            log.debug("Début validité : " + beginValidityDate.toString())
            log.debug("Expiration : " + expiryDate.toString())

            //Création du fichier de licence et assignation de son contenu
            LicenceFile licenceFile = new LicenceFile()
            licenceFile.setFileContent(licenceFileContent)
            if (!licenceFile.save(flush: true, failOnError: true)) {
                log.error("Impossible de sauvegarder le fichier de licence")
            }
        } catch (Exception e) {
            log.error("Erreur lors de la création de la licence : " + e.getMessage())
        }
    }
}

```



```

    } else {
        log.debug("Le fichier de licence a bien été créé")
        //Historisation de la ou des licence(s) active(s)
        historizeAll()

        //Création de la nouvelle licence à partir des valeurs du fichier de licence
        Licence licence = new Licence()
        licence.setAcquired(acquired)
        licence.setLicenceFile(licenceFile)
        licence.setBeginValidityDate(beginValidityDate)
        licence.setExpiryDate(expiracyDate)
        licence.setIsHistorized(false)

        //Validation et sauvegarde de la licence
        if (!updateLicence(licence)) {
            log.error("Impossible de sauvegarder la nouvelle licence")
        } else {
            log.debug("Licence ajoutée à partir du fichier de licence")
        }
    }
} catch (Exception e) {
    log.error("Une erreur est survenue lors de l'upload du fichier de licence : " + e)
    log.error("Veuillez vérifier le format et le contenu du fichier de licence")
}
} else {
    log.error("Le fichier de licence n'as pas pu être lu ou est vide")
}
}

```

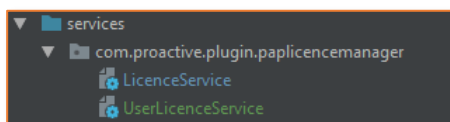
7.5.2. UserLicenceService

7.5.2.1. Description

Ce service gère les utilisateurs qui vont consommer les licences. Il doit implémenter l'interface LicenceInterface.

7.5.2.2. Création du service

La création se fait de façon identique au service précédent.



7.5.2.3. Méthodes

La méthode 'usedLicence' va devoir être écrite pour le 'UserLicenceService'. Elle va effectuer une requête et renvoyer le nombre d'utilisateurs actifs afin de pouvoir connaître le nombre de licences consommées.

```

/**
 * Retourne le nombre d'utilisateurs utilisant la licence (actifs)
 * @return User.countByIsEnabled ( true ) le nombre d'utilisateurs actifs
 */
@Override
int usedLicence() {
    return User.countByIsEnabled(true)
}

```

7.5.2.4. Injection

Afin que le `LicenceService` puisse reconnaître et utiliser ce service, il va falloir lui faire une injection Spring. Créez un fichier nommé `'resources.groovy'` dans le répertoire Spring et ajoutez-y le code suivant :

```
import com.proactive.plugin.paplicencemanager.LicenceService

/**
 * Created by vincent.praz on 03.03.2017.
 */

beans = {
    //Licence Service
    licenceService(LicenceService) {
        licenceInterface = ref('userLicenceService')
    }
}
```

Il faut également ajouter ce code au fichier `PapLicenceManagerGrailsPlugin.groovy` dans la méthode `doWithSpring` afin que l'injection s'effectue lorsque l'on exécute le plugin en 'mode plugin'. La méthode `doWithSpring` s'exécute au lancement de l'application lorsque le programme va charger Spring.

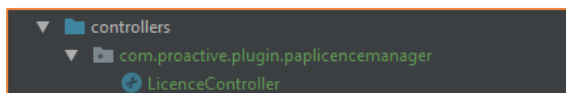
```
def doWithSpring = {
    //Licence Service
    println "Loading licence service"
    licenceService(LicenceService) {
        licenceInterface = ref('userLicenceService')
    }
}
```

7.6. Écrans

7.6.1. Contrôleur

7.6.1.1. Création du contrôleur

Dans `controllers PapLicenceManager\grails-app\controllers`, créez un package et créez-y un contrôleur nommé `LicenceController`



7.6.1.2. Exécution automatique du contrôleur

De façon à ce que le contrôleur s'exécute automatiquement au démarrage de l'application, il va falloir modifier les 'mappings' URL. Pour ce faire, ouvrez le fichier `'UrlMappings.groovy'`, commentez la ligne lançant la vue `'/index/'` et ajoutez le lien vers le contrôleur.

```
class UrlMappings {

    static mappings = {
        "/*controller/*action?/*id?(*.format)?" {
            constraints {
                // apply constraints here
            }
        }
    }
}
```

```

    }

    //"/"(view:"/index") à commenter pour empêcher la vue par défaut de s'ouvrir
    "500"(view: '/error')

    "/"(controller: 'licence') //Liens vers le contrôleur licence
}

```

7.6.1.3. Méthodes

Le contrôleur contient les méthodes qui vont interagir entre la vue et le service.

Voici la méthode qui va créer un fichier de licence et permettre de le télécharger directement à l'écran.

En premier lieu, on récupère les paramètres du formulaire de la vue. Puis le *LicenceService* est appelé afin de créer un String contenant le xml encodé correspondant aux champs entrés. Et finalement, le contrôleur envoie les informations à la page afin que le fichier apparaisse en download.

