

# Rapport de Stage de fin d'étude



Présenté par :

**Oussama ELABED**

Maître de stage : **Mr. Ludovic CHABOT**

Tuteur académique : **Mme. Luciana Arantes**

**Année Universitaire**  
2014/2015



# Résumé

Ce rapport décrit le travail que j'ai effectué à la société Coheris dans le cadre de mon stage de fin d'études en vue d'obtention du diplôme de Master Informatique spécialité systèmes et applications réparties.

Éditeur français de logiciels de gestion de la relation client (CRM), la société a décidé de lancer une nouvelle offre modulaire afin de proposer une gamme complète d'applications CRM en mode SaaS à la demande.

Intégré au sein de l'équipe de recherche et développement CRM, j'ai eu en charge de mettre en œuvre une nouvelle architecture basée sur l'approche Domain-Driven Design et d'intégrer la solution SPAD RealTime dans le nouveau module social. Au cours de cette mission, j'ai ainsi pu parfaire mes connaissances sur le langage JAVA et le Framework Java entreprise Edition dans sa dernière spécification 7, puis de réaliser une veille technologique sur les systèmes de planification de tâches et de pratiquer la conception avec l'approche Domain-Driven Design ainsi que l'approche CQRS.

## Mots clés

CRM, DDD, CQRS, Quartz, Graph API Facebook, JAVA, JSF, REST, Text mining.

## Contacts

- Christophe Debarre  
Chief Technology Officer  
4, rue du port aux Vins - F-92150 Suresnes  
[cdebarre@coheris.com](mailto:cdebarre@coheris.com) - [www.coheris.com](http://www.coheris.com)
  
- Ludovic Chabot  
Responsable R&D CRM  
4, rue du Port aux Vins - 92150 Suresnes  
[lchabot@coheris.com](mailto:lchabot@coheris.com) - [www.coheris.com](http://www.coheris.com)

# Remerciements

Je remercie, tout d'abord, toutes les personnes qui m'ont donné la chance de rejoindre une équipe de développement au sein de COHERIS, et qui ont contribuées au succès de mon stage.

Je remercie vivement mon maître de stage Monsieur Ludovic CHABOT, chef d'équipe R&D CRM, pour m'avoir accompagné au quotidien, guidé et conseillé de manière pertinente. Grâce à sa confiance, j'ai pu m'accomplir totalement sur les tâches que j'avais à réaliser

Je remercie également mon tuteur universitaire, Madame Luciana Arantes, pour son suivi tout au long de ce stage.

Enfin, je salue l'ensemble des collaborateurs de COHERIS pour leur accueil et leur esprit d'équipe qui a facilité mon intégration dans l'entreprise.

# Sommaire

<b>INTRODUCTION.....</b>	<b>6</b>
<b>1 PRESENTATION DU STAGE.....</b>	<b>7</b>
<b>    1.1 SOCIETE COHERIS.....</b>	<b>8</b>
1.1.1 Présentation .....	8
1.1.2 Produits.....	8
1.1.3 Organisation.....	9
1.1.4 En chiffres.....	9
<b>    1.2 MISSION.....</b>	<b>9</b>
1.2.1 Sujet de stage .....	9
1.2.2 Besoin.....	10
1.2.3 Équipe .....	10
1.2.4 Environnement de travail .....	11
<b>2 TRAVAUX EFFECTUES.....</b>	<b>14</b>
<b>    2.1 ETUDE DE LA MISE EN ŒUVRE DE L'APPROCHE DOMAIN DRIVEN DESIGN .</b>	<b>15</b>
2.1.1 Objectifs.....	15
2.1.2 Etude et réalisation.....	16
2.1.3 Bilan .....	20
<b>    2.2 MODULE SCHEDULER .....</b>	<b>20</b>
2.2.1 Objectifs.....	20
2.2.2 Etude détaillée.....	22
2.2.3 Réalisation.....	25
2.2.4 Bilan .....	28
<b>    2.3 MODULE SOCIAL ET INTEGRATION SPAD REAL TIME .....</b>	<b>29</b>
2.3.1 Objectifs.....	29

2.3.2 Etude de l'existant .....	30
2.3.3 Réalisation.....	34
2.3.4 Bilan .....	37
<b>3 BILAN .....</b>	<b>38</b>
<b>3.1 RESULTATS OBTENUS .....</b>	<b>39</b>
3.1.1 Mise en œuvre de l'approche Domain-driven design .....	39
3.1.2 Planificateur de tâche de hautes disponibilités.....	39
3.1.3 Intégration du Text mining dans le module social.....	40
<b>3.2 DIFFICULTES RENCONTREES.....</b>	<b>40</b>
3.2.1 Compilation des modules découpés.....	40
3.2.2 Tester l'efficacité de l'algorithme de Text mining .....	42
<b>CONCLUSION.....</b>	<b>44</b>
<b>RESSOURCES.....</b>	<b>45</b>
<b>GLOSSAIRE .....</b>	<b>46</b>
<b>ANNEXES .....</b>	<b>51</b>
<b>A. PLANNING .....</b>	<b>52</b>
<b>B. ARCHITECTURE D'UN MODULE AVEC DOMAIN DRIVEN DISIGN.....</b>	<b>54</b>

# Introduction

Dans un contexte où Coheris cherchent en permanence à être plus efficace et plus réactive aux besoins du marché, la société a décidé de lancer une nouvelle offre modulaire à la fin cette année 2015. Elle cherche à proposer une gamme complète d'applications CRM en mode SaaS à la demande.

Coheris souhaite constamment évoluer et proposer des modules basés sur une architecture technique et sur des technologies de pointes afin de répondre parfaitement aux attentes des utilisateurs.

Mon stage de fin d'études se déroula dans le cadre de la nouvelle offre. En effet, au sein de l'équipe de recherche et développements CRM et dans une période de six mois, les objectifs étaient découpés en deux parties.

Dans une première partie, le but était de comprendre la nouvelle architecture des modules Coheris, la mise en œuvre de l'approche « Domain Driven Design » dans un module prototype qui gère un système de planification des tâches. Et dans une seconde partie, l'objectif était de mettre en place une solution d'intégration du Datamining dans le nouveau module social et d'étendre l'existant connecteur social avec des fonctionnalités de tests, afin d'évaluer l'efficacité des algorithmes de la solution SPAD realtime sur la qualification des messages provenant des canaux sociaux Facebook et Twitter.

Le présent rapport trace les phases du déroulement du stage. Je commencerai par décrire le contexte dans lequel se situent ce stage, notamment le diplôme que je prépare et l'entreprise qui m'a accueilli. J'aborderai, ensuite, en détail le travail que j'ai réalisé en prenant soin de décrire les objectifs, la méthodologie utilisée et les résultats obtenus. Enfin, je conclurai en faisant un bilan critique de mes réalisations par rapport aux attentes de l'entreprise.

## **Partie 1**

# **Présentation du stage**

### **Sommaire**

---

<b>1.1 SOCIETE COHERIS .....</b>	<b>8</b>
1.1.1 Présentation.....	8
1.1.2 Produits.....	8
1.1.3 Organisation.....	9
1.1.4 En chiffres.....	9
<b>1.2 MISSION .....</b>	<b>9</b>
1.2.1 Sujet de stage .....	9
1.2.2 Besoin.....	10
1.2.3 Équipe .....	10
1.2.4 Environnement de travail .....	11

---

## 1.1 Société Coheris

### 1.1.1 Présentation

Editeur de logiciels Français multinational dédié à la maîtrise et l'optimisation de la relation client : CRM [1] (ventes, marketing, service client, connaissance client) et Business Intelligence (analytics, pilotage, data Mining).

Coheris est un des leaders dans la gestion des forces de vente nomades sur le marché européen du CRM. Elle compte environ 150 collaborateurs et plus de 1000 entreprises ont fait confiance à Coheris dans plus de 80 pays.

Coheris s'appuie sur des partenaires intégrateurs et sur ses propres experts pour offrir à ses clients des solutions à la fois opérationnelles, analytiques et prédictives au service de leurs performances.

### 1.1.2 Produits

Coheris propose une offre globale s'appuyant sur une gamme complète de logiciels :

Type	Produit
<b>Relation &amp; Service Client</b>	Coheris CRM Suite Coheris CRM Care
<b>Pilotage des forces de vente</b>	Coheris CRM Sales Coheris CRM Sales Trade Coheris CRM Sales Merch
<b>Pilotage des campagnes marketing</b>	Coheris CRM Marketing
<b>Datamining &amp; Business Analytics</b>	Coheris Analytics Spad Spad Deployment Server
<b>Business Intelligence &amp; Reporting</b>	Coheris Analytics Liberty Coheris Analytics Liberty Insight
<b>Cloud</b>	CRM Cloud

Tableau 1:Produit Coheris

## PARTIE 1. PRÉSENTATION DU STAGE

### 1.1.3 Organisation

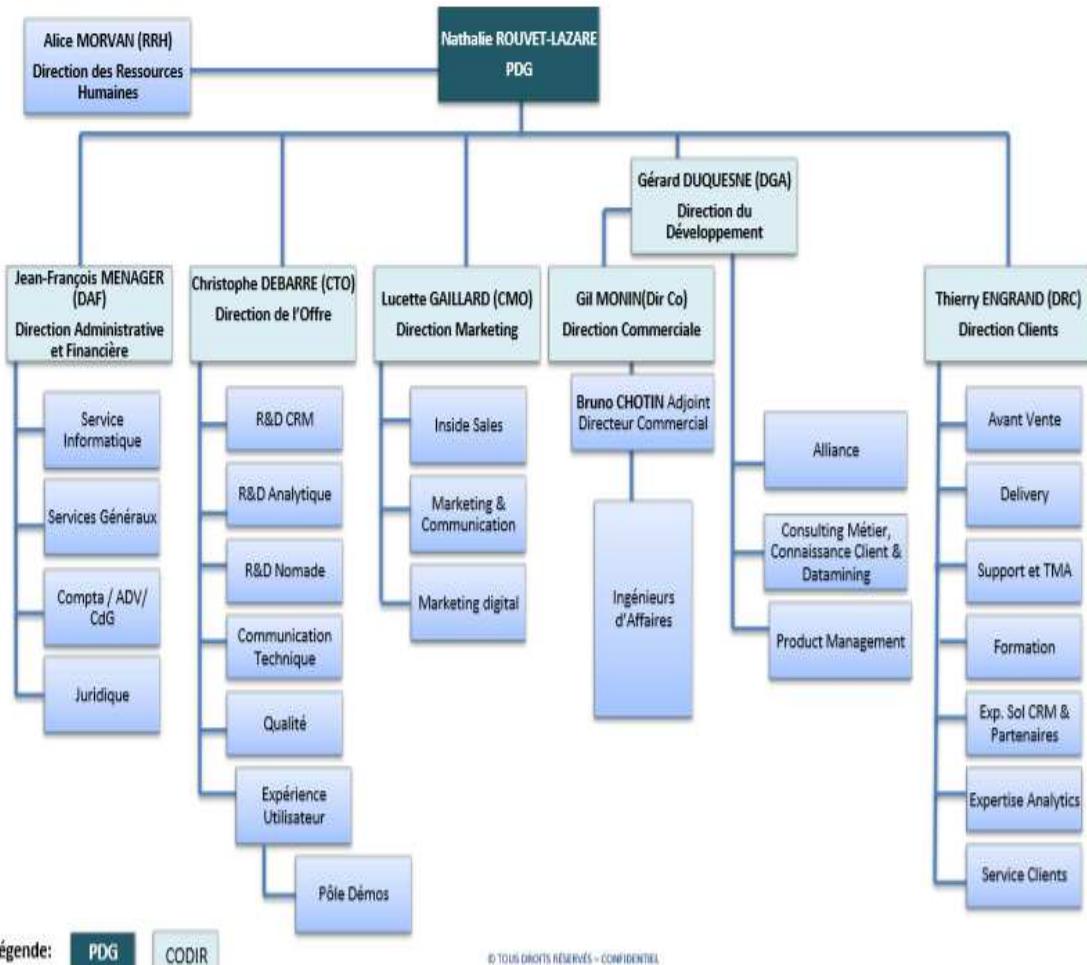


Figure 1: Organigramme de l'entreprise COHERIS

### 1.1.4 En chiffres

Le chiffre d'affaire de l'entreprise Coheris, éditeur français de référence de solutions CRM et analytiques, est de 14,58 M€ en 2014.

## 1.2 Mission

### 1.2.1 Sujet de stage

Intégration de SPAD, l'outil de datamining de Coheris, dans la distribution de CRM, afin d'enrichir le CRM d'indicateurs (KPI) métiers reposant sur des algorithmes datamining.

L'intégration de SPAD va être dans la nouvelle offre SaaS de Coheris. Ce qui ajoute des fonctionnalités innovantes au CRM tel que : La classification

## PARTIE 1. PRESENTATION DU STAGE

automatique des POSTS réseaux sociaux (text mining) pour faciliter le travail du Community Manager et de gagner le temps dans la classification des messages.

### 1.2.2 Besoin

Pour sa nouvelle offre, Coheris a choisi de se baser sur une nouvelle approche pour la conception de son offre: afin de collaborer et de communiquer plus efficacement avec les experts fonctionnels le choix s'est porté sur l'approche « Domain-Driven Design ». Une étude fonctionnelle et technique a été déjà effectuée afin de définir les orientations techniques et fonctionnelles d'un module « nouvelle offre » de Coheris.

Basé sur une nouvelle architecture, l'existant connecteur social du CRM sera migré vers un module social NO<sup>1</sup> SaaS [2] indépendant activable à la demande. De plus, il y aura des évolutions fonctionnelles qui seront mis en place pour répondre aux besoins des clients comme l'intégration de Text mining. Le passage à la nouvelle structure du module doit être précédé par une étude de l'existant (le connecteur social CRM), ainsi qu'une autoformation sur l'approche DDD [3].

