



ECOLE MAROCAINE DES SCIENCES DE L'INGÉNIEUR

PROJET DE FIN D'ETUDES

THEME

Conception et développement d'une suite d'applications pour l'automatisation des déploiements sur les machines des développeurs

FILIÈRE

Ingénierie Informatique et Réseaux

OPTION

MIAGE

RÉALISÉ PAR

Mr. SAID EL IMAM Said

ENCADRÉ PAR

Mr. OULAD HAJ THAMI Rachid (EMSI)

Mr. YEKHLEF Youcef (Capgemini)

Année universitaire : 2012 - 2013

SUJET

Conception et développement d'une suite d'applications
pour l'automatisation des déploiements sur les machines
des développeurs du projet SNCF Fret.



Directrice de projet

- **Madame ZAKRAOUI Mounia (Capgemini Casa)**

Soutenu par :

- **Monsieur SAID EL IMAM Said**

Encadré par :

- **Monsieur OULAD HAJ THAMI Rachid (EMSI)**
- **Monsieur LAMOURI Saad (Capgemini Casa)**
- **Monsieur YEKHLEF Youcef (Capgemini Lille)**

Dédicaces

A ma très chère mère

A mon très cher père

A ma très chère sœur Meryem

A toute ma famille

A tous mes amis

A tous mes professeurs

A madame la directrice du projet

A mes encadrants

Et à tous ceux qui me sont chers

Remerciements

En guise de reconnaissance, il me paraît opportun de commencer ce rapport par l'expression de mes sincères remerciements à toute personne ayant participé de près ou de loin à l'élaboration de mon projet de fin d'études, et à favoriser son aboutissement.

Ainsi, je tiens à remercier mon encadrant interne Mr. OULAD HAJ THAMI Rachid pour les conseils qu'il m'a prodigués, qui était tout le temps disponible pour me guider et m'assister tout au long de ma période de stage et pour m'avoir aiguillé sur la rédaction de ce présent rapport.

J'exprime ma profonde gratitude à Mme ZAKRAOUI Mounia, directrice de projet, pour m'avoir accueilli au sein de Capgemini, ainsi que pour la qualité de l'encadrement dont j'ai pu bénéficier au travers de son assistance.

Il m'apparaît également légitime d'accorder toute ma gratitude à monsieur YEKHLEF Youcef, chef d'équipe, pour son aide précieuse accordée tout au long de ce stage à travers des points d'avancement et qui a toujours trouvé le temps de répondre à mes questions concernant l'application.

Je tiens ensuite à remercier toute l'équipe du projet SNCF FRET avec qui j'ai travaillé et qui s'est volontairement montrée disponible pour m'aider.

Je remercie enfin l'ensemble du personnel de Capgemini, pour leur accueil appréciablement convivial.

Que le corps professoral et administratif de l'EMSI trouve ici mes vifs remerciements, pour tout le travail effectué durant ma scolarité.

Enfin, je souhaite que mon travail soit à la hauteur des attentes de toutes ces personnes.

Liste des abréviations :

<i>Abréviation</i>	<i>Description</i>
ANT	Another Neat Tool
BAT	Build Automation Tool
CRUD	Create, Read, Update, Delete
DOM	Document Oriented Model
FIFO	First In First Out
GUI	Graphical User Interface
IDE	Integrated Development Environment
IHM	Interface Homme Machine
J2SE	Java 2 Platform, Standard Edition
J2EE	Java 2 Platform, Enterprise Edition
JAXP	Java API for XML Parsing
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
MSI	MicroSoft Installer
PAD	Portable Application Description
RIA	Rich Internet Application
SAX	Simple API for XML
SI	Système d'Information
SNCF	Société Nationale des Chemins de Fer
StAX	Streaming API for XML
UML	Unified Modeling Language
VFS	Virtual File System
VCS	Version Control System
XML	eXtended Markup Language

Table des figures

Figure 1-5-1 Organisation de l'équipe du pôle TEC	22
Figure 1-5-2 Organisation de l'équipe du projet	22
Figure 1-5-3 Montage des étapes générales du projet	23
Figure 1-6-1 Graphique de répartition du travail sur les tâches	24
Figure 1-6-2 Ligne de temps du planning	25
Figure 1-6-3 Diagramme de <i>Gantt</i>	26
Figure 2-1-1 Environnement fonctionnel du projet	29
Figure 2-1-2 Interface principale du <i>Setup Maker</i>	31
Figure 2-1-3 tab Scan du <i>Setup Maker</i>	31
Figure 2-1-4 tab Set du <i>Setup Maker</i>	32
Figure 2-1-5 tab Set, fonction Sort du <i>Setup Maker</i>	33
Figure 2-1-6 tab Tweak du <i>Setup Maker</i>	34
Figure 2-1-7 tab Build du <i>Setup Maker</i>	35
Figure 2-1-8 Interface principale du <i>Dashboard</i>	36
Figure 2-3-1 Cas d'utilisation global	39
Figure 3-1-1 Diagramme global des dépendances entre les applications	43
Figure 3-1-2 Diagramme d'activité global	44
Figure 3-1-3 Structure de déploiement - <i>Package SNCF Fret</i>	45
Figure 3-2-1 Diagramme de classe - <i>Application</i>	45
Figure 3-2-2 Diagramme de classe - <i>Setup</i>	46
Figure 3-2-3 Diagramme de classe – <i>Pack</i>	47
Figure 3-2-4 Diagramme de classes du <i>Dashboard</i>	48
Figure 3-2-5 Diagramme de classes global	49
Figure 3-3-1 Diagramme d'activité du <i>Package</i>	50
Figure 3-4-1 Diagramme de séquence, connexion client	51
Figure 3-4-2 Diagramme de Séquence, connexion serveur	51
Figure 4-2-1 Capture d'écran de l'étape 1 du package	59
Figure 4-2-2 Capture d'écran de l'étape 2 du package	60
Figure 4-2-3 Capture d'écran de l'étape 3 du package	60
Figure 4-2-4 Capture d'écran de l'étape 4 du package	61
Figure 4-2-5 Capture d'écran de l'étape 5 du package	61
Figure 4-2-6 Capture d'écran de l'étape 6 du package	62
Figure 4-2-7 Capture d'écran de l'étape 7 du package	62
Figure 4-2-8 Capture d'écran de l'étape 8 du package	63
Figure 4-2-9 Capture d'écran de l'étape 1 de l'application <i>Setup Maker</i>	64
Figure 4-2-10 Capture d'écran de l'étape 2 de l'application <i>Setup Maker</i>	65
Figure 4-2-11 Capture d'écran des propriétés de l'application <i>Setup Maker</i>	66
Figure 4-2-12 Capture d'écran de l'étape 3 de l'application <i>Setup Maker</i>	67
Figure 4-2-13 Capture d'écran de l'étape 4 de l'application <i>Setup Maker</i>	68
Figure 4-2-14 Capture d'écran de l'authentification de l'application <i>Dashboard</i>	69
Figure 4-2-15 Capture de l'écran principal de l'application <i>Dashboard</i>	70
Figure 5-3-1 <i>Setup Maker</i> référencé sur le site d'IzPack	73
Figure 5-3-2 <i>Setup Maker</i> référencée sur le site SoftPedia	73
Figure 5-4-1 Statistiques du nombre de téléchargements du <i>Setup Maker</i>	74
Figure 5-4-2 Statistiques des pays de téléchargements du <i>Setup Maker</i>	74
Figure 5-4-3 Statistiques des systèmes d'exploitation des téléchargements du <i>Setup Maker</i>	75

Figure 5-5-1 Statistiques de progression des lignes de code de Setup Maker.....	75
Figure 5-5-2 Statistiques de langages utilisés pour Setup Maker	76

Liste des tableaux

Tableau 1-4-1 Contraintes majeurs.....	21
Tableau 1-4-2 Risques majeurs	21
Tableau 1-4-3 Sorties des étapes du projet	23
Tableau 1-5-1 Options du calendrier de planning.....	24
Tableau 2-1-1 Entrées de l'interface <i>Scan</i> du <i>Setup Maker</i>	32
Tableau 2-1-2 Entrées de l'interface <i>Set</i> du <i>Setup Maker</i>	33
Tableau 2-1-3 Entrées de l'interface <i>Tweak</i> du <i>Setup Maker</i>	34
Tableau 2-1-4 Entrées et sortie de l'interface <i>Build</i> du <i>Setup Maker</i>	35
Tableau 2-1-5 Environnement technique du projet.....	37
Tableau 2-3-1 Description des cas d'utilisation.....	39
Tableau 3-1-1 Résultat des actions globales	43

Table des matières

Introduction générale	12
Chapitre 1 : Projet et management du projet.....	14
Introduction.....	14
1. Contexte	15
1.1. Le groupe Capgemini.....	15
1.2. Le métier de Capgemini.....	15
1.3. Le projet SNCF Fret.....	16
1.4. Organisation du centre de service Fret	16
2. Problématique	17
2.1. Etude de l'existant et problèmes liés	17
2.2. Solution proposée	17
2.3. Amélioration proposée.....	18
3. Etude théorique.....	18
3.1. Périmètre du projet.....	18
3.2. Objectifs	19
3.3. Les retombés sur l'organisme	20
4. Management du projet	20
4.1. Méthode adoptée	20
4.2. Plan Assurance Qualité.....	21
4.2.1. Contraintes et risques majeurs	21
4.2.2. Organisation	22
4.2.3. Démarche globale de production et d'assurance qualité	23
5. Le Planning prévisionnel.....	24
5.1. Répartition du travail	24
5.2. Diagramme de Gantt.....	25
5.3. Produits tangibles.....	27
5.4. Autres prestations	27
Conclusion	27
Chapitre 2 : Analyse et spécifications.....	28
Introduction.....	28
1. Analyse fonctionnelle et technique.....	29
1.1. Environnement fonctionnel	29
1.2. Présentation des applications du projet	30
1.2.1. SNCF Package	30

1.2.2.	DCP Setup Maker.....	31
1.2.3.	DCP Dashboard.....	36
1.3.	Environnement technique.....	37
2.	Présentation des acteurs.....	38
2.1.	Développeur.....	38
2.2.	Modérateur	38
2.3.	Administrateur	38
3.	Présentation synthétique des cas d'utilisation	39
4.	Présentation détaillée des cas d'utilisation.....	40
	Conclusion	41
	Chapitre 3 : Conception	42
	Introduction.....	42
1.	Conception globale.....	43
2.	Diagrammes de classe	45
2.1.	Générer package	45
2.2.	Installer.....	46
2.3.	Surveiller.....	48
	Diagramme global	48
3.	Diagramme d'activité	50
4.	Diagrammes de séquence	51
	Conclusion	52
	Chapitre 4 : Réalisation	53
	Introduction.....	53
1.	Présentation des outils de réalisation.....	54
1.1.	Technologie de développement.....	54
1.2.	Environnement de développement	54
1.3.	Librairies graphiques	55
1.3.1.	Apache Pivot.....	55
1.3.2.	Vaadin.....	55
1.4.	Librairies de construction automatisée.....	56
1.4.1.	Apache Ant	56
1.4.2.	Apache Maven.....	56
1.5.	Outils d'emballage.....	57
1.5.1.	Izpack.....	57
1.5.2.	TrueZip.....	57
1.6.	Gestion de données.....	58

1.6.1.	XML.....	58
1.6.2.	Stax	58
1.6.3.	Staxmate.....	58
2.	Etapes de réalisation	59
2.1.	Développeur	59
	Etape 0 : Choix de langue	59
	Etape 1 : Ecran de bienvenue	59
	Etape 2 : Choix du pole.....	60
	Etape 3 : Choix des packs à installer.....	60
	Etape 4 : Ecran de résumé.....	61
	Etape 5 : Installation.....	61
	Etape 6 : Lancement des scripts et exécutables	62
	Etape 7 : Installation des raccourcis.....	62
	Etape 8 : Ecran de fin.....	63
2.2.	Administrateur	64
	Etape 1 : Scan du répertoire.....	64
	Etape 2 : Paramétrage des packs	65
	Etape 3 : Paramétrage du package.....	67
	Etape 4 : Construction du package final.....	68
2.3.	Modérateur	69
	Authentification.....	69
	Dashboard	70
	Conclusion	70
	Chapitre 5 : Vers un produit Open Source	71
	Introduction.....	71
1.	Licence d'utilisation	72
2.	Annuaire de code source.....	72
3.	Stratégie Marketing.....	73
4.	Statistiques de téléchargement	74
5.	Statistiques techniques	75
	Conclusion	76
	Conclusion générale	77
	Webographie	79
	Annexes.....	80
	Licence Apache 2.0.....	80

Introduction générale

Le problème de la gestion des parcs informatiques tant au niveau des installations ou des configurations est une étape importante et préalable dans les grands projets. En effet, avant de démarrer un projet, une phase d'installation des machines et des outils nécessaires au projet est primordiale pour préparer l'environnement de travail au développeur. D'habitude, une telle tâche consomme un nombre important de budget jour/homme, surtout pour les grands projets où on trouve des dizaines, voire des centaines de développeurs et de postes de travail à gérer.

A Capgemini, il y a toute une équipe dédiée à cette tâche.

Le présent stage de fin d'étude s'inscrit dans cette problématique et principalement dans le cadre du projet SNCF Fret pris en charge par Capgemini pour le compte de la société SNCF en France.

Ainsi, l'objectif de mon projet est la conception et le développement d'un système pour :

- Générer un package développeur en mode installateur qui permettra d'installer tous les outils nécessaires au projet pour les développeurs et les configurations requises pour le projet ;
- Générer des patchs de mise à jour ;
- Gestion des versions ;
- Etc.

Ces tâches qui se font actuellement manuellement et consomment plusieurs jours/homme, doivent au terme de mon projet être totalement automatisées. En plus, j'ai proposé d'enrichir le projet par une autre fonctionnalité de :

- Monitoring et surveillance réseau des machines ;

À son terme, le projet servira comme support à l'équipe de travail dans ses activités de développement et de test. Elle permettra en effet de gérer localement les différents outils et programmes utilisés par les développeurs sur chaque poste. Ainsi, qu'une gestion avancée des versions, des fichiers de travail et des paramétrages des postes informatiques.

Pour garantir le bon déroulement de ce projet et la réalisation des objectifs, dans le respect du budget alloué, le choix d'une méthode était obligatoire. Dans ce sens, j'ai opté pour suivre un cycle de développement semi-itératif, une démarche qui a fait ses preuves dans le domaine des projets informatiques dont les besoins sont constants.

Ce rapport, qui relate ce projet, est axé sur cinq grands chapitres :

Le premier chapitre englobe l'étude des processus existants et la délimitation du champ de travail afin de permettre de fixer les objectifs.

Le deuxième chapitre consiste en la collecte des informations disponibles au niveau fonctionnel, ainsi qu'une étude pour la partie technique.

Le troisième chapitre détaille la partie conception du projet qui a pour objectif de proposer des solutions adaptées.

Le quatrième chapitre présente la réalisation de ces solutions qui contient, entre autres, les outils de mise en œuvre ainsi que la plupart des interfaces illustrant l'utilisation des applications développées.

Le cinquième et dernier chapitre porte sur la publication du produit réalisé pour une communauté internationale de l'open source.

Dans une conclusion générale, je fais le bilan de mon travail et en donne des perspectives.

Pour plus d'informations, une annexe jointe à la fin de ce document détaille les contraintes de la licence *Apache 2.0* choisie pour l'application *Setup Maker* qui est développée et publiée en open source.

Chapitre 1 :

Projet et management du projet

Introduction

Ce chapitre aborde la présentation du contexte de mon stage, de la société d'accueil, puis, le sujet et les objectifs de mon projet, ainsi que la méthodologie adoptée pour sa réalisation. Aussi, le planning prévisionnel sera présenté.

1. Contexte

Mon stage de fin d'études s'est déroulé à la société Capgemini, au siège de Casablanca Nearshore, au sein du projet SNCF.

1.1. Le groupe Capgemini

Fort d'environ **120 000 collaborateurs** et présent dans **40 pays**, Capgemini est l'un des leaders mondiaux du conseil, des services informatiques et de l'infogérance. Le Groupe a réalisé en 2011 un chiffre d'affaires de **9,7 milliards d'euros**. Avec ses clients, Capgemini conçoit et met en œuvre les solutions business et technologiques qui correspondent à leurs besoins et leurs apportent les résultats auxquels ils aspirent. Profondément multiculturel, Capgemini revendique un style de travail qui lui est propre, la « *Collaborative Business Expérience* », et s'appuie sur un mode de production mondialisé, le « *Rightshore®* ».

Capgemini est la deuxième SSII dans le secteur des **services informatiques** en France, derrière IBM Global Services, et est parmi les **5 premiers mondiaux** en **externalisation**. Elle a été créée par Serge Kampf le 1er octobre 1967 à Grenoble (France) sous le nom de Sogeti (Société pour la gestion de l'entreprise et traitement de l'information).



Capgemini Consulting est la filiale du groupe Capgemini dont les consultants sont spécialisés dans le conseil en management, le conseil en stratégie et la transformation de l'entreprise. Elle se classe **quatrième en Europe**.

L'entité **Capgemini Maroc** intervient sur l'ensemble des métiers : Banques, Assurance, Telecom, Services, Industrie, Energies & Utilities. Elle a pour ambition d'accompagner ses clients dans la transformation et l'évolution de leur système d'information.

1.2. Le métier de Capgemini

En plus des activités de Capgemini, nous pouvons dire que l'intervention de Capgemini auprès de ses clients, a pour finalité de faire évoluer et maintenir le patrimoine applicatif des Clients à travers :

- *fusion de systèmes d'information et de migration*
- *évolution des grands domaines fonctionnels*
- *refonte des systèmes d'information*

1.3. Le projet SNCF Fret

Le **FRET SNCF** est le **premier transporteur de marchandises** en France. Fret SNCF est également le **2ième transporteur ferroviaire** en Europe.

L'ouverture du marché à la concurrence ainsi



qu'un contexte économique difficile poussent le FRET à **automatiser ces processus** afin de **réduire ces coûts de fonctionnement**.

Ainsi, les SI gèrent les processus métiers de la prise en compte de la commande client à la consolidation du plan de production des solutions d'acheminement.

1.4. Organisation du centre de service Fret

J'ai effectué mon stage au centre de service **Fret**. Ce service est organisé en fonctionnement front office (Casablanca) / back office (Lille).

Sur le site de la **SNCF**, les équipes de Capgemini effectuent les tâches liées aux **aspects fonctionnels** tels que la rédaction des spécifications fonctionnelles et les tests de non régression. Leur proximité avec le client facilite grandement leur travail.

Ce mode de fonctionnement n'est cependant pas figé; la tendance actuelle vise à **rapatrier sur le site de Casablanca** une partie du projet. La principale raison de cette réorganisation est le coût, bien moins important à *Casablanca* qu'en *France*.

Les locaux sont sécurisés (norme site sûr niveau deux) car Capgemini a accès directement au réseau SNCF et ses équipes travaillent sur leurs serveurs grâce à une ligne sécurisée.

Ce projet, débuté à la mi-2006, est un **projet à long terme**. Il s'appuie sur un **ensemble d'applications déjà existantes** et doit permettre de les faire **évoluer** tout en effectuant une **migration régulière vers des applications J2EE**.

2. Problématique

La **migration constante des machines** vers de nouvelles versions d'outils et de nouvelles technologies, ainsi que les quelques **transferts de nouveaux employés** ou de stagiaires dans l'une des équipes du projet demande une **constante gestion des installations** pour chaque poste et, pour la plupart du temps, une **incompatibilité** d'outils et de versions entre les machines des développeurs du projet peut survenir.

2.1. Etude de l'existant et problèmes liés

Pour pallier au problème cité ci-dessus, plusieurs solutions disponibles ont été mises en place dont :

- ***Une image du système***

Cette image pourrait contenir tous les outils nécessaires, ainsi que les différents paramétrages. Celle-ci serait constamment mise à jour pour utiliser les derniers programmes intégrés au projet.

Mais l'installation de l'image pour une mise à jour pourrait provoquer une perte de données du projet, et peut s'avérer couteux en terme de temps nécessaire pour une simple copie.

De même, vu le nombre important de configurations différentes nécessaires pour chaque poste de travail, il était nécessaire de gérer un nombre non négligeable d'images machine.

- ***Un ensemble de scripts batch avec outils inclus***

Cette solution est aussi d'un grand usage dans les sociétés, permettant en même temps un contrôle complet sur la configuration du système d'exploitation ainsi que le lancement d'installation des outils inclus.

Par contre une gestion des versions déjà installées pourrait être assez compliquée à implémenter, de plus le lancement des scripts demande l'intervention d'un utilisateur avancé qui examinera le fichier trace de l'exécution du batch pour s'assurer que ce dernier s'est déroulé avec succès.

2.2. Solution proposée

Pour remédier à ces problèmes, un **package global** pour développeurs pourrait être la solution adéquate pour éviter une perte importante en terme de temps et de ressources à chaque installation de matériel ou pendant la synchronisation des postes entre employés.

2.3. Amélioration proposée

Bien évidemment, et au-delà des exigences fonctionnelles et techniques de la société, je me suis proposé de développer en plus un tableau de bord, sous forme d'application RIA, qui sera l'objet d'une plateforme de monitoring pour surveiller en permanence le réseau de l'entreprise ainsi que les différentes installations appliquées sur chacun des postes informatiques des développeurs pour une meilleure maîtrise.

La solution que je mettrai en œuvre sera déployée sur n'importe quel réseau de machines.

3. Etude théorique

3.1. Périmètre du projet

Basé sur la maîtrise des outils de conception et de développement, ce projet permettra de découvrir toutes les phases de la construction d'un projet de système d'information : compréhension des besoins, conception technique, prototypes, développement spécifiques en **Java**, intégration et validation globale du système.

L'objet de ce stage est d'une part de **définir la liste des fonctionnalités à développer** ainsi **qu'outils d'industrialisation et de productivité** afin de disposer d'un logiciel, notamment un **générateur de package**, destiné à être utilisé par les développeurs du projet.

Ce package doit être en mode « **Installeur** » et a pour but d'harmoniser et mettre à jour les postes des développeurs. Le package doit contenir tous les outils utilisé sur le projet, et prendre en compte les paramétrages nécessaires pour se connecter au réseau du client.

