Stage Ingénieur CRM



Présenté par :

**Oussama ELABED**

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 des études en vue d’obtention d’un 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, ainsi de réaliser une veille technologique sur les systèmes de planification de tâches et de pratiquer la conception avec l’approche Domain-Driven Disign.

#### Mots clés

CRM, DDD (Domain-Driven Disign), 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 -[**www.coheris.com**](http://www.coheris.com/)

* Ludovic Chabot

R&D CRM

4, rue du Port aux Vins - 92150 Suresnes

lchabot@coheris.com -[**www.coheris.com**](http://www.coheris.com/)

Remerciements

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

Je remercie vivement mon maitre 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 m’a facilité mon intégration dans l’entreprise.

Liste des figures

[Figure 1: Organigramme de l’entreprise COHERIS 11](#_Toc428988162)

[Figure 2: Organigramme de la direction de l'offre 13](#_Toc428988163)

[Figure 3:Socle J2EE Coheris 14](#_Toc428988164)

[Figure 4:Communication entre le backend et le frentend dans un mme serveur 18](#_Toc428988165)

[Figure 5: Communication entre le backend et le frentend déployé sur deux serveurs distincts 19](#_Toc428988166)

[Figure 6:Mode1 20](#_Toc428988167)

[Figure 7:Mode 2 21](#_Toc428988168)

[Figure 8: Mode 3 22](#_Toc428988169)

[Figure 9:Architecture du module scheduler 24](#_Toc428988170)

[Figure 10: Concepts Quartz 26](#_Toc428988171)

[Figure 11: Architecture du module Scheduler 28](#_Toc428988172)

[Figure 12:capture d'ecran du dashbord coheris 29](#_Toc428988173)

[Figure 13:capture écran du gestionnaire de taches 30](#_Toc428988174)

[Figure 14:Capture d'ecran du gestionnaire des traitements 30](#_Toc428988175)

[Figure 15:Capture d'écran sur le gestionnaire d'historiques 31](#_Toc428988176)

[Figure 16:Architecture du module social 32](#_Toc428988177)

[Figure 17:Architecture de serveur de déploiement SPAD RealTime 34](#_Toc428988178)

[Figure 18:Capture d'écran 38](#_Toc428988179)

[Figure 19:Capture d’écran 39](#_Toc428988180)

[Figure 20:Capture écran 42](#_Toc428988181)

[Figure 21: Les couches DDD 50](#_Toc428988182)

[Figure 22:Architecture DDD avec CQRS 52](#_Toc428988183)

Liste des tableaux

[Tableau 1:Produit Coheris 10](#_Toc428988156)

[Tableau 2: Tableau comparatif des Framework étudiés 25](#_Toc428988157)

[Tableau 3:Table de propriété d'un fichier modèle SPAD REALTIME pour le textmining 35](#_Toc428988158)

Sommaire

[Introduction 8](#_Toc428988184)

[1 Présentation du stage 9](#_Toc428988185)

[1.1 Société Coheris 10](#_Toc428988186)

[1.1.1 Présentation 10](#_Toc428988187)

[1.1.2 Produits 10](#_Toc428988188)

[1.1.3 Organisation 11](#_Toc428988189)

[1.1.4 En chiffres 11](#_Toc428988190)

[1.2 Mission 11](#_Toc428988191)

[1.2.1 Sujet de stage 11](#_Toc428988192)

[1.2.2 Besoin 11](#_Toc428988193)

[1.2.3 Équipe 12](#_Toc428988194)

[1.2.4 Environnement de travail 13](#_Toc428988195)

[2 Travaux effectués 16](#_Toc428988196)

[2.1 Etude de la mise en œuvre de l’approche Domain Driven design 17](#_Toc428988197)

[2.1.1 Objectifs 17](#_Toc428988198)

[2.1.2 Etude et réalisation : 18](#_Toc428988199)

[2.1.3 Bilan 23](#_Toc428988200)

[2.2 Module Scheduler 23](#_Toc428988201)

[2.2.1 Objectifs 23](#_Toc428988202)

[2.2.2 Etude détaillée 25](#_Toc428988203)

[2.2.3 Réalisation 27](#_Toc428988204)

[2.2.4 Bilan 31](#_Toc428988205)

[2.3 Module Social et l’Intégration SPAD RealTime 31](#_Toc428988206)

[2.3.1 Objectifs 31](#_Toc428988207)

[2.3.2 Etude de l’existant 32](#_Toc428988208)

[2.3.3 Réalisation 36](#_Toc428988209)

[2.3.4 Bilan 39](#_Toc428988210)

[3 Bilan 40](#_Toc428988211)

[3.1 Résultats obtenus 41](#_Toc428988212)

[3.1.1 Mise en œuvre de l’approche Domain-driven design 41](#_Toc428988213)

[3.1.2 Planificateur de tache de hautes disponibilités 41](#_Toc428988214)

[3.1.3 Intégration de la Textmining dans le module social 41](#_Toc428988215)

[3.2 Difficultés rencontrées 42](#_Toc428988216)

[3.2.1 Compilation des modules découplés 42](#_Toc428988217)

[3.2.2 Tester l’efficacité de l’algorithme de Textmining 44](#_Toc428988218)

[Conclusion 45](#_Toc428988219)

[Ressources 46](#_Toc428988220)

[Glossaire 47](#_Toc428988221)

[Annexes 48](#_Toc428988222)

[A. Planning 49](#_Toc428988223)

[B. Architecture d’un module avec Domain Driven Disign 49](#_Toc428988224)

# Introduction

Dans un contexte où Coheris cherchent en permanence à être plus efficaces et plus réactives 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 basées sur une approche d’architecture technique et des nouvelles technologies de pointes qui répondent 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 dirven disign » 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 des canaux sociaux Facebook et Twitter.

Le présent rapport trace les phases du déroulement du stage. Je commercerai 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**

[1.1 Société Coheris 12](#_Toc428908795)

[1.1.1 Présentation 12](#_Toc428908796)

[1.1.2 Produits 12](#_Toc428908797)

[1.1.3 Organisation 13](#_Toc428908798)

[1.1.4 En chiffres 13](#_Toc428908799)

[1.2 Mission 13](#_Toc428908800)

[1.2.1 Sujet de stage 13](#_Toc428908801)

[1.2.2 Besoin 13](#_Toc428908802)

[1.2.3 Équipe 14](#_Toc428908803)

[1.2.4 Environnement de travail 15](#_Toc428908804)

## Société Coheris

### Présentation

Editeur de logiciels Français multinational dédié à la maîtrise et l’optimisation de la relation client : CRM (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.

### Produits

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

|  |  |
| --- | --- |
| Type | Produit |
| Relation & Service Client | Coheris CRM Suite  Coheris CRM Care |
| Pilotage des forces de vente | Coheris CRM Sales  Coheris CRM Sales Trade  Coheris CRM Sles Merch |
| Pilotage des campagnes marketing | Coheris CRM Marketing |
| Datamining & Business Analytics | Coheris Analytics Spad  Spad Deployment Server |
| Business Intelligence & Reporting | Coheris Analytics Liberty  Coheris Analytics Liberty Insight |
| Cloud | CRM Cloud |

Tableau :Produit Coheris

### Organisation

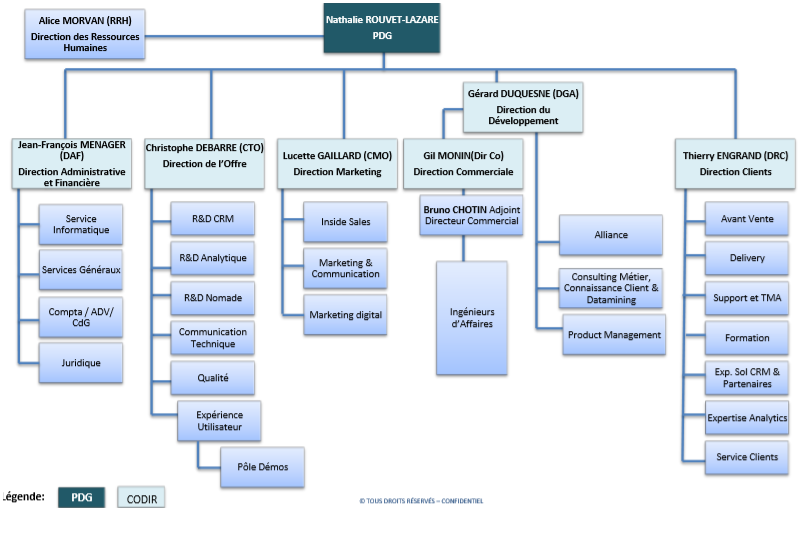


Figure : Organigramme de l’entreprise COHERIS

### 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.

## Mission

### 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.

### Besoin

Pour sa nouvelle offre, Coheris a choisi de se baser sur une nouvelle manière de penser à la conception autour du code, de collaborer et de communiquer avec les experts fonctionnels via l’approche « Domain-Driven Design ». Une étude fonctionnelle et technique a été déjà faite qui définit les limites technique et fonctionnel 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 [] SaaS 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 Textmining. Le passage à la nouvelle structure du module doit être précédé par une étude de l’existant, le connecteur social CRM et une autoformation sur l’approche DDD.