Le but de mon stage est de préparer cette évolution, en participant à l'intégration du SPAD Real Time dans le module social NO et en prenant soin de proposer des fonctionnalités pour tester cette intégration. Mon rôle consiste aussi à enrichir le socle NO de briques techniques transverses : notamment la conception et le développement d'un module scheduler NO prototype, pour planifier des tâches de fond pour la nouvelle offre, cette brique technique étant nécessaire dans le cadre de l'intégration Spad Real Time.

### 1.2.3 Équipe

Au sein de l'entreprise, la direction de l'offre est l'organisme qui rassemble tous les équipes qui travaillent sur les différents produits de Coheris. Je me suis intégré au sein de l'équipe du pôle R&D CRM qui est une équipe de taille composée de 10 ingénieurs.

---

<sup>1</sup> Nouvelle Offre

## PARTIE 1. PRÉSENTATION DU STAGE

Cette équipe travaille en mode agile. Elle a mis en place la méthode SCRUM depuis 4 ans. Son objectif est de faire évoluer le produit phare Coheris CRM, en produisant plusieurs releases évolutifs par an (2 à 3), tout en maintenant les versions précédentes du produit qui sont toujours en production chez ses clients (environ 10 releases correctives par an).

Le contexte du projet, ainsi que la diversité des tâches qui m'ont été confiées, m'ont amené à collaborer avec l'équipe de Coheris SPAD dans le pôle R&D Analytique.

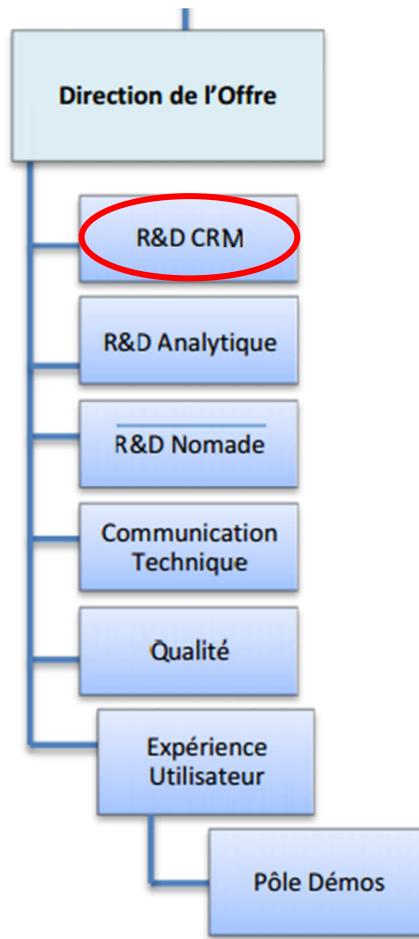


Figure 2: Organigramme de la direction de l'offre

### 1.2.4 Environnement de travail

#### Plate-forme de développement

Comme pour CRM, l'architecture de la nouvelle offre de Coheris est entièrement basée sur la plateforme Java J2EE [4] dans sa dernière version 7.

## PARTIE 1. PRESENTATION DU STAGE

La plateforme de cette nouvelle offre intègre aussi certaines briques techniques opensource répandues, comme:

- Spring social : pour la gestion des appels webservices vers l'api Graph Facebook.
- HazelCast : gestionnaire de cache réparti de deuxième niveau

### **Le socle J2EE Coheris**

L'étude a été faite en février 2014 par les membres de l'équipe R&D CRM ce qui a mené à l'aboutissement de l'implémentation du socle (framework) J2EE propre à Coheris et qui apporte les briques techniques nécessaires, non seulement pour la nouvelle offre mais aussi pour les autres applications à venir tel que Coheris SPAD RealTime.

Les briques techniques sont développées de manière innovante selon les spécifications techniques de EJB 3.1 (dont les EJB Lite)[5], JPA 2.0 [6], JSF 2.0 [7], CDI 1.0 [8], @Inject 1.0, Interceptors 1.1, JAX-RS 1.1 [9], etc.\*

Pour l'IHM, le choix s'est porté sur les frameworks JSF, PrimeFaces, JQuery et Twitter Bootstrap.

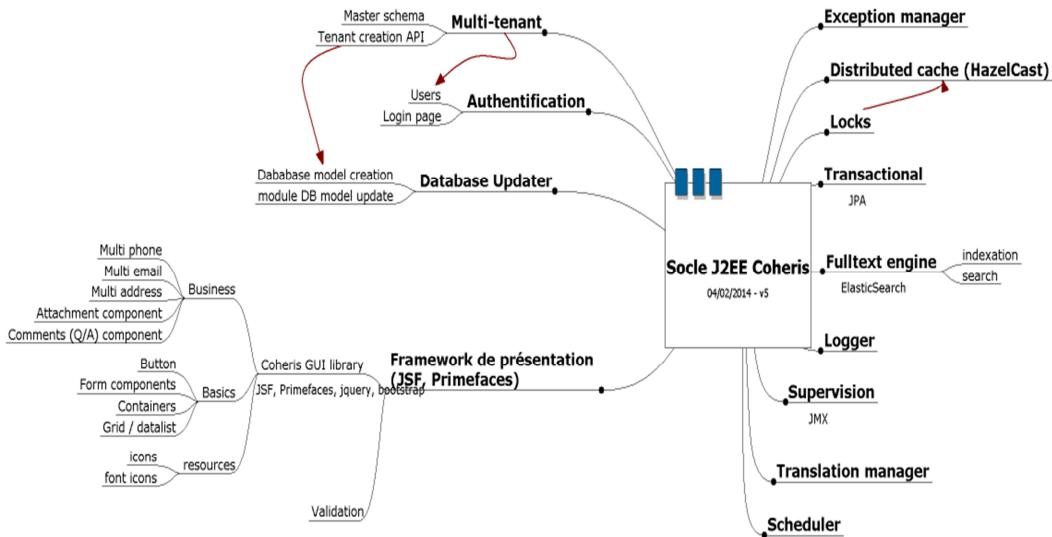


Figure 3:Socle J2EE Coheris

### **Serveur d'application :**

L'implémentation est effectuée afin de permettre l'utilisation de l'application sur les différents serveurs d'applications du marché supportant la plateforme J2EE. En R&D, le serveur d'application utilisé pour le développement est Wildfly 8.2.0 (anciennement JBoss).

### **Outils de développement**

L'équipe Recherche et Développement CRM utilise l'environnement de développement intégré (IDE en anglais) **Eclipse** dans sa version Luna.

Pour gérer la compilation des sources et l'obtention d'un produit exécutable, l'équipe a choisi l'outil de build **Gradle** [10].

Pour l'intégration continue, et comme pour le produit CRM, le choix était sur **Jenkins**. Il s'agit d'un programme permettant de mettre en place un processus d'intégration continue. Cela consiste à lancer et à programmer des compilations à distance. Ainsi, une compilation est réalisée très régulièrement (chaque commit peut être déclencheur d'une build), son résultat est consultable sur une page web afin de vérifier le bon déroulement de l'opération ou, à défaut, les fichiers responsables de l'échec. En cas de nécessité, une compilation peut être forcée par un administrateur. Le concept d'intégration continue est à la base de la méthode de développement agile.

### **Travail collaboratif**

Pour le travail collaboratif, l'équipe R&D CRM a opté pour un serveur **Subversion SVN**. Les fichiers versionnés, c'est-à-dire, les fichiers gérés par SVN, sont directement manipulables dans l'Explorateur Windows par le biais d'une extension, **TortoiseSVN**.

## **Partie 2**

# **Travaux effectués**

### **Sommaire**

---

<b>2.1 ETUDE DE LA MISE EN ŒUVRE DE L'APPROCHE DOMAIN DRIVEN DESIGN .</b>	<b>15</b>
2.1.1 Objectifs.....	15
2.1.2 Etude et réalisation.....	16
2.1.3 Bilan.....	20
<b>2.2 MODULE SCHEDULER .....</b>	<b>20</b>
2.2.1 Objectifs.....	20
2.2.2 Etude détaillée.....	22
2.2.3 Réalisation.....	25
2.2.4 Bilan.....	28
<b>2.3 MODULE SOCIAL ET INTEGRATION SPAD REALTIME .....</b>	<b>29</b>
2.3.1 Objectifs.....	29
2.3.2 Etude de l'existant .....	30
2.3.3 Réalisation.....	34
2.3.4 Bilan.....	37

---

## 2.1 Etude de la mise en œuvre de l'approche Domain Driven design

Les travaux réalisés dans cette partie sont les plus intéressants par rapport aux autres tâches qui m'ont été confié pendant le stage. En effet, les conceptions réalisées à la fin de cette partie vont être mis en œuvre sur les différents modules développés par la suite.

### 2.1.1 Objectifs

Afin d'avoir une démarche unifiée de construction des modules et afin d'en assurer (ou améliorer) leur pérennité les développeurs Coheris se doivent d'adopter une approche de conception de logiciel plus agile, plus modulaire et plus maintenables et évolutifs. L'équipe R&D CRM, a ainsi fait le choix d'adopter l'approche « Domain Driven Design » au travers du design pattern « CQRS-Event Sourcing » pour répondre aux besoins mentionnés.

Le premier travail qui m'a été attribué était de faire la conception à partir d'un module servant d'exemple, en appliquant l'approche DDD et en prenant en compte les différents types de déploiements.

Afin de viser le plus grand nombre de client du marché, et en se basent sur les expériences de la direction client de Coheris, les modules de la nouvelle offre doivent être plus flexible en terme de déploiement et de mise en œuvre. Les modes de déploiement adoptés pour la mise en œuvre d'un module sont :

- Mode 1 : le Frontend et le Backend dans le même serveur
- Mode 2 : le Frontend et le Backend sur les deux différents serveurs
- Mode 3 : Backend en mode API REST.

Le choix du mode de déploiement d'un module dépendra du fonctionnel du module lui-même. Par exemple suivant la nécessité ou non (en terme de sécurité notamment) d'avoir un front-end en DMZ et un backend sur un serveur protégé)

Il faudra aussi prendre en compte, d'une part, la solution qui peut être soit en mode SAAS, soit en mode ON-PREMISE, et d'autre part la mise en cluster du module.

Pour que les travaux soient bien cadrés, une spécification fonctionnelle a été réalisée par le Product Manager qui décrit les fonctionnalités attendus du module.

L'objectif de cette première étape est multiple :

- La documentation de la conception des différents types de déploiement
- La compréhension de l'approche Design Driven Developpment,
- La mise en œuvre du design pattern « CQRS-Event Sourcing », tout en respectant les modes de déploiement.

### 2.1.2 Etude et réalisation

L'architecture du module est en multicouche (N-Layered). Elle assure les concepts d'inversion de contrôle, et d'injection de dépendances du socle technique JEE de Coheris. En suivant les besoins mentionnées par rapport aux modes de déploiements, le module va être divisé en 7 couches. Chaque couche est encapsulée dans un « .jar » à part.

#### Les différents couches du module :

L'architecture du module et de ses différentes couches est détaillée en annexes (Annexe B). Il faut cependant savoir que des refontes ont été faites dans la partie présentation sur l'approche DDD pour qu'on puisse répondre aux différentes types de déploiements. En effet, la couche « GUI », qui est basée sur le Framework MVC [11] standard de la spécification JEE7 « JavaServer Faces 2.2», exploite les services exposés dans la couche applications soit via un appel direct vers ce dernier ou par un appel à une autre couche « api-client » qui implémente les mêmes fonctionnalités que l'application mais via des appels REST.

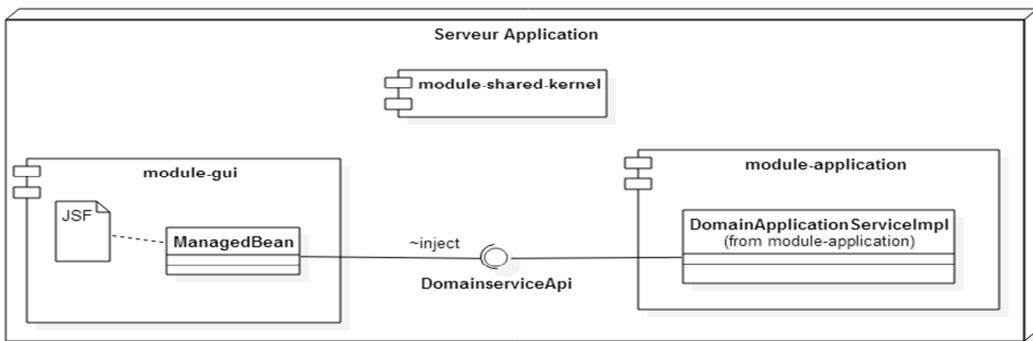


Figure 4: Communication entre le backend et le frontend dans un même serveur

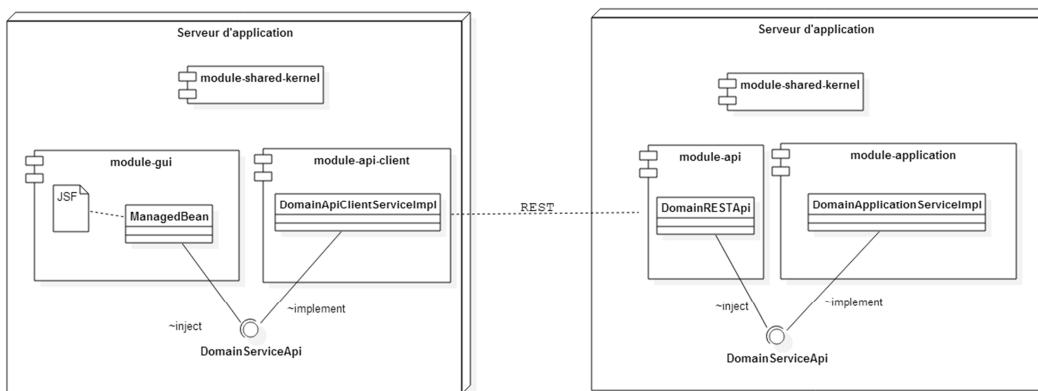


Figure 5: Communication entre le backend et le frontend déployé sur deux serveurs distincts

Pour assurer la mise en œuvre du mode saas du module, un composant « module-saas » doit être rattaché au module lors du processus de build qui assure l'architecture multi-tenant du module. Ce composant repose sur le module core-saas qui gère le partitionnement virtuellement des données et la configuration requise.