Le livrable consiste en un **package** englobant tous les outils, programmes, scripts et paramétrages à exécuter sur les différentes machines des développeurs du projet, pour une homogénéité et une compatibilité de versions et d'outils entre les équipes.

Le projet sera donc la mise en place d'un programme de couche supérieure, permettant de **générer** le dit 'package' de façon simple et intuitive, tout en permettant **un suivi des différentes installations sur le réseau** de l'entreprise avec une application réseau de monitoring se connectant aux installateurs clients générés, avec une possibilité **d'appliquer des mises-à-jours** sur ces packages déjà installées via un installeur sous forme de patch.

3.2. Objectifs

Les objectifs principaux de ce projet se présentent ainsi :

- *Lister les différents outils et versions utilisés par les développeurs*
- *Etudier la priorité de chaque outil par pôle*
- *Rassembler les archives ou installateurs des outils*
- *Générer un package global à installer sur les machines*
- *Monitoring des versions installées sur les postes*
- *Création de patchs pour les futures mises-à-jours des programmes*

La fonction principale étant d'automatiser toutes ces tâches avec une intervention humaine absente ou presque.

Les fonctions du produit :

- Grouper les outils par Pôles/Types
- Forcer l'utilisation d'un chemin d'installation commun
- Inclure vérification base de registre pour versions installées
- Mettre à jour package si version antérieure installée
- Créer raccourcis pour les outils installés
- Effectuer paramétrages réseau nécessaires sur la machine
- Surveiller les outils installés sur les postes en réseau
- Générer des patchs de mises à jour
- Planifier des mises à jour sur les postes

3.3. Les retombés sur l'organisme

Le livrable final permettra à l'entreprise d'avoir un **contrôle sur les outils** présents sur les postes des développeurs du projet et ainsi surveiller de près la cohérence des programmes, numéros de versions installées et mises-à-jours à appliquer.

Ceci permettra une **meilleure gestion des ressources** du projet, tout en évitant les quelques risques possibles, relatifs à des incompatibilités entre développeurs ou autre.

Et pour finir, **moins de temps perdu** à créer les différents installeurs, regrouper les informations sur les outils présents sur chaque poste ou mettre en place des patchs sans perdre les données déjà présentes. Autrement, tout le processus sera automatisé grâce à ce projet, soit l'équivalent de **3 jours/homme** minimum pour **chaque nouveau collaborateur** qui intègre un projet au sein de l'entreprise.

4. Management du projet

Les informations qui suivent concernent le développement de l'application noyau du projet responsable de génération des packages pour développeurs.

4.1. Méthode adoptée

Etant intégré à une équipe de développement au pôle *TEC* du projet *SNCF Fret*, je me dois de respecter une méthode agile propre à *Capgemini* qui permet un suivi continu de l'évolution du projet grâce à des « *Daily meetings* » chaque matin et des points d'avancement chaque semaine.

Le développement de l'application adopte un modèle **semi-itératif**, grâce à un cycle où les deux premières phases classiques, **expression des besoins** et **conception de la solution**, sont initialement exécutées avant d'entrer dans la notion d'itérations courtes depuis la troisième et quatrième phase du projet, relatives à l'étape de **construction et validation du produit**.

Les deux premières phases non itératives ont permis un premier contact pour la **définition des spécifications** demandées, **conception du projet**, **étude comparative** des outils à intégrer et **formation** sur ces produits et outils pour enfin passer à l'étape itérative.

A défaut **d'une semaine par itération**, constituée des différentes phases de développement : **réalisation**, **tests** et **validation** du livrable avec le responsable Capgemini du projet, le produit a connu une évolution progressive au fil des semaines.

4.2. Plan Assurance Qualité

Le *Plan Assurance Qualité (PAQ)* est rédigé et appliqué à l'initialisation du projet sous ma responsabilité, et validé par les parties concernées, à savoir l'encadrant.

4.2.1. Contraintes et risques majeurs

- Contraintes majeurs :

	Contrainte	Action mises en place
Contrainte de délai	L'application doit être terminée en l'espace de 4 mois	<i>Adopter une méthode agile itérative</i>
Contrainte de budget	Le projet doit être réalisé à 0 Dirhams budget	<i>Utiliser des outils libres</i>

Tableau 1-4-1 Contraintes majeurs

- Risques majeurs :

	Impact	Actions correctives
Panne inattendue du matériel	Perte de données	<i>Prévoir des commits sur un serveur SVN</i>
Bug logiciel	Programme instable	<i>Réaliser des analyses du code avec l'aide de programmes comme Checkstyle ou Findbugs</i>
Compétences insuffisantes	Développement interrompu	<i>Autoformation et recherches préalables sur les technologies et outils à utiliser</i>
Absence ou maladie	Non-respect du temps imparti	<i>Prévoir une longueur d'avance sur le planning prévisionnel</i>
Non compréhension des spécifications	Projet non conforme	<i>Planifier des points de validation avec l'encadrant pour suivre l'avancement</i>
Plateformes incompatibles	Utilisation limitée du projet	<i>Utiliser des technologies portables comme Java et gérer les différents systèmes et plateformes valables</i>

Tableau 1-4-2 Risques majeurs

4.2.2. Organisation

- Organisation de l'équipe

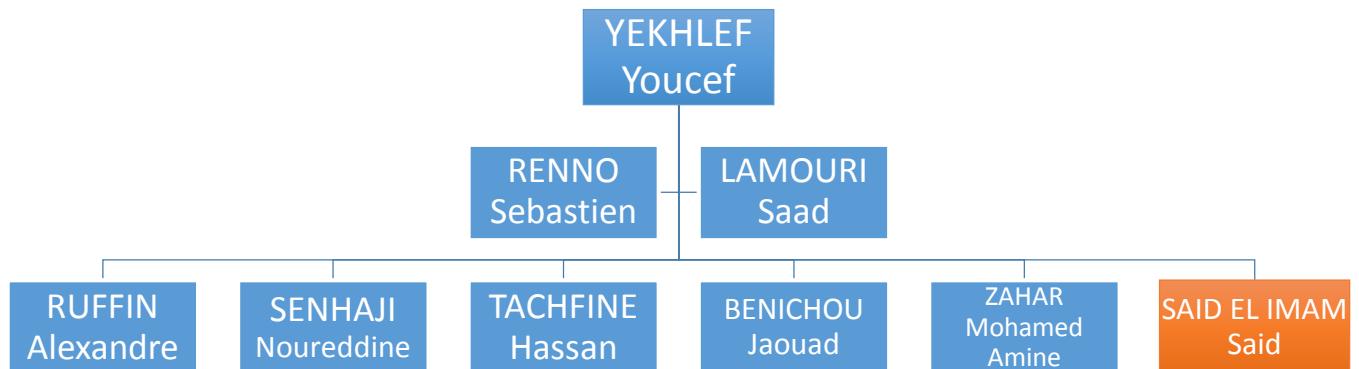


Figure 1-4-1 Organisation de l'équipe du pôle TEC

Le Pôle TEC est principalement constitué de consultants et architectes, ainsi que développeurs dont je fais partie, développant autour de la technologie Java, Java EE, langage de programmation principalement utilisé sur l'ensemble des applications SNCF Fret.

Mon projet s'adapte essentiellement à l'aide de deux encadrants externes, dont **LAMOURI Saad** sur le site de Casablanca, et **YEKHLEF Youcef** sur le site de Lille en France.

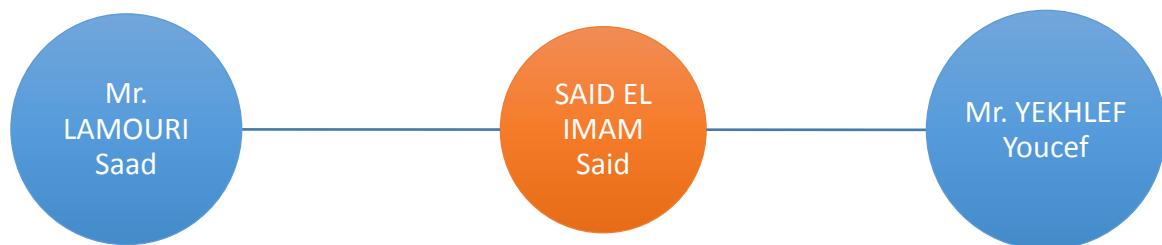


Figure 1-4-2 Organisation de l'équipe du projet

- Organisation de la logistique

Capgemini met à disposition des collaborateurs **un poste informatique connecté** sur le réseau du projet, ainsi qu'un **casque** et éventuellement un accès à la **Visio** et à une salle sous réservation pour communiquer avec l'équipe de Lille.

4.2.3. Démarche globale de production et d'assurance qualité

- Montage général du projet

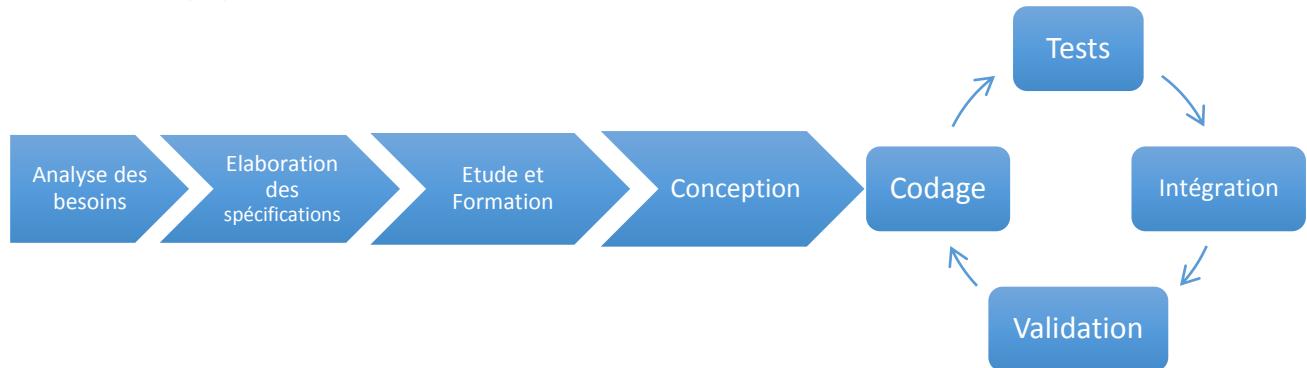


Figure 1-4-3 Montage des étapes générales du projet

Selon le graphe ci-dessus (*Figure 1-4-3*), on peut constater que le projet suit une méthode semi-itératrice grâce à une première étape préliminaire d'analyse et conception, suivie par un modèle itératif de réalisation et validation.

- Montage détaillé du projet par étapes

○ **Sorties**

Analyse des besoins

analyse détaillée des besoins du projet

- Cahier de charges

Elaboration des spécifications

définir les spécifications relatives au cahier de charges

- Spécifications fonctionnelles
- Spécifications techniques

Etude et Formation

réaliser une étude sur les outils/librairies à inclure pour le développement du projet, et s'auto-former dessus

- Etude comparative des outils à utiliser
- Compétences suffisantes

Conception

modéliser les différents diagrammes UML du projet ainsi que les maquettes graphiques

- Diagrammes UML
- Maquettes IHM

Codage

codage et mise en place des spécifications de l'itération en cours

- Itération

Tests

appliquer des tests unitaires et corrections de bugs sur l'ensemble des classes de l'itération

- Résultats de tests et corrections

Intégration

déployer l'itération pour utilisation autonome

- Livrable itération

Validation

validation du livrable par les parties concernées

- Livrable validé

Tableau 1-4-3 Sorties des étapes du projet

5. Le Planning prévisionnel

5.1. Répartition du travail

Un calendrier de progression du processus de développement du projet est défini selon les caractéristiques présentées ci-dessous :

Une itération	Une semaine
Une semaine	5 jours homme (<i>Lundi – Vendredi</i>)
Un jour homme	8 heures (<i>9h – 17h</i>)
Jours fériés	<i>Mercredi 1^{er} Mai 2013</i>

Tableau 1-5-1 Options du calendrier de planning

Cependant, selon les étapes définies pour le projet (*Figure 1-4-3*) ainsi que leurs priorités par rapport au livrable final, les durées de travail, représentées en heures et jours hommes, ont été réparties sur les tâches comme suit :

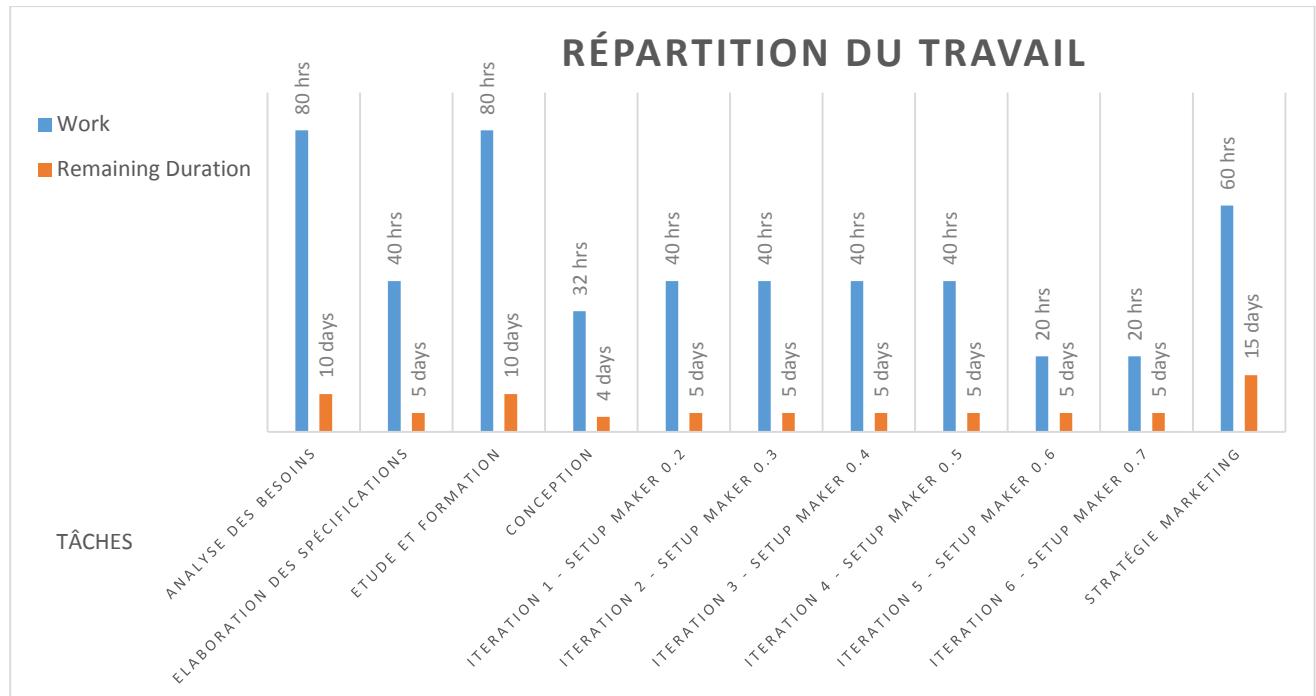


Figure 1-5-1 Graphique de répartition du travail sur les tâches

On remarque l'apparition d'une nouvelle tâche à la fin, la **stratégie marketing**, qui ne fait pas partie du développement mais qui s'annonce comme activité parallèle à partir de la **5^{ème} itération** pour publier le produit sur internet et avoir des retours et avis d'amélioration du logiciel de la part d'une large communauté.

Cette tâche étant exécutée en parallèle par moi-même, les deux dernières itérations ne reçoivent plus que **50 %** du travail.

5.2. Diagramme de Gantt

Le projet ayant débuté à la date du 25 Mars 2013, ci-dessous les différentes phases du projet au fil du temps :

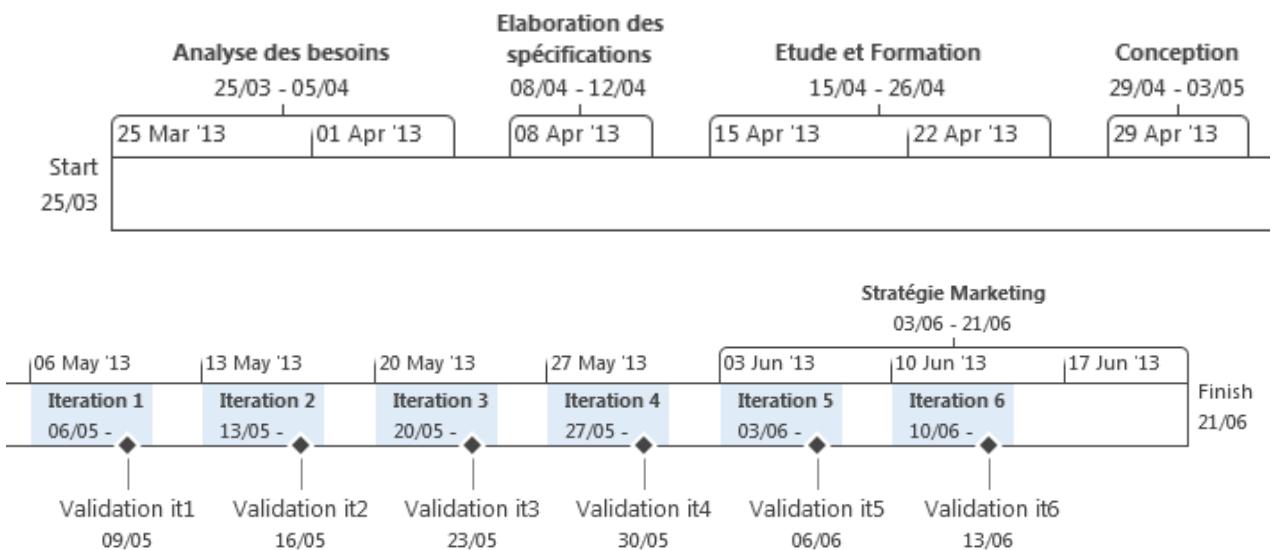


Figure 1-5-2 Ligne de temps du planning

Cette ligne de temps (*Figure 1-5-2*) précise la date de début et de fin pour chaque phase du projet, à partir du point de départ. De plus, on remarque les **jalons de validation** pour chaque itération, soit la dernière étape de l'itération, s'exécuter un jour avant la fin de celle-ci.

Ceci permet d'avoir un ***jour de risque*** pour les différentes étapes de l'itération (**codage, test et intégration**) pour ***éviter un dépassement de charge***, et ainsi ***valider le livrable à temps***.

Dans le diagramme de Gantt suivant (*Figure 1-5-3*), seule la première itération est détaillée à titre d'exemple ; Le reste des itérations respectent le même processus.

Chapitre 1 :
Projet et management du projet

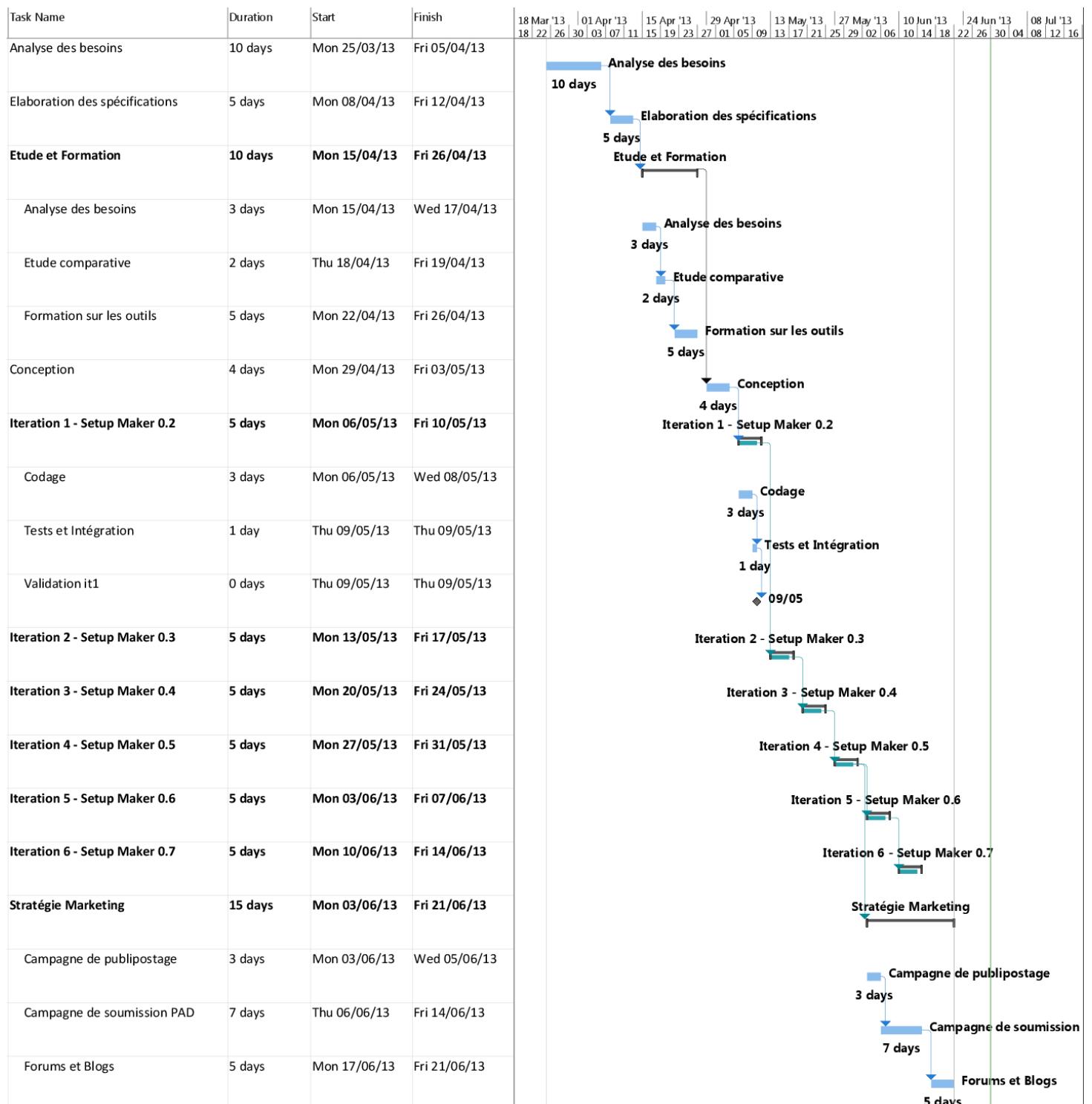


Figure 1-5-3 Diagramme de Gantt

5.3. Produits tangibles

Les différents produits livrables du projet se présentent comme suit :

- ✓ Package installateur global contenant tous les outils
- ✓ Générateur d'installateurs
- ✓ Application RIA de monitoring des installations sur le réseau

5.4. Autres prestations

Autres documents et produits divers à livrer, dont :

- ✓ Liste des outils inclus dans l'installateur par pôle
- ✓ Archive Eclipse prête à l'emploi, avec Workspace et plugins préinstallés

Conclusion

Ce chapitre a permis de déterminer le périmètre global du projet, la gestion de celui-ci selon une méthode modélisée spécialement pour les besoins du projet et une planification détaillée des tâches à réaliser, ainsi que les différentes fournitures livrables du projet.