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 ainsi à assister au développement d’autres modules, comme la conception et le développement d’un module scheduler NO prototype, pour planifier des tâches de fond pour la nouvelle offre.

### 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.

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 ramené à collaborer avec l’équipe de Coheris SPAD dans le pôle R&D Analytique.

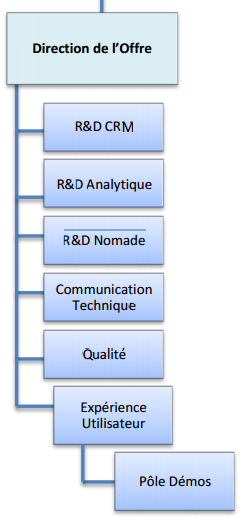


Figure : Organigramme de la direction de l'offre

### 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 (Java2 Enterprise Edition) dans sa dernière version 7. De plus, la plateforme de cette nouvelle ofrre supporte d’autre framework opensource,comme:

* Spring social : pour la gestion des appels webservices vers l’api Graph Facebook.
* HazelCast : gestionnaire de cache répartie 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 permis d'aboutir 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), JPA 2.0, JSF 2.0, CDI 1.0, @Inject 1.0, Interceptors 1.1, JAX-RS 1.1, etc.\*

Pour l’IHM, le choix était sur les fameworks JSF, PrimeFaces, JQuery et Twitter Bootstrap.

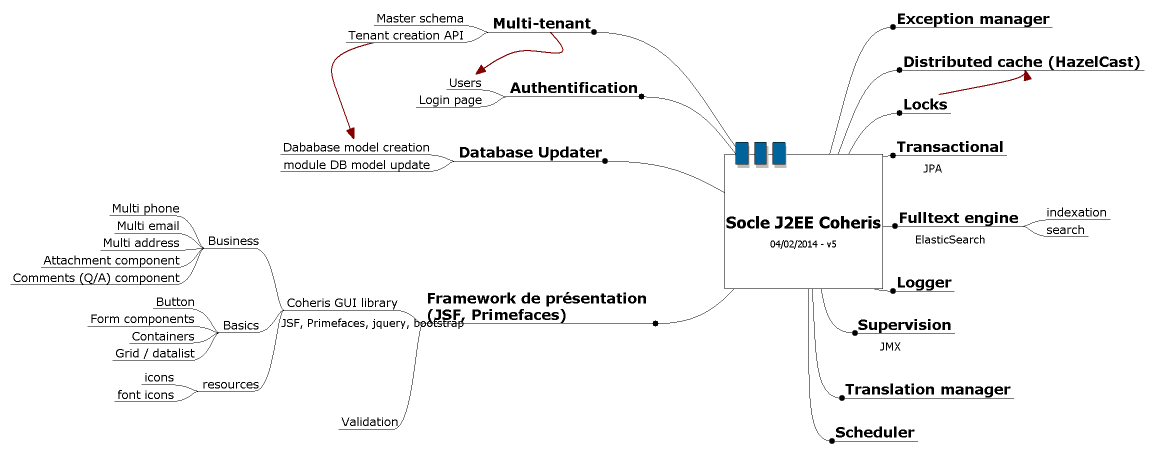


Figure :Socle J2EE Coheris

#### Serveur d’application :

L’implémentation est effectuée afin de permettre l’utilisation de l’application sur un maximum de serveur d’application supportant la plateforme J2EE. En effet, le serveur d’application sur lequel l’application est testée était le servuer 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**[].

Pour l’intégration contenue, 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 chaque nuit, 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 nécessite, 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 visionnés, 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**

[2.1 Etude de mise en œuvre de l’approche Domain Driven design 19](#_Toc428910432)

[2.1.1 Objectifs 19](#_Toc428910433)

[2.1.2 Etude et réalisation : 20](#_Toc428910434)

[2.1.3 Bilan 24](#_Toc428910435)

[2.2 Module Scheduler 25](#_Toc428910436)

[2.2.1 Objectifs 25](#_Toc428910437)

[2.2.2 Etude détaillée 26](#_Toc428910438)

[2.2.3 Réalisation 29](#_Toc428910439)

[2.2.4 Bilan 31](#_Toc428910440)

[2.3 Module Social et l’Intégration SPAD RealTime 31](#_Toc428910441)

[2.3.1 Objectifs 31](#_Toc428910442)

[2.3.2 Etude de l’existant 32](#_Toc428910443)

[2.3.3 Réalisation 36](#_Toc428910444)

[2.3.4 Bilan 38](#_Toc428910445)

## 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 appliquée sur les différents modules développés par la suite.

### Objectifs

Afin d’avoir une démarche unifiée de construction des modules et de prendre en compte la complexité croissante dans la future, il faudra adopter une approche de conception de logiciel plus agile, modulaire et simple à évoluer. L’équipe R&D CRM, ont mis d’accord d’adopter l’approche « Domain Driven Disign » à travers le 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 exemplaire, 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. En effet, les modes adoptés pour la mise en œuvre du module étaient :

* 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.

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.

### 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, page 80). Il faut cependant savoir que des refontes ont était 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 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.

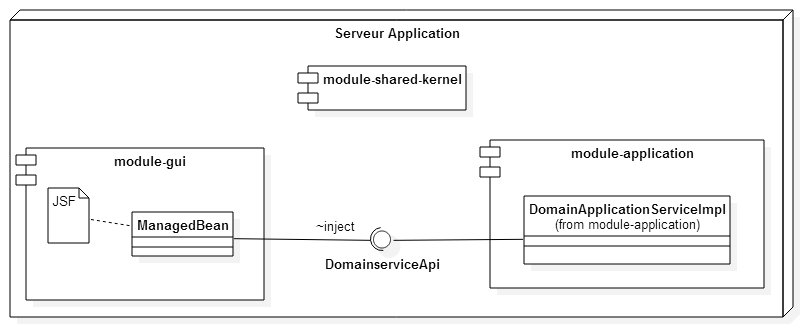


Figure :Communication entre le backend et le frentend dans un mme serveur

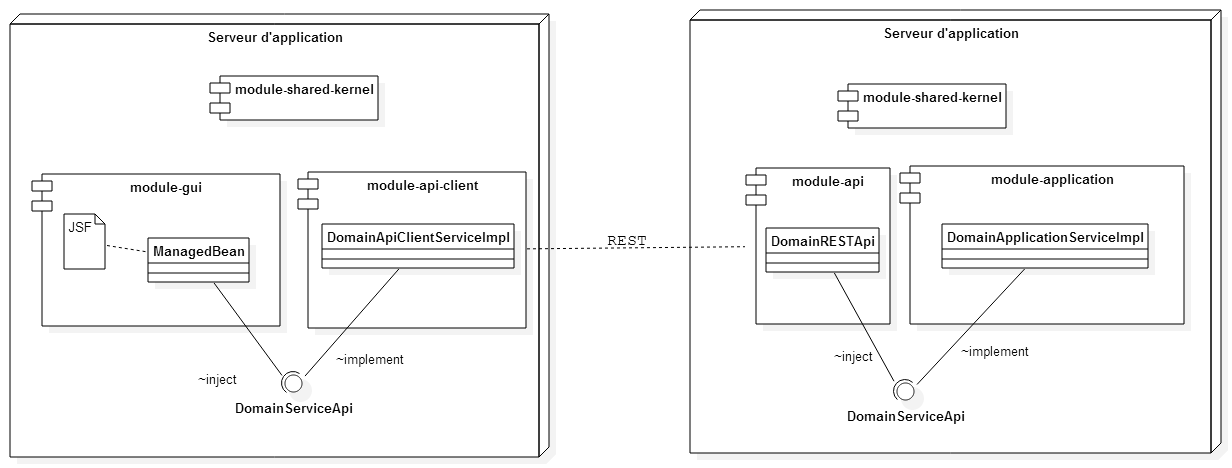


Figure : Communication entre le backend et le frentend déployé sur deux serveurs distincts

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.