```

def createLicenceFile() {
    //Récupération des paramètres
    String acquired = params.acquired
    Date beginValidityDate = params.beginValidityDate
    Date expiryDate = params.expiryDate

    //Création du contenu de licence en xml et encodé blowfish
    String xmlEncoded = licenceService.createLicenceXmlEncoded(acquired,
beginValidityDate, expiryDate)

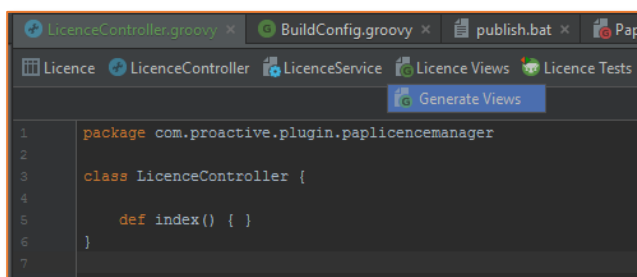
    //Lancement download
    //Envoi à la page des informations du fichier et download
    response.setHeader("Content-Disposition", "attachment; filename=licenceFile.xml")
    response.setContentType("text/xml")
    OutputStream outputStream = response.getOutputStream()
    outputStream.write(xmlEncoded.getBytes())
    outputStream.flush()
    outputStream.close()
}

```

7.6.2. Vue

7.6.2.1. Création de la vue

Une fois le contrôleur créé, ouvrez celui-ci, cliquez sur Licence views > Generate views.



Les vues par défaut de grails sont générées



7.6.2.2. Index

Nous avons décidé de concentrer la démonstration de façon à ce que toutes les informations soient présentées sur 1 seule page.

The screenshot shows the LicenceManager web application. The interface includes a header, a main content area with several panels, and a footer. The panels are annotated with numbers 1 through 6:

- 1**: A table listing all licenses with columns: Acquired, Begin Validity Date, Expiry Date, Is Historized, Licence File.
- 2**: A panel for modifying a selected license, showing fields for Acquired, Begin Validity Date, Expiry Date, and Is Historized.
- 3**: A panel showing information about the license in progress, including fields for Id, Valid, Acquired, Used, Remaining, Begin Validity Date, and Expiry.
- 4**: A panel for managing users, showing a table with columns: Login, Password, isEnabled, and buttons for Disable and Ajouter.
- 5**: An upload panel for creating a new license file, with a file selection button and an Upload button.
- 6**: A panel for creating a license file encoded in Blow Fish, with fields for Acquired, Begin validity, and Expiry, and an Encoder button.

1. Liste de toutes les licences
2. Modification de le licence sélectionnée à l'aide du lien dans la liste
3. Information sur la licence en cours (non historisée)
4. Liste des utilisateurs avec possibilité de création et activation/désactivation
5. Upload d'un fichier de licence afin de créer une licence supplémentaire
6. Création d'un fichier de licence encodé en Blow Fish

Voici le code nécessaire à l'affichage de la liste des licences :

```
<tbody>

  <g:set var="licenceList" value="${Licence.getAll()}" />

  <g:each in="${licenceList}" status="i" var="licence">
    <tr class="${(i % 2) == 0 ? 'even' : 'odd'}">

      <td><g:link action="select" id="${licence.id}">${fieldValue(bean: licence, field:
"acquired")}</g:link></td>

      <td><g:formatDate format="dd-MM-yyyy" date="${licence.beginValidityDate}" /></td>

      <td><g:formatDate format="dd-MM-yyyy" date="${licence.expiryDate}" /></td>

      <td><g:formatBoolean boolean="${licence.isHistorized}" /></td>

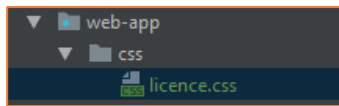
      <td>${fieldValue(bean: licence, field: "licenceFile.id")}</td>

    </tr>
  </g:each>
</tbody>
```

7.6.3. CSS

7.6.3.1. Création et liaison du CSS

Le fichier CSS va devoir être créé dans PapLicenceManager\web-app\css et se nommera licence.css



Pour lier le CSS à notre vue, il suffit d'y ajouter la ligne suivante :

```
<link rel="stylesheet" href="{resource(dir:'css',file:'licence.css')}" type="text/css" />
```

7.7. Fichier de licence

7.7.1. XML

Voici un aperçu du fichier de licence non-crypté au format XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<Licence>
  <acquired>20</acquired>
  <beginValidityDate>2017-03-09</beginValidityDate>
  <expiryDate>2020-03-09</expiryDate>
</Licence>
```

7.7.2. XML encodé en Blow Fish

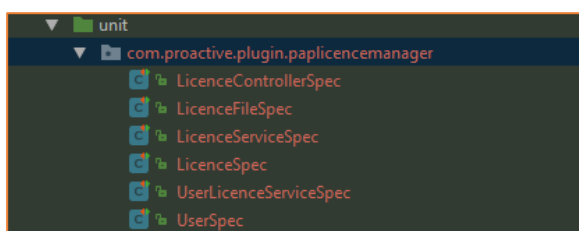
Voici le contenu du même fichier encodé en Blow Fish

```
PrGSrACwTX/eoJ/VqPSBoicfWZTDcXl50k0fEy6n0OxXRNzIcYu6X/PMGKuUp+9+F5/YWTRl80TzR0VzoCK
mnq12K7h0G2LThTf8q5oMx+0Rj0MtSUjdr2u9SGwdcSQq/n4UfsLfECRLt1BIsOTOBfrd3odPA7x4qOzI0f
+D/0sdySgh5GiHp4RfWr0yrvyYdICaAqxKCehXD+ziwk8F3QmkGMbrR2pUFaL91noJG1WEX1q9Mq78mPrLu
CTa0ahbWl80Dat7uxILbyoQ/TC7K8bwQdAOc0a12dGRmXk6S30Y/qwCOJnjug==
```

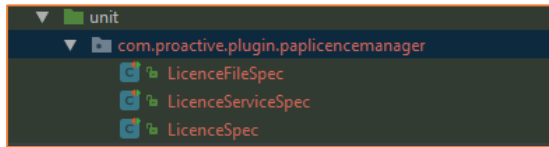
7.8. Tests Unitaires

Les tests unitaires sont primordiaux lors de la création d'une application afin de valider le code actuel et les modifications futures de celui-ci.