Chapitre 2 :

Analyse et spécifications

Introduction

Ce chapitre présente en détail l'analyse du projet, situé au sein du service SNCF Fret, les applications et maquettes IHM, ainsi que les spécifications fonctionnelles et techniques propres aux cas d'utilisations et aux règles de gestion.

1. Analyse fonctionnelle et technique

1.1. Environnement fonctionnel

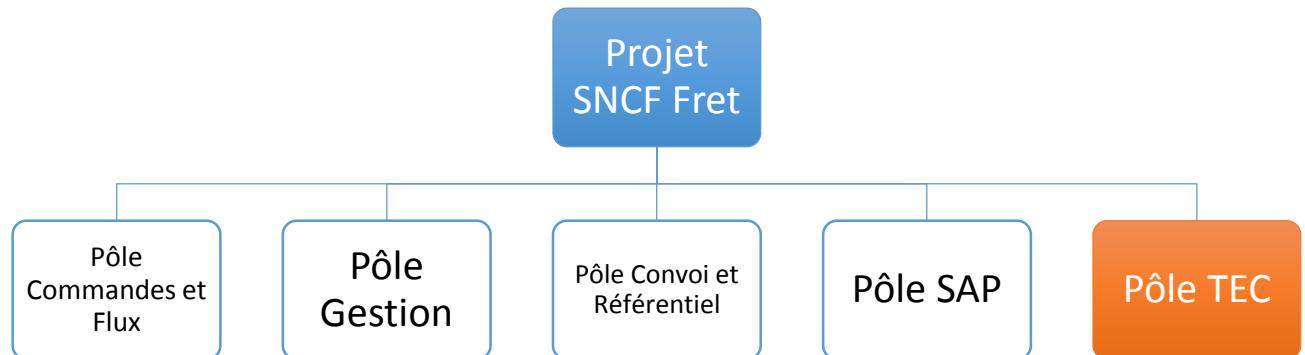


Figure 2-1-1 Environnement fonctionnel du projet

Le projet *SNCF Fret* est divisé en plusieurs pôles, chacun représentant une équipe responsable du développement, suivi et maintenance d'une ou plusieurs applications constituant une des parties du SI global :

- *Pôle Commandes et Flux* pour la gestion interne du processus d'une commande ;
- *Pôle Gestion* pour le suivi de celle-ci jusqu'au point de livraison ;
- *Pôle Convoi et Référentiel* pour les différentes ressources disponibles ;
- puis le *pôle SAP* pour la gestion ERP.

Concernant le **pôle TEC**, n'intervenant pas au niveau du système du client, l'équipe le composant s'intéresse essentiellement à garantir un meilleur environnement de travail pour tous les autres pôles, plus spécialement les développeurs, en fournissant un support ainsi qu'une gestion des postes et outils utilisés.

C'est à ce niveau du pôle TEC qu'intervient mon projet pour garantir une gestion plus simplifiée et un suivi des différents développeurs sur les pôles du projet à travers **trois applications à développer** (1.2voir §2 1.2).

Ainsi, **l'installeur généré** devra regrouper **tous les programmes et outils utilisés** lors du développement d'une des applications SNCF Fret, ainsi que **leurs paramétrages** si nécessaire, en prenant en compte les compatibilités entre différentes versions de ceux-ci et leurs librairies respectives.

1.2. Présentation des applications du projet

Dans cette section, je présente les différents modules du projet à développer et qui sont : SNCF Package, DCP Setup Maker, DCP Dashboard.

1.2.1. SNCF Package

Le package développeur, livrable principal du projet, permet l'accès aux fonctionnalités basiques d'un installateur en plus de quelques améliorations, à savoir :



- « **Choix d'une langue** » : Le développeur a accès à un choix multiple de langues d'affichage pour les écrans d'assistance du package installateur.
- « **Choix d'un groupe d'installation relatif au pôle en question** » : Le développeur a ensuite un choix entre plusieurs groupes d'installations, chacun représentant un ensemble d'applications, programmes et archives propres au pôle de l'intéressé.
- « **Sélection des packs facultatifs à installer** » : Une possibilité s'ajoute à l'utilisateur de sélectionner ou désélectionner les packs qu'il désire déployer sur sa machine, sauf pour ceux requis qui seront grisés, et donc impossible à désélectionner.
- « **Installer raccourcis vers applications/chemins installé(e)s** » : Une fois l'installation de l'ensemble des packs terminée, ainsi que l'exécution d'autres, un invite à rajouter des raccourcis sur le menu démarrer ou/et sur le bureau s'affiche pour l'utilisateur avec le choix de les désactiver.
- « **Générer script d'installation automatisée du processus définis** » : A la fin de l'installation du package, un dernier écran de confirmation présentant une fonctionnalité de générer un script automatisé de l'installation réalisée par ce package qui enregistrera tous les choix réalisés par le développeur ayant installé ce package pour une possibilité future d'installer le même package, avec les mêmes configurations et choix réalisés directement par le script, sans avoir à repasser par les différentes étapes de l'installation à chaque poste.

1.2.2. DCP Setup Maker

L'application de couche supérieure au package développeur, **DCP Setup Maker**, dont le préfixe (*DCP > Developers' Company Package*) traduit ‘Package Développeur pour Entreprise’, permet essentiellement de générer le package présenté ci-dessus, de façon dynamique et configurable, grâce à une interface intuitive et ergonomique (*Figure 2-1-2*), pour un meilleur contrôle sur les packs à inclure et les différents paramétrages du package.

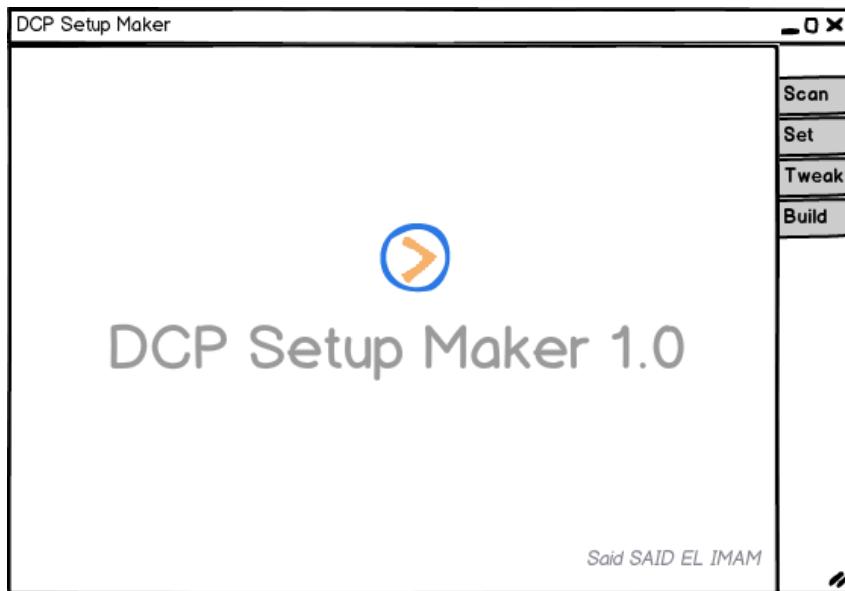


Figure 2-1-2 Interface principale du *Setup Maker*

Les principales vues se présentent comme suit :

1. Scan : Scanner un répertoire contenant les fichiers et programmes à inclure dans le package

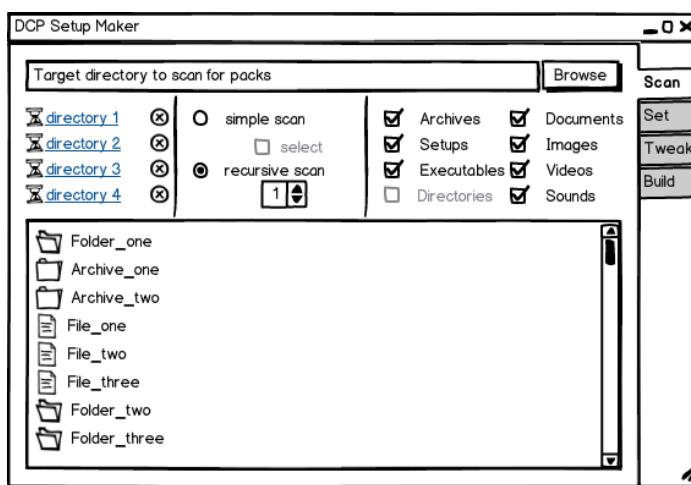


Figure 2-1-3 tab Scan du *Setup Maker*

La fenêtre initiale demande **un chemin vers un dossier** contenant les fichiers à empaqueter comme entrée, pour **afficher son contenu** de façon **singulière** ou **récursive**, avec la possibilité de **filtrer** quelques types de fichiers à travers des boutons check.

Le **mode de scan simple** présente une possibilité de choisir quelques packs seulement du dossier scanné par **sélection**, tandis que le **mode récursif** permet d'attribuer une valeur de **profondeur du scan** à travers les répertoires. Une fois la limite définie atteinte, les répertoires non accédés sont considérés comme packs.

Les **filtres** se déparentagent en deux colonnes, dont la première pour ceux de type **pack** (*archive, exécutable, dossier*), et la deuxième pour **les fichiers médias** (*document, son, image, vidéo*).

Parallèlement, chaque dossier scanné est **enregistré** pour un futur accès plus rapide, un nombre qui peut monter à cinq répertoires sous forme d'une liste **FIFO**, avec possibilité **d'effacer** un répertoire de l'historique.

Entrées	Dossier de packs à scanner Fichier de configuration Fichier de sauvegarde
----------------	---

Tableau 2-1-1 Entrées de l'interface Scan du Setup Maker

2. Set : Paramétrage des packs scannés et mise en hiérarchie à travers des groupes d'installation

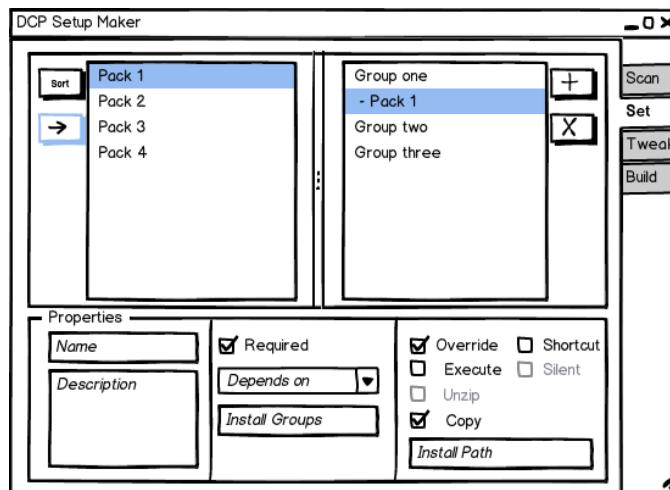


Figure 2-1-4 tab Set du Setup Maker

La deuxième étape est la plus importante, puisqu'elle permet d'établir tous les **paramétrages** nécessaires pour les fichiers scannés dans la fenêtre précédente.

Les paramètres se décomposent en trois parties (organisées sous forme de 3 colonnes dans les propriétés) qui sont :

- **Informations** : Les données du pack comme le nom et la description, affichées lors de la sélection de packs dans le package durant l'installation.
- **Options** : Attributs divers du pack pour le situer par rapport aux autres packs du point de vue dépendances, groupes d'installation et obligation d'installer.

- **Déploiement** : Durant l'installation, et une fois les packs choisis, l'étape déploiement commence à traiter chaque pack selon les paramétrages dans cette partie qui définissent le chemin cible, le mode de déploiement, actions pour le cas d'un pack déjà installé, création d'un raccourci, etc.

Une autre fonctionnalité **d'organisation par hiérarchie** se présente à l'utilisateur pour structurer les packs selon des **groupes**, que l'on pourrait utiliser pour créer **une dépendance** d'un ou plusieurs packs par rapport à un groupe de packs ou bien pour structurer le déploiement d'un ensemble de fichiers sous forme de groupes et sous-groupes représentant dans ce cas **les dossiers cibles** de l'installation.

Tous ces paramétrages doivent bien entendu être **applicables pour un seul comme pour un ensemble de packs sélectionnés** pour une gestion plus efficace et rapide.

Une dernière fonctionnalité très importante implémente un système de **priorité**. Ainsi les packs peuvent être triés par l'utilisateur selon **des modes de tris prédéfinis** agissant sur les attributs des packs, dont le mode '**custom**' pour un tri personnalisé.

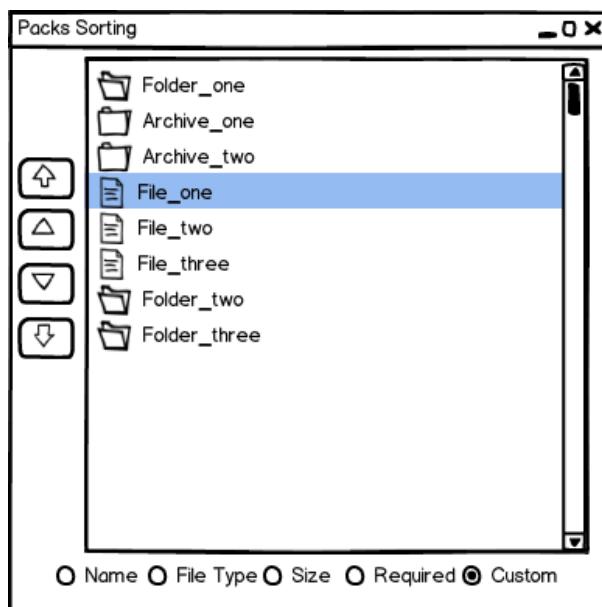


Figure 2-1-5 tab Set, fonction Sort du *Setup Maker*

Ce **tri de priorité** permettra donc de définir **l'ordre d'exécution des packs** durant l'installation si quelques packs doivent s'installer avant d'autres, chose qui est très fréquente dans le cas d'un package global d'outils pour développeurs.

Entrées

Packs scannés, choisis
Structure des dossiers

Tableau 2-1-2 Entrées de l'interface Set du *Setup Maker*

3. Tweak : Retouches et configurations du package en entier

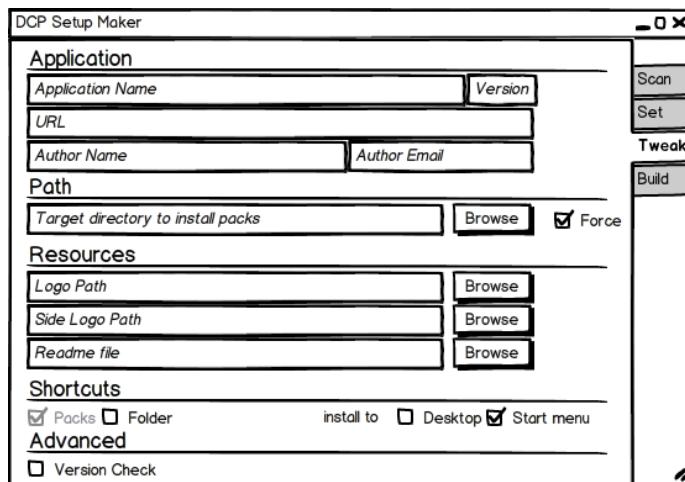


Figure 2-1-6 tab Tweak du Setup Maker

La troisième étape, et avant-dernière, complète les quelques informations restantes essentielles pour la création du package à travers une interface sous forme d'un formulaire à remplir avec les données demandées.

Ces informations sont décomposées en plusieurs parties dont :

- **Application** : les informations de l'application comme le *nom, version, url*.
- **Auteur** : Quelques coordonnées facultatives concernant l'*auteur* du package.
- **Chemin d'installation** : Préciser le *chemin d'installation* du package par défaut, et possibilité de *forcer* le choix de ce chemin, chose qui va éliminer la demande du chemin lors de l'installation.
- **Ressources** : Inclure quelques *ressources* au package comme *images, fichier d'informations, licence*.
- **Raccourcis** : Quelques options de *raccourcis* pour définir l'*emplacement* sur la machine.
- **Avancé** : Enfin quelques *options avancées* comme la *vérification de version installée sur le système* à l'initialisation de l'installation pour éviter de le réinstaller deux fois.

Entrées	Nom d'utilisateur de la machine Packs avec raccourci activé
---------	--

Tableau 2-1-3 Entrées de l'interface Tweak du Setup Maker

4. Build : Compilation et construction du package final

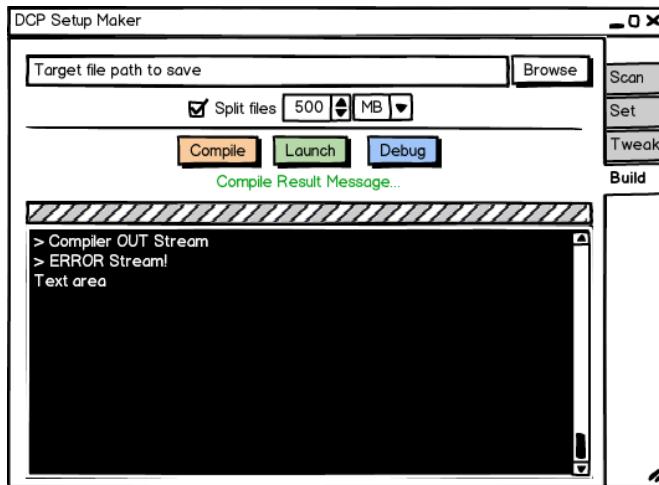


Figure 2-1-7 tab Build du Setup Maker

La dernière fenêtre complète le processus de création du package en **compilant** le tout vers le **chemin** du fichier package donné, tout en affichant un **log** des instructions réalisées durant la compilation pour tracer le processus si une erreur est générée.

Une fois le package généré, l'utilisateur a une possibilité de le **lancer**, ou le **débugger** depuis l'application pour avoir un log détaillé au fur et à mesure que l'on avance dans l'installation.

Pour le cas d'un package contenant plusieurs packs de grande taille, le package pourrait avoir une taille de fichier énorme, ce qui présente une problématique pour graver le package sur un disque de taille limitée comme un *CD* ou *DVD*.

Pour remédier à ce problème, générer le package sous forme d'un **ensemble de fichiers de taille limitée** est possible. Ce qui permettra de traiter un ensemble de fichier, que l'on pourra exécuter comme étant un seul package en regroupant tous les fichiers dans un même dossier et en exécutant le premier fichier exécutable.

Entrées	Nom et Version application Chemin racine de l'application
Sortie	Package

Tableau 2-1-4 Entrées et sortie de l'interface Build du Setup Maker

1.2.3. DCP Dashboard

En remontant une autre couche supérieure, on retrouve **DCP Dashboard**, l'application web de surveillance et statistiques réseau, qui permet un contrôle et suivi des installations réalisées sur les différentes machines de l'entreprise, ou des postes appartenant à un projet quelconque, comme pour notre cas, le projet SNCF Fret (*Figure 2-1-8*).

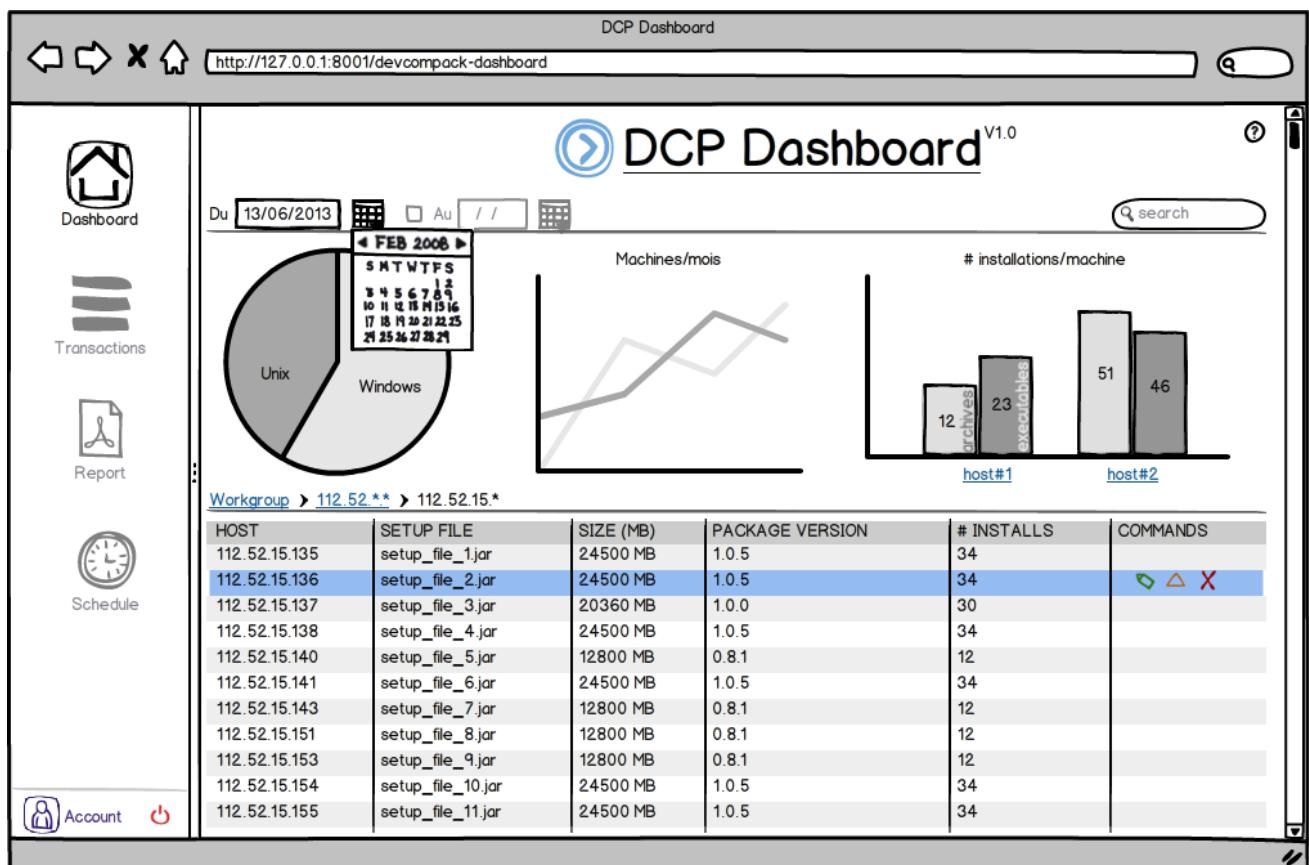


Figure 2-1-8 Interface principale du *Dashboard*

Cette application web est définie comme étant une application **serveur**, se connectant aux différents **clients**, à savoir les packages générés, en enregistrant ainsi les informations écrites par ceux-ci lors de l'installation dans la base de registre, et qui concernent les **packs installés, version, mises à jour**, etc.