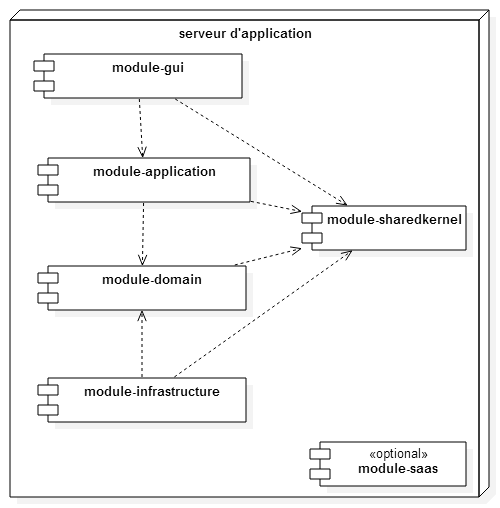
* **Mode1**:
* ****

Figure :Mode1

* **Mode 2**

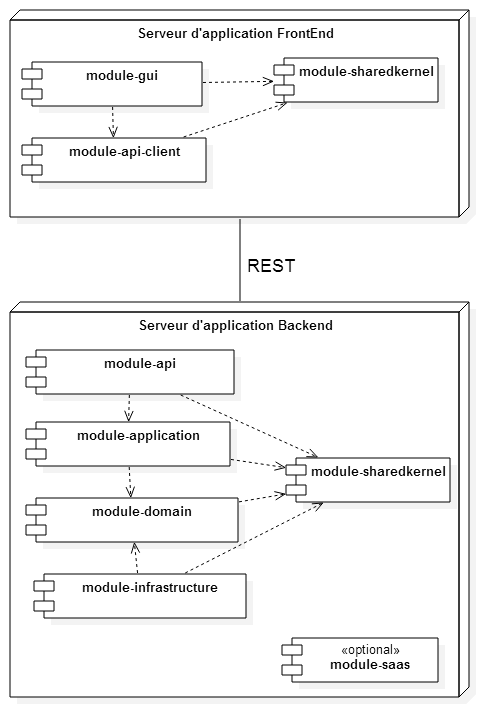


Figure :Mode 2

* **Mode 3 :**

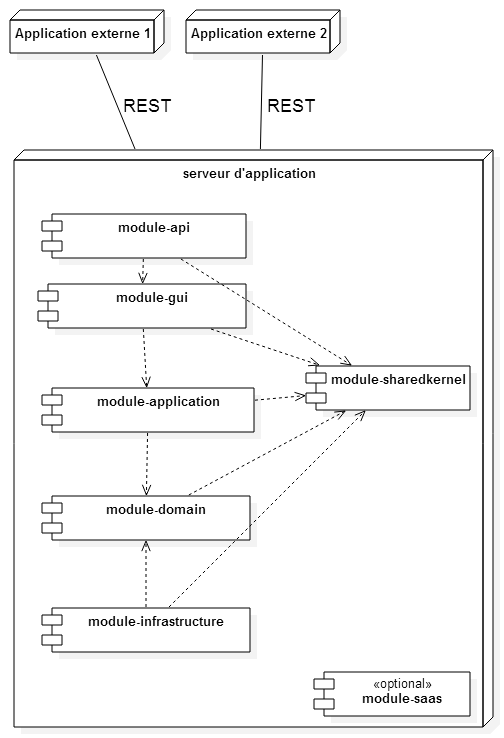


Figure : Mode 3

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

### 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 des bonnes conditions, le développement des modules avec la nouvelle architecture, j’étais formé sur l’outil de build gradle pour pouvoir écrire les tâches de build, en utilisant le langage de scripts JAVA Groovy.

## Module Scheduler

### Objectifs

Afin de planifier des traitements souvent longs  dans les différents modules de la nouvelle offre, on a 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 qualifié après.

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

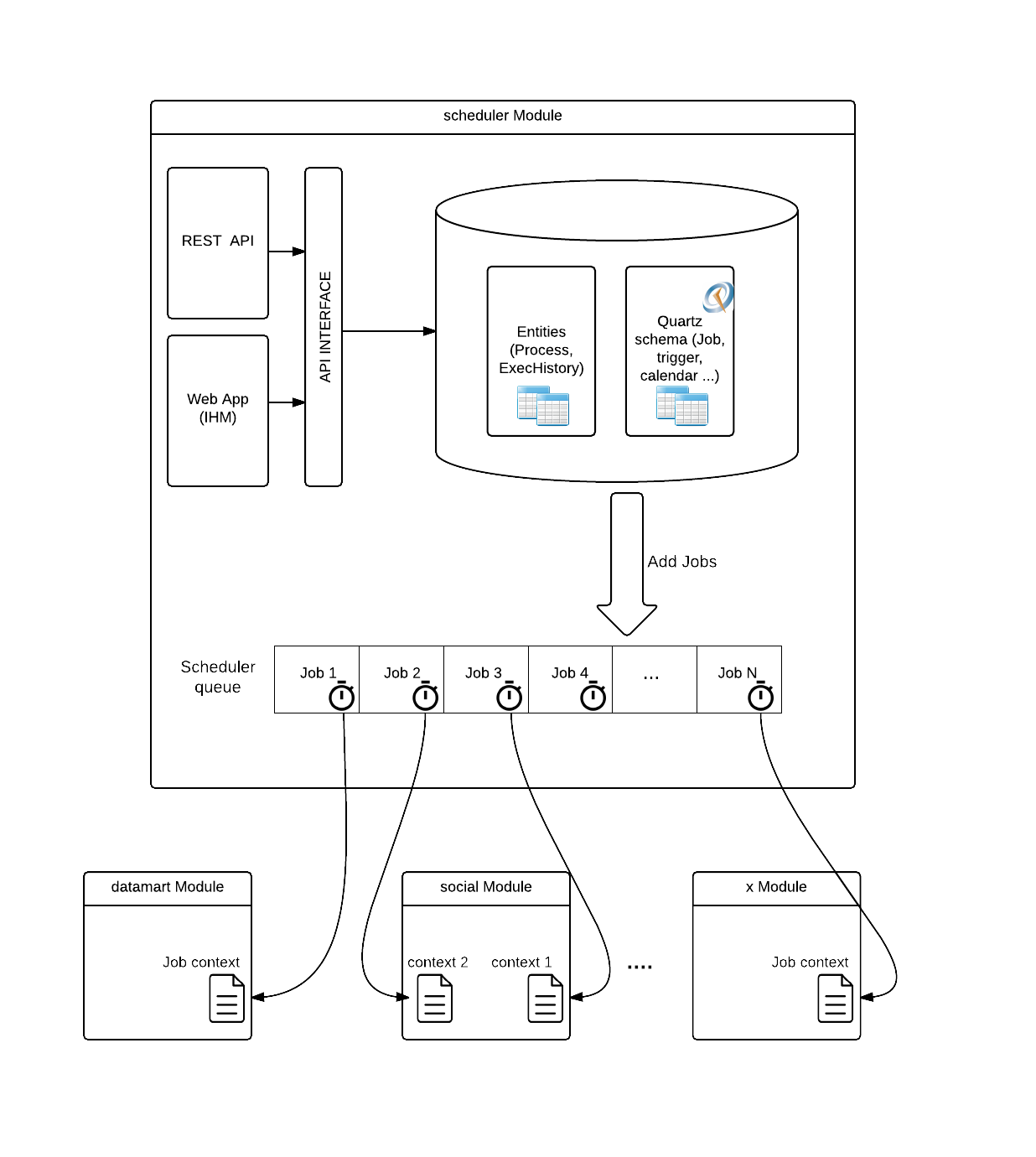


Figure :Architecture du module scheduler

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.