Par défaut IntelliJ va créer les classes permettant de configurer les tests unitaires dans : D:\Vincent\PapLicenceManager\test\unit\com\proactive\plugin\paplicencemanager



Supprimer les classes non-voulues :



7.8.1. Type de test

Les classes de tests peuvent être de Spécifications ou d'intégration. Les classes d'intégrations vont effectuer des tests plus axés sur les actions d'une base de données physique contrairement aux spécifications qui elles toucheront plutôt aux bases telles que H2 et au code.

Pour ma part, j'ai choisi d'écrire uniquement des tests unitaires simples avec des interactions simples avec la base de données. C'est pourquoi j'ai choisi d'utiliser une classe de spécification.

7.8.2. Structure

La classe se compose de deux méthodes importantes : setup et cleanup qui sont exécutées au lancement et à la fin du test. Les autres méthodes seront des méthodes de tests.

7.8.3. Méthode de test

7.8.3.1. Écriture

Une méthode de test s'écrit comme suit :

```
void "showLicenceInfoVideTest"() {}
```

7.8.3.2. Vérification

Pour effectuer un test simple ne nécessitant pas que le code s'exécute au préalable, il suffit d'écrire le code suivant :

```
void "historizeAllTest"() {  
    //Historisation donc aucune licence non historisée restantes  
    expect:  
    assertTrue(licenceService.historizeAll())  
}
```

Si du code doit être exécuté avant de tester cela, il va falloir utiliser l'écriture when/then.

```
void "showLicenceInfoVideTest"() {  
    //Historisation de toutes licences donc aucune infos  
    when:  
    licenceService.historizeAll()  
    then:  
    assertNull(licenceService.showLicenceInfo())  
}
```

7.8.4. Exemple complet

Voici la classe de test LicenceFileSpec qui va tester la classe de domaine Licence.

```
package com.proactive.plugin.paplicencemanager

import grails.test.mixin.TestFor
import spock.lang.Specification

/**
 * See the API for {@link grails.test.mixin.domain.DomainClassUnitTestMixin} for usage
 * instructions
 */
@TestFor(LicenceFile)
class LicenceFileSpec extends Specification {
    LicenceFile licenceFileBlank
    LicenceFile licenceFileNull
    LicenceFile licenceFile

    def setup() {
        licenceFileBlank = new LicenceFile(fileContent: "")
        licenceFileNull = new LicenceFile(null)
        licenceFile = new LicenceFile(fileContent: "uzfktfdgvliligilghuigkuzfgkjvfkfgvfuj")
    }

    def cleanup() {
        licenceFileBlank?.delete()
        licenceFileNull?.delete()
        licenceFile?.delete()
    }

    void "Creation LicenceFile"() {
        expect:
        assertNull(licenceFileBlank.save())
        assertNull(licenceFileNull.save())
        assertNotNull(licenceFile.save())
    }
}
```

7.9. Scripts

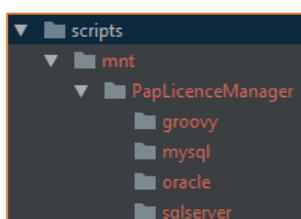
7.9.1. Configuration

Pour commencer, il va falloir activer le plugin SQL-Maintenant. Pour ce faire, décommenter la ligne correspondante dans le fichier BuildConfig.groovy.

```
compile "sql-maintenance:16.6.0" //Laisser commenté jusqu'à la création des bases de données
```

Ensuite il faut appliquer ces changements : [Apply Grails changes to IDEA project structure](#)

Une fois les changements effectués, des répertoires sont apparus dans D:\Vincent\PapLicenceManager\scripts.



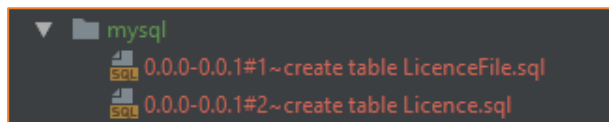
Un fichier supplémentaire est créé : PapLicenceManagerDefault.groovy. Il contient les informations sur la version du composant.

Afin d'éviter que les scripts se lancent lorsque l'on exécute le plugin séparément de l'application tierce, il faudra ajouter le code suivant au fichier config.groovy :

```
//Disable MNT when run in standalone mode
mnt.deactivateAutoUpdate = true
```

7.9.2. Création des scripts

Les scripts doivent respecter la convention de nommage suivante :



0.0.0 : Numéro de version actuelle

0.0.1 : Numéro de la version cible

#x : Numéro servant à l'ordre de passage du script s'il y en a plusieurs par version

~texte : Description du script

Il faut créer les scripts pour les différents types de bases des applications de Proactive : Mysql, oracle, sqlserver selon les standards de Proactive.

7.9.2.1. Exemple MySQL

```
1 CREATE TABLE LICENCE_FILE (
2   id          BIGINT(20) AUTO_INCREMENT NOT NULL PRIMARY KEY,
3   file_content TEXT NOT NULL
4 )
5 ENGINE = InnoDB
6 DEFAULT CHARACTER SET = utf8;
```

```
1 CREATE TABLE LICENCE (
2   id          BIGINT(20) AUTO_INCREMENT NOT NULL PRIMARY KEY,
3   acquired    BIGINT(20) NOT NULL,
4   begin_validity_date DATE NOT NULL,
5   expiry_date  DATE NOT NULL,
6   historized   BIT(1) NOT NULL,
7   licence_file_id BIGINT(20),
8   CONSTRAINT FK_LICENCE_FILE_ID FOREIGN KEY (licence_file_id) REFERENCES LICENCE_FILE (id),
9   INDEX IDX_LICENCE_LICENCE_FILE_ID(licence_file_id)
10 )
11 ENGINE = InnoDB
12 DEFAULT CHARACTER SET = utf8;
```

7.9.3. Mappings

Afin que Grails puisse utiliser ces tables, il faut ajouter des mappings aux classes de domaines.