1.3. Environnement technique

L'environnement technique du projet, présenté sous forme de spécifications techniques globales ainsi que celles relatives à chacun des produits livrables du projet, est défini à travers les informations suivantes :

Configuration du matériel de développement	Windows 7 Professionnel (64 bits)
Environnement de développement	Eclipse Indigo
Technologie de développement	Java
Configuration du matériel d'exploitation	Windows (32/64 bits), Unix, Mac OS
VCS	Subversion
Méthode	SCRUM
<hr/>	
SNCF Package	
Langues	Français, Anglais
Programmes utilisées	<ul style="list-style-type: none"> • DCP Setup Maker - <i>Création package</i> • DCP Dashboard - <i>Monitoring réseau</i>
<hr/>	
DCP Setup Maker	
Langues	Anglais
Langage de développement	J2SE 6.0
Librairies utilisées	<ul style="list-style-type: none"> • IzPack 4.3.5 - <i>Setup Builder</i> • Apache Pivot 2.0.2 - <i>GUI</i> • Apache Ant 1.9.0 - <i>BAT</i> • Staxmate 2.2.0 - <i>JAXP</i> • TrueZip 7.7.1 - <i>VFS</i>
Système de gestion de données	XML Fichiers binaires
VCS	Subversion
<hr/>	
DCP Dashboard	
Langage de développement	J2EE 6.0
Serveur de déploiement	Apache Tomcat 7.0.41
Librairies utilisées	<ul style="list-style-type: none"> • Vaadin 7.0.7 - <i>RIA Framework</i> • Apache Maven 2 - <i>BAT</i> • Staxmate 2.2.0 - <i>JAXP</i> • TrueZip 7.7.1 - <i>VFS</i>
Application client	DCP Setup Maker
Système de gestion de données	XML JSON
VCS	Git

Tableau 2-1-5 Environnement technique du projet

2. Présentation des acteurs

Dans mon projet, j'ai identifié trois acteurs qui sont : le **développeur**, l'**administrateur** et le **modérateur**, dont les différents rôles sont présentés dans le cas d'utilisation global (*Figure 2-3-1*).

2.1. Développeur

L'utilisateur final du package, s'occupera de l'**installation du package** sur sa machine, ainsi que la mise à jour de celui-ci.



Sa machine enregistrée sur le réseau une fois le package installé est suivie par le modérateur au niveau des outils et programmes installés, ceux qui ont besoin d'être mis à jour, ceux incompatibles avec d'autres, etc.

2.2. Modérateur

Responsable de **surveillance réseau**, utilisateur de l'application réseau **Dashboard** de monitoring.



Cet acteur jouera un rôle crucial dans la maintenance des machines des développeurs ainsi que les packs disponibles sur ceux-ci pendant tout le long d'un projet en cours.

Le modérateur peut hériter des cas d'utilisation d'un développeur, si le modérateur est en effet un développeur d'origine, lui permettant ainsi d'inclure sa machine au réseau d'installations.

2.3. Administrateur

Responsable de **génération des packages**, l'**administrateur** est l'acteur principal, utilisateur de l'application noyau de création du package. Il devra suivre une formation sur l'utilisation de l'application **Setup Maker** pour générer des packages ou patchs de mise à jour selon le besoin et puis les affecter ensuite aux développeurs pour les installer.



L'administrateur peut hériter des deux acteurs précédents pour ainsi réaliser toutes les tâches et permettre une réduction des coûts sur l'utilisation de ressources supplémentaires en affectant toutes les tâches de gestion des installations à cet acteur seul.

3. Présentation synthétique des cas d'utilisation

Dans la figure suivante, je présente le cas d'utilisation global du projet.



Figure 2-3-1 Cas d'utilisation global

Acteur	Cas d'Utilisation ¹	Description
Développeur	Installer package	Lancer l'installation du fichier package
	<ul style="list-style-type: none"> • choisir packs à installer ○ générer script automatisé 	Choisir des programmes/groupes pendant le processus d'installation Générer un script d'installation automatisée à la fin de l'installation, sans avoir à repasser par les mêmes étapes
	Appliquer patch mise à jour	Installer patch de mise à jour pour le package déjà installé sur la machine
Modérateur	Monitoring sur réseau	Surveiller les différentes installations appliquées sur les postes et machines du réseau du projet.
Administrateur	Générer package <ul style="list-style-type: none"> • gestion des packs • gestion groupes • gestion package ○ Créer patch mise à jour 	Créer package englobant des packs à installer Configuration de chaque pack à installer Attribuer les packs à des groupes d'installation Définir les informations et configuration du package Générer un patch de mise à jour à partir d'un ancien package déjà généré

Tableau 2-3-1 Description des cas d'utilisation

¹ • → include | ○ → extend

4. Présentation détaillée des cas d'utilisation

- **Développeur**

- *Installer package*

Le développeur peut lancer l'installation ainsi qu'interagir avec les différents écrans de l'installation.

- *Choisir packs à installer*

Pendant l'installation, le développeur a le choix de choisir un groupe d'installation relatif à son pôle puis, parmi les packs facultatifs non grisés, ceux qu'il désire installer sur sa machine.

- *Générer script automatisé*

Une fois l'installation terminée, il est capable de générer un script contenant des informations sur les choix effectués pendant l'installation.

- *Appliquer patch mise à jour*

Il est possible d'appliquer un patch sur une installation déjà réalisée sur la machine pour modifier les programmes déployés, ou en ajouter d'autres.

- **Modérateur**

- *Monitoring réseau*

Le modérateur a la capacité d'avoir un accès aux ressources machines ainsi qu'aux différentes installations réalisées sur celles-ci, pour permettre un suivi des versions installées sur les postes des développeurs.

- **Administrateur**

- *Générer package*

L'administrateur a la capacité primordiale de pouvoir générer un package contenant tous les outils, informations et paramétrages utiles pour chaque pôle, et chaque poste développeur.

- *Gestion des packs*

Le package doit pouvoir inclure une certaine gestion des packs à inclure dans celui-ci avec une affectation des propriétés de ceux-ci pour le déploiement.

- *Gestion groupes*

Les packs gérés sont, de même, organisés à travers une hiérarchie de groupes pour les différents types d'outils et groupes d'installation pour les pôles.

- *Gestion package*

Ces packs et groupes sont inclus dans un package global, défini par un nom et une version, et configurable selon des paramètres accessibles à l'administrateur pour personnaliser l'aperçu de celui-ci ainsi que les fonctionnalités qu'il offre en plus.

- *Créer patch mise à jour*

La création d'un package peut aussi mener à une création d'un patch, de numéro de version supérieur à celui du package à mettre à jour, contenant les mêmes packs et groupes avec quelques modifications de version pour certains packs, modification de structure, ou ajout de nouveaux outils.

Conclusion

Ce chapitre a permis de définir le périmètre du projet du côté fonctionnel ainsi que technique, tout en présentant les spécifications et cas d'utilisation de manière synthétique et détaillée.

Ainsi, on remarque que ce projet peut être utilisé par trois types d'acteurs, le Développeur, Modérateur et Administrateur qui s'unissent pour une gestion parfaite des machines du parc informatique.

Chapitre 3 :

Conception

Introduction

Ce chapitre présente les différents diagrammes qui constituent la conception du projet, tout en mettant le point sur les connections et dépendances entre les applications.

1. Conception globale

Les dépendances entre les différentes applications du projet sont représentées comme suit :

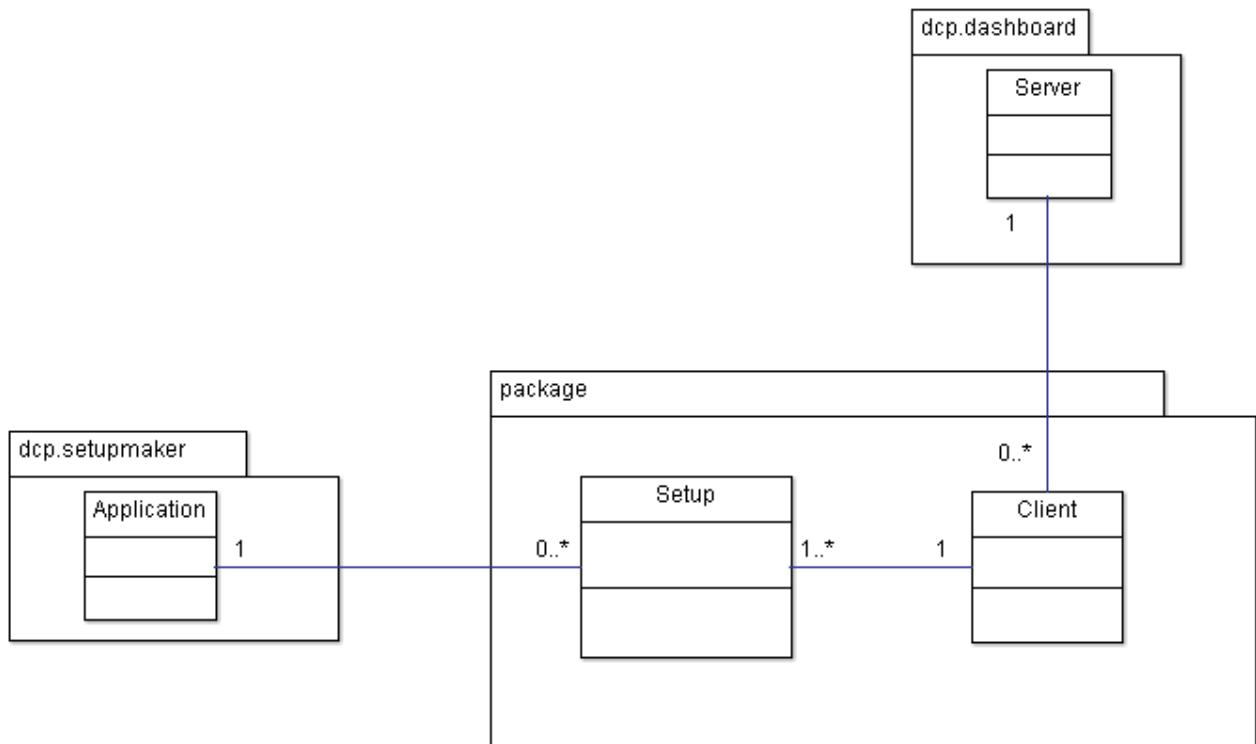


Figure 3-1-1 Diagramme global des dépendances entre les applications

Ainsi, l'application '**DCP Setup Maker**' génératrice d'installateurs, exécutée en premier, donnera lieu à un ensemble de setups, chacun contenant un programme *client* qui sera déployé sur la machine lors de l'installation, sauf pour les patchs qui ne contiendront aucun *client* puisqu'ils mettent à jour une installation déjà réalisée sur la machine, et donc un *client* est déjà présent.

Le *serveur* pourra se connecter à plusieurs *clients* déployés sur différentes machines du serveur de l'entreprise, tandis que le *client* ne peut se connecter qu'à un et un seul *serveur*. Ce *client* sera donc le point d'entrée pour l'application *serveur*, **DCP Dashboard**, à travers un numéro de port réservé pour l'application.

Un tableau résumant les étapes qui connectent les trois applications est présenté ci-dessous :

Application	Action	Résultat
DCP Setup Maker	Générer package	✓ Package installateur créé
Package	Installer	✓ Client déployé sur la machine ✓ Machine enregistrée sur le serveur
DCP Dashboard	Surveiller	✓ Serveur connecté aux programmes clients

Tableau 3-1-1 Résultat des actions globales

On peut résumer cette suite d'actions sous forme d'un diagramme d'activité à trois entrées/sorties, chacune correspondante à l'un des acteurs, à savoir par ordre de gauche à droite : *l'administrateur*, le *développeur* et le *modérateur*.

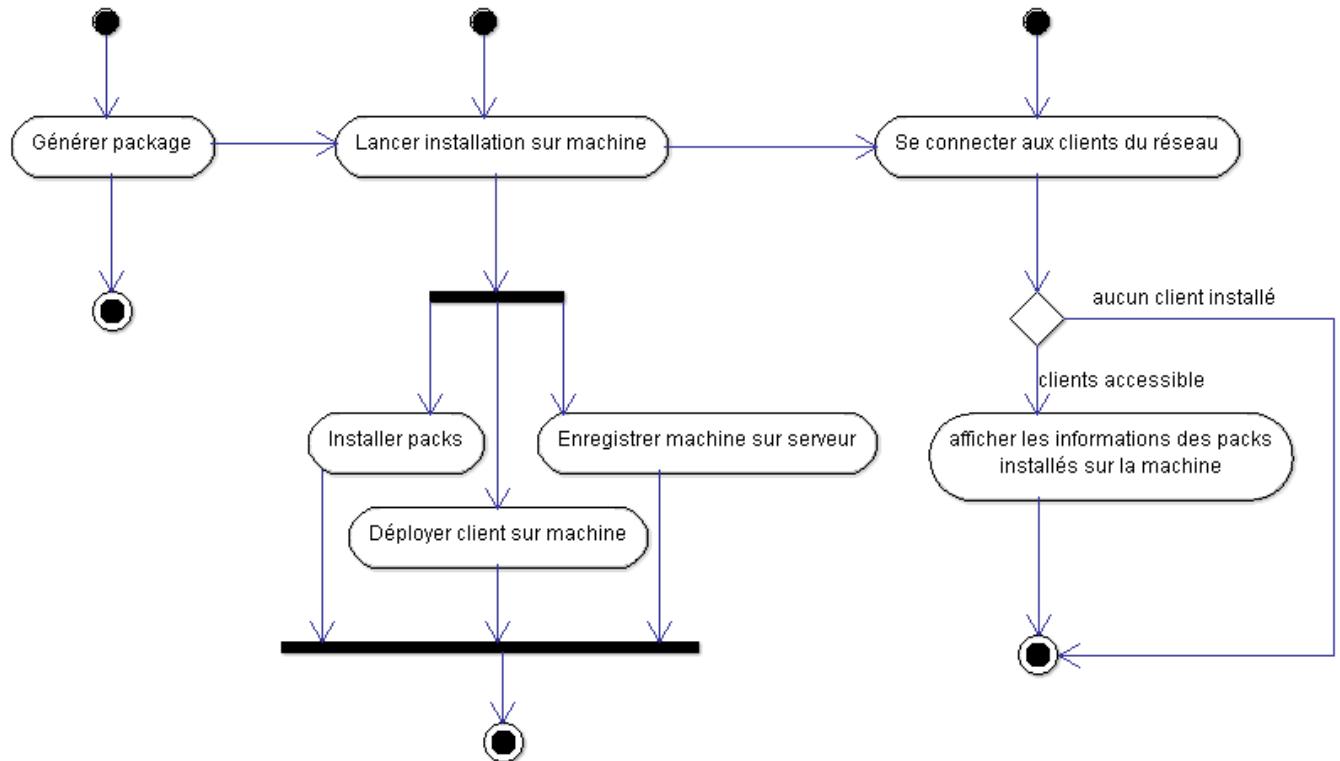


Figure 3-1-2 Diagramme d'activité global

Le **package** étant donc le point essentiel, faisant l'objet de l'utilisation du générateur d'installateur et permettant de connecter les machines avec le serveur contenant l'application serveur. A remarquer que dans le diagramme, il appartient à l'administrateur de passer par tous les processus, et ainsi utiliser les trois applications si nécessaire.

Pour le cas du projet *SNCF Fret*, une organisation des programmes déployés sur la machine a été conçue, sur un chemin, pour être standardisée sur toutes les machines des développeurs. Celle-ci concerne la structure de placement des outils empaquetés à travers les dossiers :

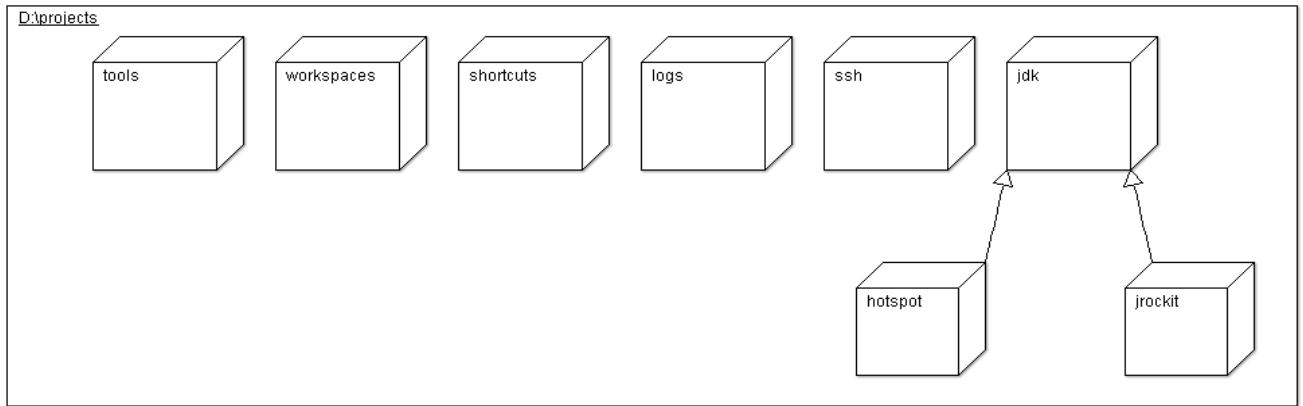


Figure 3-1-3 Structure de déploiement - Package SNCF Fret

2. Diagrammes de classe

2.1. Générer package

Les classes représentant l'application primaire du projet pour la génération de l'installeur sont définies sur le diagramme qui suit :

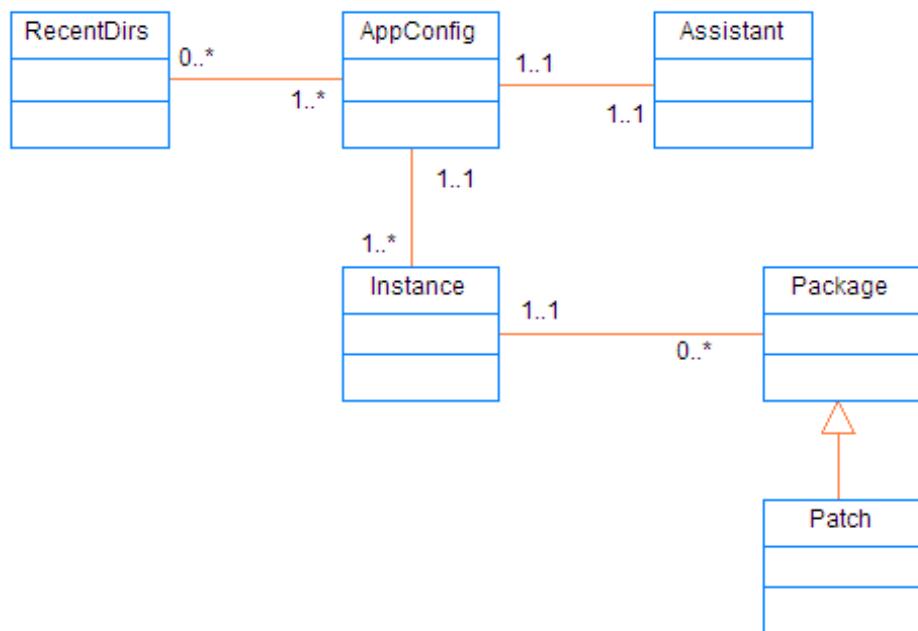


Figure 3-2-1 Diagramme de classe - Application

L'**application**, sous forme d'un ensemble **d'instances** chacune s'exécutant sur une JVM à part, dispose d'une **configuration** qui permet d'enregistrer les **informations et paramètres d'utilisation**, ainsi qu'une possibilité de générer un ou plusieurs **Setups** (installeurs) ou **Patchs** relatifs à un installeur déjà créé.

2.2. Installer

Un installateur est composé de **groupes d'installation**, dont chacun est composé de **groupes**, dont chacun est composé de **packs**. Sans oublier une **configuration** de cet installateur concernant les différentes ressources à rajouter comme le logo, licence, fichier d'information, chemin d'installation, etc.

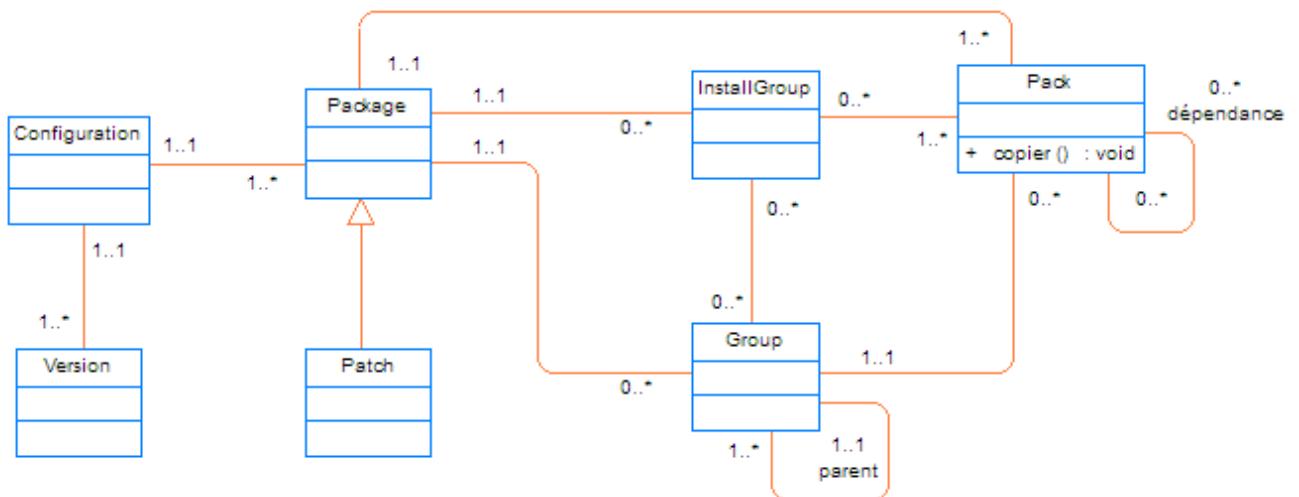


Figure 3-2-2 Diagramme de classe - Setup

La **configuration** permet de définir des images à intégrer dans les écrans d'installation, le chemin de déploiement par défaut (ou obligatoire), quelques ressources supplémentaires ainsi qu'une version du package.