### 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 tache avec l’expression cron []. En revanche, Il existe plusieurs scheduler disponible 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 [].

|  |  |  |
| --- | --- | --- |
| 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 |
| Clusturing | Non | Non |
| schéma de donnée | Non | Inclus et gérer automatiquement |
| configuration | Pas de configuration | Un fichier « properties » |

Tableau : Tableau comparatif des Framework étudiés

Le choix a été  sur Quartz Scheduler vu qu’il gère lui-même sa base de données des taches et il  supporte le mode Clusturing qui est une contrainte impérative pour les modules nouvelles offre de Coheris. A noter,  Quartz ne gère pas l’historique des taches 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 :

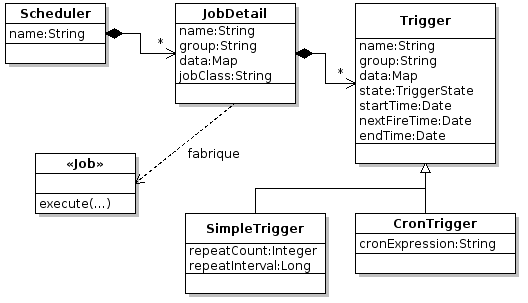


Figure : Concepts Quartz

* **Trigger:** le *quand exécuter*, ça 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 taches : 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 Domain de fonctionnalités dans le module

* **JobInfo :** l'entité qui contient les informations nécessaire pour créer un job
* **TriggerInfo :** l'entité qui collecte les informations nécessaire pour créer un Trigger quartz
* **SchedulerManager:** l'entité qui représente l'entité scheduler du quartz
* **ExecHistory :** l'entité qui contienne les informations catchées du listener du quartz sur les exécutions
* **Process :** l'entité qui représente les infos nécessaire 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 ces composants IHM (couche gui).

Ce qui concerne le branchement du framework 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’externalisée la configuration du Framework d’une façon programmatique à partir des paramètres définie dans le fichier standalone.xml du serveur.

### 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.

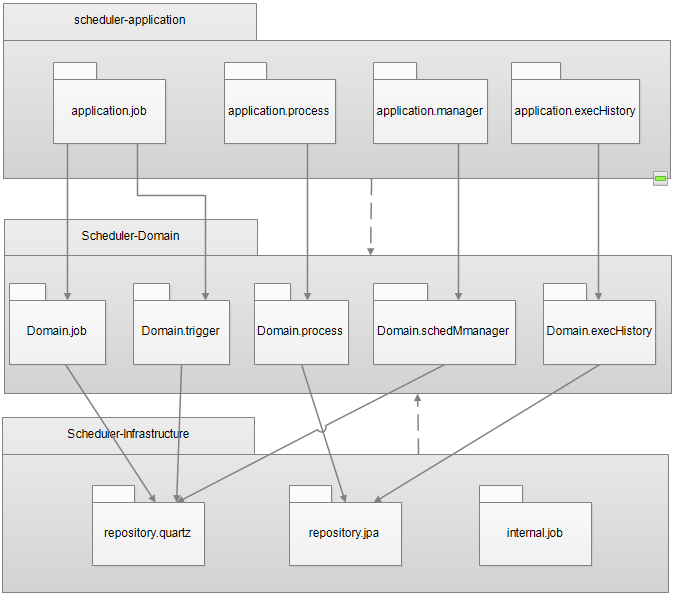


Figure : Architecture du module Scheduler

La configuration de quartz, mentionnée précédemment dans la partie étude, doit prendre ces paramètres de démarrage à partir les 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</userTransactionURL>

<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 désigner des maquettes simple et basique pour visualiser et tester tous les fonctionnalités développer dans le module. Avant de commencer les travaux IHM, les maquettes ont été validées par le Product Manager. Il m’a demandé de se baser sur les Framework :

* 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 taches en cours d’exécution, et d’autre part une vue global sur l’état du scheduler par des statistiques représentés dans un diagramme.