### Modes de déploiements du module Coheris

Comme expliqué dans l'objectif de partie des travaux, le déploiement de l'ensemble des ressources du module doit respecter le mode choisi. On a rencontré des difficultés pendant cette phase du à la séparation des encapsulations des différentes couches du module et dans quelle extensions livrable doit-on les mettre.

En effet, un serveur d'application JEE (conteneur Web + conteneur EJB) nous permet de déployer 3 différents types de livrable « .ear », « .war » ou « .jar ». Le choix était sur un livrable « .war » pour qu'on puisse assurer la mise en œuvre d'une part, la couche *module-gui* comme frontend et d'autre part la couche *module-api* qui expose les fonctionnalités du module via une API REST comme Backend. En fin, chaque mode de déploiement est défini dans un script de Build Gradle.

- **Mode1 :**

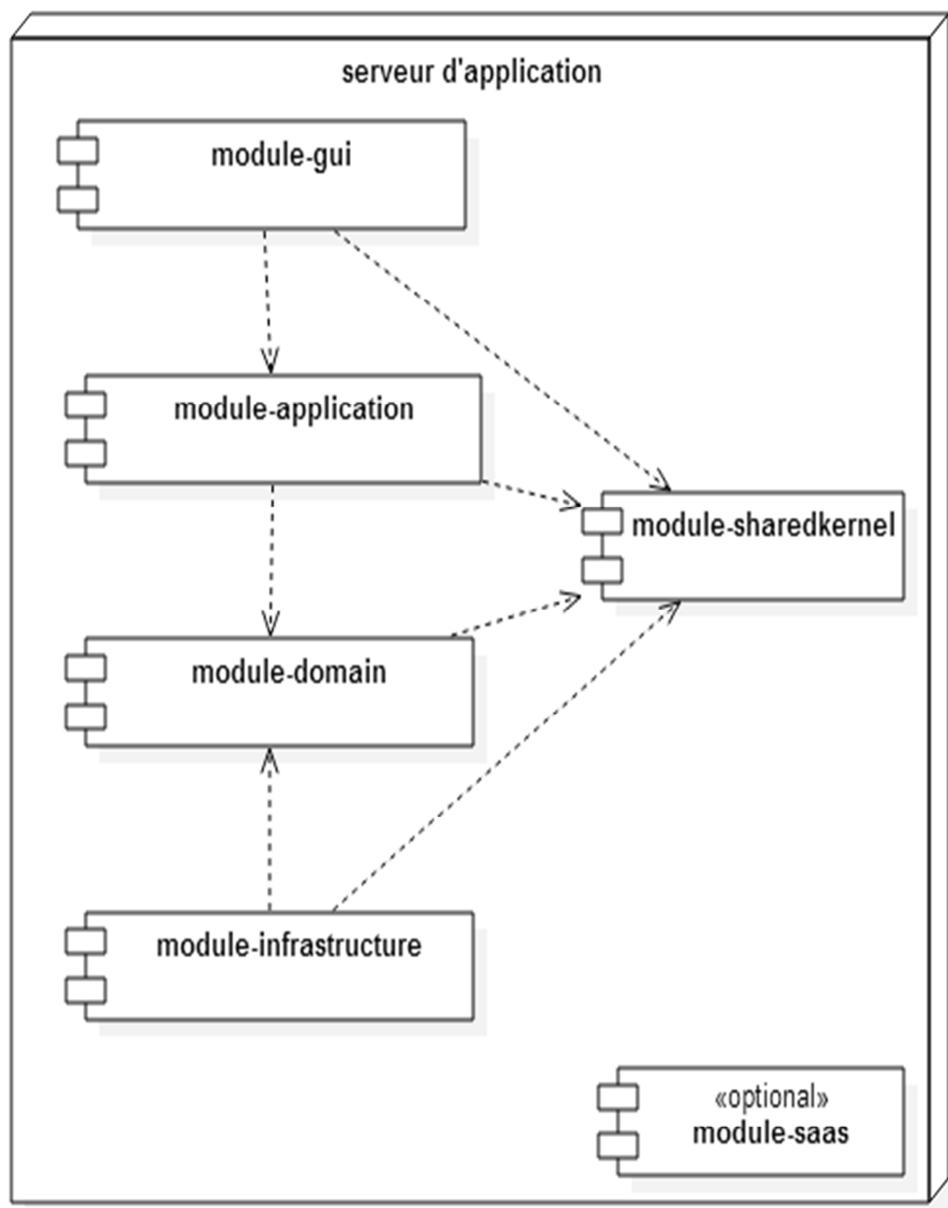


Figure 6:Mode1

- Mode 2

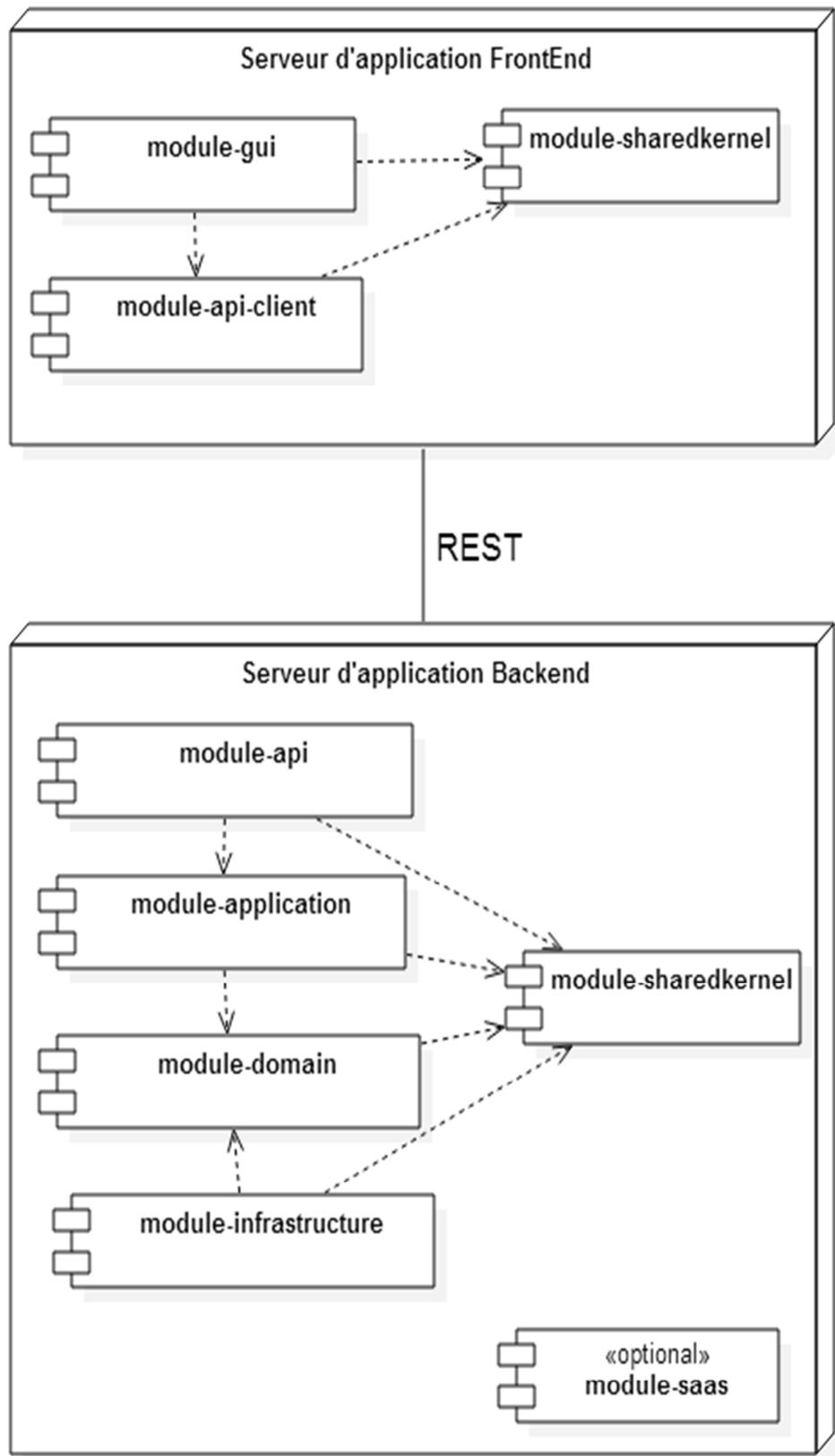


Figure 7:Mode 2

- **Mode 3 :**

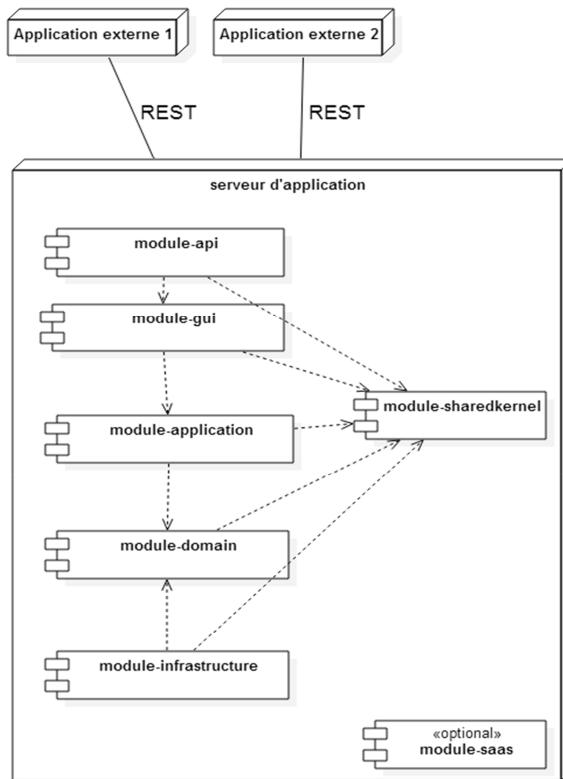


Figure 8: Mode 3

Pour le moment et dans le cadre de mon stage, il a été convenu de déployer en « mode 1 » comme une application « ON-PREMISE ».

### 2.1.3 Bilan

L'objectif de cette première étape était une excellente introduction au sujet principal de mon stage. La conception réalisée m'a servi à organiser le développement selon les différentes couches pour chaque module. En effet, afin de réaliser, dans de bonnes conditions, le développement des modules avec la nouvelle architecture, j'ai été formé sur l'outil de build gradle pour pouvoir écrire les tâches de build, en utilisant le langage de scripts JAVA Groovy [12].

## 2.2 Module Scheduler

### 2.2.1 Objectifs

Afin de planifier des traitements souvent longs dans les différents modules de la nouvelle offre, nous avons besoin d'un module Scheduler. C'est

un gestionnaire de tâches, paramétrable et intégrable selon le besoin. En effet, le module social a besoin de planifier des tâches de fond qui collecte les posts et les commentaires Facebook dans les différentes Fanpage pour pouvoir les qualifier après.

L'objectif de ce travail est, d'une part, de maîtriser la nouvelle architecture adoptée pour les modules, et d'autre part, le développement d'un module prototype complet qui peut être déployé sur les différents modes.

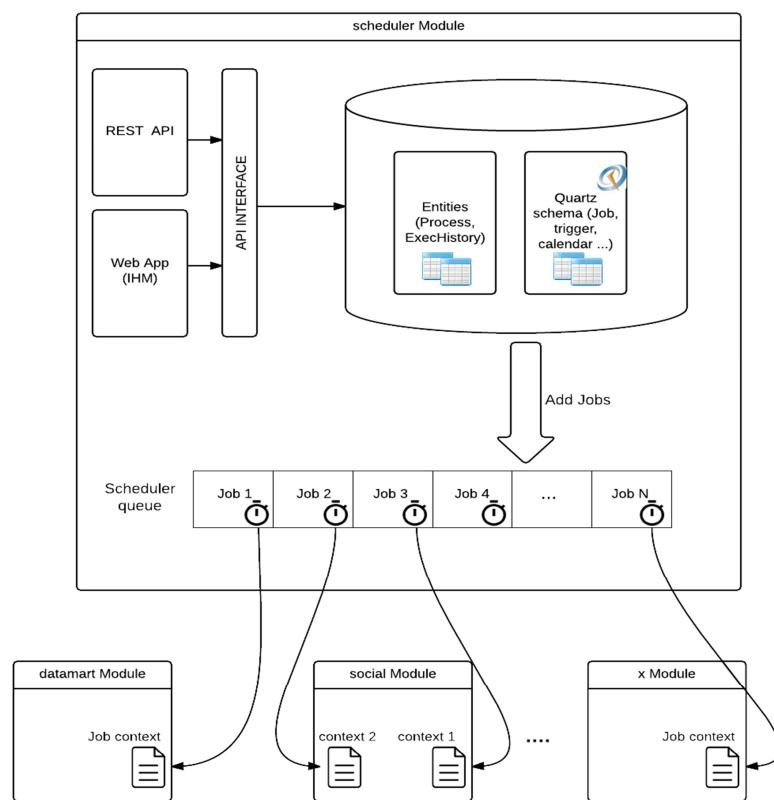


Figure 9 : Architecture du module scheduler

Avant de commencer les travaux, une étude complète et détaillée doit être réalisée, en prenant soin d'effectuer des comparatifs avec les solutions existantes des Framework qui gère les systèmes de planification. Dans cette étude, il faut mettre en valeur les points jugés intéressants et les éventuelles

régressions pour chaque solution. De plus, un certain nombre de points techniques doivent être vérifiés.

La mission a été découpée en plusieurs tâches :

- Etude, installation et manipulation de la nouvelle architecture,
- Intégration du framework
- Développement d'un prototype.