La **version**, gravée dans la base de registre de Windows, joue un rôle très important pour la génération d'un **patch** futur, sous forme d'installateur aussi avec une version incrémentée et une modification partielle des programmes installés à mettre à jour ou un ajout de nouveaux packs à l'installation précédente.

- ✓ *Le patch doit absolument être exécuté sur une machine qui contient déjà une installation complète.*
 - ✓ *Une installation peut avoir plusieurs patchs.*
 - ✓ *Un patch ne met à jour qu'une seule installation pour laquelle il a été créé.*

Chaque **Pack** est représenté physiquement sur la machine par un ou un ensemble de **fichiers** empaquetés dans une **archive**, un **exécutable** ou un simple **répertoire**.

Un mode de déploiement est donc implémenté pour tous les types de packs cités, à savoir la possibilité de **copier** le pack, le **décompresser** ou **l'exécuter** durant l'installation.

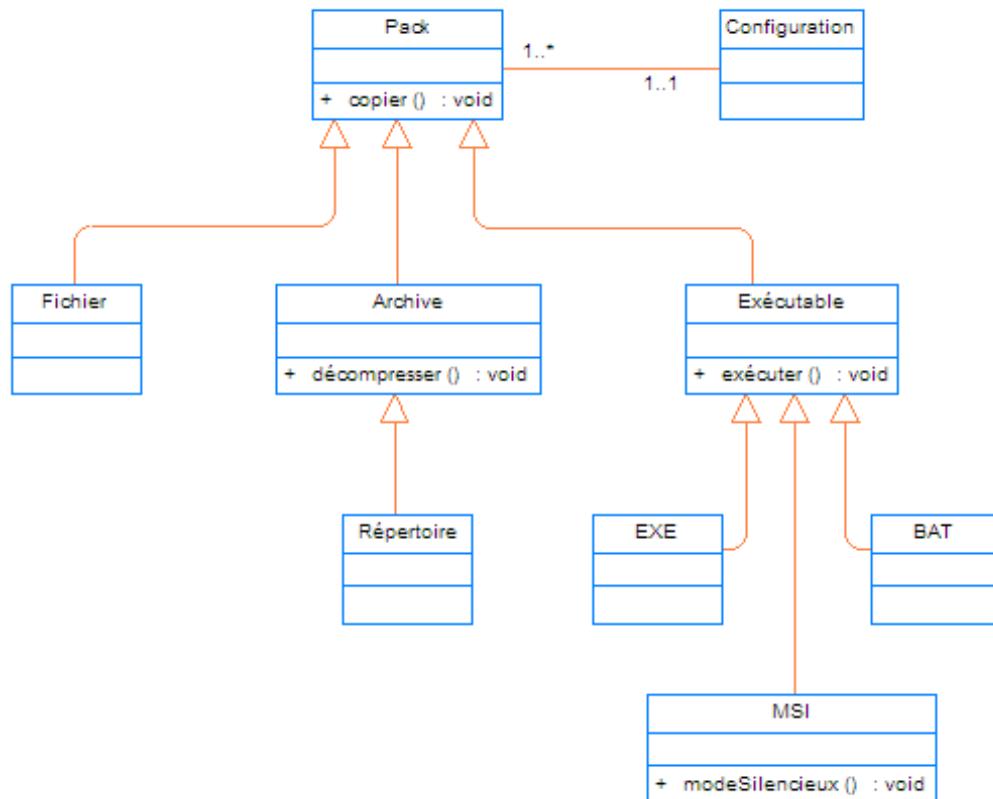


Figure 3-2-3 Diagramme de classe – *Pack*

Tous les types de packs héritent donc de la fonctionnalité basique de déploiement, la **copie**.

Le **répertoire** hérite des fonctions de **l'archive** vu qu'il est aussi considéré comme étant un conteneur de fichiers qui peuvent être extraits du dossier vers un chemin quelconque.

L'exécutable peut faire allusion à trois types de fichiers qui sont le fichier ***.exe**, ***.bat** et ***.msi** pour les installateurs Windows qui bénéficient d'une possibilité de s'exécuter en **mode silencieux**, sans présenter d'écrans d'interaction.

Tous ces modes d'installation sont **configurés** lors de la création du package, et appliqués au niveau de l'installation.

2.3. Surveiller

Une fois notre package généré et installé sur les machines des développeurs, on passe à la partie surveillance de réseau, dont le diagramme de classes se présente comme suit :

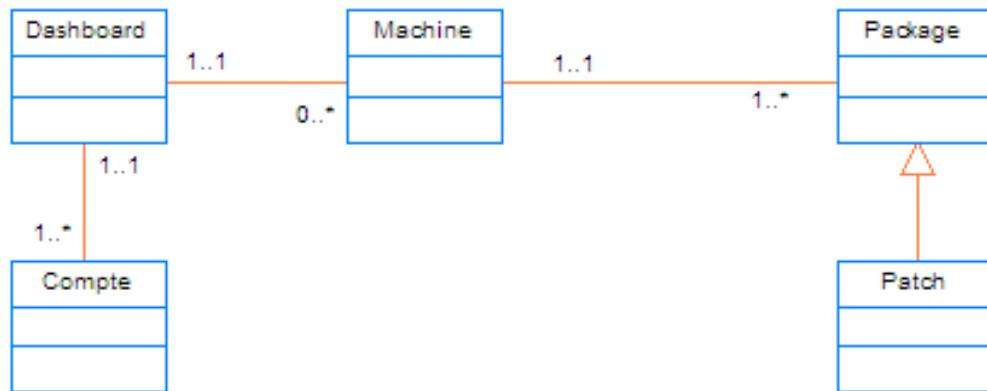


Figure 3-2-4 Diagramme de classes du *Dashboard*

Ainsi, l'application Dashboard se connecte à plusieurs machines hôtes, où le client est déployé, permettant l'accès à nombre d'informations sur la machine en relation avec l'installation ainsi que les mises-à-jour exécutées sur celle-ci.

Chaque machine a donc un ou plusieurs packages installés dessus, chacun ayant déployé bon nombre de **packs**, et des **patches** appliqués sur ces packages.

Diagramme global

On retrouve donc le diagramme de classes englobant toutes ces parties (*Figure 3-2-5*) ;

Ce diagramme rajoute la déclaration de quelques *interfaces* générales pour faciliter l'implémentation de quelques classes comme pour celles de **configuration**.

On remarque aussi l'ajout d'une classe **RegistryKey** relative à la clé de registre enregistrée dans la base de registre de Windows, cette clé est associée à la **version** d'une configuration, vu qu'elle est différente d'une version à l'autre. De même, la classe **Raccourci** permet l'ajout d'un raccourci pour le package et les packs respectifs.

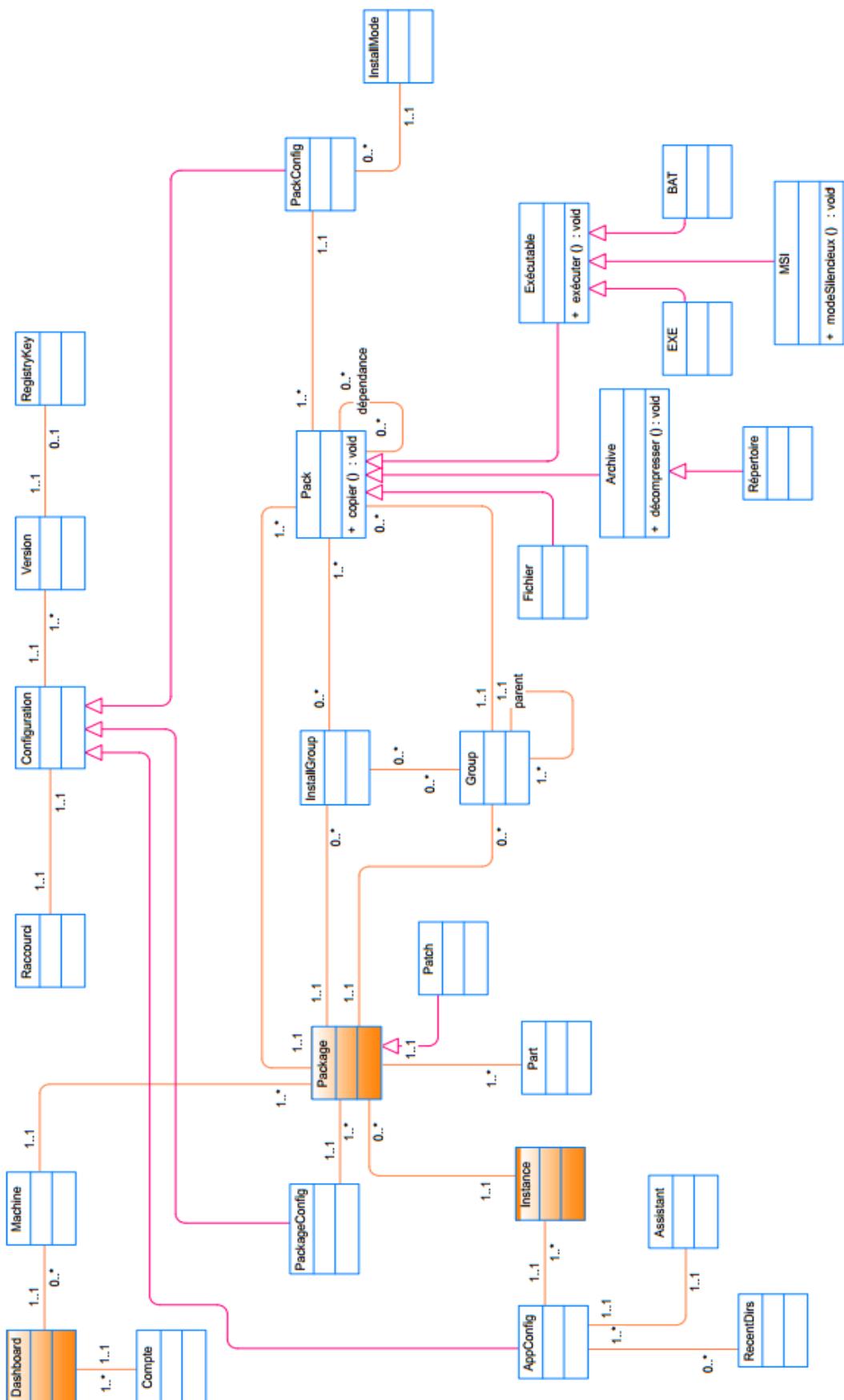


Figure 3-2-5 Diagramme de classes global

3. Diagramme d'activité

Le diagramme d'activité suivant détaille le processus d'installation du package généré :

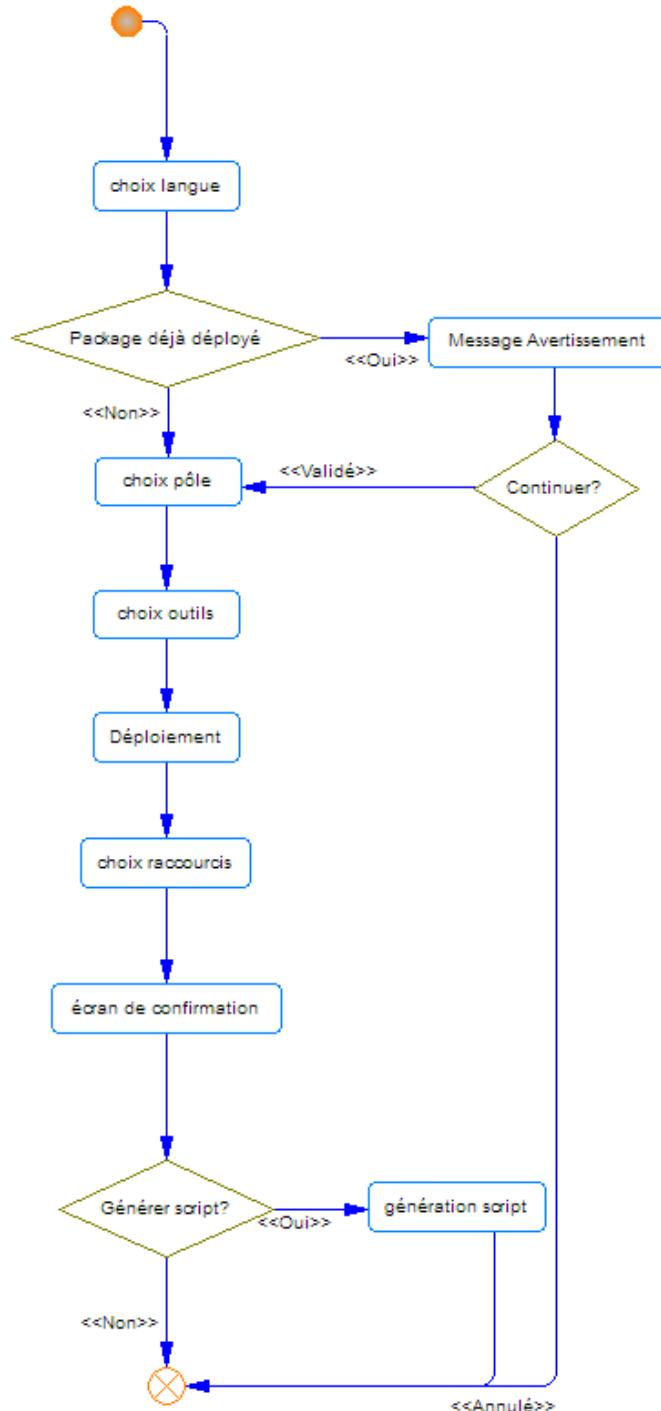


Figure 3-3-1 Diagramme d'activité du Package

4. Diagrammes de séquence

Ces deux diagramme de séquence expliquent le processus de connexion du package client au serveur ainsi que l'utilisation de cette ressource par l'application Dashboard sur le serveur pour lister les machines connectées à l'ouverture de celle-ci.

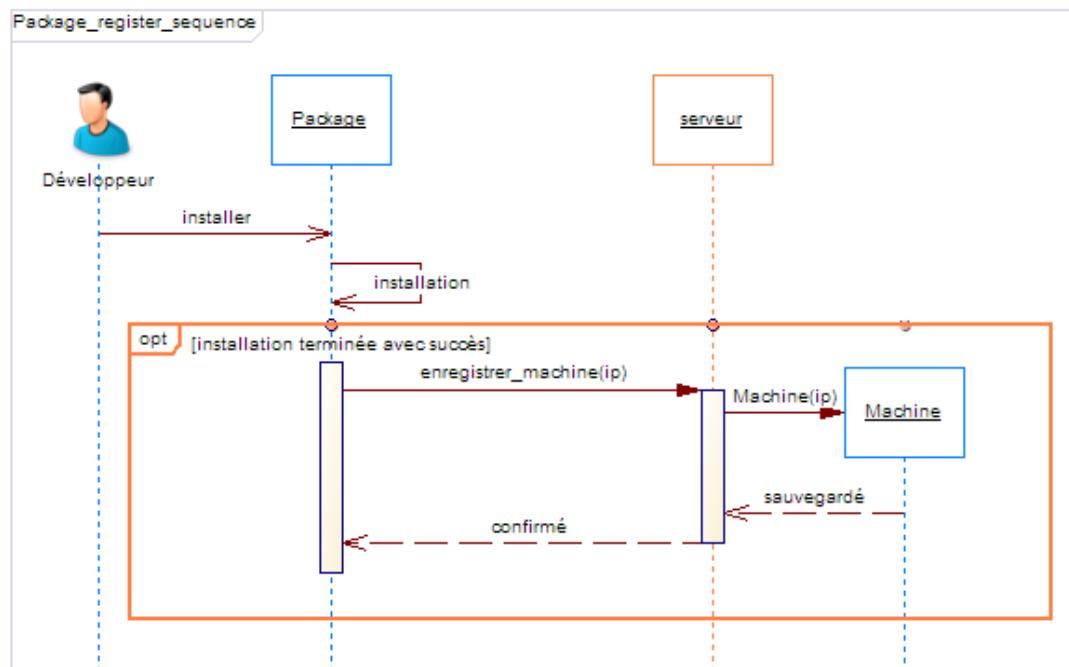


Figure 3-4-1 Diagramme de séquence, connexion client

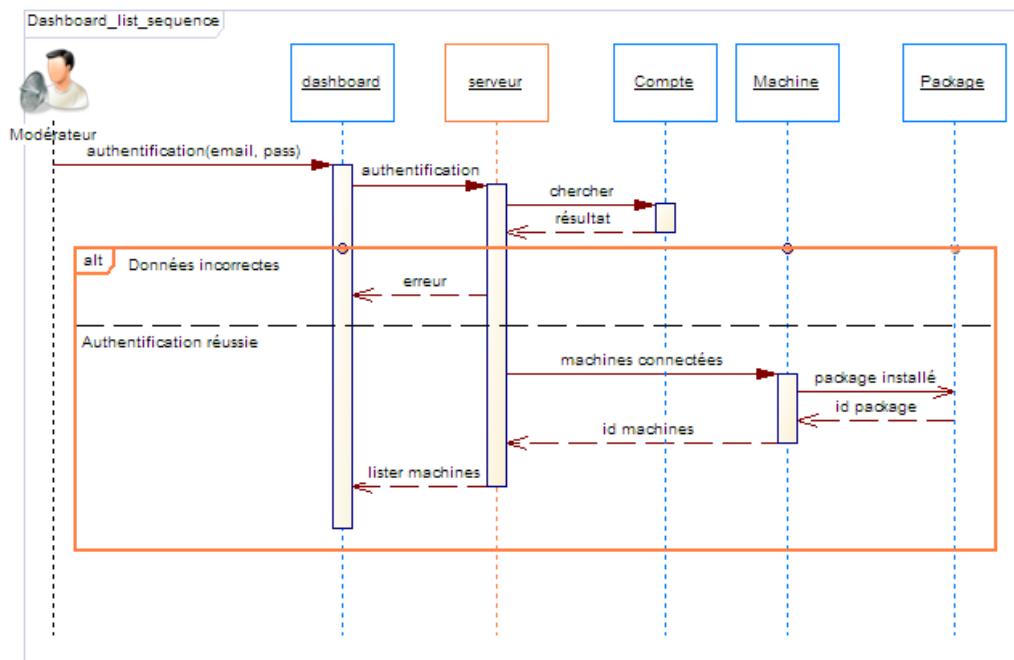


Figure 3-4-2 Diagramme de Séquence, connexion serveur

Conclusion

Ce chapitre a décrit en détail les informations de conception du projet, ainsi que les quelques diagrammes le constituant et qui ont permis ensuite une réalisation plus fluide de l'ensemble des applications du projet.

Chapitre 4 :

Réalisation

Introduction

Ce chapitre détail l'étape cruciale de la réalisation des applications du projet. Ceci est illustré à l'aide de captures d'écran pour chacune des applications.

1. Présentation des outils de réalisation

1.1. Technologie de développement



Java est un des termes les plus connus du monde de l'informatique et de l'Internet, que ce soit des professionnels comme du grand public.

Défini à l'origine comme un langage, « Java » a évolué au court du temps pour devenir un ensemble cohérent d'éléments techniques et non techniques. Ainsi, la technologie Java regroupe :

- Des standards (la plate-forme Java) définis sous forme de spécification par le Java Community Process (JCP), en trois éditions :
 - **Java SE (standard edition)** – utilisé pour l'application *Setup Maker*
 - **Java EE (enterprise edition)**, s'appuyant sur Java SE – utilisé pour l'application *Dashboard*
 - Java ME (micro edition), indépendante des deux précédentes
- Des logiciels (langages informatiques, bibliothèques, frameworks, serveurs d'application, outils d'aide au développement).
- Des communautés d'entreprises, organisations à but non lucratif (fondations, Java User Groups, universités) et indépendants, membres ou non du JCP, possédant tout ou partie des marques, brevets, parts de marché liés à la technologie Java.

La version utilisée pour le développement du projet est la **JDK 6 - JRE 1.6**.

1.2. Environnement de développement



Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la Fondation Eclipse visant à développer un environnement de production de logiciels libres qui soit extensible, universel et polyvalent, en s'appuyant principalement sur **Java**.

Eclipse est un environnement de développement (**IDE**) historiquement destiné au langage Java, même si grâce à un système de plugins il peut également être utilisé avec d'autres langages de programmation.

Il a été notamment choisi pour le développement des applications du projet grâce à sa puissante intégration au langage Java ainsi que ses nombreux plugins disponibles gratuitement sur le net pour permettre une programmation professionnelle et constamment optimisée.

La version utilisée pour le projet est **Eclipse Indigo**.

1.3. Librairies graphiques

Une étude comparative a permis de choisir les différentes librairies efficaces à implémenter pour chacune des applications, dont, au niveau graphique, deux librairies ont pu sortir du lot : **Apache Pivot** et **Vaadin**.

1.3.1. Apache Pivot



Apache Pivot est une plate-forme open-source pour construire des applications Internet installables (RIA). Il combine l'amélioration de la productivité et des caractéristiques d'utilisation d'une boîte à outils d'interface utilisateur moderne avec la robustesse de la plate-forme Java.

Pivot permet aux développeurs de construire facilement des applications liées, visuellement attrayantes et multi plates-formes, sous **Java** ou tout autre langage JVM, tels que *JavaScript*, *Groovy* ou *Scala*. Pivot est également le seul véritable framework IIA libre: il est complètement open source, et est géré entièrement par la communauté de développement de logiciels *Apache*.