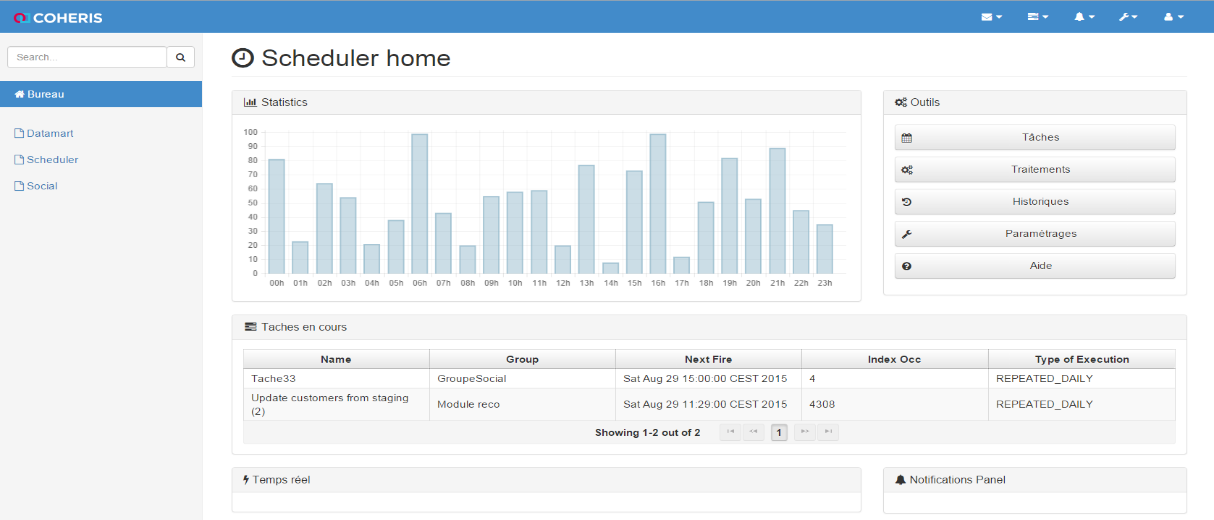


Figure :capture d'ecran du dashbord coheris

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

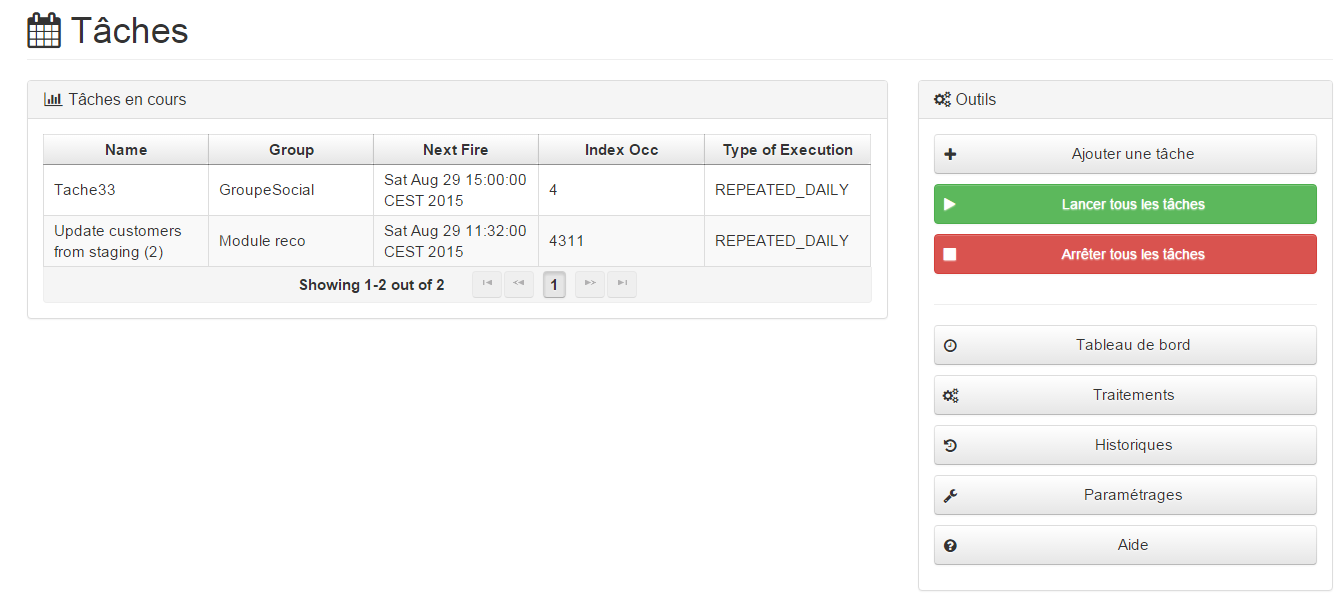


Figure :capture écran du gestionnaire de taches

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

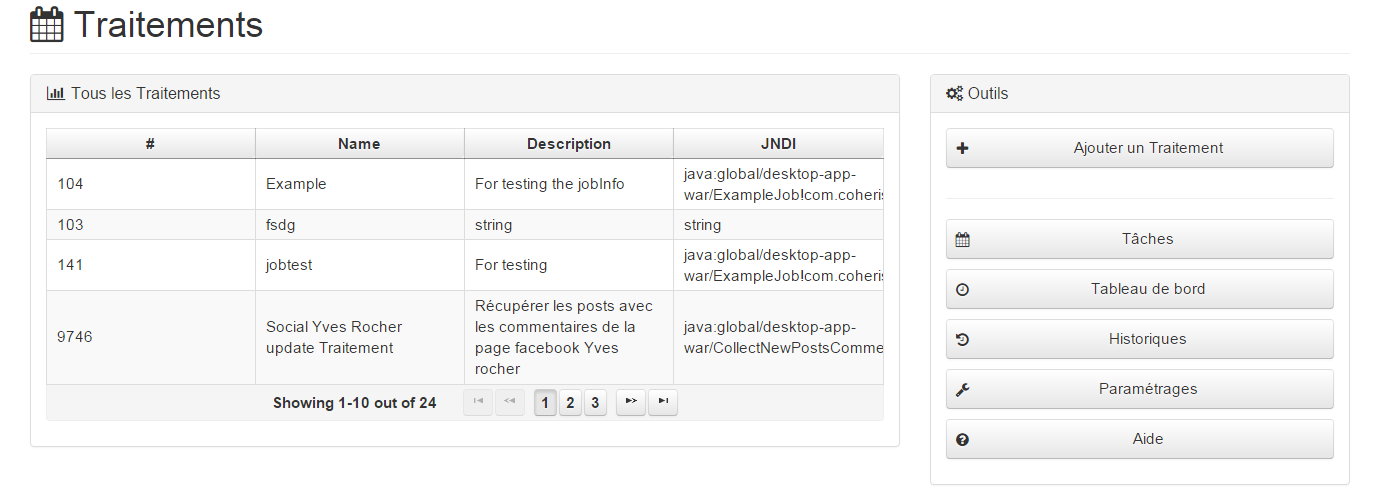


Figure :Capture d'ecran du gestionnaire des traitements

Gestionnaire d’Historiques : où se trouve l’historique de tous les taches exécutées.

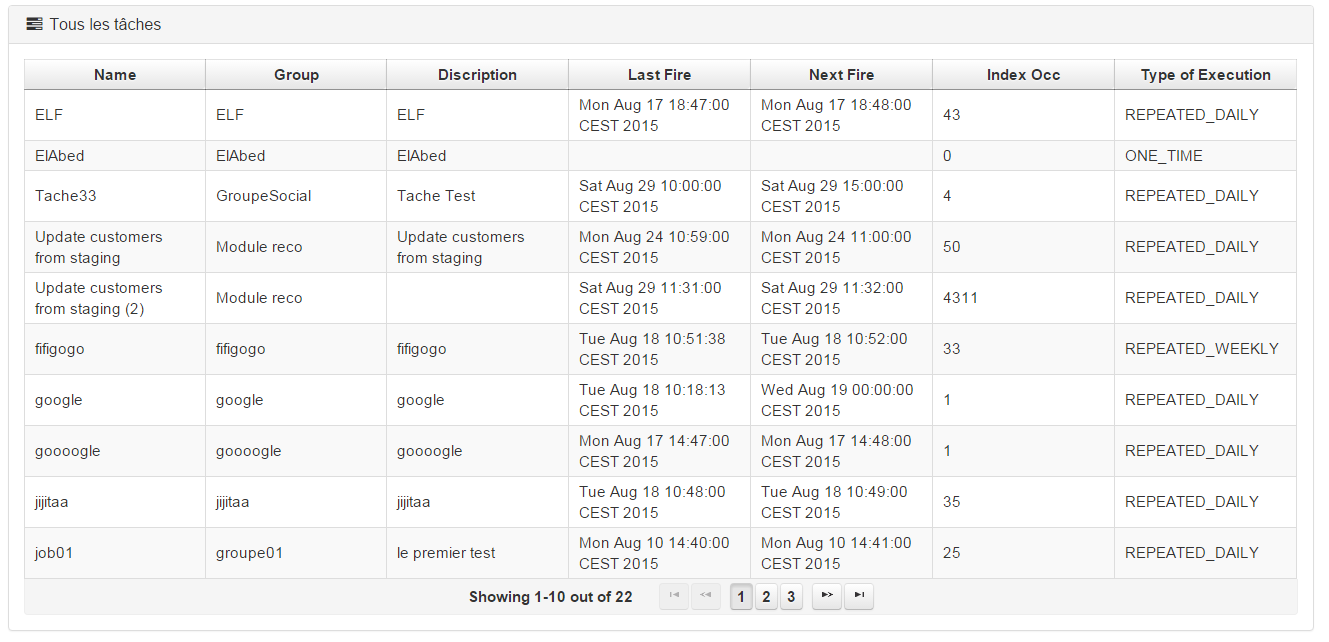


Figure :Capture d'écran sur le gestionnaire d'historiques

### Bilan

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

## Module Social et l’Intégration SPAD RealTime

### 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 qui 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 textmining dans le module était une des primordiales évolutions planifiées sur le module social à travers l’appel aux fonctionnalités de la solution SPAD RealTime.

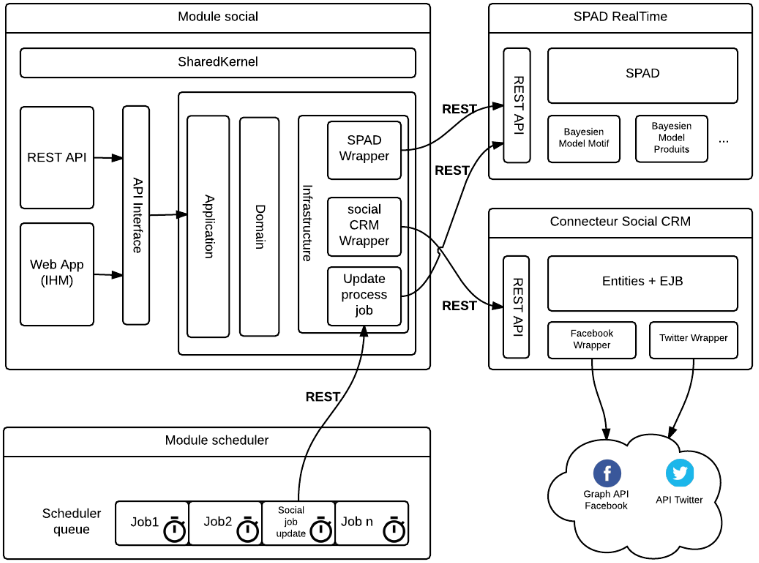


Figure :Architecture du module social

Les différents objectifs de cette partie, était dans un première 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 textmining dans la solution en se basant sur le connecteur social existant.

### 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, 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 ciblag.

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(Textmining) basées sur des algorithmes d’apprentissage bayésiennes.