### 2.2.2 Etude détaillée

#### Etude du choix du Framework

Depuis la Java Entreprise Edition 7, l'API Concurrency Utilities1.0 (JSR 236) de Java propose un système de planification de tâches riches et simple à utiliser. On peut donc spécifier des annotations pour définir une tâche avec l'expression cron [13]. En revanche, Il existe plusieurs scheduler disponibles sur plateforme J2EE qui sont plus évolués et qui sont des open source. Quartz est un projet de Terracota qui propose des composants orientés entreprise JAVA. Mon rôle a été donc de réaliser une veille technologique, afin d'acquérir des informations techniques et établir la liste des points forts et faibles pour chaque API [14].

Framework	Concurrency Utilities 1.0	Quartz Scheduler
documentation	Peu de documentation: documentation standard	Documentation riche : exemples, tutoriaux...
Communauté autour	Encore nouveau pas beaucoup	Beaucoup plus
Clustering	Non	Oui
schéma de données	Non	Inclus et gérer automatiquement
configuration	Pas de configuration	Un fichier « properties »

Tableau 2: Tableau comparatif des Framework étudiés

Le choix a été sur Quartz Scheduler vu qu'il gère lui-même sa base de données des tâches et il supporte le mode Clustering qui est une contrainte impérative pour les modules « nouvelle offre » de Coheris. A noter, Quartz ne gère pas l'historique des tâches exécutées mais il fournit un listener qui sert à sauvegarder l'historique d'exécution pour chaque tâche.

## Architecture

En se basant sur le DDD, il faudra trouver une approche pour brancher l'API Quartz dans cette architecture. En effet, l'api se base sur la structure de donnée suivante :

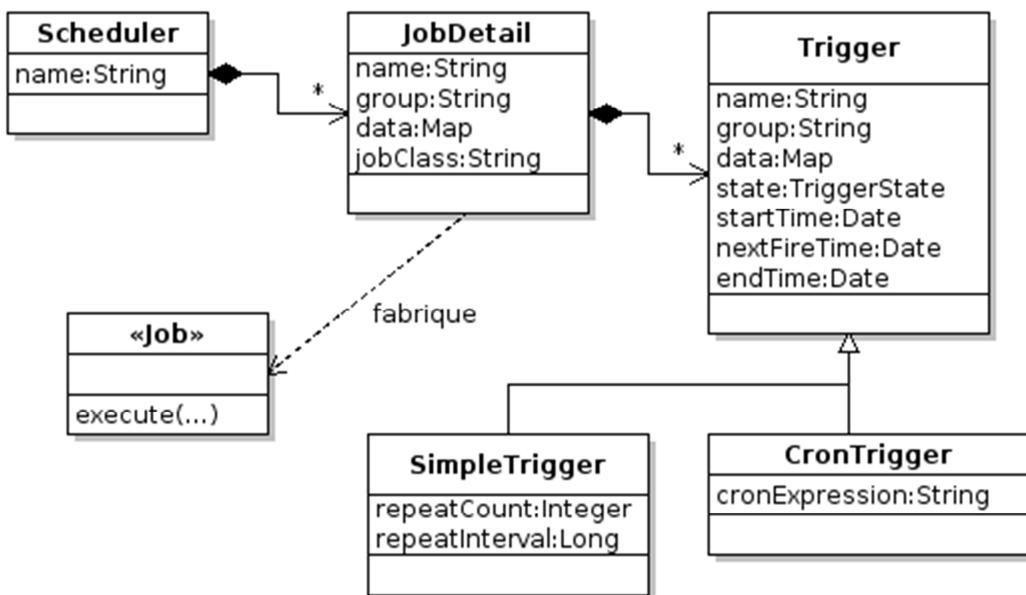


Figure 10: Concepts Quartz

- **Trigger**: le *quand exécuter*, il décrit une ou une série de moments, il y a plusieurs variantes: Cron, périodique, one shot...
- **Job**: le *quoi exécuter*, c'est un bout de code
- **Job Detail**: l'instance de Job, elle a un ou plusieurs triggers et éventuellement des paramètres
- **Job Group**: un ensemble de Job Details ou de Triggers que l'on manipule ensemble: arrêt, démarrage, annulation...
- **Scheduler**: le moteur chargé de la planification et de l'exécution des tâches planifiées.

On constate que le modèle de données quartz offre une manipulation libre et flexible du Framework. Selon les spécifications fonctionnelles, le module nous permet de :

- Planifier des tâches : quotidienne, hebdomadaire, mensuelle et annuelle.
- Définir un traitement qui pourra être exécuté par plusieurs tâches planifiées.
- Consulter l'historique de chaque tâche exécutée et son état d'exécution.

Cela ramène à définir 5 domaines de fonctionnalités dans le module

- **JobInfo** : l'entité qui contient les informations nécessaire pour créer un job
- **TriggerInfo** : l'entité qui collecte les informations nécessaires pour créer un Trigger quartz
- **SchedulerManager**: l'entité qui représente l'entité scheduler de Quartz
- **ExecHistory** : l'entité qui contient les informations catchées du listener de Quartz sur les exécutions
- **Process** : l'entité qui représente les infos nécessaires pour le traitement à exécuter.

Chaque modèle a son propre service (couche application et api), son propre repository (couche Infrastructure), ses propres commandes, ses queries et enfin ses composants IHM (couche gui).

En ce qui concerne le branchement du framework Quartz dans le module, la meilleure approche était de mettre ses fonctionnalités, dans la couche infrastructure du module. D'abord, car suivant l'approche DDD la couche infrastructure est le composant qui assure le dialecte avec des ressources externe par exemple: base de données, l'exploitation d'un API REST ou soap externe. Alors, on peut considérer Quartz comme étant une ressource externe

du module en traduisant les modèles de la couche Domain en modèle de données Quartz.

Finalement, il m'a été demandé aussi d'externalisé la configuration du Framework d'une façon programmatique à partir des paramètres définis dans le fichier standalone.xml du serveur.

### 2.2.3 Réalisation

Les réalisations relatives à cette mission correspondent en grande partie à brancher le Framework Quartz dans le module scheduler et à développer les classes java nécessaires, en se basant sur un module exemple [voir lien] qui applique l'approche DDD avec CQRS event sourcing. Dans une autre partie des travaux, j'ai été mené à réaliser un prototype IHM qui expose les fonctionnalités nécessaire du module.

#### Backend du module :

C'est la partie du travail qui englobe les développements faites sur les couches : shared-kernel, application, Domain, infrastructure, api et api-client.

Chaque modèle de conception, selon DDD, à son propre package dans chacun de ces couches.

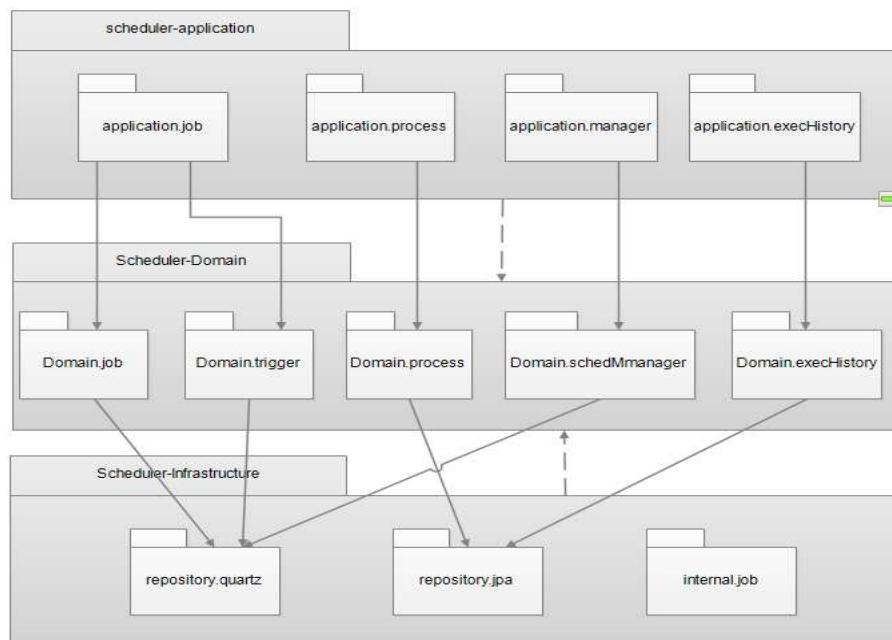


Figure 11: Architecture du module Scheduler

La configuration de quartz, mentionnée précédemment dans la partie étude, doit prendre ces paramètres de démarrage à partir des configurations standard du serveur d'application. Mais, il était impérativement nécessaire de créer un fichier .xml associé à la ressource de la couche infrastructure pour mettre en place les scripts de création de schéma de données Quartz. On trouve aussi dans ce fichier, par exemple, l'activation du scheduler en mode cluster ou non, le nombre de thread

```
<?xml version="1.0" encoding="UTF-8"?>
<schedulerQuartzConfiguration
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="SchedulerQuartzConfiguration.xsd">

  <userTransactionURL>java:/jboss/UserTransaction</userTransactionU
RL>
    <threadCount>25</threadCount>
    <threadPriority>5</threadPriority>
    <clusterCheckinInterval>20000</clusterCheckinInterval>
    <clustered>false</clustered>
</schedulerQuartzConfiguration>
```

### **Prototype IHM:**

Dans une première partie de ces travaux, j'ai été mené à « designer » des maquettes simple et basique pour visualiser et tester tous les fonctionnalités développées dans le module. Avant de commencer les travaux IHM, les maquettes ont été validées par le Product Manager. Ces maquettes respectent les choix des briques techniques GUI pour le socle :

- Bootstrap : pour que les composant soient responsives,
- Primefaces : qui propose une bibliothèque riche de composants et des exemples qui se basent sur JSF facile à intégrer.

**Tableau de bord:** où on peut trouver d'une part, les tâches en cours d'exécution, et d'autre part une vue globale sur l'état du scheduler par des statistiques représentés dans un diagramme.

## PARTIE 2. TRAVAUX EFFECTUES

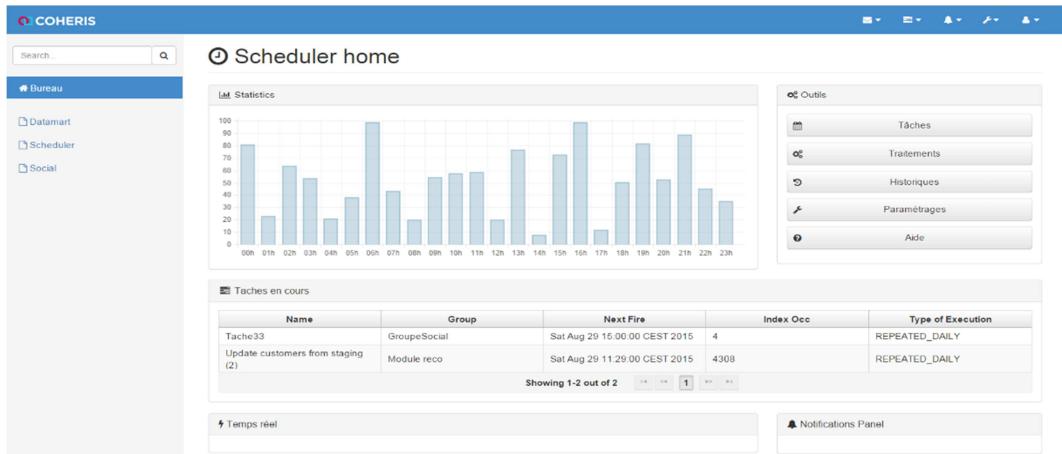


Figure 12: capture d'écran du dashboard Coheris

**Gestionnaire de tâches :** Dans cette vue, on peut ajouter des tâches selon un formulaire dynamique. Et Aussi on peut trouver plus de détail sur chaque tâche.

### Tâches

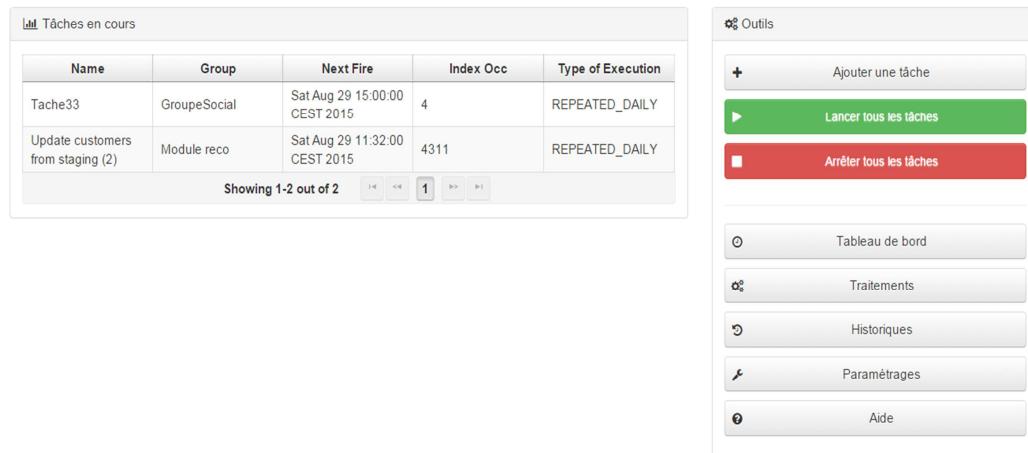


Figure 13: capture écran du gestionnaire de tâches

**Gestionnaire des traitements :** On peut ajouter, dans cette vue, des traitements selon un formulaire dynamique.

## PARTIE 2. TRAVAUX EFFECTUÉS

#	Name	Description	JNDI
104	Example	For testing the jobInfo	java:global/desktop-app-war/Example.JobCom.coheri
103	fsdg	string	string
141	jobtest	For testing	java:global/desktop-app-war/Example.JobCom.coheri
9746	Social Yves Rocher update Traitement	Récupérer les posts avec les commentaires de la page facebook Yves rocher	java:global/desktop-app-war/CollectNewPostsComme