7.9.3.1. LicenceFile

```
static mapping = {
  table 'LICENCE_FILE'
  version false
  id generator: 'native', params: [sequence: 'LICENCE_FILE_ID_SEQ', max_lo: 100]
```



```
//Pour oracle utilisation d'une séquence pour incrémenter les ids  
}
```

7.9.3.2. Licence

```
static mapping = {  
    table 'LICENCE'  
    version false  
    id generator: 'native', params: [sequence: 'LICENCE_ID_SEQ', max_lo: 100] //Pour  
    oracle utilisation d'une séquence pour incrémenter les ids  
}
```

7.10. Déploiement

Le plugin étant terminé, il faut à présent le déployer dans l'Artifactory de Proactive.

Ouvrir le fichier PapLicenceManagerGrailsPlugin.groovy choisir comme numéro de version 0.0.1 :

```
// the plugin version  
def version = "0.0.1"
```

Modifier la méthode doWithSpring tel que suit :

```
def doWithSpring = {  
    Map mntConfig = application.mergedConfig.asMap()  
  
    String serviceToInject = mntConfig.papLicence.customLicenceService  
  
    //Licence Service  
    log.info "Loading licence service: [$serviceToInject]"  
    licenceService(LicenceService) {  
        licenceInterface = ref(serviceToInject)  
    }  
}
```

Le code maintenant présent dans cette méthode, permet de récupérer **dans le fichier de configuration de l'application qui utilise le plugin** le service '*CustomLicenceService*' à injecter.

7.10.1. Exclusions

Certains fichiers doivent être exclus du déploiement car ils servent uniquement pour la démonstration ou sont utiles seulement si le plugin est lancé en mode standalone. Pour ce faire, il faut modifier pluginExcludes dans ce même fichier et y ajouter le code suivant :

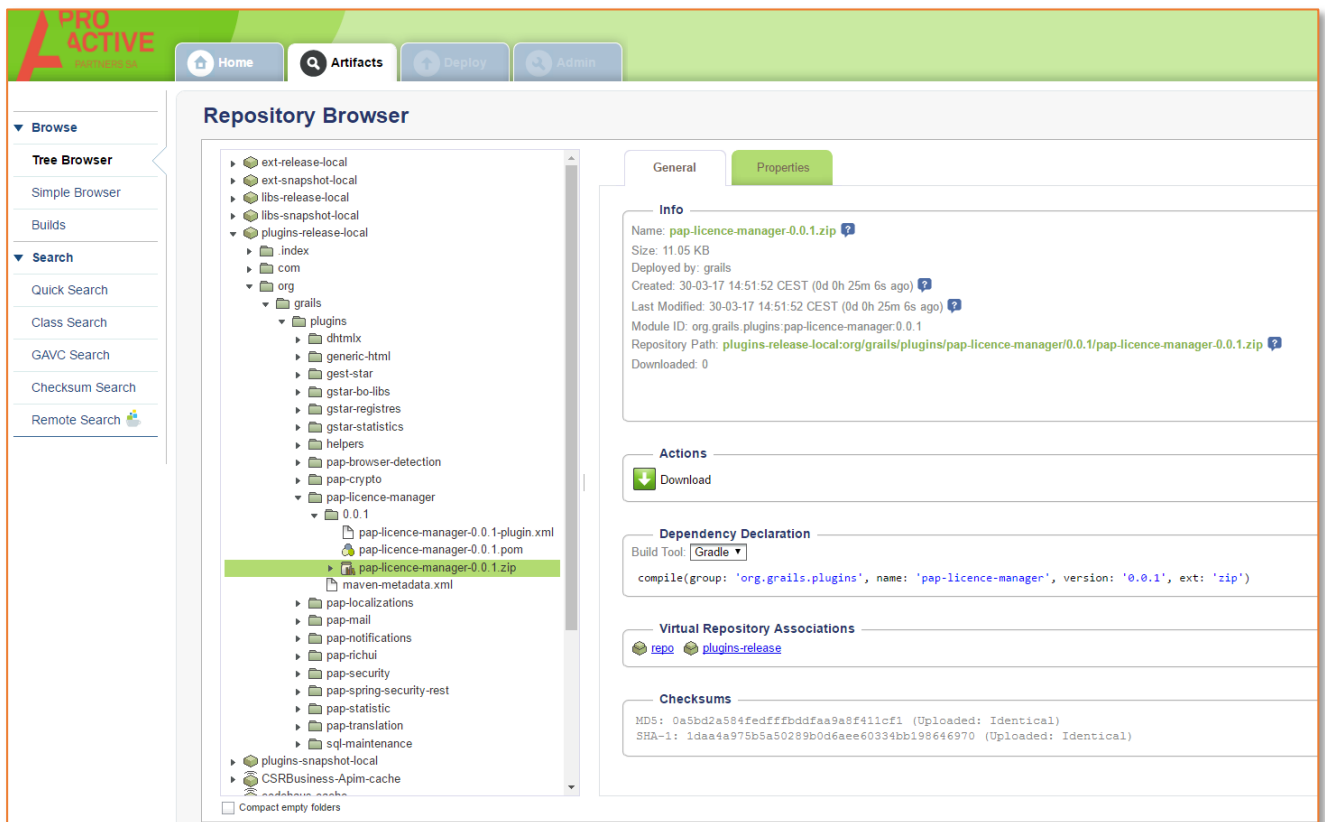
```
def pluginExcludes = [  
    "grails-app/controllers/**",  
    "grails-app/views/**",  
    "grails-  
app/services/com/proactive/plugin/paplicencemanager/UserLicenceService.groovy",  
    "grails-app/domain/com/proactive/plugin/paplicencemanager/User.groovy",  
    "web-app/css/**"  
]
```

7.10.2. Publication

Les repositories ont déjà été configurés lors de la mise en place de l'architecture. Il suffit de lancer le fichier publish.bat précédemment importé afin de commencer la publication du plugin.