#### 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.

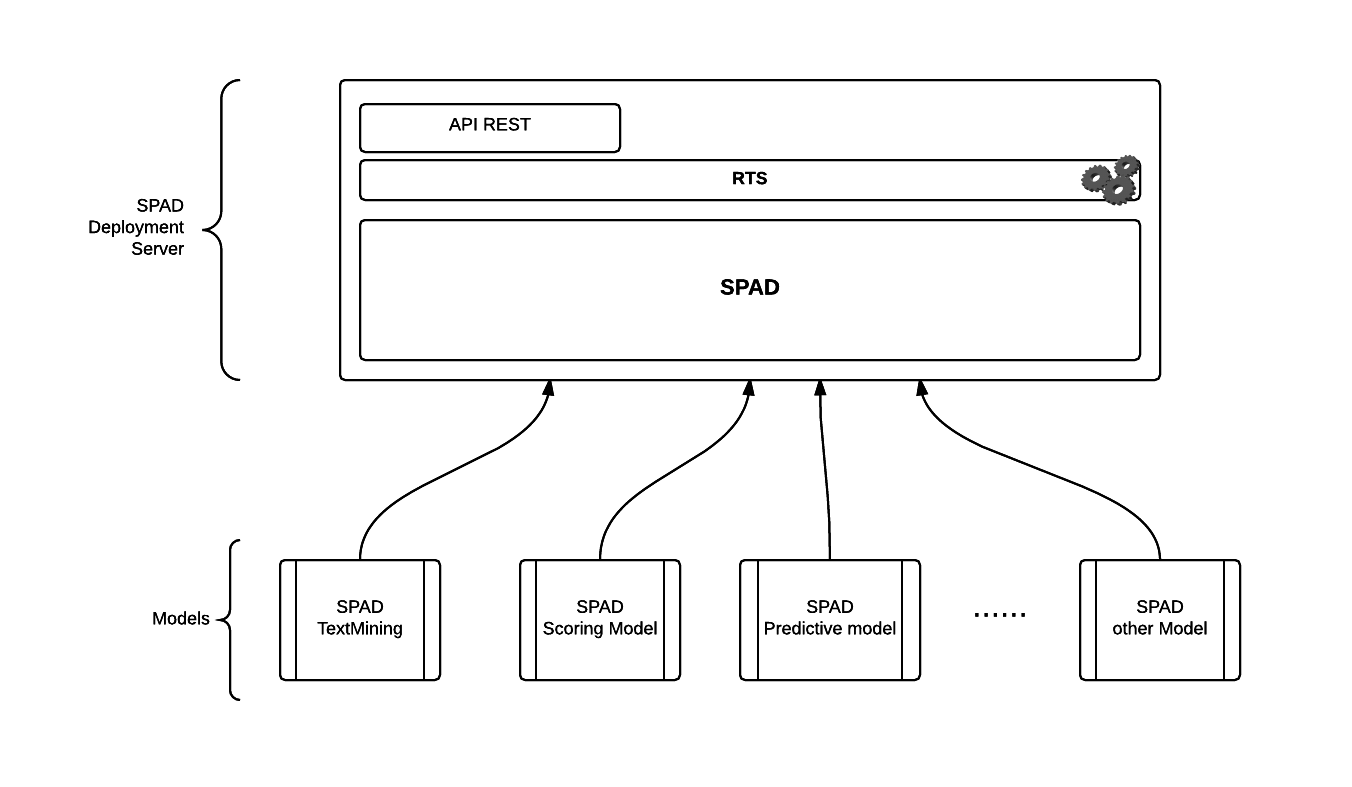


Figure :Architecture de serveur de déploiement SPAD RealTime

#### Model Bayesien pour champs texte:

Le modèle bayésien est utilisé pour affecter un texte libre dans une catégorie parmis 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,opportunity,lead,intervention

minLength=3

lang=fr

targetName=target

keysToParse=text

learnFile=<com.spad.modelsdir>\text\posts-synopsis.txt

##Modéle 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 textmining SPAD realtime

|  |  |  |  |
| --- | --- | --- | --- |
| Propriété | Valeur | Défaut | commentaire |
| modelClass | maestro.web.bayes.TextBayesModel | **requis** | **Obligatoire** : Il indique la classe qui implémente ce modèle |
| modelName | <libre> | **requis** | nom du modèle dans le serveur : c’est un identifiant pour les appels REST et il doit être unique |
| possibleValues | liste de valeur séparée par des "," | **requis** | Le nombre de valeur possible que prédit ce modèle.  Il est aujourd'hui impossible de prendre en compte des valeurs contenant des "," |
| keysToParse | 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". |
| targetName | 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" |
| lang | 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). |
| minLength | taille minimum d'un mot | 3 | Tous les mots de taille inférieure à cette valeur sont supprimés après le parsing. |

Tableau :Table de propriété d'un fichier modèle SPAD REALTIME pour le textmining

#### Base de données de tests

Afin de tester fonctionnellement l’efficacité de l’algorithme bayésien des model textmining, 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.

### Réalisation

L’essentiel du travail a consisté d’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 alogrithmes 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 aux web services externes ont été définies dans la couche infrastructure comme étant une ressource externe du nouvel module social.

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

Tous ces appels ajoutés, prennent le nom de la fanpage comme un paramètre de recherche dans le graph API facebook. De plus, il était primordiale de migrer les sources des entités métier, les porteuses des informations, du connecteur social CRM vers la couche shared-kernel, 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 des nouvelles posts et commentaires d’une page public.

#### 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 « bayesien motif » :

Fichier bayesienMotif.model

modelClass=maestro.web.bayes.TextBayesModel

modelName=bayesien motif

possibleValues=demande,opportunity,lead,intervention

minLength=3

lang=fr

targetName=target

keysToParse=text

##Modéle 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 je l’ai ajouté pour l’entité commentaire et l’entité Post.