Showing 1-10 out of 24

Figure 14: Capture d'écran du gestionnaire des traitements

**Gestionnaire d'Historiques :** où se trouve l'historique de tous les tâches exécutées.

Name	Group	Description	Last Fire	Next Fire	Index Occ	Type of Execution
ELF	ELF	ELF	Mon Aug 17 18:47:00 CEST 2015	Mon Aug 17 18:48:00 CEST 2015	43	REPEATED_DAILY
EIAbed	EIAbed	EIAbed			0	ONE_TIME
Tache33	GroupeSocial	Tache Test	Sat Aug 29 10:00:00 CEST 2015	Sat Aug 29 15:00:00 CEST 2015	4	REPEATED_DAILY
Update customers from staging	Module reco	Update customers from staging	Mon Aug 24 10:59:00 CEST 2015	Mon Aug 24 11:00:00 CEST 2015	50	REPEATED_DAILY
Update customers from staging (2)	Module reco		Sat Aug 29 11:31:00 CEST 2015	Sat Aug 29 11:32:00 CEST 2015	4311	REPEATED_DAILY
fifigogo	fifigogo	fifigogo	Tue Aug 18 10:51:38 CEST 2015	Tue Aug 18 10:52:00 CEST 2015	33	REPEATED_WEEKLY
google	google	google	Tue Aug 18 10:18:13 CEST 2015	Wed Aug 19 00:00:00 CEST 2015	1	REPEATED_DAILY
gooogle	gooogle	gooogle	Mon Aug 17 14:47:00 CEST 2015	Mon Aug 17 14:48:00 CEST 2015	1	REPEATED_DAILY
jijitaa	jijitaa	jijitaa	Tue Aug 18 10:48:00 CEST 2015	Tue Aug 18 10:49:00 CEST 2015	35	REPEATED_DAILY
job01	groupe01	le premier test	Mon Aug 10 14:40:00 CEST 2015	Mon Aug 10 14:41:00 CEST 2015	25	REPEATED_DAILY

Showing 1-10 out of 22

Figure 15: Capture d'écran sur le gestionnaire d'historiques

### 2.2.4 Bilan

Les principales fonctionnalités, comme l'ajout d'une tâche qui se déclenche une seule fois, quotidiennement, hebdomadaire ou mensuellement, ont été testées et répondent parfaitement aux besoins. Le même cas pour l'observation en temps réel, des tâches en cours d'exécution et pour la visualisation de l'historique de l'exécution d'une tâche, ont été aussi bien testées. Inversement, dans la vue Tableau de bord il manquait le composant temps réel pour visualiser la barre de progression pour chaque

tâche. En effet, le temps de développement nécessaire pour la partie était relativement faible. Dans cette partie, une formation était nécessaire sur le framework bootstrap, afin de bien organiser les composants pour qu'ils soient parfaitement responsive.

Enfin, le module scheduler était un bon exercice pour entamer le développement sur le module social qui est le but de mon stage de fin d'étude.

## 2.3 Module Social et intégration SPAD RealTime

### 2.3.1 Objectifs

Les réseaux sociaux interviennent en tant que nouveaux canaux de communication au sein de Coheris CRM. Ce qui mène l'équipe R&D CRM à développer leur propre connecteur social qui répond aux fonctionnalités attendues par Coheris CRM.

Dans le cadre de la nouvelle offre que propose Coheris, il a été prévu une migration totale du connecteur social de Coheris CRM en un module indépendant qui s'appuie sur la nouvelle architecture DDD, en ajoutant certaines évolutions. L'intégration du Text mining dans le module était une des évolutions primordiales planifiées sur le module Social au travers de l'appel des fonctionnalités de la solution SPAD RealTime.

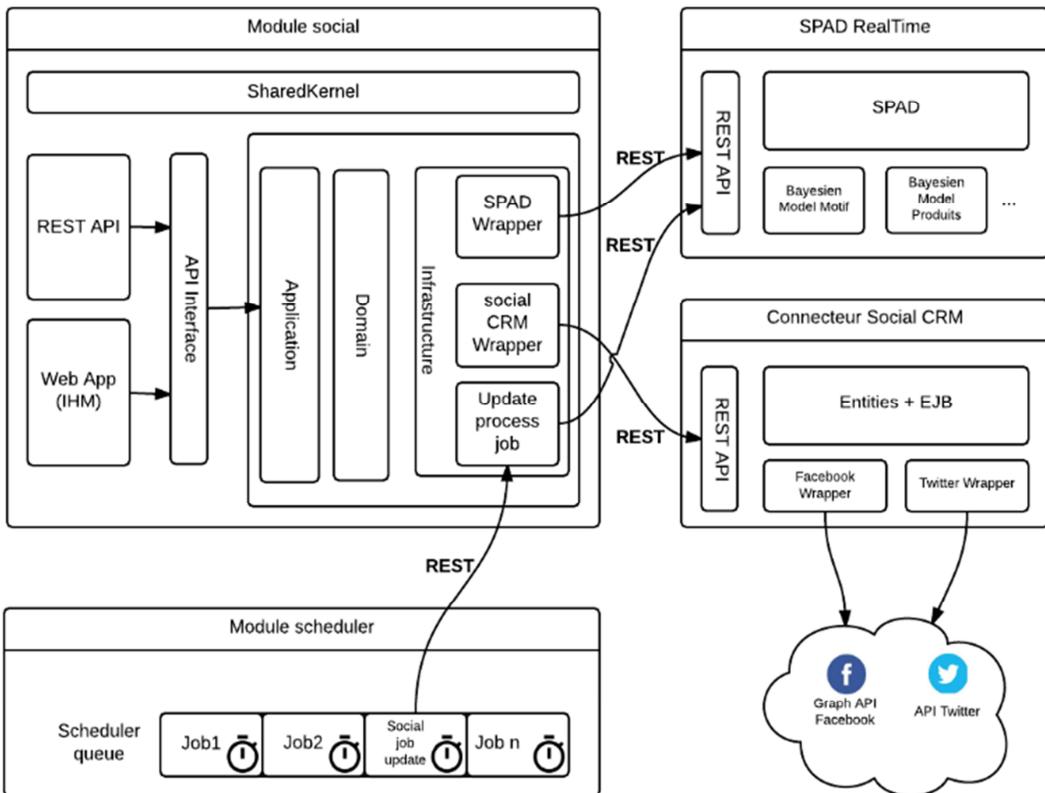


Figure 16: Architecture du module Social

Les différents objectifs de cette partie, était dans un premier temps l'initiation des développements sur le module par la création de la structure nécessaire basée sur l'architecture DDD. Et Dans un second temps, il était de trouver une approche pour intégrer les modèles de Text mining dans la solution en se basant sur le connecteur social existant.

### 2.3.2 Etude de l'existant

#### Connecteur social Coheris CRM :

La brique Social apporte la connectivité à diverse fonctionnalité dont on peut citer par exemple :

- Administration des profils Facebook et Twitter des utilisateurs CRM,
- Management des Fanpages Facebook: Consultation des Posts publiés sur les Fanpages administrées, réponse aux posts sur le mur, identification et gestion du cycle de vie des statuts des publications.
- Alimentation de dossier client sur CRM : Recherche du profil Facebook d'une personne, Recherche du profil Twitter d'une personne,

## PARTIE 2. TRAVAUX EFFECTUÉS

Récupération des indicateurs sociaux (Facebook : Likes, Friends, Post/Comment, Fanpage | Twitter : Followers, Tweets) pour la mesure d'influence de la personne.

- Ciblage et Reporting: Usage des données sociales dans le Ciblage ou le reporting, usage des indicateurs sociaux dans le ciblage.

Le connecteur est une brique applicative indépendante (on-premise, saas). Il offre une API REST qui répond aux fonctionnalités décrit ci-dessus.

En effet, il a sa propre base de données qui stocke les informations et les données associées aux profils Facebook et Twitter renseignés par les utilisateurs.

### **SPAD RealTime**

Ce module permet d'utiliser les modèles SPAD pour répondre en temps réel aux requêtes de scoring ou de typologie. En effet, et après des travaux réalisés en parallèle dans mon stage, le serveur était enrichi par des modèles de fouille de textes (Text mining) basées sur des algorithmes d'apprentissage bayésiens.

### **Architecture**

Le module offre une API REST qui utilise un dictionnaire JSON pour les paramètres en entrée et en sortie et qui répond aux fonctionnalités décrit-ci dessus.

La mise en œuvre du modèle se fait par la création d'un fichier « .model ». Chaque type de modèle à une représentation qui est différente aux autres types.

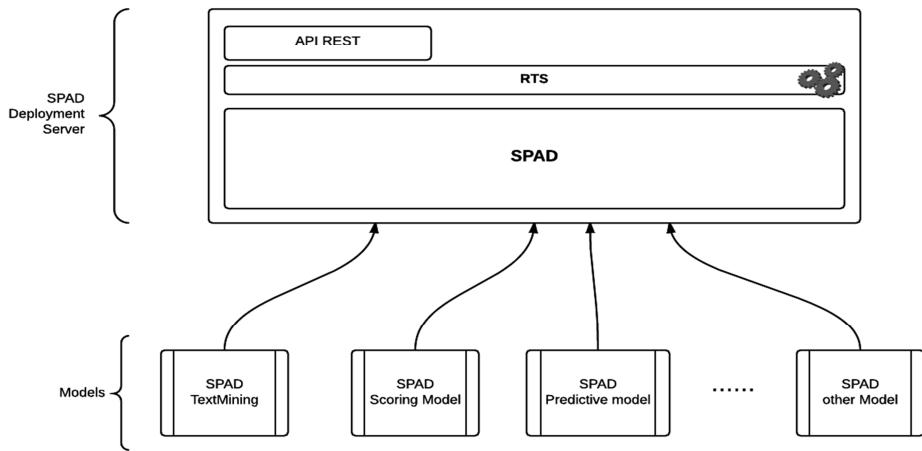


Figure 17: Architecture de serveur de déploiement SPAD RealTime

### Model Bayésien pour champs texte:

Le modèle bayésien est utilisé pour affecter un texte libre dans une catégorie parmi N catégories prédéfinis. Dans le cas des commentaires et les Feed Facebook, on peut classifier automatiquement le message selon une catégorie de sollicitation : Demande, Opportunité, Intervention, lead. Le fichier conteneur du modèle doit être sous le format suivant :

```

modelClass=maestro.web.bayes.TextBayesModel
modelName=bayesien Motif
possibleValues=demande,opportunité,lead,intervention
minLength=3
lang=fr
targetName=target
keysToParse=text
learnFile=<com.spad.modelsdir>\text\posts-synopsis.txt
##Modèle de bayésien naïfs en mode random.
##pour tester l'intégration avec CRM.
##renvoie une des valeurs au hasard parmis la liste des possible

```

Le tableau ci-après décrit des propriétés d'un fichier modèle Text mining SPAD realtime

Propriété	Valeur	Défaut	commentaire
<b>modelClass</b>	maestro.web.bayes.TextBayesModel	<b>requis</b>	<b>Obligatoire :</b> Il indique la classe qui implémente ce modèle
<b>modelName</b>	<libre>	<b>requis</b>	nom du modèle dans le serveur : c'est un identifiant pour les appels REST et il doit être unique
<b>possibleValues</b>	liste de valeur séparée par des ","	<b>requis</b>	Le nombre de valeur possible que prédit ce modèle. Il est aujourd'hui impossible de prendre en compte des valeurs contenant des ","
<b>keysToParse</b>	liste des clefs à parser	"text", "title"	Ensemble des variables du dictionnaire passé en paramètre qui doivent être parsées pour l'analyse textuelle. Toutes les valeurs sont considérées comme le même texte concaténé. S'il n'est pas spécifié on parsera les valeurs associées aux clefs "text" et "title".
<b>targetName</b>	nom de la variable cible pour l'apprentissage	"target"	nom de la variable cible dans le dictionnaire passé à la méthode d'apprentissage. Les valeurs associées à la clef doivent être dans possibleValues. S'il n'est pas spécifié on regarde la valeur associée à la clef "target"
<b>lang</b>	langage pour le parsing	fr	Langage pour les stops Word du parsing du texte. Par défaut, on est en français, si le code est différent de "fr" on est en anglais. (2 langages supportés seulement pour l'instant).
<b>minLength</b>	taille minimum d'un mot	3	Tous les mots de taille inférieure à cette valeur sont supprimés après le parsing.

Tableau 3:Table des propriétés d'un fichier modèle SPAD REALTIME pour le Text mining

### **Base de données de tests**

Afin de tester fonctionnellement l'efficacité de l'algorithme bayésien des model Text mining, on a besoin d'une base de données de test qui contient des commentaires et des posts réels.

En effet, L'existant connecteur social a été branché sur des pages Facebook de tests. Elle ne contient pas de véritable message qui ont de sens sur la page. Alors elle nécessite de choisir une page Facebook publique pour alimenter nos bases de données. Cela nécessite une connaissance de l'API Graph du Facebook qu'elle nous permet d'extraire les données publique sur une page choisie. De plus, on pourra même afficher le nom de l'internaute et son image publique sur Facebook.

#### **2.3.3 Réalisation**

L'essentiel du travail a consisté à appliquer les mêmes travaux réalisés dans le module scheduler. Mais la différence était de ne pas aborder les travaux de migration du connecteur social CRM vers le nouveau module DDD. Ces derniers ont été planifiés pour des sprints qui ont commencé à la fin de mon stage.

Finalement, le but était de mettre en œuvre la testabilité des algorithmes de textminig du serveur RealTime SPAD sur des sources de donnée réel.

### **Branchement avec l'existant connecteur social CRM**

D'un point de vue structurel, et comme dans l'exemple du Framework Quartz dans le module Scheduler, les interfaces responsables aux appels des web services externes ont été définies dans la couche infrastructure comme étant une ressource externe du nouveau module Social.

Afin de récupérer les posts Facebook et les commentaires d'un fanpage publique, j'ai eu besoin d'étendre l'api du connecteur social CRM avec de nouvelles fonctionnalités.

Tous ces appels ajoutés, prennent le nom de la Fanpage comme paramètre de recherche dans le graph API Facebook. De plus, il était primordial de migrer les sources des entités métier, porteuses des informations, du

connecteur social CRM vers la couche shared-kernel (voir annexe B), afin de ne pas tomber sur des conflits entre les objets passés en JSON et les entités Domain de la nouvelle architecture.