Résultat :

```
| Plugin packaged grails-pap-licence-manager-0.0.1.zip
| POM generated: D:\Vincent\PapLicenceManager\target/pom.xml..
Maven deploy complete.
```



The screenshot shows the Repository Browser interface with the following details:

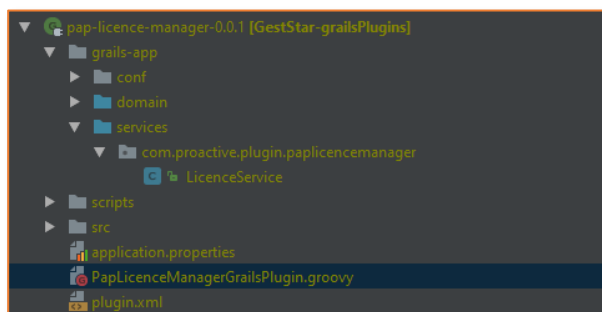
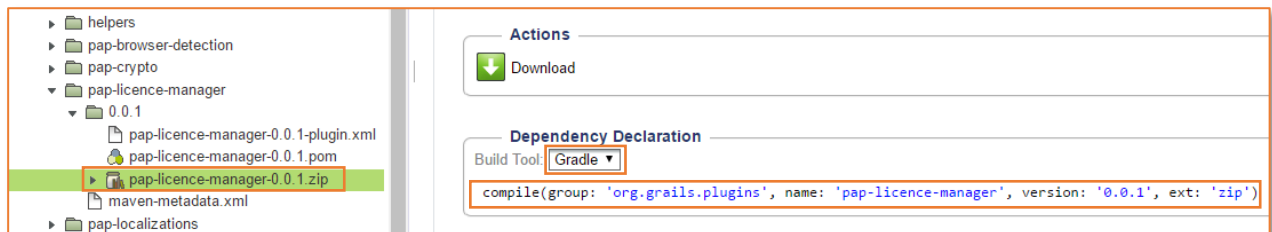
- Left Sidebar:** Contains navigation links for Browse, Tree Browser, Simple Browser, Builds, Search, Quick Search, Class Search, GAVC Search, Checksum Search, and Remote Search.
- Repository Browser:** Displays a tree view of the repository structure. The path `org.grails.plugins.pap-licence-manager-0.0.1.pap-licence-manager-0.0.1.zip` is highlighted.
- General Tab:**
 - Info:**
 - Name: `pap-licence-manager-0.0.1.zip`
 - Size: 11.05 KB
 - Deployed by: `grails`
 - Created: 30-03-17 14:51:52 CEST (0d 0h 25m 6s ago)
 - Last Modified: 30-03-17 14:51:52 CEST (0d 0h 25m 6s ago)
 - Module ID: `org.grails.plugins.pap-licence-manager:0.0.1`
 - Repository Path: `plugins-release-local/org/grails/plugins/pap-licence-manager/0.0.1/pap-licence-manager-0.0.1.zip`
 - Downloaded: 0
 - Actions:** Includes a `Download` button.
 - Dependency Declaration:**
 - Build Tool: `Gradle`
 - compile(group: 'org.grails.plugins', name: 'pap-licence-manager', version: '0.0.1', ext: 'zip')
 - Virtual Repository Associations:** Shows associations for `repo` and `plugins-release`.
 - Checksums:**
 - MD5: `0a5bd2a584fedffdbddfaa9a8f411cf1` (Uploaded: Identical)
 - SHA-1: `1daa4a975b5a50289b0d6aee60334bb198646970` (Uploaded: Identical)

Les fichiers suivants seront créés lors de la publication. Il faut les supprimer de subversion. Pour éviter que ceux qui seront ajoutés lors des prochains déploiements ne soient ajoutés à subversion, supprimez de subversion tous les *.zip ainsi que le fichier plugin.xml.

```
grails-pap-licence-manager-0.0.1.zip
grails-pap-licence-manager-0.0.1.zip.sha1
PapLicenceManager.iml
PapLicenceManager-grailsPlugins.iml
PapLicenceManagerGrailsPlugin.groovy
plugin.xml
publish.bat
```

7.10.3. Intégration dans une application

Pour intégrer ce plugin dans une application, il suffit d'ajouter à son BuildConfig.groovy la ligne générée en Gradle sur Artifactory.



Il ne reste plus qu'à coder la classe 'CustomLicenceService' et de rajouter dans son fichier de configuration un champ 'papLicence.customLicenceService' dont la valeur sera le nom de ce service afin qu'il soit injecté correctement dans le plugin.

8. Test

8.1. Sélection de la licence

En cliquant sur le lien situé sur le nombre de licences acquises dans le bloc liste, le bloc modifications de la licence : X doit être modifié et afficher les valeurs de la licence sélectionnée.

Liste						Modifications de la licence : 1					
Acquired	Begin	Validity	Date	Expiry	Date	Is	Historized	Licence	File		
5	21-02-2013	14-03-2014	True								
10	24-02-2014	11-12-2020	False					1			
25	01-06-2015	04-02-2015	True					1			

Liste						Modifications de la licence : 2					
Acquired	Begin	Validity	Date	Expiry	Date	Is	Historized	Licence	File		
5	21-02-2013	14-03-2014	True								
10	24-02-2014	11-12-2020	False					1			
25	01-06-2015	04-02-2015	True					1			

8.2. Édition d'une licence

8.2.1. Licence avec champs corrects

Cliquer sur « edit » et entrer des champs corrects (date début < date fin etc) puis « Valider ». La licence doit être mise à jour.

Modifications de la licence : 2				
Acquired	Begin Validity Date	Expiracy Date	Is Historized	
10	24-02-2014	11-12-2020	False	Edit

Modifications de la licence : 2				
Acquired	Begin Validity Date	Expiracy Date	Is Historized	
2	24 février 2014 11	décembre 2046	false	
<input type="button" value="Valider"/> <input type="button" value="Annuler"/>				

La liste est mise à jour et voici le résultat :

Liste				
Acquired	Begin Validity Date	Expiracy Date	Is Historized	Licence File
5	21-02-2013	14-03-2014	True	
2	24-02-2014	11-12-2046	False	1
25	01-06-2015	04-02-2015	True	1

8.2.2. Licence avec champs incorrects

Entrez cette fois-ci une date de fin antérieure à la date de début. La licence ne doit pas être sauvegardée.