Pivot permet aux développeurs de créer des solutions utilisant les outils qu'ils connaissent déjà, en diminuant le temps de livraison et réduisant l'étalement de technologie.

La version utilisée pour l'application **Setup Maker** est la **2.0.2**.

1.3.2. Vaadin

The Vaadin logo consists of the word "vaadin" in a lowercase, sans-serif font, followed by a blue right-pointing arrow icon. To the right of the arrow, the word "Vaadin" is repeated in a larger, bold, lowercase, sans-serif font.
Vaadin est un framework d'application Web open source pour les applications Internet riches. Contrairement aux bibliothèques JavaScript et des solutions basées sur un navigateur plug-in, il dispose d'une architecture côté serveur, ce qui signifie que la majorité de la logique s'exécute sur les serveurs. La technologie *Ajax* est utilisée au niveau du navigateur pour assurer une expérience utilisateur riche et interactive.

Vaadin utilise **Java** comme langage de programmation pour la création de contenu web. Le framework intègre la programmation événementielle et widgets, ce qui permet un modèle de programmation qui est plus proche du développement de logiciels *GUI* que du développement web traditionnel avec *HTML* et *JavaScript*.

La version utilisée pour l'application web **Dashboard** est la **7.1.0**.

1.4. Librairies de construction automatisée

Le développement de projets de grande envergure demande une certaine gestion approfondie des différentes configurations de construction et dépendances. De plus, plus grand et intéressant est le projet, plus nombreux sont ses dépendances externes.

Ainsi, une librairie de gestion de construction (**Apache Ant**) et une pour les dépendances (**Apache Maven**) sont nécessaires.

1.4.1. Apache Ant



Apache Ant est un logiciel créé par la fondation *Apache* qui vise à automatiser les opérations répétitives du développement de logiciel telles que la compilation, la génération de documents (*Javadoc*) ou l'archivage au format *JAR*, à l'instar des logiciels *Make*.

Ant est écrit en *Java* et son nom est un acronyme pour « *Another Neat Tool* » (un autre outil chouette). Il est principalement utilisé pour automatiser la construction de projets en langage *Java*, mais il peut être utilisé pour tout autre type d'automatisation dans n'importe quel *langage*.

Parmi les tâches les plus courantes, *Ant* permet la *compilation*, la génération de pages HTML de document (*Javadoc*), la génération de rapports, l'exécution d'outils annexes (*checkstyle*, *FindBugs* etc), l'archivage sous forme distribuable (*JAR* etc.)

Ant a donc été utilisé pour l'automatisation des processus de génération du package développeur. La version utilisée pour l'application **Setup Maker** est la **1.9.0**

1.4.2. Apache Maven

Maven **Apache Maven** est un outil *logiciel libre* pour la *gestion* et *l'automatisation* de production des projets logiciels *Java* en général et *Java EE* en particulier. L'objectif recherché est comparable au système *Make* sous *Unix* : produire un logiciel à partir de ses sources, en optimisant les tâches réalisées à cette fin et en garantissant le bon ordre de fabrication. Il est semblable à l'outil *Ant*, mais fournit des moyens de configuration plus simples, eux aussi basés sur le format *XML*. *Maven* est géré par l'organisation *Apache Software Foundation*. Précédemment *Maven* était une branche de l'organisation *Jakarta Project*.

Maven est utilisé pour la gestion des dépendances par rapport aux différentes librairies incluses. La version utilisée pour l'application **Dashboard** est la **2.1.0**

1.5. Outils d'emballage

La génération de packages étant l'action principale de l'ensemble des applications du projet, une sélection d'outils intéressants pour compléter cette tâche a été nécessaire.

1.5.1. Izpack

IzPack est un outil largement utilisé pour des applications d'emballage sur la plate-forme **Java**. Il permet de réaliser facilement des **installateurs** qui fonctionnent parfaitement sur *Microsoft Windows, Linux, Solaris et Mac OS X*.



Le même installateur fonctionne donc sur n'importe quel système d'exploitation où une machine virtuelle **Java SE 6+** existe. Les installateurs *IzPack* générés fournissent également une intégration native d'options comme la création de **raccourcis**, la manipulation d'un système de **registre** ou la gestion d'un compte **administrateur** lors de l'installation.

IzPack est publié sous les termes de la licence **Apache Software, Version 2.0**, ce qui signifie que vous pouvez **l'adapter à vos besoins** avec des contraintes très minimales.

IzPack a été utilisé dans le projet pour la génération finale de l'installateur. Une modification du code source a été nécessaire pour implémenter de nouvelles fonctionnalités ou corriger quelques bugs trouvés.

La version utilisée pour l'application **Setup Maker** ainsi que le **package** est la **4.3.5**

1.5.2. TrueZip

TrueZIP est un système de fichiers virtuel basé sur **Java (VFS)** qui permet aux applications clientes d'effectuer les opérations **CRUD** (Create, Read, Update, Delete) sur les fichiers **d'archives** comme si elles étaient des répertoires virtuels, même avec des fichiers d'archives imbriquées dans des environnements multithreads.

TrueZIP est une librairie Open Source et est couverte par la licence publique **Eclipse, Version 1.0**.

TrueZip a été utilisée pour accéder aux différents fichiers sous formes d'archives, à convertir des types d'archives en d'autres, à décompresser ou à accéder tout simplement à ces fichiers en lecture.

La version utilisée pour le projet est la **7.7.1**

1.6. Gestion de données

Comme tout bon programme qui se respecte, une gestion interne des données est nécessaire.

Cependant le projet a la contrainte de devoir s'exécuter rapidement sur n'importe quelle machine, sans besoin de paramétrages préalable ou d'installations.

Ainsi, l'utilisation de bases de données a été dépréciée pour ce projet, laissant la place au langage de balisage extensible : **XML**.

1.6.1. XML



L'**Extensible Markup Language** (XML, « langage de balisage extensible) est un langage informatique de **balisage générique** qui dérive du *SGML*. Cette syntaxe est dite « **extensible** » car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme *XHTML*, *XSLT*, *RSS*, *SVG*... Elle est reconnaissable par son usage des *chevrons* (<>) encadrant les balises. L'objectif initial est de faciliter l'échange automatisé de contenus complexes (*arbres*, *texte riche...*) entre systèmes d'informations hétérogènes (*interopérabilité*).

1.6.2. Stax

StAX est une API de **traitement XML** standard qui vous permet de **diffuser** des données XML depuis et vers votre application. Celle-ci reprend les avantages de *SAX*, sa rapidité et faible utilisation, tout en fournissant un moyen de créer des documents *XML* à l'instar de *DOM*.

StAX est publié sous les termes de la licence **Apache Software**, Version 2.0.

La version utilisée pour le projet est la **2 – API 3.1.1**

1.6.3. Staxmate

StaxMate est un framework léger (livré en un seul fichier jar) de couche supérieure à StAX, qui vise à ajouter une **commodité** de traitement XML, tout en conservant l'**efficacité** et s'assurant que l'**exactitude** n'est pas compromise.

La version utilisée pour le projet est la **2.2.0**

2. Etapes de réalisation

Les différentes étapes par lesquelles chaque acteur devra passer sont présentées en détail dans cette partie.

2.1. Développeur

Le développeur s'occupe, comme précisé avant (§2 2.1), de l'installation du package développeur généré dont les principales étapes d'installation sont présentées sous forme des captures d'écran suivantes :

Etape 0 : Choix de langue



Une fois le package démarré, une première fenêtre s'ouvre, affichant le logo ainsi qu'un choix entre deux langues : *Français* ou *Anglais*.

Etape 1 : Ecran de bienvenue

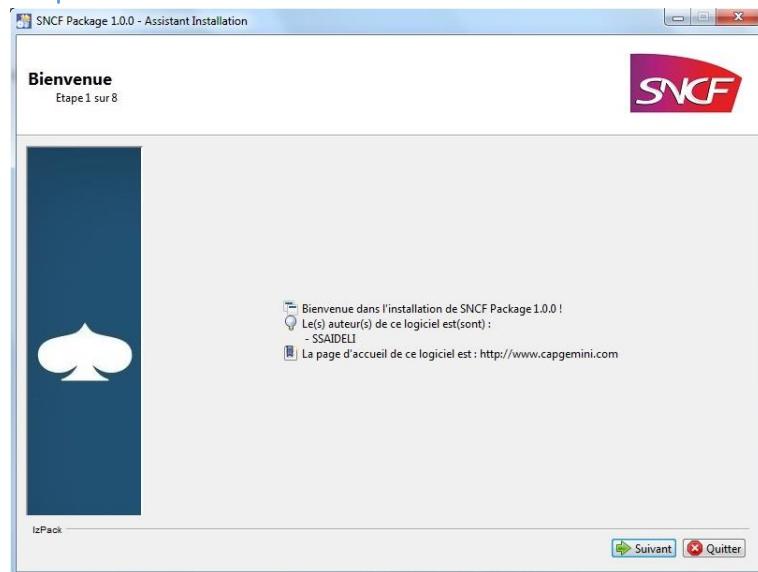


Figure 4-2-1 Capture d'écran de l'étape 1 du package

Ensuite l'installateur démarre, avec pour commencer un écran de bienvenue contenant les informations du **package**, la **version**, les informations de **l'auteur** ainsi qu'un **url** vers une page web définie.

Etape 2 : Choix du pole

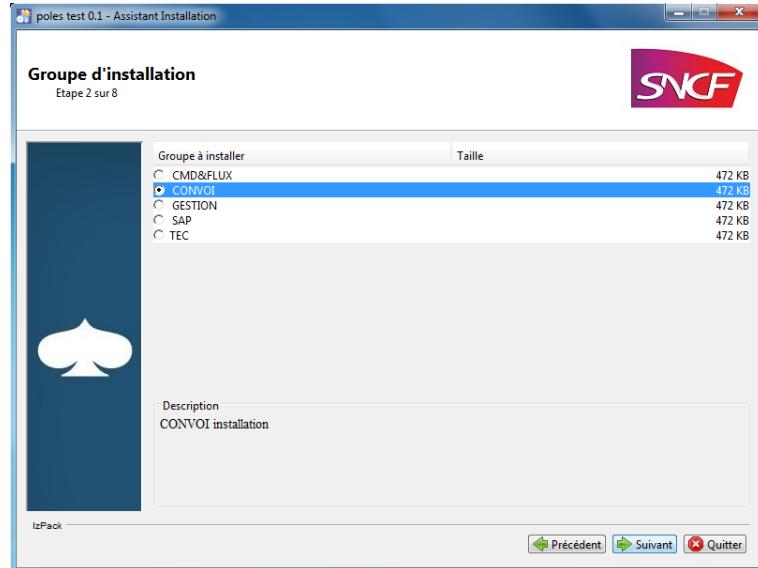


Figure 4-2-2 Capture d'écran de l'étape 2 du package

La deuxième étape présente un choix du **pôle** auquel le développeur appartient, pour avoir une liste personnalisée des outils et paramétrages propres au pôle seul, en plus de quelques packs standards présents dans tous ces pôles.

Etape 3 : Choix des packs à installer

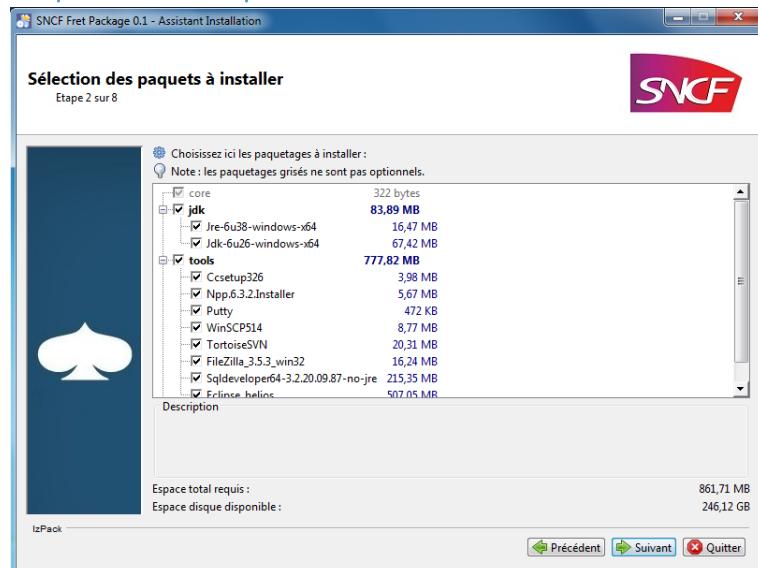


Figure 4-2-3 Capture d'écran de l'étape 3 du package

Une fois le pôle défini, la **liste des outils** s'affiche, organisée sous forme d'un arbre hiérarchique de **groupes** d'outils. Chaque **pack** est suivi de sa **taille** à droite, les groupes de la taille globale des packs le composant, et tout pack sélectionné affiche ses informations dans la partie **description**.

Le développeur a la possibilité de **désélectionner** quelques packs facultatifs, avant de valider en cliquant sur '**suivant**'.

Etape 4 : Ecran de résumé

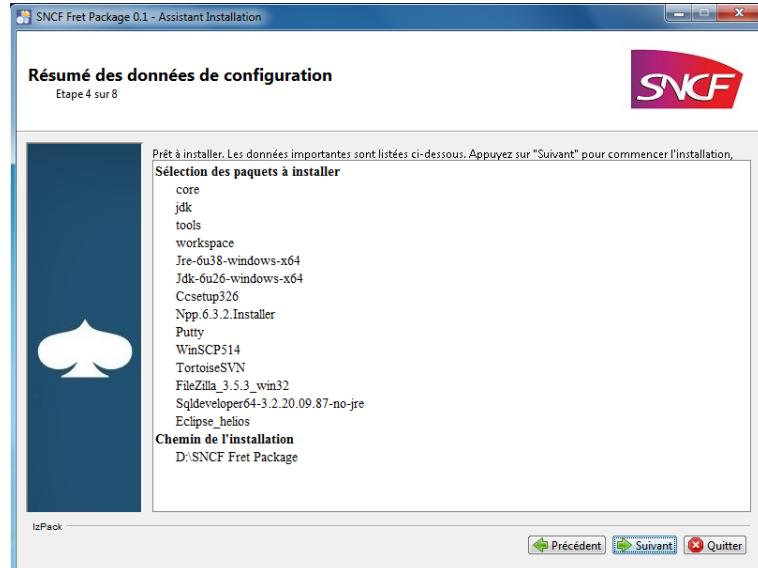


Figure 4-2-4 Capture d'écran de l'étape 4 du package

Une avant dernière étape présente un **résumé** des packs et groupes sélectionnés, ainsi que le **chemin** d'installation défini par défaut pour l'installateur (*qui ne peut pas être modifié*).

Après une revue de ces informations, l'utilisateur peut soit revenir en arrière en appuyant sur '**précédent**' pour modifier la sélection, ou confirmer en appuyant sur '**suivant**'.

Etape 5 : Installation

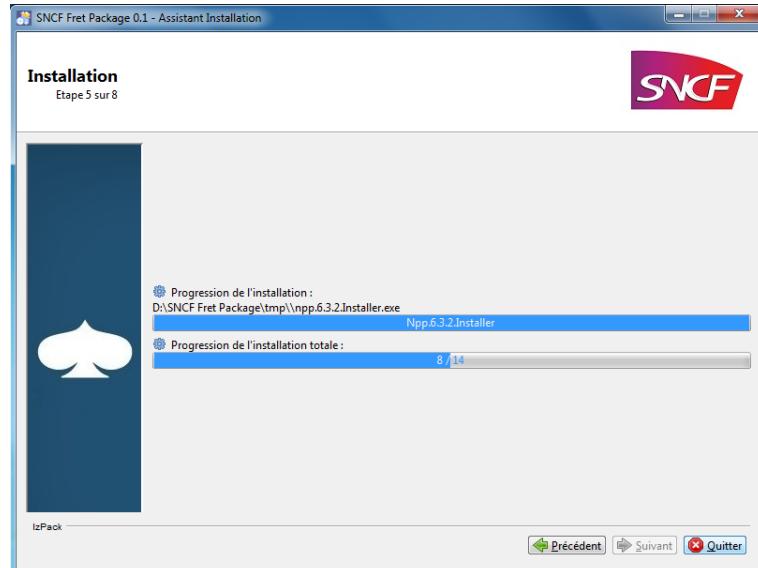


Figure 4-2-5 Capture d'écran de l'étape 5 du package

Le **déploiement** des programmes à **copier** ou **extraire** commence, avec une **barre de progression** globale ainsi qu'une barre de progression pour le traitement de chaque pack.

Une fois terminé, le bouton '**suivant**' devient accessible pour passer à l'étape suivante.

Etape 6 : Lancement des scripts et exécutables

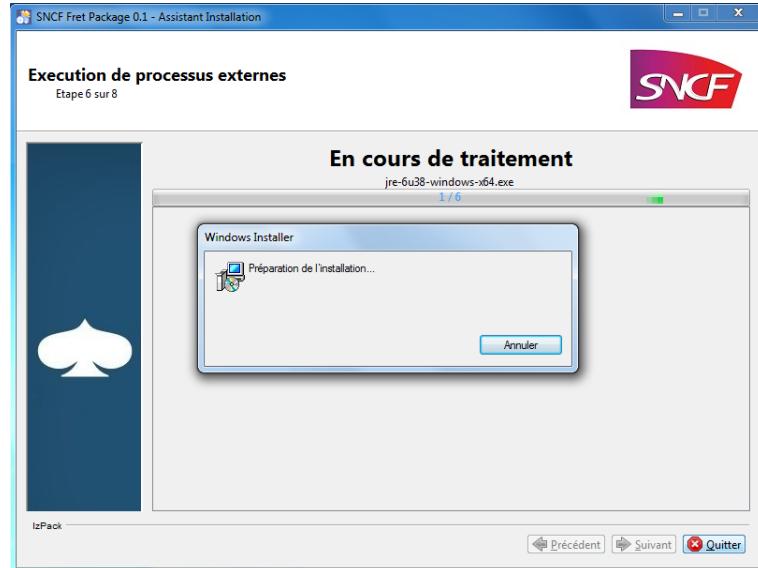


Figure 4-2-6 Capture d'écran de l'étape 6 du package

Après que le déploiement soit terminé, l'exécution de **processus externe** débute. Cette étape prend en charge l'exécution des **installateurs externes intégrés** et des **scripts batch** pour enfin réaliser un **nettoyage** des fichiers temporaires créés sur la machine.

Etape 7 : Installation des raccourcis

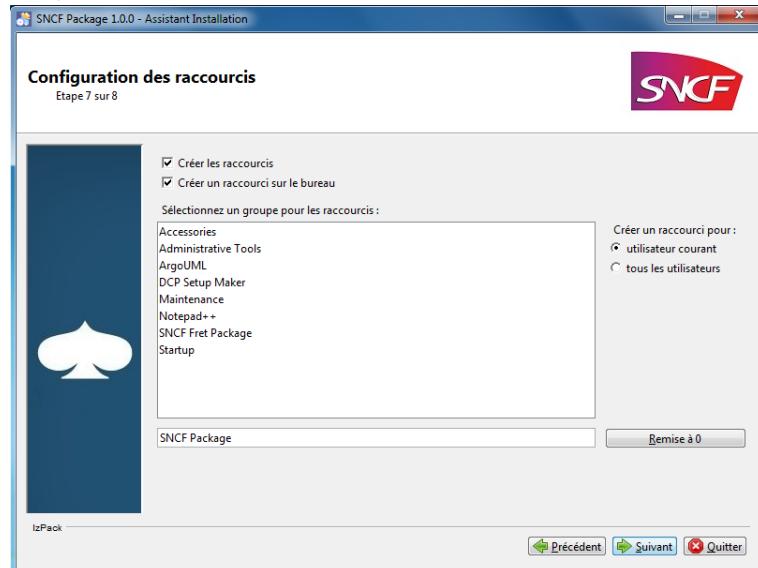


Figure 4-2-7 Capture d'écran de l'étape 7 du package

L'installation est terminée. Le développeur n'a plus qu'à choisir d'installer des **raccourcis** sur sa machine en précisant le chemin cible pour le **menu démarrer**, et s'il le désire les ajouter sur le **bureau**.

Une dernière possibilité de rendre le package accessible par tous les **utilisateurs** de la machine, ou seulement par l'utilisateur actuel.

Etape 8 : Ecran de fin

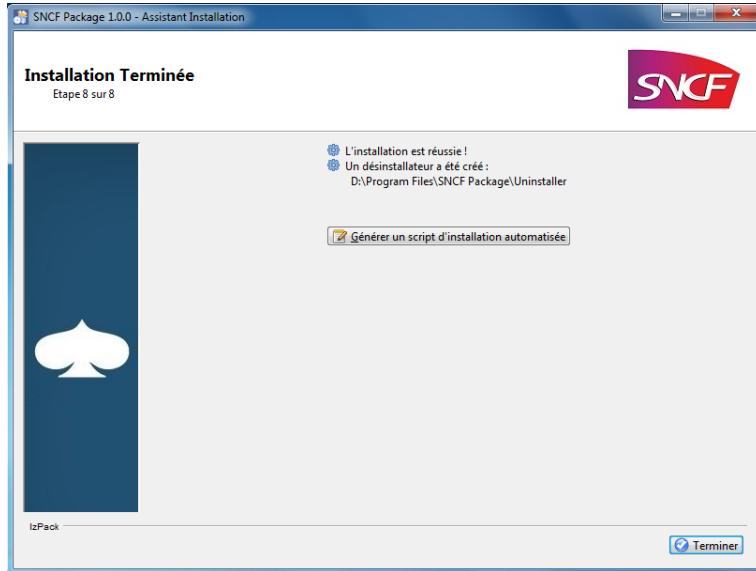


Figure 4-2-8 Capture d'écran de l'étape 8 du package

Finalement, le package est installé, paramétré et prêt à l'emploi !

Un écran de fin **confirme l'installation avec succès**, en rappelant le **chemin** où les fichiers ont été déployés. De plus, toutes les étapes et choix du développeur pour réaliser l'installation ont été enregistrés et peuvent donc être générés sous un fichier au même emplacement du package grâce à un bouton '**Générer un script d'installation automatisée**'.

Ainsi, un autre développeur du même pôle, travaillant sur la même partie du projet n'aura qu'à lancer le package **accompagné du script** pour avoir le même résultat, sans avoir à passer par toutes les étapes une fois de plus.