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

Image classe interface wrapper liste

#### Prototype IHM

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

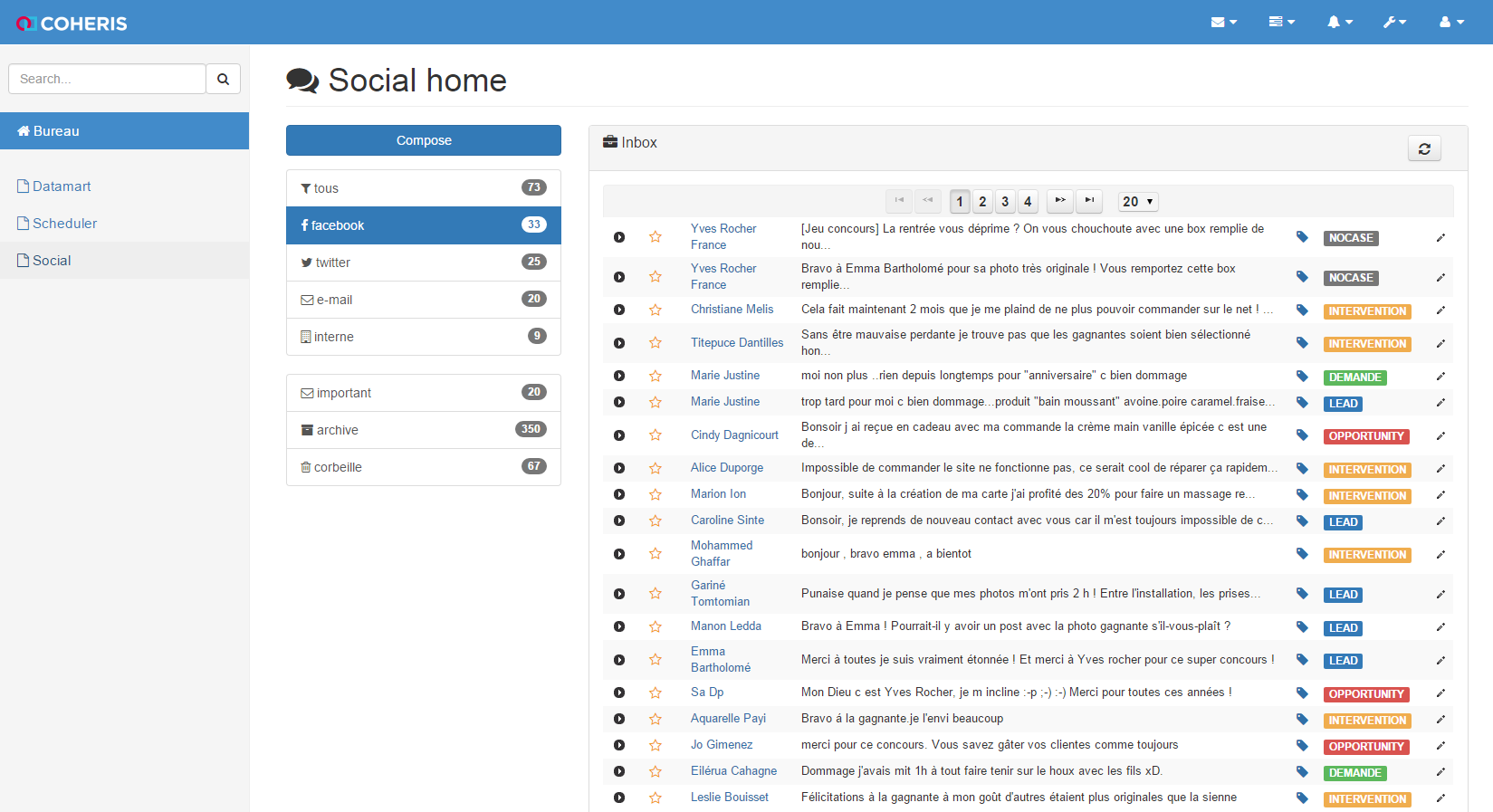


Figure :Capture d'écran

De plus, il m’a été demandé de rendre le champ de la qualification éditable pour requalifier une fonction dans le champ qui affiche

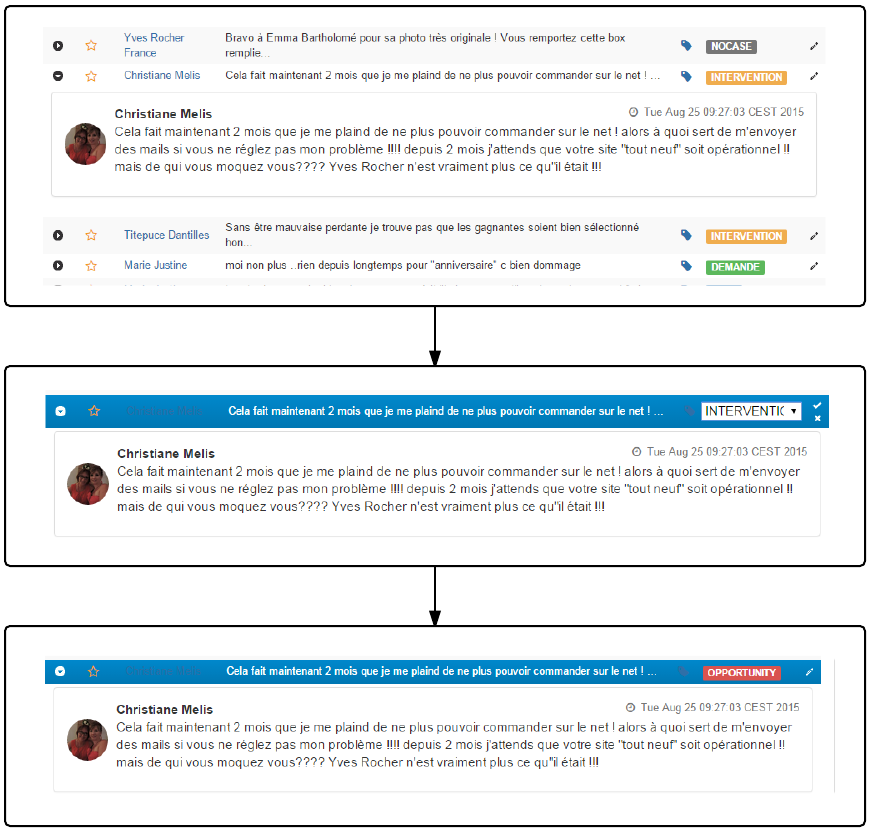


Figure :Capture d’écran

### Bilan

L’intégration de SPAD Real Time dans le module social DDD était un travail extrêmement complet et qui permettra dans les prochaines itérations quel que soit pour l’équipe SPAD ou 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 textmining qui répond aux attentes des clients.

Partie 3

# Bilan

**Sommaire**

[3.1 Résultats obtenus 41](#_Toc428917000)

[3.1.1 Mise en œuvre de l’approche Domain-driven design 41](#_Toc428917001)

[3.1.2 Planificateur de tache de hautes disponibilités 41](#_Toc428917002)

[3.1.3 Intégration de la Textmining dans le module social 41](#_Toc428917003)

[3.2 Difficultés rencontrées 42](#_Toc428917004)

[3.2.1 Compilation des modules découplés 42](#_Toc428917005)

[3.2.2 Tester l’efficacité de l’algorithme de Textmining 43](#_Toc428917006)

## Résultats obtenus

Durant mon stage, j’ai travaillé sur des 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

### Mise en œuvre de l’approche Domain-driven design

Ce premier travail qui 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 faudra le mettre en œuvre. N’oublions pas la possibilité de lancer le module soit en mode on-premise soit en mode multi-tenant.

### Planificateur de tache 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 a permis d’apporter un système de planification de tâche qui peuvent être déployé sur différent mode. 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 CRM Coheris : La gestion des retards, le multithreading, la mise en mode cluster, et la persistance des tâches. Le module scheduler semble donc répondre aux attentes.

### Intégration de la Textmining dans le module social

Le module social DDD, comme on l’appel, 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 de mettre en place la structure de ce module en se basant sur l’approche DDD, de préparer la migration du connecteur socail 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 précédé d’une étude complète de l’existant connecteur social afin d’extraire les fonctionnalités que j’ai besoin pour bien répondre aux besoins.

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

## 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 textmining de SPAD RealTime

### Compilation des modules découplés

#### Structure d’un module DDD des projets dans …….

En utilisant la plateforme JAVA EE de coheris, et après une étude faite sur la nouvelle architecture, l’équipe R&D CRM sont mis d’accord que chaque couche du module (chaque rôle de l’approche DDD) est encapsulée d’un projet java déployer après comme une archive jar.

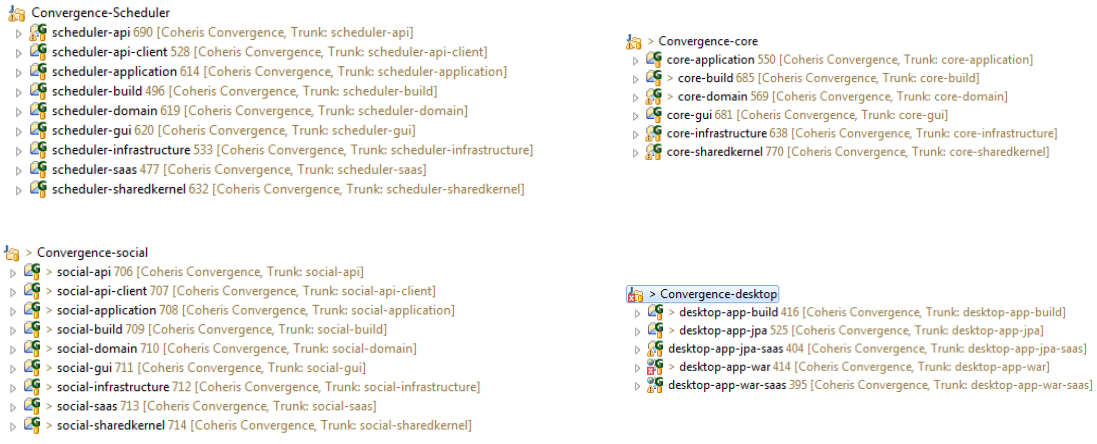


Figure :Capture écran

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[]. 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’entrée 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.

### Tester l’efficacité de l’algorithme de Textmining

Malheureusement, je n’ai pas arrivé à tester l’efficacité des algorithmes de Textemining 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. Ainsi, les tests demandent beaucoup de temps.

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

target text

demande je veux

demande j'aimerai

opportunity Vous ne faite 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 !

….

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 textmining dans ce dernier toute en mise en œuvre des fonctionnalités additionnel pour faciliter les tests dans les prochains travaux de Coheris. De plus, les travaux supplémentaires qui m’ont déjà affecté sont aboutis d’une part à une solution technique pour mettre en œuvre l’architecture d’un module nouvelle offre basée sur l’approche Domain Driven Driven et d’autre part à une système de planification de taches basées sur le framework Quartz.