Enfin, pour alimenter la base de données du connecteur Social, j'ai créé une tâche scheduler qui fait la mise à jour périodique avec les nouveaux posts et commentaires de la Fanpage ciblée.

### **Intégration de SPAD RealTime**

De la même manière, et comme pour l'existant connecteur Social CRM, le branchement était côté infrastructure. Mais avant de commencer, il était nécessaire de faire des tests sur l'api de Spad RealTime pour savoir les paramètres entrées/sortie des requêtes.

Prenant l'exemple du modèle de qualification des messages selon leur motif «bayésien» :

Fichier bayesienMotif.model

```
modelClass=maestro.web.bayes.TextBayesModel
modelName=bayesien motif
possibleValues=demande,opportunity,lead,intervention
minLength=3
lang=fr
targetName=target
keysToParse=text
##Modèle de bayésien naïfs en mode random.
##pour tester l'intégration avec CRM.
##renvoie une des valeurs au hasard parmis la liste des possible
```

Les paramètres d'entrées sous format json

```
{modelName=bayesien Motif, nameToValueMap={text=Cela fait maintenant 2 mois que je me plaind de ne plus pouvoir commander sur le net}}
```

Retour sous format json

```
[ "intervention", 32.38115350352977, 3.8666739556296896, 22.402458516347743,
41.349714024492805 ]
```

On s'intéresse plutôt au retour fourni par le serveur pour la qualification d'un message. En effet, la liste comme on la constate contient la décision du serveur à propos du contenu de message « *intervention* » suivi de pourcentage de chaque Target dans le même ordre écrit dans le fichier du model. Dans un premier temps, et suivant la spécification fonctionnelle du module, on a juste besoin de sauvegarder le motif décisif de la requête dans un champ *BusinessMotif* que j'ai ajouté pour l'entité commentaire et l'entité Post.

A noter dans les développements des interfaces responsables aux appels web service au serveur SPAD RealTime, qu'on peut passer une liste de model à jouer sur un message.

Image classe interface wrapper liste (Manque l'image !)

### Prototype IHM

Enfin, et afin de visualiser le résultat de mon travail dans le module Social, j'ai été amené à réaliser une IHM qui affiche les messages provenant du post ou du commentaire avec leur qualification selon le champ *BusinessMotif*.

The screenshot shows the COHERIS Social Home interface. On the left, there's a sidebar with 'Bureau' selected, followed by 'Social' which is also highlighted. Below that are 'Datamart', 'Scheduler', and 'Archive'. The main area has tabs for 'Compose' and 'Inbox'. Under 'Compose', there are sections for 'tous' (33), 'facebook' (33), 'twitter' (25), 'e-mail' (20), and 'interne' (9). Under 'Archive', there are sections for 'important' (20), 'archive' (350), and 'corbeille' (47). The 'Inbox' tab is selected, showing a list of messages from various users. Each message includes a snippet of text and a 'BusinessMotif' classification on the right. The classifications shown are 'NOCASE', 'INTERVENTION', 'DEMANDE', 'LEAD', 'OPPORTUNITY', and 'INTERVENTION' again. The interface has a clean, modern design with a light blue header and a white background.

Figure 18: Capture d'écran du tableau de bord du module Social

## PARTIE 2. TRAVAUX EFFECTUES

De plus, il m'a été demandé de rendre le champ de la qualification éditabile pour requalifier le motif et pour appeler en même temps la fonction d'apprentissage du serveur SPAD.

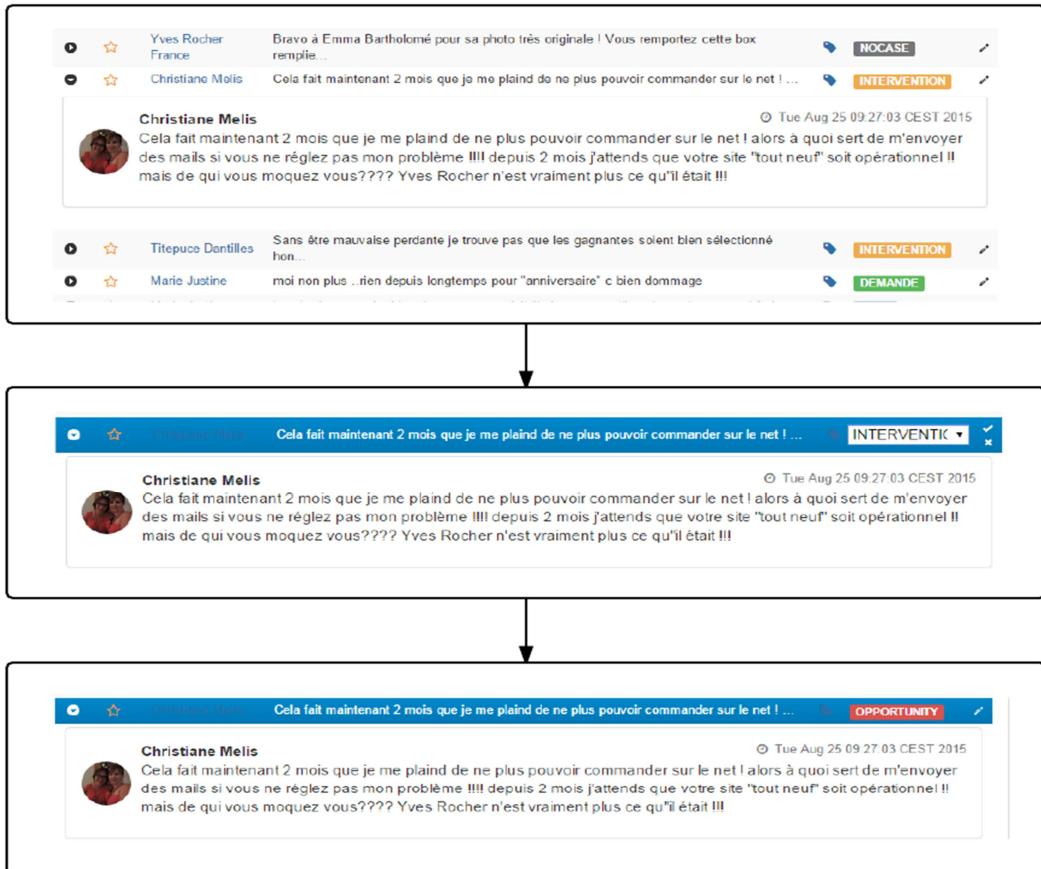


Figure 19: Capture d'écran de l'édition de la qualification

### 2.3.4 Bilan

L'intégration de SPAD Real Time dans le module social DDD était un travail extrêmement complet et qui permettra dans les prochaines itérations de l'équipe SPAD ou de l'équipe R&D CRM de tester l'efficacité des algorithmes Bayésiens naïves sur la qualification. En effet, les réalisations étaient fragmentées en plusieurs étapes afin d'accompagner les travaux d'études menés en parallèle.

Toujours en cours de développement, la finalité de ce projet est de remplacer l'existant connecteur Social par un module indépendant avec des fonctions plus avancé comme le Text mining [15] qui répond aux attentes des clients.

## **Partie 3**

# **Bilan**

### **Sommaire**

---

<b>3.1 RESULTATS OBTENUS .....</b>	<b>39</b>
3.1.1 Mise en œuvre de l'approche Domain-driven design.....	39
3.1.2 Planificateur de tache de hautes disponibilités.....	39
3.1.3 Intégration de la Text mining dans le module social.....	40
<b>3.2 DIFFICULTES RENCONTREES.....</b>	<b>40</b>
3.2.1 Compilation des modules découplés.....	40
3.2.2 Tester l'efficacité de l'algorithme de Text mining.....	42

---

### 3.1 Résultats obtenus

Durant mon stage, j'ai travaillé sur différentes tâches. Cette partie synthétise le travail effectué ainsi que les résultats obtenus afin de les replacer par rapport aux objectifs initiaux qui avaient été fixés

#### 3.1.1 Mise en œuvre de l'approche Domain-driven design

Ce premier travail avait avant tout un rôle introductif vis-à-vis de l'environnement logiciel et technique de Coheris, a conduit à mettre en œuvre la nouvelle architecture DDD avec le design pattern CQRS et Event-Sourcing. La structure d'un module nouvelle offre Coheris permet d'offrir plus de flexibilité lors de l'ajout des nouvelles fonctionnalités. De plus, il permet la séparation des rôles technique en encapsulant chacun dans une couche applicative.

De même, les différentes modes de déploiements nous permettent selon les besoins du client, de choisir qu'elle approche de déploiement il faudra mettre en œuvre. N'oublions pas la possibilité de lancer le module soit en mode on-premise soit en mode multi-tenant.

#### 3.1.2 Planificateur de tâche de hautes disponibilités

Le module Scheduler était une véritable formation sur le Domain-driven design. Il m'a permis de mieux comprendre le principe de l'architecture des modules étudiés dans la première partie des travaux. En effet, mon travail s'est étalé sur 2 mois et a permis d'apporter un système de planification de tâche qui peut être déployé suivant différents modes. N'oublions pas l'apport de l'intégration du Framework Quartz dans la solution. Ce dernier nous a apporté plusieurs avantages par rapport au scheduler de Coheris CRM : La gestion des retards, le multithreading, la mise en mode cluster, et la persistance des tâches. Le module Scheduler semble donc répondre aux attentes.

### **3.1.3 Intégration du Text mining dans le module social**

Le module social DDD, comme on l'appelle en interne, est la nouvelle génération du connecteur Social existant. Coheris souhaite le vendre comme étant une application indépendante à la fin de l'année 2015. Mon travail a d'une part consisté à mettre en place la structure de ce module en se basant sur l'approche DDD, de préparer la migration du connecteur Social CRM. Et d'autre part, j'ai eu en charge de créer les fonctionnalités nécessaires, d'abord pour intégrer SPAD RealTime dans le module, et ensuite de créer les interfaces de Tests que la solution nécessite.

Cette partie technique a été précédée d'une étude complète du connecteur Social existant afin d'en extraire les fonctionnalités utiles pour bien répondre aux besoins.

Enfin, mon travail a ainsi permis de mettre en place une stratégie pour tester le Text mining sur les posts et les commentaires Facebook.

## **3.2 Difficultés rencontrées**

Cette partie a pour but de présenter les éléments qui ont pu ralentir ma progression dans le travail. Je vais principalement aborder des points techniques. D'abord, je présenterai les structures des projets java et le système de build adopté. Ensuite, j'attirerai l'attention sur les difficultés rencontrées pour tester les algorithmes de Text mining de SPAD RealTime

### **3.2.1 Compilation des modules découplés**

#### **Structure des projets d'un module DDD dans la plateforme JE22**

En utilisant la plateforme JAVA EE de Coheris, et après une étude faite sur la nouvelle architecture, l'équipe R&D CRM s'est accordée sur le fait que chaque couche du module (chaque rôle de l'approche DDD) est encapsulée dans un projet java et déployé en tant qu'archive jar.

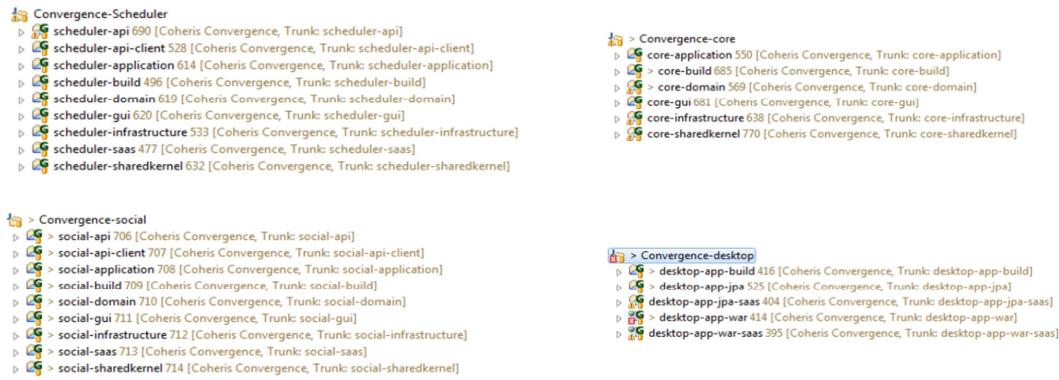


Figure 20: Capture d'écran des projets java des différents modules

Il faut prendre en compte aussi que chaque couche de module dépend de la même couche dans le module core. C'est-à-dire, si on prend par exemple la couche module-application, elle dépend de la couche core-application. La couche module-infrastructure dépend de la couche core-infrastructure et ainsi de suite. De plus, chaque module a son Tronc de développement dans le gestionnaire de version SVN. Le fait d'ajouter une fonctionnalité ou d'ajouter une classe dans le core, il doit être d'abord compilé et encapsulé dans un jar et après publié sur Artifactory [16]. Pour le module scheduler ou le module social, et avant de lancer son build, il faut s'assurer qu'on dépend de la bonne version du module core. Sinon, il faut refaire tout l'enchaînement décrit pour le module core.

### Tester le module

Afin de pouvoir tester les fonctionnalités d'un module, on doit mettre en œuvre une autre approche d'encapsulation. Pour assurer les points d'entrées vers ce dernier soit à travers son api REST, soit à travers son IHM. Alors, un projet web dynamique (war) sous le nom «desktop-application» assure l'encapsulation des modules développés. Ce qui m'a permis de tester les mises à jour des messages Facebook dans le module Social avec une tâche planifiée créé à partir de l'IHM du module Scheduler.

Le problème ici apparait lorsqu'on fait une modification dans le code du module core et dans le module social par exemple. En effet, on ne peut pas utiliser la modification de ce code à chaud. Il faut donc suivre l'enchainement suivant :

- 1. compiler et publier le module core,
- 2. compiler le module social
- 3. republier les nouveaux jars dans le serveur d'application

De plus, j'étais obligé de migrer tous les pages JSF de chaque module de son couche gui vers le desktop-application pour que je puisse assurer la modification du code HTML. A la fin des tests, il faut les ré-migrer vers la couche gui de son module initiale.