L'utilisation du script se fait à l'aide d'une commande java avec le script en paramètre:

Java –jar ./script.xml

2.2. Administrateur

L'administrateur, ressource principale du projet, s'occupe de la génération des packages (comme celui présenté au §4 2.1) à l'aide de l'application **Setup Maker**, dont voici les captures d'écrans :

Etape 1 : Scan du répertoire

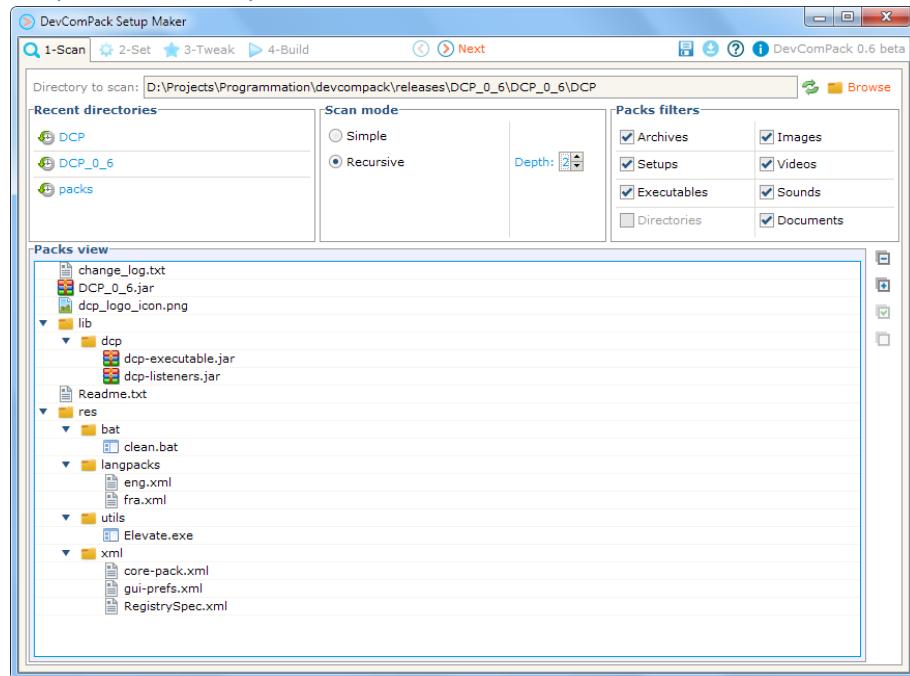


Figure 4-2-9 Capture d'écran de l'étape 1 de l'application *Setup Maker*

On commence tout d'abord par présenter l'interface globale de l'application, composée essentiellement en haut d'un groupe de **Tabs**, un pour chaque étape, de deux boutons pour **naviguer** entre ces étapes (*les tabs ne sont pas accessibles par clic de souris à cause de la dépendance de chaque tab par rapport au précédent*) ainsi qu'un groupe de boutons à droite pour la **sauvegarde**, **chargement**, lancement de **l'assistant** d'aide et **informations** de l'application.

Cette première étape présente donc à l'administrateur un écran simple qui lance un **assistant** d'utilisation de l'application si celle-ci est lancée pour la première fois sur la machine, pour présenter de manière simple les fonctionnalités du programme.

Une fois l'assistant fermé, l'administrateur commence par appuyer sur le bouton '**Browse**' pour avoir un navigateur intégré lui permettant de naviguer jusqu'au dossier contenant les fichiers à intégrer dans le package.

Après que le répertoire est choisi, le chemin de celui-ci est enregistré pour un futur accès rapide à partir de la partie '**Recent directories**' contenant jusqu'à 5 *liens* vers des dossiers précédemment scannés organisés par algorithme *FIFO*.

Le **scan** est donc lancé pour le répertoire, un scan qu'on pourrait **rafraîchir** si le dossier physique a été modifié depuis le dernier scan, selon le mode choisi dans la partie du milieu '**Scan mode**' :

- **Simple** : Le dossier est scanné simplement en listant les différents fichiers et dossiers le composant. Tous sont considérés comme des packs. Une option de sélection s'ajoute au mode pour pouvoir sélectionner une partie des fichiers et dossiers à prendre en compte.
- **Récursif** : Un scan récursif, où les dossiers à l'intérieur du répertoire scanné sont aussi bien accédés selon le niveau de profondeur attribué, pour enfin intégrer tous les fichiers trouvés comme packs, ainsi que les dossiers à la limite de la profondeur de scan.

Finalement, des **filtres**, à droite, permettent un meilleur contrôle sur la sélection des packs grâce à des filtres prédefinis pour les types de packages ainsi que d'autres pour les fichiers medias.

Les packs sont représentés sous forme d'une structure hiérarchique, avec à droite quelques boutons pour simplifier la lecture (**expand**, **collapse**) ou pour une sélection rapide (**select all**, **select none**).

Un clic sur le bouton '**Next**' pour passer à la prochaine étape :

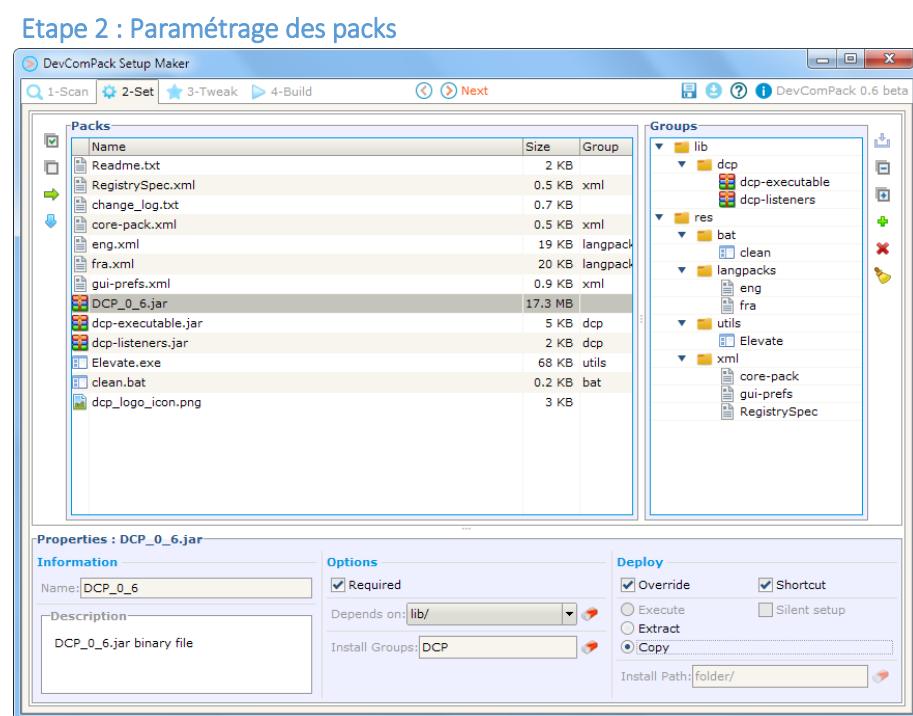


Figure 4-2-10 Capture d'écran de l'étape 2 de l'application Setup Maker

Le deuxième écran s'affiche, avec tous les packs sélectionnés dans l'écran précédent, permettant à l'administrateur de configurer chaque pack, ou ensemble de packs, selon quelques critères.

L'écran est décomposé en trois parties, redimensionnables, dont la partie gauche pour les **packs**, la partie droite pour la **hiérarchie packs/groupes**, puis la partie inférieure pour les **propriétés** des packs.

Les deux parties supérieures contiennent une **barre d'outils** traitant les données de la partie en question.

L'utilisateur commence par organiser les différents packs qu'il a choisis (à gauche) en les affectant à des groupes (à droite) grâce à un bouton '**add**' dans la partie Packs, ou simplement ou **glissant-déposant** le/les packs vers un groupe créé par le bouton '**new**'. Pour le cas d'un scan récursif préalable, il est possible d'importer la structure des dossiers scannés et ainsi avoir des groupes représentant les dossiers, grâce au bouton '**import**' de la partie Groupes.

Une fois la structure complète, une possibilité d'ouvrir une fenêtre de tri des packs pour **changer la priorité d'exécution** est possible avant de commencer les paramétrages nécessaires.

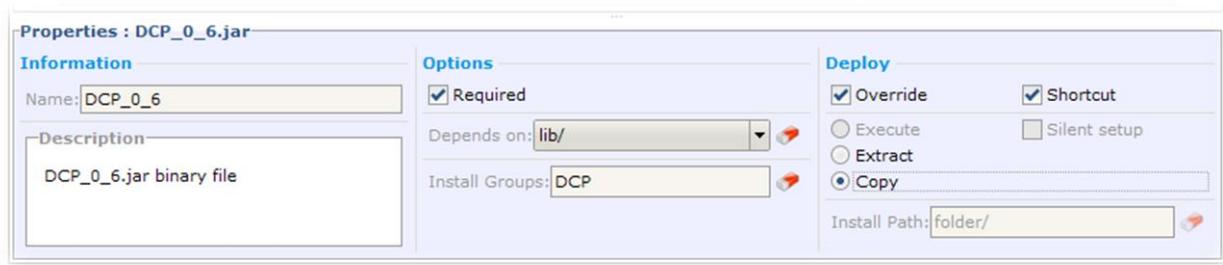


Figure 4-2-11 Capture d'écran des propriétés de l'application Setup Maker

La partie '**propriétés**' permet de configurer l'affichage des packs pendant l'installation, leur relation par rapport au contenu du package ainsi que le mode de déploiement pour ceux-ci.

Le **nom** du pack, qui par défaut reçoit le nom du fichier sans extension, est essentiel pour identifier le pack pendant l'installation. En effet une modification de celui-ci sera nécessaire dans le cas où deux packs situés dans des dossiers différents ont le même nom de fichier et sont tous les 2 inclus dans le package.

La **description** n'est rien d'autre qu'un simple texte affiché lors de la sélection du pack pendant l'installation pour informer le développeur de l'utilisation de celui-ci.

Un pack peut être **obligatoire** à installer, **dépendant** d'un autre pack ou groupe, et faisant partie d'un ou plusieurs **groupes d'installations**.

De même, une fois le pack sélectionné, il sera déployé par **copie**, **extraction** ou **exécution** de ce dernier, avec un **écrasement** possible si le pack est déjà présent sur le **chemin cible** défini, et l'installation d'un **raccourci** pour celui-ci à la fin de l'installation.

A noter que ces paramètres peuvent être appliqués pour un seul pack sélectionné, comme pour un ensemble de packs sélectionnés !

Etape 3 : Paramétrage du package

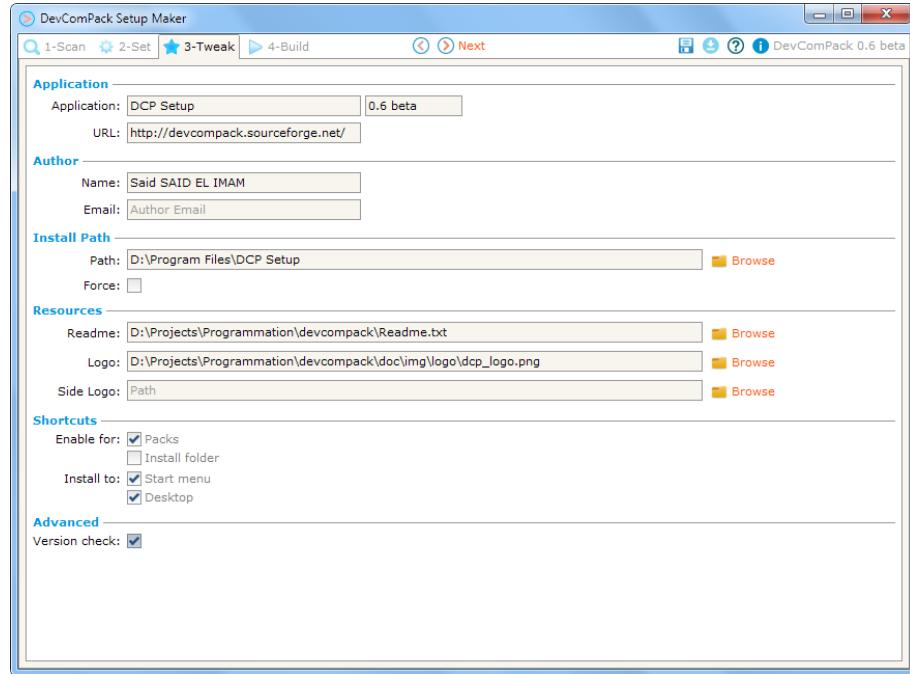


Figure 4-2-12 Capture d'écran de l'étape 3 de l'application *Setup Maker*

L'avant dernière étape de génération concerne la configuration du package en général, à commencer par le **nom** et la **version** du package, et pourquoi pas une **adresse web**. Quelques informations sur **l'auteur** du package, à savoir le nom de l'administrateur, remplis par défaut par le nom d'utilisateur de la session en cours.

Les paramétrages qui suivent sont :

- Un **chemin d'installation** par défaut, pouvant être **forcé**.
- Des **ressources** pour les écrans de l'installateur comme le fichier d'informations affiché au début, un logo et une image affichée de côté.
- Des options pour l'installation des raccourcis vers les packs, ou un raccourci vers le dossier global déployé sur la machine, ainsi que l'endroit d'installation de ces raccourcis.
- D'autres **options avancées** peuvent s'ajouter comme la gestion des versions du package installé.

Une fois les paramétrages et préférences définis, un clic sur le bouton '**next**' renvoi vers la dernière étape.

Etape 4 : Construction du package final

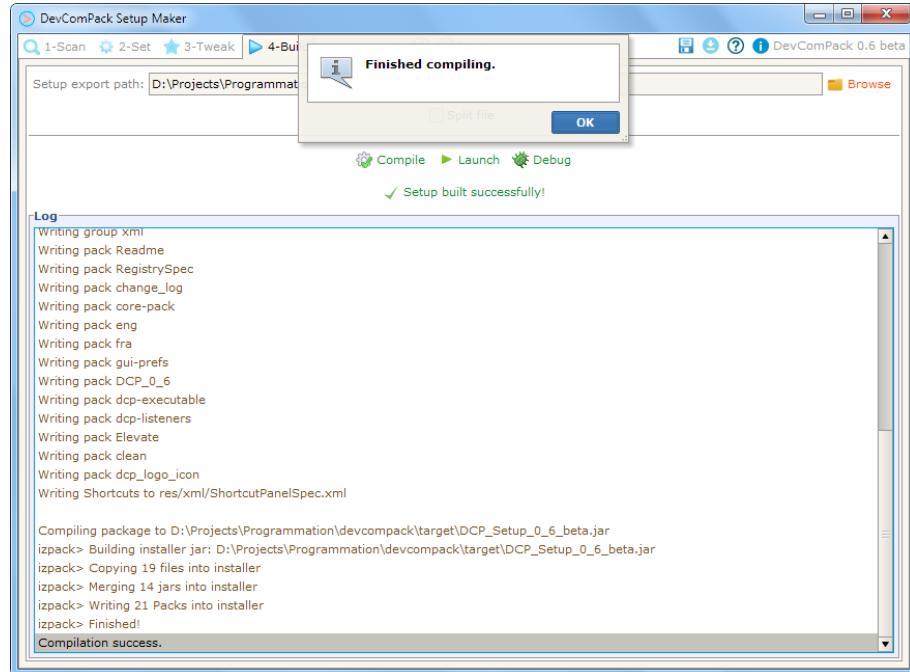


Figure 4-2-13 Capture d'écran de l'étape 4 de l'application *Setup Maker*

Nous voici à la dernière étape, avant de lancer la génération de notre package.

Un chemin d'export par défaut vers la racine de l'application avec un nom de package à partir du **nom** et **version** du package définis dans l'écran précédent.

L'administrateur a finalement un dernier choix de **décomposer le fichier** en plusieurs fichiers de taille définie pour enfin appuyer sur le bouton '**compile**' et suivre le **log** détaillant les étapes de compilation par lesquelles l'application passe pour enfin afficher un message de confirmation ou d'erreur à l'utilisateur.

Une fois la compilation terminée avec succès, une possibilité soit de **lancer** le package généré ou le **débugger** est finalement accessible.

Notre package est prêt à l'emploi !

2.3. Modérateur

C'est là où le rôle du modérateur devient intéressant. Celui-ci intervient une fois le package généré et installé sur toutes les machines pour lancer un suivi continu de l'avancement des installations sur le réseau.

L'application utilisée par cet acteur, sous forme d'application internet riche, peut-être détaillée rapidement en deux étapes :

Authentification

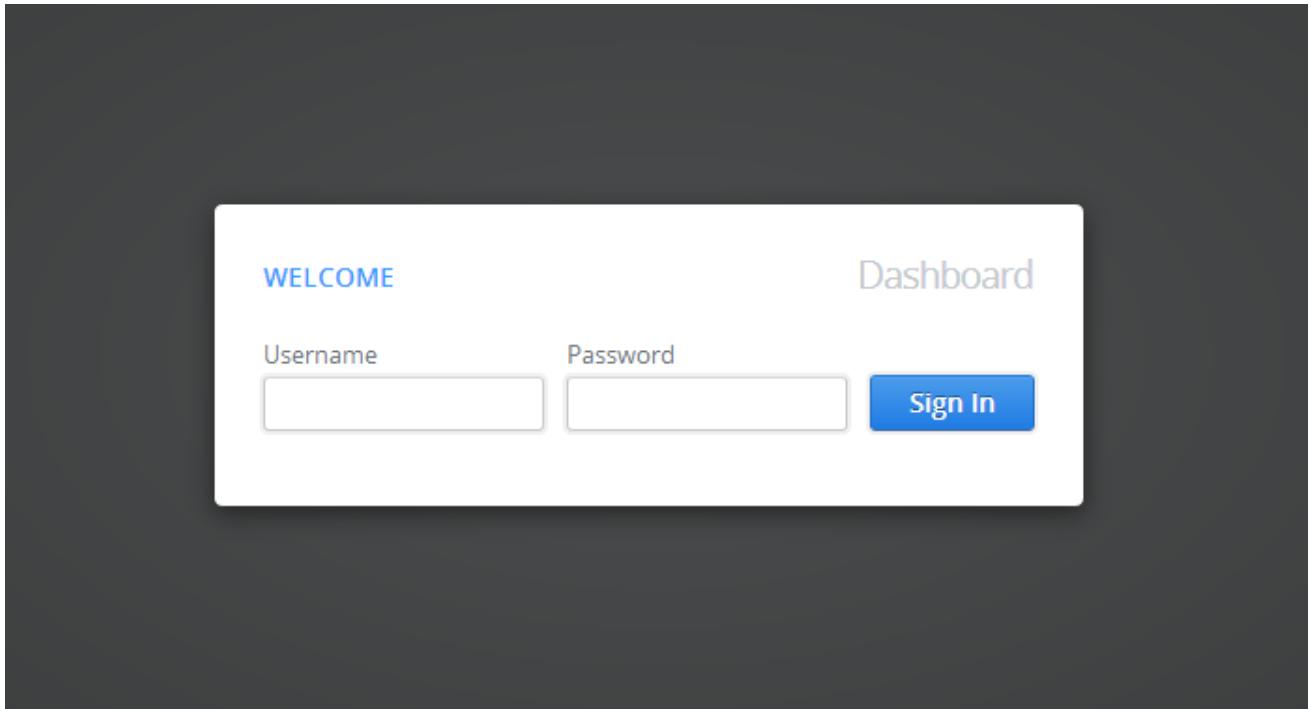


Figure 4-2-14 Capture d'écran de l'authentification de l'application **Dashboard**

Le modérateur commence par fournir les informations **d'authentification** pour pouvoir accéder à l'application.

Cette étape est primordiale pour éviter le fait que n'importe quelle personne accède à l'application et fasse des modifications critiques sur les machines accessibles.

Cependant, plusieurs modérateurs peuvent être affectés aux droits de monitoring, et donc auront un compte créé pour pouvoir y accéder.

Une fois l'utilisateur connecté, l'application ouvre l'écran **Dashboard**.

Dashboard

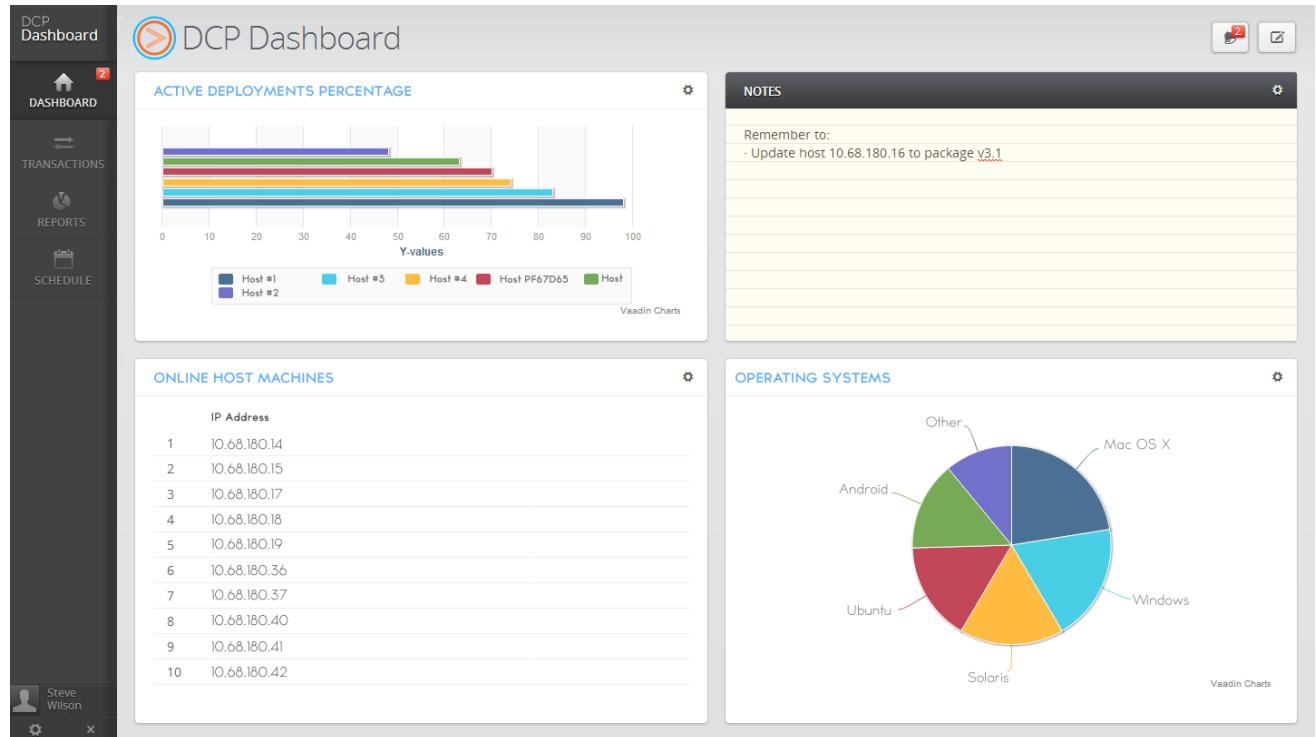


Figure 4-2-15 Capture de l'écran principal de l'application Dashboard

Le **Dashboard** est donc ouvert, avec comme première interface un **résumé** des déploiements en cours sur les machines, des **notes** à ne pas oublier définies par l'utilisateur lui-même, les **machines** connectées au serveur et puis quelques **statistiques** des installations dans le réseau.