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

Ce stage, qui ´était mon premier contact avec le milieu professionnel et qui conclut mes ´études supérieures, me conforte dans l’idée de d´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 une équipe dynamique et accueillante qui contient des collaborateurs professionnels.

# Ressources

# Glossaire

# Annexes

**Sommaire**

[A. Planning 49](#_Toc428984340)

[B. Architecture d’un module avec Domain Driven Disign 49](#_Toc428984341)

## 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 | Avril | Mai | Juin | Juillet | Août |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Architecture d’un module avec Domain Driven Disign

#### Présentation du DDD

Le Domain Driven Disign 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

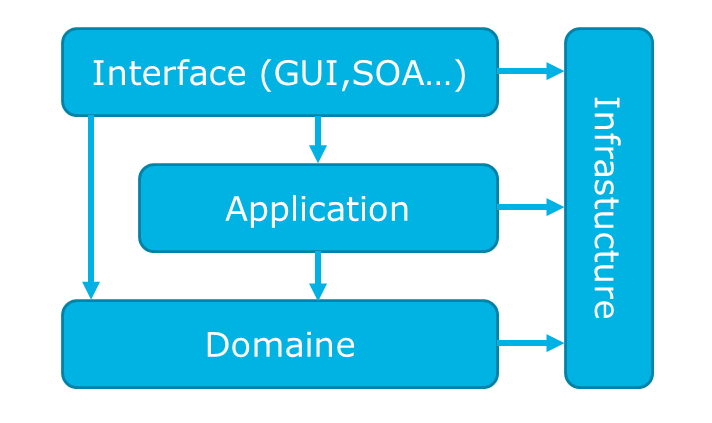


Figure : Les couches DDD

#### 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 En fin, 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 corréler 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**

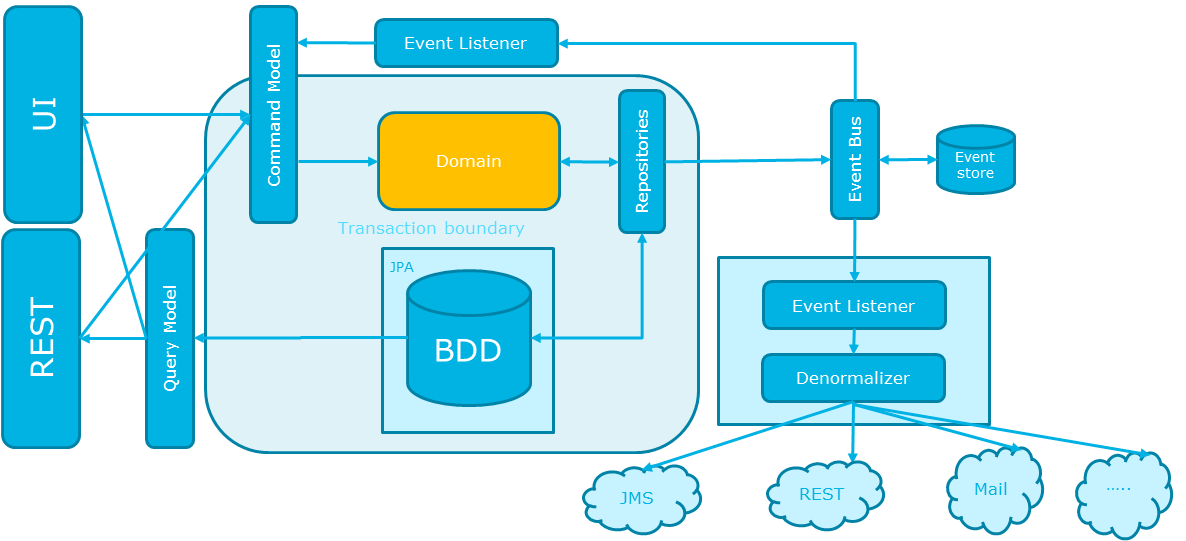
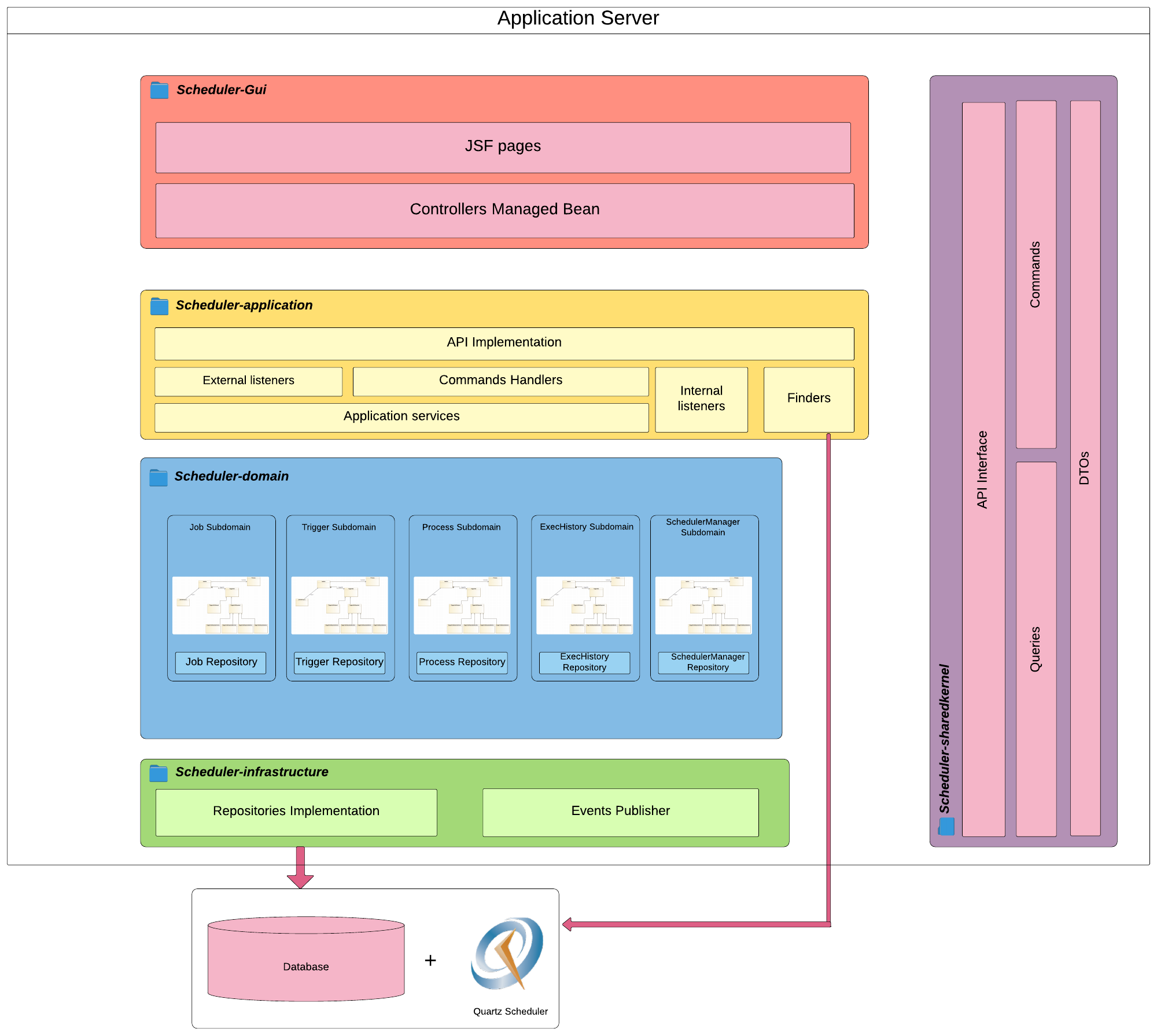


Figure :Architecture DDD avec CQRS

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

**Mode 1 : Backend et frontEnd dans le meme serveur**



#### N:\Stage\rapportPFE\Print screen\Mode 2 -723-9c60f2df-0.pngMode 2 : backend et front end sur 2 serveur différents

#### Mode 3 : Backend avec api REST