### 3.2.2 Tester l'efficacité de l'algorithme de Text mining

Malheureusement, je n'ai pas eu l'opportunité de confirmer l'efficacité des algorithmes de Text mining SPAD RealTime sur les messages Facebook avec le model «bayesien motif». En effet, les développeurs sont encore en cours de développement sur le serveur SPAD. Aussi, les tests permettant la validation de ces algos demanderont un certain temps.

Il faudra d'abord apprendre l'algorithme m par des milliers de message bien qualifier et les mettre dans un fichier d'amorçage au démarrage du serveur. A titre illustratif, le code relatif à ce dernier fichier est le suivant :

Cette dernière partie devrait être revue pour cadrer avec les derniers choix effectués : Pas de fichier d'initialisation, mais plutôt un amorçage via la qualification manuelle d'un certain nombre de post (en se servant de l'ihm développée pour).

```
target text

demande      je veux
demande      j'aimerai
opportunity   Vous ne faites plus rien pour les anniversaires?! Je n'ai même pas eu de mail
               avec une ptite offre
intervention   moi non plus ..rien depuis longtemps pour ""anniversaire"" c bien dommage
lead         Merci à toutes je suis vraiment étonnée ! Et merci à Yves rocher pour ce super
               concours !
...

```

### PARTIE 3. BILAN

On constate ici qu'il faut alimenter ce fichier pour qu'à chaque démarrage du serveur, ce dernier soit prêt à donner des bons résultats de qualification. Il faudra que toute l'équipe R&D CRM assiste à l'alimentation de ce fichier par des messages bien qualifiés pendant des jours.

# Conclusion

Mon stage s'est inscrit dans une problématique d'intégration du Serveur SPAD RealTime dans le nouveau module Social de la nouvelle offre de Coheris. Mon travail a conduit à la mise en place d'une solution d'intégration du text mining dans ce dernier, ainsi que le développement de fonctionnalités additionnelles pour faciliter les tests dans les prochains travaux de Coheris. Les travaux ont répondu aux résultats attendus. L'intégration du Text Mining a un impact important dans le mode du travail des Community Manager. La perte du temps dans la classification des messages sociaux est résolue grâce au Text Mining.

Au cours de mon stage, Coheris a changé son stratégie vers une offre modulaire simple à déployer qui répond aux attentes de ses clients. J'avais de la chance à avoir bien assisté au développement des ces nouvelles modules qui se base sur des nouvelles technologies de pointe : Java EE7, gradle, JSF...

Ce stage, était mon premier contact avec le milieu professionnel et conclut mes études supérieures, me conforte dans l'idée de débuter ma future carrière dans le domaine de l'informatique et plus particulièrement dans les technologies et les architectures orientées web. Cette expérience a été très enrichissante et m'a permis de parfaire mes connaissances dans la programmation objet, la conception et la modélisation.

J'ai apprécié la grande autonomie et la confiance que l'on m'a accordées. Je remercie une nouvelle fois la société Coheris pour m'avoir accueilli et je suis heureux d'avoir travaillé avec l'équipe de Recherche et développements CRM, qui est une équipe dynamique et accueillante et qui contient des collaborateurs professionnels.

# Ressources

## **Domain-Driven Design Vite fait**

<http://blog.infosaurus.fr/public/docs/DDDViteFait.pdf>

Article écrit par Abel Avram & Floyd Marinescu, 2006, C4Media.

## **DDD and CqRS Leaven**

<http://ddd-cqrs-leaven.blogspot.fr/>

Groupes de discussion en anglais du projet DDD and CqRS Sample Leaven J2EE.

## **Java EE 7 Development with WildFly**

Livre écrit par Francesco Marchioni, Michal Cmil et Michal Matloka. 2014, isbn: 978-1-78217-198-0. Résume les méthodologies et les fonctionnalités implémentées par le nouveau serveur wildfly

## **Quartz Scheduler Product Documentation 2.2.1**

<http://quartz-scheduler.org/documentation>

Documentation officielle du framework Terracota Quartz scheduler.

## **Java(TM) EE 7 Specification APIs**

<http://docs.oracle.com/javaee/7/api/>

Documentation officielle de la spécification JEE 7.

## **Bootstrap**

<http://getbootstrap.com/>

Le site et la documentation officielle du framework Twitter Bootstrap.

# Glossaire

## [1] **CRM- Customer Relationship Management**

La gestion de la relation client est l'ensemble des outils et techniques destinés à capter, traiter, analyser les informations relatives aux clients et aux prospects, dans le but de les fidéliser en leur offrant le meilleur service

## [2] **SaaS - Software as a Service**

C'est un modèle d'exploitation commerciale des logiciels dans lequel ceux-ci sont installés sur des serveurs distants plutôt que sur la machine de l'utilisateur.

## [3] **DDD-Domain Driven Design**

La conception pilotée par le domaine est une approche de la conception de logiciel basée sur le principe des conceptions complexes qui doivent être basées sur un modèle et sur le principe du focus qui doit être sur le domaine et la logique associée, sans égard à l'implémentation.

## [4] **J2EE Java Entreprise Edition**

C'est une spécification pour la technique Java d'Oracle plus particulièrement destinée aux applications d'entreprise.

## [5] **EJB – Entreprise Java Bean**

Les Entreprise Java Bean ou EJB sont des composants serveurs donc non visuels qui respectent les spécifications d'un modèle éditées par Sun. Ces spécifications définissent une architecture, un environnement d'exécution et un ensemble d'API.

## [6] **JPA – Java Persistence API**

C'est est une interface de programmation Java permettant aux développeurs d'organiser des données relationnelles dans des applications utilisant la plateforme Java.

## [7] **JSF – JavaServer Faces**

C' est un framework Java, pour le développement d'applications Web. Il est basé sur la notion de composants, comparable à celle de Swing

ou SWT, où l'état d'un composant est enregistré lors du rendu de la page, pour être ensuite restauré au retour de la requête.

[8] **CDI – Contexts and Dependency Injection**

C'est une spécification (JSR 299) de Java EE définissant une interface de programmation (ou API) pour l'injection de dépendances. Son implémentation de référence est Weld.

[9] **JAX-RS – Java API for RESTful Web Services**

C'est une interface de programmation Java permettant de créer des services Web avec une architecture REST.

[10] **Gradle**

Gradle est un moteur de production fonctionnant sur la plateforme Java. Il permet de construire des projets en Java, Scala, Groovy voire C++. Gradle permet d'écrire des tâches de constructions dans un fichier de construction en utilisant le langage Groovy.

[11] **MVC - Model-View-Controller**

C'est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective.

[12] **Groovy**

C'est un langage de programmation orienté objet destiné à la plate-forme Java. Il s'intègre et est entièrement compatible avec la JVM étant donné que le bytecode est le même. Il peut donc utiliser les bibliothèques Java et être utilisé dans des classes Java.

[13] **cron**

Le syntaxe cron est utilisé pour écrire un script de planification des tâches sous Unix

[14] **API - Application Programming Interface--**

Interface servant de base pour concevoir un programme capable d'interagir avec un autre logiciel. Ensemble de fonctions qui a pour but de faciliter le travail du développeur.

[15] **Text mining**

Elle désigne un ensemble de traitements informatiques consistant à extraire des connaissances selon un critère de nouveauté ou de similarité dans des textes produits par des humains pour des humains

## [16] Artifactory

C'est un gestionnaire de référentiels de binaires. Il permet la gestion des repository pour Apache Maven, mais bien au-delà de Maven, c'est l'entrepôt unique pour tous les artefacts d'une usine logicielle. Artifactory est capable de conserver de façon pérenne, distribuée et sécurisée tous les livrables.

# Liste des figures

Figure 1: Organigramme de l'entreprise COHERIS .....	9
Figure 2: Organigramme de la direction de l'offre .....	11
Figure 3:Socle J2EE Coheris .....	12
Figure 4:Communication entre le backend et le frontend dans un même serveur .....	17
Figure 5: Communication entre le backend et le frontend déployé sur deux serveurs distincts .....	17
Figure 6:Mode1 .....	18
Figure 7:Mode 2 .....	19
Figure 8: Mode 3 .....	20
Figure 9 : Architecture du module scheduler.....	21
Figure 10: Concepts Quartz.....	23
Figure 11: Architecture du module Scheduler.....	25
Figure 12:capture d'ecran du dashboard Coheris .....	27
Figure 13:capture écran du gestionnaire de tâches .....	27
Figure 14:Capture d'écran du gestionnaire des traitements .....	28
Figure 15:Capture d'écran sur le gestionnaire d'historiques .....	28
Figure 16:Architecture du module Social .....	30
Figure 17:Architecture de serveur de déploiement SPAD RealTime.....	32
Figure 18:Capture d'écran du tableau de bord du module Social .....	36
Figure 19:Capture d'écran de l'édition de la qualification .....	37
Figure 20:Capture d'écran des projets java des différents modules.....	41
Figure 21: Les couches DDD.....	55
Figure 22:Architecture DDD avec CQRS .....	57

# Liste des tableaux

Tableau 1:Produit Coheris .....	8
Tableau 2: Tableau comparatif des Framework étudiés .....	22
Tableau 3:Table des propriétés d'un fichier modèle SPAD REALTIME pour le Text mining.....	33

# **Annexes**

## **Sommaire**

---

<b>A. PLANNING .....</b>	<b>52</b>
<b>B. ARCHITECTURE D'UN MODULE AVEC DOMAIN DRIVEN DESIGN.....</b>	<b>54</b>

---

## A. Planning

Le planning reporté ci-dessous est le planning effectif. Les numéros des semaines correspondent à ceux du calendrier civil. A noter que l'équipe R&D CRM à Coheris utilise l'outil **JIRA** afin d'assurer le suivi des projets. Cela permet, d'une part, d'affecter une tâche à un utilisateur et d'autre part, de permettre aux supérieurs hiérarchiques d'être tenus informés du bon déroulement des opérations. Je devais ainsi inscrire au quotidien le travail effectué avec le volume horaire associé et le pourcentage d'avancement des travaux affectés.

### Mars

#### **Semaine 10 - Découverte du Connecteur social CRM**

Prise en main de l'environnement technique et logiciel de la R&D Coheris.  
Etude sur le connecteur social CRM et l'API Graph Facebook.

#### **Semaine 11 – Découverte de SPAD Analytics**

Formation sur SPAD Analytics. Réalisation de la maquette des IHM dédié à l'affichage des messages et leurs qualifications.

#### **Semaine 12, 13 – Etudes des fonctionnalités de tests de Text mining**

Etendre l'API REST de connecteur social CRM par des fonctionnalités de test de collection des posts et commentaires d'une page publique.

### Avril

#### **Semaine 16 – Interface IHM de collection des**

Création d'un composant IHM pour visualiser les postes et des commentaires Facebook sur l'existant CRM Cloud.

#### **Semaine 17- Ecriture de la Pré-rapport**

Préparer le pré-rapport, faire un point à mi-parcours, sur l'avancée de mon travail de stage.

**Semaine 18 - Formation sur la nouvelle architecture**

Etudier l'approche Domain-Driven Design. Etudier les design pattern CQRS et Event-Sourcing. Modélisation de l'architecture.

**Semaine 19 - Etudes du module scheduler**

Réaliser une étude comparative sur les solutions existantes pour un système de planification des tâches.

**Mai**

**Semaine 20, 21, 22 -Backend du module scheduler NO**

Etudier et Intégrer le Framework Quartz dans le module. Développer les différentes couches applicatives du module. Mise en place de l'api REST du module.

**Semaine 23- Test et Documentation du module Scheduler NO**

Documenter les travaux faits et les solutions utilisées du module scheduler NO.

**Juin**

**Semaine 24- Amélioration de la partie IHM Desktop-application**

Refonte graphique et technique du Layout de l'application conteneur des module «desktop-application». Remonter en version tous les Framework: Bootstrap, Font-Awesome, jquery.

**Semaine 25,26 - Frontend du module scheduler NO**

Création des composants JSF de la couche gui du module, tableau de Bord, formulaire d'ajout d'un traitement, formulaire d'ajout d'une tache. Affichage des historiques des taches.

**Semaine 26 -SPAD RealTime**

Etudier la première version du serveur de déploiement SPAD Realtime. Tester l'api du serveur avec apache jmeter.

**Semaine 27 - Etude du module social NO**

Etudier l'intégration de connecteur social CRM dans le module. Préparer les interfaces responsables aux appels web services vers l'api du SPAD.

**Juillet**

**Semaine 28, 29,30 - Backend Module social NO**

Développer les différentes couches applicatives du module. Mise en place de l'api REST du module. Branchement de SPAD RealTime. Alimentation de la base donnée du connecteur social par des réel post et commentaire Facebook importer depuis une page publique quelque que.

**Semaine 31- Test et Documentation du module social NO**

Documenter les travaux faits et les solutions utilisées du module social NO.

**Août**

**Semaine 32, 33 - Frontend du module social**

Création les composants JSF de la couche gui du module afin d'afficher les qualifications des messages avec le serveur SPAD. Ajouter la fonction d'apprentissage Learn dans l'IHM.

**Semaine 34,35 – Documentation des travaux et transfert de compétence**

**B. Architecture d'un module avec Domain Driven Design**

**Présentation du DDD**

Le Domain Driven Design n'est ni une méthode ni une technologie. C'est est une manière de penser sur la conception autour du code, de collaborer et de communiquer avec les experts fonctionnels.

Il a été introduit par Eric Evans en 2003. Il a construit cette approche de conception suite à des retours d'expérience. Le DDD est centré sur le métier de l'application et le code source qui l'implémente.

En effet, la conception est conduite par un modèle. Ce modèle est en partie constitué d'un langage de communication commun aux experts fonctionnels et aux équipes de développement appelé *Ubiquitous*.