Un **menu à gauche** permet l'accès aux différentes fonctionnalités de l'application, comme pour effectuer des **transactions** entre machines, générer des **rapports** ou planifier des **mises à jour**.

Avec, au coin inférieur-gauche, la possibilité d'accéder aux données du **compte** pour les modifier ou bien se **déconnecter**.

Finalement, un système de **notifications** est intégré à l'application pour notifier l'utilisateur des quelques évènements récents qui pourraient l'intéresser, comme pour un déploiement d'une mise à jour terminé ou bien la déconnexion d'une machine du serveur, etc.

Conclusion

Ce chapitre a permis de présenter les outils utilisés, avec leurs versions respectives, puis démontrer l'utilisation des différentes applications du projet, étape par étape, en passant par les différentes interfaces IHM réalisées.

Chapitre 5 :

Vers un produit Open Source

Introduction

Ce chapitre conclut le processus de développement de l'application noyau du système réalisé (Setup Maker) par la publication de celle-ci sur le marché mondial à travers une stratégie marketing définie.

1. Licence d'utilisation

L'application **Setup Maker** a dû prendre avantage d'un nombre de programmes open source, la plupart sous licence **Apache 2.0**.

En résumé, cette licence permet la réutilisation du code pour un programme libre ou propriétaire, mais impose le **copyright** lors de toute modification en stipulant quels fichiers ont été changés. Il est aussi obligatoire de faire figurer le fichier Notice provenant du code utilisé dans l'application. La version dérivée doit aussi avoir un nom différent, et le nom des auteurs ne peut pas être utilisé pour promouvoir un logiciel dérivé.

Une fois l'application stable et prête pour la production, j'ai jugé opportun de la publier aussi sous licence libre. Ceci ne posant aucun problème pour **Capgemini**, qui se réserve tous les droits sur le **package** généré. L'application a été publiée sur internet sous licence Apache 2.0, selon une stratégie marketing, pour avoir le plus de retours possibles de la part d'une communauté internaute intéressée, en plus de bénéficier des nombreux services disponibles pour les projets open-source, dont un **système de gestion de versions** et un **hébergement gratuit**.

2. Annuaire de code source

Le site **Sourceforge.net** a été celui choisi comme annuaire de code source pour l'application **Setup Maker**.

« SourceForge.net permet aux développeurs d'héberger des projets logiciels et propose des outils pour leur gestion. Il fournit plusieurs systèmes de gestion de versions comme CVS, SVN, Bazaar, Git ou Mercurial. Le site propose aussi un wiki, assure l'accès à une base de données MySQL et offre un sous-domaine pour chaque projet (comme nom-de-projet.sourceforge.net).

Environ 240 000 projets étaient hébergés sur le site en août 2010. En mars 2013 le site revendique 3 400 000 développeurs utilisant la plateforme, 324 000 projets hébergés et 4 000 000 téléchargements de logiciels par jour. » wikipedia

L'application a pu donc bénéficier de plusieurs services offerts par celui-ci, en plus d'un système de suivi des téléchargements effectués de partout dans le monde avec les statistiques.

3. Stratégie Marketing

Une fois le programme prêt à être utilisé, à partir de la 4^{ème} itération, un plan marketing a été lancé pour un soutien plus vaste à détecter les quelques problèmes et bugs lors de l'utilisation du programme.

A commencer par une **campagne de publipostage**, une multitude de mails ont été envoyés à des personnes spécifiques responsables d'un programme ou librairie en relation avec l'application.

Le premier à répondre étant le fondateur du projet **IzPack**, qui a été assez intéressé par l'application basée sur son projet et n'a pas hésité à la référencer directement sur la page de téléchargement officielle de son site : <http://izpack.org/downloads>



Graphical user interfaces

DCP Setup Maker

DCP Setup Maker is an open source project, powered by IzPack and developed by Said El Imam, that generates java installers easily for a set of files, without the need of coding knowledge. It automatically writes the xml code, and compiles it for you.

DCP Setup Maker is not affiliated with the IzPack project, and is released under the terms of the [Apache license v2](#)

Figure 5-3-1 Setup Maker référencé sur le site d'IzPack

Une deuxième **campagne de soumission de fichiers PAD** a visé différents sites annuaires de téléchargements, dont la liste est grande, pour avoir un meilleur référencement sur le web.

PAD, abréviation de **Portable Application Description**, est un fichier XML permettant une manière claire et facile pour les auteurs de logiciels de fournir des sources en ligne avec des informations sur leurs produits. L'avantage d'utiliser un fichier PAD est qu'il permet de stocker toutes les descriptions et les caractéristiques en un seul lieu - la configuration système requise, prix, coordonnées, noms de fichiers, URL et plus.

Cette campagne a permis de s'approprier le mot clé '*DCP Setup Maker*' ainsi que '*DevComPack*' sur internet.

La société **Softpedia** a été l'une des premières à s'intéresser à l'application, en l'intégrant à son annuaire avec une description et un Review complet du programme réalisé sur son site :

<http://www.softpedia.com/get/Authoring-tools/Setup-creators/DevComPack-Setup-Maker.shtml>

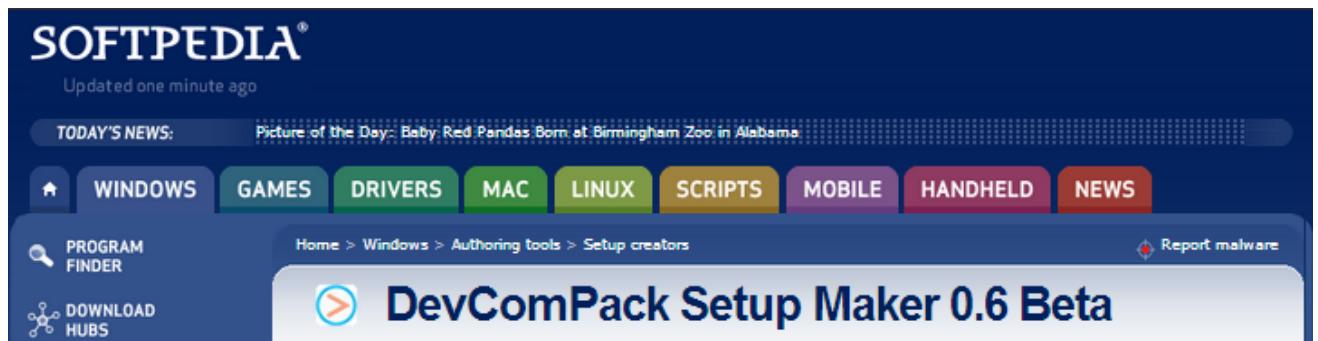


Figure 5-3-2 Setup Maker référencée sur le site SoftPedia

L'application peut aussi être accessible à partir de ces quelques liens importants, parmi d'autres :

- SourceForge.net - <https://sourceforge.net/projects/devcompack/>
- cnet Download.com - http://download.cnet.com/DCP-Setup-Maker/3000-2216_4-75938724.html
- ZDnet.com - <http://downloads.zdnet.com/product/2216-75938724/>
- Softbuz.com - <http://www.softbuz.com/windows/authoring-tools/setup-creators/dcp-setup-maker>
- Filedudes.com - http://www.filiedudes.com/DCP_Setup_Maker-download-181930.html

4. Statistiques de téléchargement

L'application, ayant reçu un soutien de plusieurs sites ; Une hausse des nombres de téléchargements a tout de suite été remarquée dès la première semaine.

Cette partie présente quelques statistiques rassemblées pour le mois de **Juin 2013**. A commencer par le nombre de téléchargements, dépassant les **300** téléchargements!

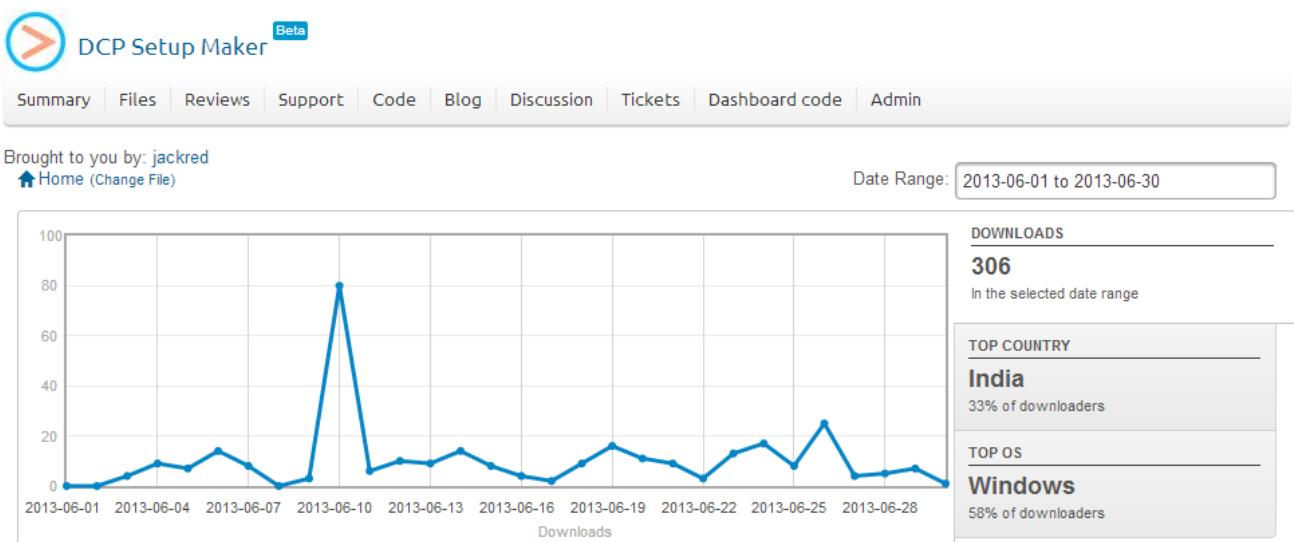


Figure 5-4-1 Statistiques du nombre de téléchargements du Setup Maker

Le pic remarqué au **10/06** correspond au jour où **Softpedia** a référencé l'application sur son site.

On peut aussi remarquer l'**Inde** comme premier pays utilisant l'application, avec **33%** des téléchargements, le reste dispatché sur l'ensemble des pays restants.

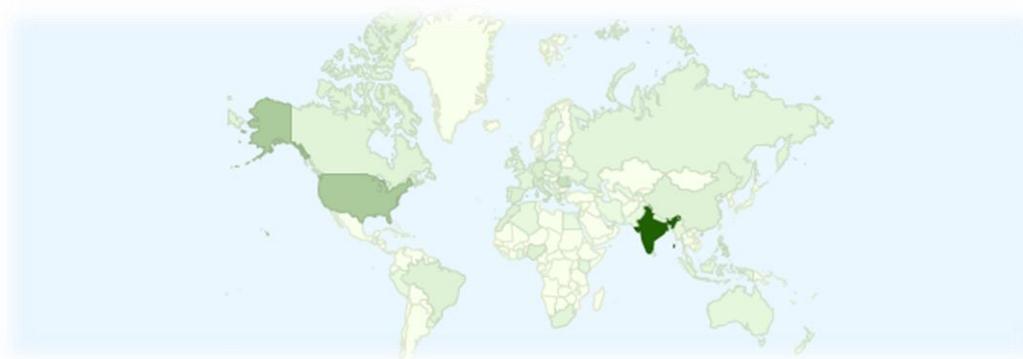


Figure 5-4-2 Statistiques des pays de téléchargements du Setup Maker

De même, on remarque un plus grand nombre de téléchargements à partir du système d'exploitation Windows avec **58%** des téléchargements, soit **185**, par rapport aux autres systèmes :

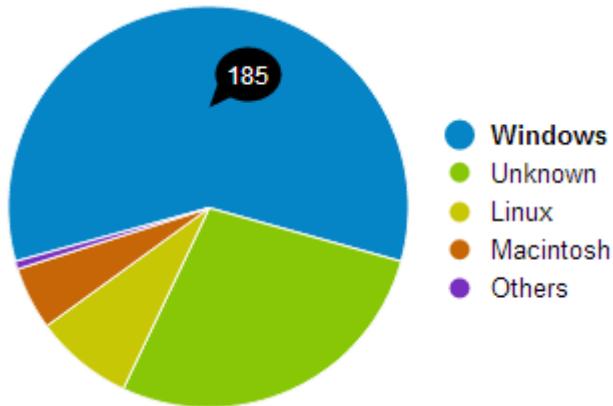


Figure 5-4-3 Statistiques des systèmes d'exploitation des téléchargements du Setup Maker

Par contre, le système d'exploitation d'une grande partie des téléchargements n'a pas pu être identifié.

5. Statistiques techniques

Un autre avantage pour les projets Open-source, c'est **Ohloh** :

« *Ohloh est un site web qui catalogue les projets open source et fournit des informations purement objectives sur ces derniers, principalement par le biais d'une analyse statistique des dépôts de code source.* » [wikipedia](#)

L'application a donc été intégrée sur ce site pour avoir de plus amples informations sur la progression du code, dont voici les statistiques :

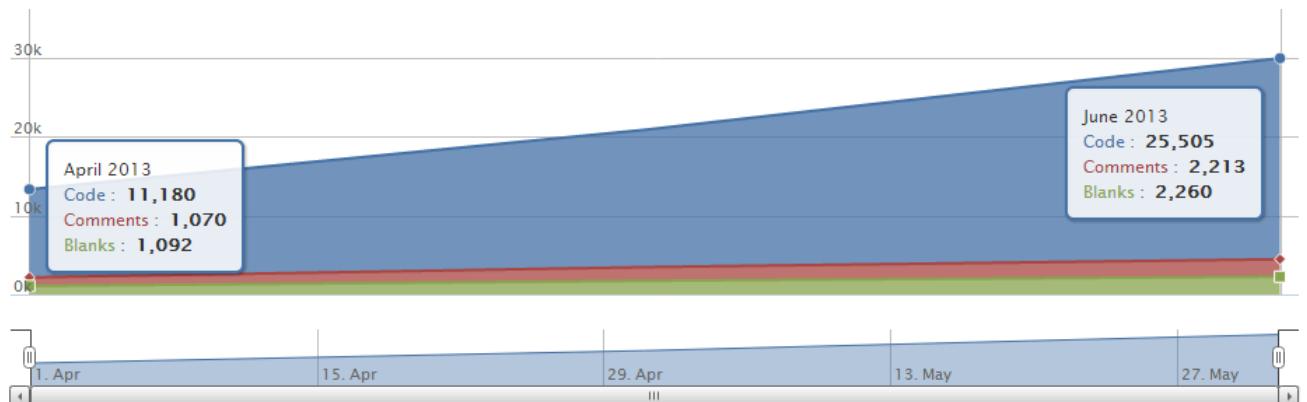


Figure 5-5-1 Statistiques de progression des lignes de code de Setup Maker

Ces informations expliquent la **progression du nombre de lignes de code** à partir du premier commit, en **Avril 2013** avec **11 180** lignes, jusqu'à la première version stable publiée en **Juin 2013** avec **25 505** ligne, soit une progression de **14 325** lignes en l'espace de **2 Mois**.

Ces lignes de code sont détaillées ci-suit selon les langages utilisés :

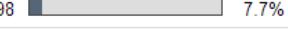
Language	Code Lines	Comment Lines	Comment Ratio	Blank Lines	Total Lines	Total Percentage
XML	16,324	327	2.0%	572	17,223	 57.5%
Java	6,386	1,098	14.7%	1,263	8,747	 29.2%
XML Schema	1,775	340	16.1%	183	2,298	 7.7%
shell script	347	99	22.2%	77	523	 1.7%
C++	276	131	32.2%	75	482	 1.6%
Python	205	136	39.9%	35	376	 1.3%
DOS batch script	138	80	36.7%	49	267	 0.9%
JavaScript	35	2	5.4%	6	43	 0.1%
Make	19	0	0.0%	0	19	 0.1%
Totals	25,505	2,213		2,260	29,978	

Figure 5-5-2 Statistiques de langages utilisés pour Setup Maker

Une grande partie du code revient à **XML** avec 57.5%, utilisé essentiellement pour les interfaces graphiques ainsi que la configuration du package, **Java** faisant le lien entre ces deux partie XML.

Les langages restants sont utilisés pour les quelques ajouts et améliorations du package.

Conclusion

Ce chapitre a démontré un certain intérêt d'une large communauté pour le projet, essentiellement l'application *Setup Maker*, ce qui démontre une stabilité évidente du projet ainsi qu'un besoin des utilisateurs dans le monde pour ce type de logiciel.

Conclusion générale

Le problème de la gestion des parcs informatiques tant au niveau des installations ou des configurations est une étape importante et préalable dans les grands projets. En effet, avant de démarrer un projet, une phase d'installation des machines et des outils nécessaires au projet est primordiale pour préparer l'environnement de travail au développeur. D'habitude, une telle tâche consomme un nombre important de budget Homme/jour surtout pour les grands projets où on trouve des dizaines voire des centaines de développeurs et de postes de travail à gérer.

Mon projet était justement l'automatisation du travail de cette équipe en concevant et en développant un système d'installation des outils et des configurations sur les machines et la synchronisation de ces machines.

Dans cet objectif, l'application que j'ai développée, *Setup Maker*, répond aux besoins suivants :

- *Génération d'un package en mode installateur*
- *Génération d'un patch de modification d'un package*

Ce projet de fin d'études a été une véritable opportunité pour découvrir le monde du travail en équipe, et pour approfondir mes connaissances en informatique ainsi que, et surtout, en conduite de projet.

De plus, et tout au long de ce stage au sein de Capgemini, j'ai pu améliorer mes capacités de communication, notamment grâce aux réunions et aux différents points de communication et présentations avec les responsables et les chefs de projet, que ce soit à Casablanca ou en France, pour rassembler des informations pour le projet ou valider une itération de celui-ci.

D'autre part, les difficultés rencontrées sont essentiellement d'ordre fonctionnel et proviennent de la complexité à mesurer les différentes étapes de la phase de développement.

Le déploiement et gestion des parcs informatiques est, certes, un projet déjà présent sur le marché informatique à travers toute une collection de solutions professionnelles, certaines développées par des géants informatiques.

Ceci dit, il est quasi-impossible pour un nouvel arrivant dans ce domaine de concurrencer ces systèmes informatiques, encore moins les dépasser en terme d'utilisateurs. Cependant, l'avantage primordial de la suite d'applications de ce projet est le fait qu'ils soient open source, et donc accessibles à un public encore plus large, en plus d'être facile d'utilisation pour n'importe quel utilisateur, sans besoin d'être préalablement formé dessus.

Toutefois des améliorations de ce travail sont encore possibles ! Les quelques retours d'utilisateurs et sites web ont créé une certaine motivation et volonté à continuer à travailler dessus et à l'améliorer pour répondre aux demandes du nombre d'utilisateurs qui augmente à une moyenne de 60 nouvel utilisateur par semaine au moment de la rédaction de ce rapport.

Webographie

IzPack
TrueZip
StaxMate
Vaadin
Apache Pivot
Apache Ant
Apache Maven
Licence Apache V2.0
Sourceforge
Ohloh

<http://izpack.org/>
<https://truezip.java.net/>
<http://staxmate.codehaus.org/>
<https://vaadin.com/home>
<http://pivot.apache.org/>
<http://ant.apache.org/>
<http://maven.apache.org/>
<http://www.opensource.org/licenses/apache2.0.php>
<http://sourceforge.net/>
<http://www.ohloh.net/>

Annexes

Licence Apache 2.0

Version 2.0, January 2004

<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work

or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that you meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that you changed the files; and
3. You must retain, in the Source form of any Derivative Works that you distribute, all copyright, patents, trademarks, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add your own attribution notices within Derivative Works that you distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add your own copyright statement to your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of your modifications, or for any such Derivative Works as a whole, provided your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless you explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by you to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, you may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Résumé

Le projet réalisé vise à améliorer la procédure de gestion des machines de développeurs du projet SNCF Fret, à l'aide de deux applications développées *Setup Maker* et *Dashboard*.

La première application, *Setup Maker*, est une application open source pour la génération automatique de packages, ceux-ci englobent tous les outils et paramétrages essentiels au développeur pour travailler sur le projet.

La deuxième, *Dashboard*, est une application web pour le monitoring réseau et suivi des installations déployées sur les machines des développeurs inclus dans le réseau.

Ces applications sont développées en *Java*, à l'aide de librairies open source pour la gestion des packages, à savoir Izpack et TrueZip, ainsi que d'autres pour la gestion interne des dépendances et compilations des applications développées comme Apache Ant ou Apache Maven.

Mots clés : *déploiement, installation, gestion automatique de parc informatique, gestion des configurations, surveillance de réseaux de machines, Setup Maker, Dashboard*.

Abstract

The project is carried out to improve the management process for machines of the SNCF Fret project developers, using both *Dashboard* and *Setup Maker* applications.

The first application, *Setup Maker* is an open source application for the automatic generation of packages, these include all the essential tools and settings for the developer working on the project.

The second, *Dashboard* is a web application for monitoring and tracking packages that are deployed on developers' machines included in the network facilities.

These applications are developed in *Java*, using open source package management libraries, namely IzPack and TrueZip and others to the internal management of dependencies and compilations developed as Apache Ant and Apache Maven libraries.

Keywords: *deployment, installation, automatic deployment management, configuration management, network monitoring, Setup Maker, Dashboard*.