Modifications de la licence : 2				
Acquired	Begin Validity Date	Expiracy Date	Is Historized	
2	24 février 2057 11	décembre 2046	false	
<input type="button" value="Valider"/> <input type="button" value="Annuler"/>				

La licence n'a pas été modifiée et une trace dans les logs le confirme.

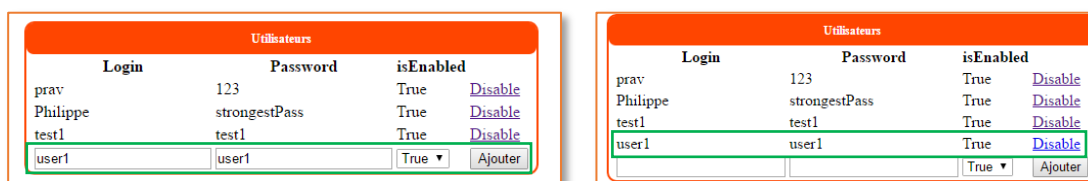
Liste				
Acquired	Begin Validity Date	Expiracy Date	Is Historized	Licence File
5	21-02-2013	14-03-2014	True	
2	24-02-2014	11-12-2046	False	1
25	01-06-2015	04-02-2015	True	1

```
....La date de début de validité ne peut pas être plus grande que la date d'expiration
Erreur lors de la sauvegarde de la licence, opération annulée
```

8.3. Utilisateurs

8.3.1. Ajout d'utilisateurs

Entrer le login et password, choisir si l'utilisateur est activé ou non.



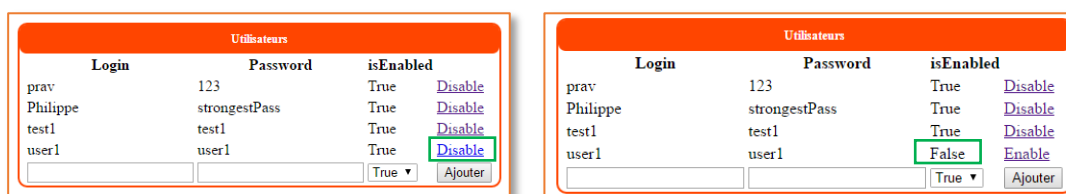
The first screenshot shows the 'Utilisateurs' form with the 'Ajouter' button highlighted. The second screenshot shows the form after clicking 'Ajouter', with the 'user1' entry added to the list.

Login	Password	isEnabled	
prav	123	True	Disable
Philippe	strongestPass	True	Disable
test1	test1	True	Disable
user1	user1	True	Disable

8.3.2. Activation/désactivation utilisateurs

Cliquer sur Enable/Disable pour apercevoir l'activation/désactivation d'un utilisateur.

8.3.2.1. Désactivation



The first screenshot shows the 'Utilisateurs' form with the 'Disable' button highlighted for 'user1'. The second screenshot shows the form after clicking 'Disable', with the 'isEnabled' status for 'user1' changed to 'False'.

Login	Password	isEnabled	
prav	123	True	Disable
Philippe	strongestPass	True	Disable
test1	test1	True	Disable
user1	user1	False	Enable

8.3.2.2. Activation



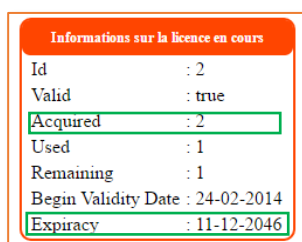
The first screenshot shows the 'Utilisateurs' form with the 'Enable' button highlighted for 'user1'. The second screenshot shows the form after clicking 'Enable', with the 'isEnabled' status for 'user1' changed to 'True'.

Login	Password	isEnabled	
prav	123	True	Disable
Philippe	strongestPass	True	Disable
test1	test1	True	Disable
user1	user1	True	Disable

8.4. Information sur la licence en cours

8.4.1. Modifications

Ce bloc est lui aussi affecté par les modifications de la licence actuelle ainsi que par l'activation d'utilisateurs supplémentaires



The screenshot shows the 'Informations sur la licence en cours' form with the following values:

Id	: 2
Valid	: true
Acquired	: 2
Used	: 1
Remaining	: 1
Begin Validity Date	: 24-02-2014
Expiracy	: 11-12-2046

8.4.2. Valeurs calculées

Les champs « Used, Remaining et Valid » sont des champs calculés par rapport au nombre d'utilisateurs

Rajouter deux utilisateurs et les activer pour faire passer la licence en invalide et remarquer le changement sur les champs de la licence en cours.

Informations sur la licence en cours
Id : 2
Valid : false
Acquired : 2
Used : 3
Remaining : -1
Begin Validity Date : 24-02-2014
Expiry : 11-12-2046

Utilisateurs

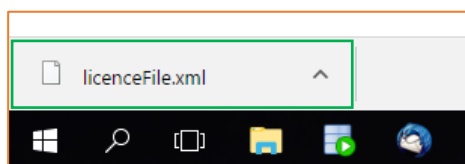
Login	Password	isEnabled	
prav	123	True	Disable
Philippe	strongestPass	True	Disable
test1	test1	True	Disable
<input type="text"/>	<input type="text"/>	True	<input type="button" value="Ajouter"/>

8.5. Création de fichier de licence encodé

Entrer les valeurs dans les champs correspondants > « Encoder »

Création de fichier de licence encodé
Acquired :
Begin validity :
Expiry :