Techniquement, C'est un ensemble de concepts et de design patterns. Il permet :

- Isolation de la couche domaine avec les concepts d'inversion de contrôle et d'injection de dépendances.
- Découpler toutes les couches d'infrastructure de la couche du domaine
- Utilisation de nombreux patterns d'architecture (MVC, CQRS...)
- Utilisation d'ORM

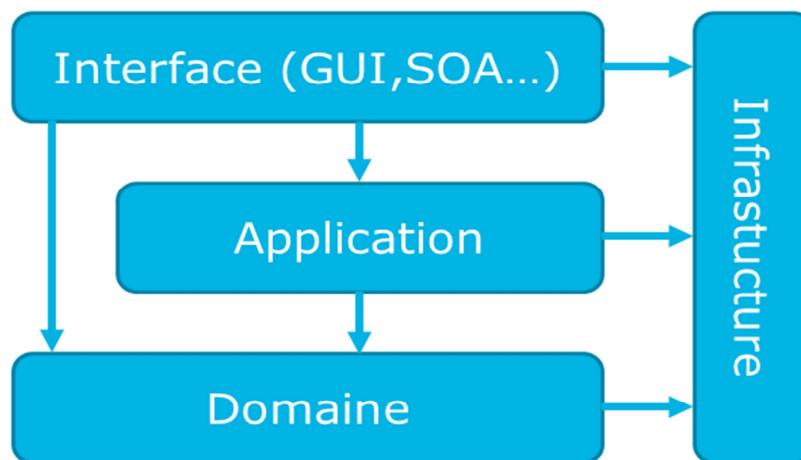


Figure 21: Les couches DDD

### Couche Domain :

C'est le cœur du logiciel et la base de discussion avec un expert métier. Elle contient tous les concepts métier «purs» et les entités associées. En effet, elle ne doit pas être contaminée avec des problématiques issues d'autres couches et doit être indépendante de toute source de données ou technologie d'accès. De plus, toutes les dépendances doivent utiliser des abstractions ou des interfaces. On utilise des conteneurs IoC (comme CDI pour java EE) pour proposer des implémentations interchangeables. Enfin, toutes les entités (au sens ORM) sont présentes dans cette couche.

**Couche Application :**

Elle contient toute la plomberie de coordination entre les différentes couches. C'est la chef d'orchestre des uses cases métiers (synchrone ou asynchrone). En effet, Elle ne contient pas de métier mais elle gère le cycle de vie de l'entité métier. De plus, elle peut envoyer des Events de niveau applicatif et peut contenir des EventListeners de niveau domaine. En fin, elle contient et implémente tous les services applicatifs.

**Couche Infrastructure :**

C'est une commune aux différentes couches. Elle contient tous ce qui existe indépendamment de l'application : les librairies externes, le « Database Engine » ou l'entité manager pour l'ORM attaché. Et le plus important, l'implémenter des Repository définies dans la couche Domain.

**Couche Interface ou présentation :**

Elle se charge de l'interprétation, la validation et la transformation des données entrantes et aussi de la serialization des données sortantes : xml, html, json, etc.

**Concepts dans DDD :**

Bounded context: Context Borné défini par son langage (Ubiquitous language) : noms+verbes présents dans le code aussi

ValueObject : Objet dont l'état ne change pas (immutable), en général c'est une classe (pas une entité) réutilisable dans différentes Entités.

Entity : Objet dont l'état peut changer mais pas son identité (mutable), il est constitué d'attributs et de valueObjects (en JPA @Embedded). Dans DDD, une entité contient du code métier.

Aggregate : Objet (Entité) du domaine constitué d'entités, d'attributs et de valueObjects, il correspond à l'atomicité d'une transaction : intégrité de l'ensemble

Repository : il gère le cycle de vie des Entités. C'est comme un DAO sans en être un, voire après dans CQRS.

DTO (Data Transfert Object) : C'est un « subset » d'une entité métier. Il est plus léger que de transférer l'entité entière (un peu comme un InfoItem configuré pour une vue).

Shared Kernel (Noyau partagé) : il permet de partager des subsets communs aux domaines (référentiel commun)

SAGA : c'est le process (events) manager entre bounded contexts. Il permet par exemple de corrélérer différents events pour une prise de décision comme gestion du workflow d'events.

## Présentation de CQRS

Le pattern CQRS (*Command Query Responsibility Seggregation*) repose sur un principe simple: la séparation, au sein d'une application, des composants de traitement métier de l'information («command» / écriture) et de restitution de l'information («query» / lecture).

Ce seul principe fournit un cadre d'architecture extrêmement intéressant pour le développement d'applications, en levant un certain nombre de contraintes et en faisant apparaître de nouvelles opportunités.

## Architecture DDD avec CQRS

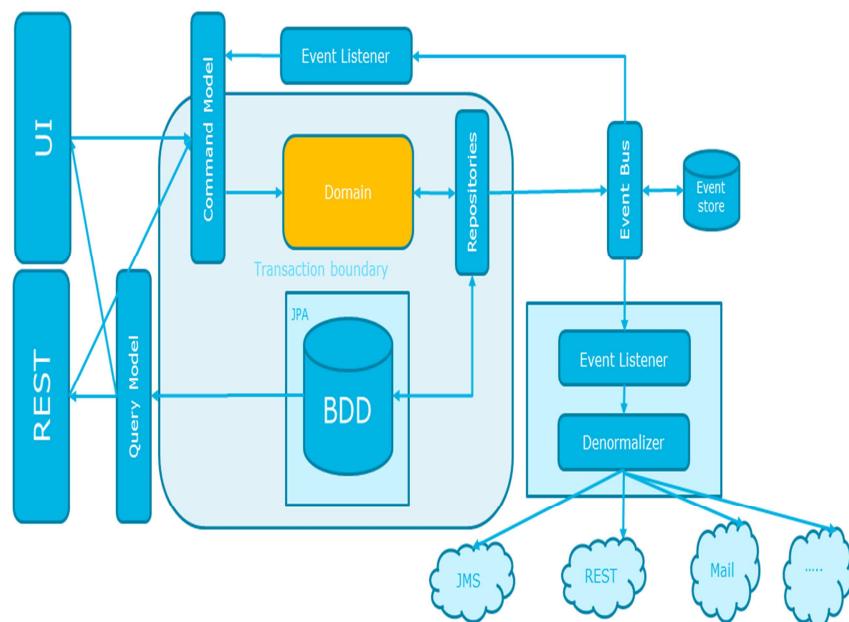
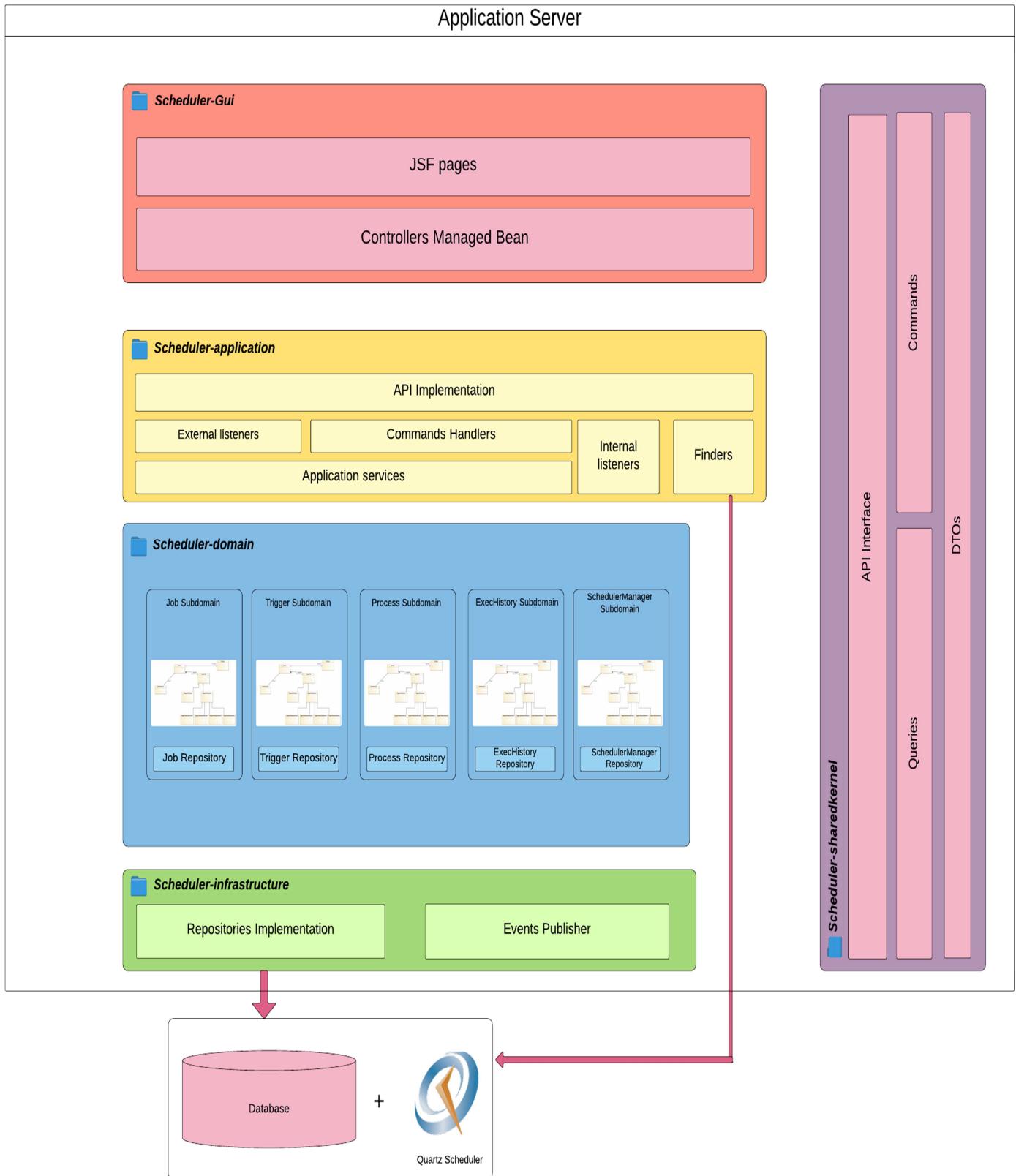


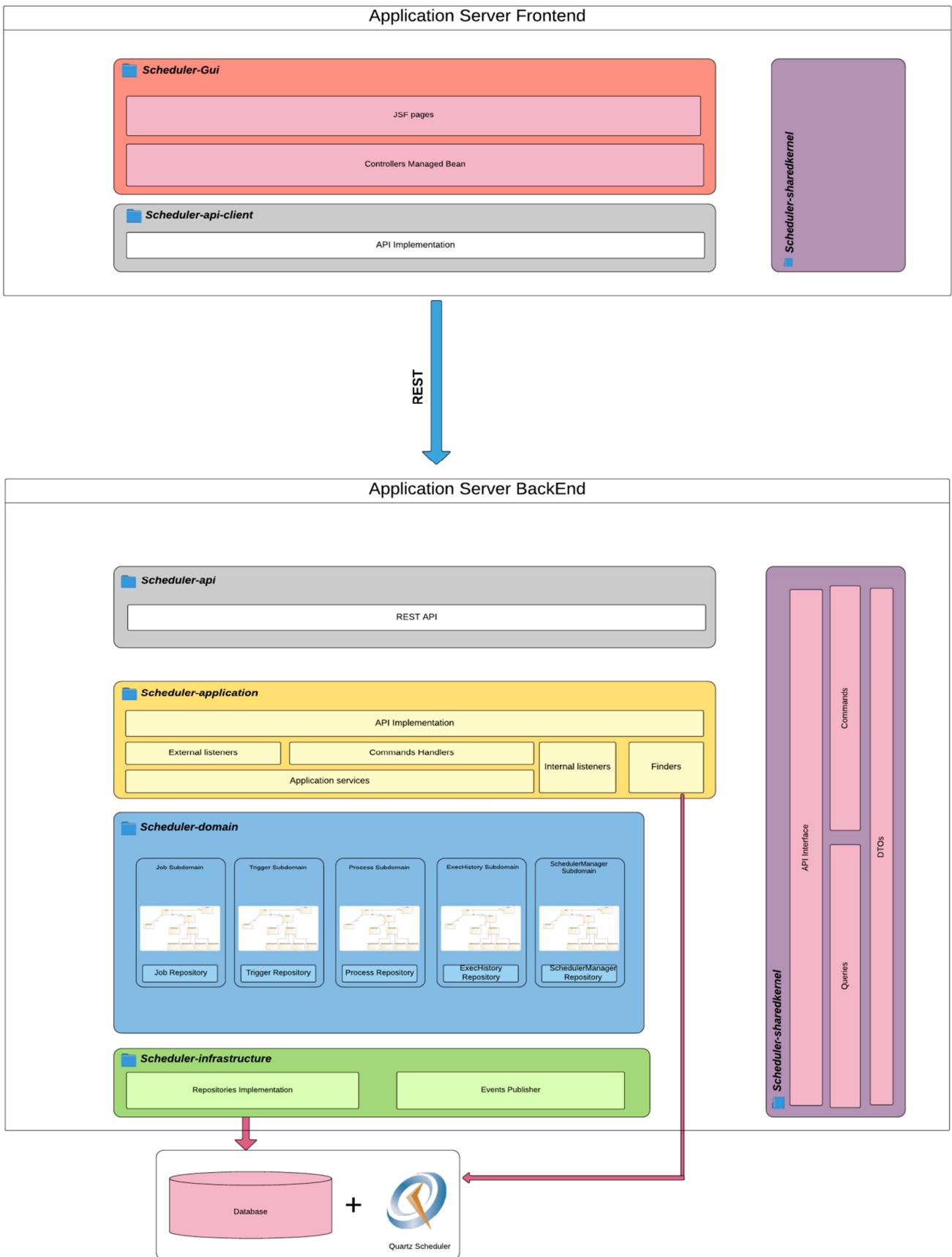
Figure 22: Architecture DDD avec CQRS

## Vues détaillées de l'architecture DDD en différents modes de déploiements: Exemple module scheduler NO

- Mode 1 : Backend et frontEnd dans le même serveur



- Mode 2 : backend et front end sur 2 serveur différents



- Mode 3 : Backend avec api REST