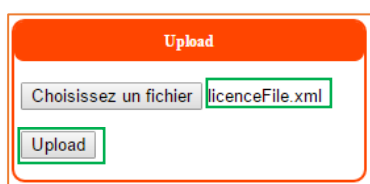
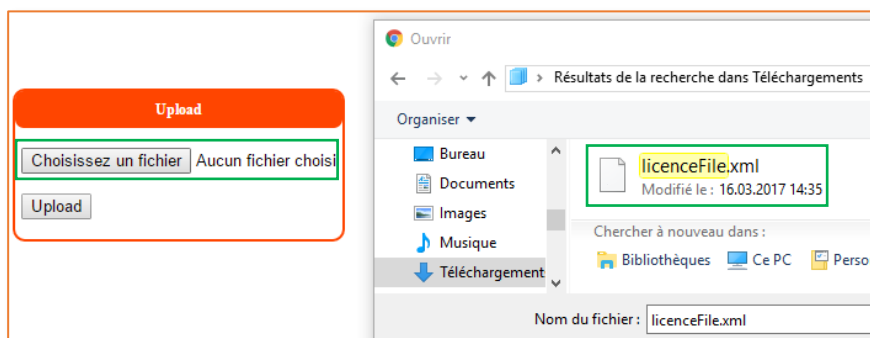
Le fichier doit être téléchargé et visible en bas du navigateur



8.6. Upload de fichier de licence

8.6.1. Upload de fichier de licence valide

Choisissez un fichier, choisissez le fichier précédemment créé > upload



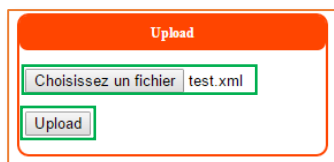
Une licence supplémentaire doit avoir été ajoutée à la liste des licences, si elle est valide elle est passée en licence en cours et les autres licences ont été historisées, un 'LicenceFile' a été créé dans la base et est lié à cette licence.

Liste					
Acquired	Begin Validity Date	Expiry Date	Is Historized	Licence File	
5	21-02-2013	14-03-2014	True		
2	24-02-2014	11-12-2046	True		1
25	01-06-2015	04-02-2015	True		1
666	04-08-2010	11-09-2030	False		2

Informations sur la licence en cours	
Id	: 4
Valid	: true
Acquired	: 666
Used	: 2
Remaining	: 664
Begin Validity Date	: 04-08-2010
Expiry	: 11-09-2030

8.6.2. Upload de fichier invalide

Créer un fichier texte : test.xml contenant uniquement le texte suivant : « test » et essayer d'upload ce fichier.



Logs :

```
Une erreur est survenue lors de l'upload du fichier de licence : java.lang.NullPointerException
Veuillez vérifier le format et le contenu du fichier de licence
```

9. Problèmes rencontrés

9.1. Génération d'un fichier XML

9.1.1. Contexte

Dans mon écran de test, je devais permettre de créer un fichier XML encodé en Blow Fish qui se télécharge dans le navigateur.

9.1.2. Problème

Pour ce faire, j'ai utilisé des fonctions fournies par Java simple. Cela m'a pris du temps et beaucoup de lignes de code pour parvenir à la création du fichier. Je devais commencer par créer un fichier temporaire, créer une arborescence XML et l'y insérer, relire le fichier pour l'encoder en Blow Fish et enfin fournir le fichier au navigateur pour le téléchargement.

En utilisant ce procédé, l'application était lente et devait créer un fichier physiquement sur le poste.

9.1.3. Résolution

Alerté par la difficulté, la complexité et la lenteur du code, j'ai discuté avec mes collègues afin de déterminer si une méthode plus simple existait.

Il se trouve que Grails permet de créer une arborescence XML à partir d'un objet sans avoir besoin de créer un fichier en une seule et simple ligne de code.

```
String content = licence as XML
```

J'ai donc utilisé cette nouvelle connaissance. Je récupère le contenu dans l'arborescence XML dans un String pour ensuite l'encoder et finalement envoyé cela au navigateur pour le téléchargement.

9.2. Lancement des tests unitaires

9.2.1. Problème

Après avoir codé les tests unitaires, en les lançant, une erreur inconnue survient :

```
java.lang.NullPointerException
    at org.springframework.transaction.support.TransactionTemplate.execute(TransactionTemplate.java:130)
    at
org.codehaus.groovy.grails.orm.support.GrailsTransactionTemplate.execute(GrailsTransactionTemplate.groovy:
85)
```

9.2.2. Tentative de résolution

Après de longues recherches et de nombreux essais, j'ai trouvé la solution en m'inspirant d'un problème identique sur <http://stackoverflow.com/questions/24622244/nullpointer-in-simple-grails-controller-unit-testcko>

Mon LicenceService ainsi que ma classe de test utilisaient la mauvaise classe pour se déclarer comme étant transactionnels.

Il faut donc remplacer :

```
import grails.transaction.Transactional
```

par

```
import org.springframework.transaction.annotation.Transactional
```

dans le `LicenceService` ainsi que la classe de test.

Mais après avoir effectué cela, certains comportements transactionnels ne s'effectuent plus correctement. Exemple : En éditant une licence et en lui assignant une date d'expiration postérieure à celle de début la modification n'est pas annulée

9.2.3. Résolution

Le problème vient de Grails et est apparemment connu. Il suffit de supprimer les '`@Transactional`' ainsi que les imports et d'ajouter au service le code suivant :

```
static transactional = true
```

Cela permet de déclarer le service et ses méthodes comme étant transactionnels sans avoir besoin d'utiliser les '`@Transactional`'.

10. Améliorations possibles

Actuellement, aucune amélioration n'est prévue mais le plugin est assez flexible pour permettre des évolutions futures.

11. Conclusion

Ce projet m'a permis de me faire une idée plus précise de la façon dont se déroule un projet du point de départ (analyse, planification) jusqu'à son achèvement. J'ai pu remplir entièrement le cahier des charges et ainsi terminer ce projet.

J'ai beaucoup appris et pris confiance en moi durant ces 120 heures, ce qui me permettra de m'affirmer un peu plus lors des futurs projets que j'aurai à réaliser.

12. Remerciements

Je souhaiterai remercier les experts Messieurs Dubuis Bernard et David Joël ainsi que Monsieur Patrick Lathion et mes collègues qui m'ont suivi et encadré durant toute la durée de ce projet.

13. Date et Signature

Sion, le vendredi, 07. Avril 2017

Praz Vincent

14. Références

14.1. Grails

[https://fr.wikipedia.org/wiki/Grails_\(technique\)](https://fr.wikipedia.org/wiki/Grails_(technique))

<https://grails.org/index.html>

14.2. MVC

<https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur>

14.3. H2

[https://fr.wikipedia.org/wiki/H2_\(base_de_donn%C3%A9es\)](https://fr.wikipedia.org/wiki/H2_(base_de_donn%C3%A9es))

14.4. IntelliJ IDEA

<https://www.jetbrains.com/idea/>

14.5. Spring

14.5.1. Spring

[https://fr.wikipedia.org/wiki/Spring_\(framework\)](https://fr.wikipedia.org/wiki/Spring_(framework))

14.5.2. IoC

https://fr.wikipedia.org/wiki/Inversion_de_contr%C3%B4le

https://fr.wikipedia.org/wiki/Injection_de_d%C3%A9pendances

15. Annexes

En annexe de ce document, se trouvent la planification du projet, le code complet du plugin ainsi que le journal de bord.

15.1. Procès-Verbaux

15.1.1. Séance du 16.02.2017

Gestionnaire de licences à l'utilisation

Séance de démarrage du projet

Séance du 16.02.2017

Visa

Membres présents :

Joël David
Bernard Dubuis
Patrick Lathion
Vincent Praz

JD
BD
PL
VP

Excusé(s) :

Secrétaire :

Vincent Praz

VP

Resp.	Délai / Etat
-------	--------------

1	Informations		
----------	---------------------	--	--

VP	La documentation utilisateur sera axée vers une documentation programmeur	VP	N/A
VP	Informations Générales sur le TPI * Les PVs doivent se trouver dans la rubrique annexe du rapport * Présentation (~30 mn) et défense (~30 mn) le temps doit être évalué en avance * Les sources doivent être citées	VP	N/A
VP	Les autres informations générales concernant l'organisation, l'administration et autres points du TPI m'ont été transmises par les experts	VP	TERMINE

2	Décisions prises		
----------	-------------------------	--	--

VP	Choix des dates de la * Visite Intermédiaire le 16.03.2017 à 8h30 * Défense le 27.04.2017 à 13h30	JD, BD, PL, VP	TERMINE
VP	La planification initiale comprend des risques clairement visibles et doit donc être modifiée afin d'éviter des retards et imprévus.	VP	23.02.2017
VP	La deuxième date de remise du projet a été modifiée sur le cahier des charges pour correspondre à la première : 07.04.2017	PL	TERMINE
VP	Le PV de la séance rédigé ainsi que la planification corrigée doivent être envoyés à tous les membres du projet.	VP	23.02.2017

3	Points ouverts		
----------	-----------------------	--	--

VP	La possibilité de soumettre le code sur une clé USB ou dans les annexes a été discutée. Je choisirai donc ce qui me convient le plus.	VP	TERMINE
----	---	----	---------

15.1.2. Séance du 23.02.2017 #1

Gestionnaire de licences à l'utilisation

Séance d'information base de données

Séance du 23.02.2017

Membres présents :

Patrick Lathion
Vincent Praz

Visa

PL
VP

Excusé(s) :
Secrétaire :

Vincent Praz

VP

		Resp.	Délai / Etat
1	Informations		
VP	Informations sur les bases de données * Informations générales sur les bases H2 et l'outil hibernate * Informations sur le plugin SQL-Maintenance	VP, PL	TERMINE
2	Décisions prises		
VP	Le nom du plugin a été choisi : PapLicenceManagerPlugin	PL	TERMINE
3	Points ouverts		
VP	Le choix reste ouvert sur certains comportements de la base de données * Travail en mémoire ou sur un fichier	PL, VP	EN COURS

15.1.3. Séance du 23.02.2017 #2

Gestionnaire de licences à l'utilisation

Séance sur les standards, l'architecture et la création d'un plugin

Séance du 23.02.2017

Membres présents :

Patrick Lathion

Visa

PL

Vincent Praz

VP

Excusé(s) :

Secrétaire :

Vincent Praz

VP

Resp.	Délai / Etat
-------	--------------

1 Informations		
-----------------------	--	--

VP	Informations sur L'architecture et la création d'un plugin * Informations sur le SVN, publication, exclusions, etc * Information sur les configurations de bases d'un plugin	VP, PL	TERMINE
----	--	--------	---------

2 Décisions prises		
---------------------------	--	--

VP	Le SVN qui contiendra les sources ainsi que l'Artifactory seront mis en place par les administrateurs de l'architecture	Administrateur architecture	TERMINE
----	---	-----------------------------	---------

15.1.4. Séance du 16.03.2017

Gestionnaire de licences à l'utilisation

Séance intermédiaire

Séance du 16.03.2017

Visa

Membres présents :

Joël David
Bernard Dubuis
Patrick Lathion
Vincent Praz

JD
BD
PL
VP

Excusé(s) :

Secrétaire :

Vincent Praz

VP

		Resp.	Délai / Etat
1	Informations		
VP	Bien remplir les objectifs du cahier des charges avant de faire des fonctionnalités supplémentaires car l'évaluation s'effectue sur son contenu.	VP	07.04.2017
VP	Impression de 3 exemplaires de la documentation, envoyer par la poste le jour de la fin du projet en recommandé avec le tampon du guichet de la poste.	VP	07.04.2017
2	Décisions prises		
VP	Prendre conseil auprès des collègues et s'entraîner avec leur aide pour la présentation.	VP	07.04.2017
3	Points ouverts		
VP	Éventuelle modification de la planification afin d'améliorer sa lisibilité	VP	07.04.2017