Page de garde 1

Résumé

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 12](#_Toc428819789)

[Figure 2: Organigramme de la direction de l'offre 13](#_Toc428819790)

[Figure 3:Mode1 18](#_Toc428819791)

[Figure 4:Mode 2 19](#_Toc428819792)

[Figure 5: Mode 3 20](#_Toc428819793)

[Figure 6: Concepts Quartz 24](#_Toc428819794)

[Figure 7: Scheduler 26](#_Toc428819795)

Liste des abréviations

Liste des tableaux

[Tableau 1:Produit Coheris 11](#_Toc428819817)

[Tableau 2: Tableau comparatif des Framework étudiés 23](#_Toc428819818)

Sommaire

[Introduction 12](#_Toc428623762)

[1 Présentation du stage 14](#_Toc428623764)

[1.1 Société Coheris 15](#_Toc428623765)

[1.1.1 Présentation 15](#_Toc428623766)

[1.1.2 Produits 15](#_Toc428623767)

[1.1.3 Organisation 15](#_Toc428623768)

[1.1.4 En chiffres 15](#_Toc428623769)

[1.2 Mission 15](#_Toc428623770)

[1.2.1 Sujet de stage 15](#_Toc428623771)

[1.2.2 Besoin 15](#_Toc428623772)

[1.2.3 Équipe 15](#_Toc428623773)

[1.2.4 Environnement de travail 15](#_Toc428623774)

[1.2.5 Planning prévisionnel 15](#_Toc428623775)

[2 Travaux effectués 16](#_Toc428623776)

[2.1 Etude de mise en œuvre de l’approche Domain Driven disign 18](#_Toc428623777)

[2.1.1 Objectifs 18](#_Toc428623778)

[2.1.2 Etude et réalisation : 19](#_Toc428623779)

[2.2 Module Scheduler 20](#_Toc428623780)

[2.2.1 Objectifs 20](#_Toc428623781)

[2.2.2 Etude de l’existant 20](#_Toc428623782)

[2.2.3 Réalisation 20](#_Toc428623783)

[2.2.4 Bilan 20](#_Toc428623784)

[2.3 Module Social 20](#_Toc428623785)

[2.3.1 Objectifs 20](#_Toc428623786)

[2.3.2 Etude de l’existant 20](#_Toc428623787)

[2.3.3 Réalisation 20](#_Toc428623788)

[2.3.4 Bilan 20](#_Toc428623789)

[2.4 Intégration SPAD RealTime dans le Module social 20](#_Toc428623790)

[2.4.1 Objectifs 20](#_Toc428623791)

[2.4.2 Etude de l’existant 20](#_Toc428623792)

[2.4.3 Réalisation 20](#_Toc428623793)

[2.4.4 Bilan 20](#_Toc428623794)

[2.5 Desktop Application 20](#_Toc428623795)

[2.5.1 Objectifs 20](#_Toc428623796)

[2.5.2 Etude de l’existant 20](#_Toc428623797)

[2.5.3 Réalisation 20](#_Toc428623798)

[2.5.4 Bilan 20](#_Toc428623799)

[3 Bilan 21](#_Toc428623800)

[3.1 Résultats obtenus 22](#_Toc428623801)

[3.1.1 Mise en œuvre de l’approche Domain-driven design 22](#_Toc428623802)

[3.1.2 Planificateur de tache de hautes disponibilités 22](#_Toc428623803)

[3.1.3 Refonte de l’IHM du module social existant 22](#_Toc428623804)

[3.1.4 Intégration de la Textmining dans le module social 22](#_Toc428623805)

[3.2 Difficultés rencontrées 22](#_Toc428623806)

[3.2.1 Compilation des modules découplés 22](#_Toc428623807)

[3.2.2 Tester l’efficacité de l’algorithme de Textmining 22](#_Toc428623808)

[3.3 Perspectives 22](#_Toc428623809)

[3.3.1 Intégration de SPAD Real Time dans le module mail 22](#_Toc428623810)

[3.3.2 Portabilité des composants IHM 22](#_Toc428623811)

[Conclusion 23](#_Toc428623812)

[Ressources 24](#_Toc428623813)

[Glossaire 25](#_Toc428623814)

[4 Annexes 26](#_Toc428623815)

[A. Planning 27](#_Toc428623816)

[B. Architecture d’un module avec Domain Driven Disign, Event Sourcing et CQRS 27](#_Toc428623817)

# 

# Introduction

Partie 1

# Présentation du stage

**Sommaire**

[1.1 Société Coheris 12](#_Toc428613324)

[1.1.1 Présentation 12](#_Toc428613325)

[1.1.2 Produits 12](#_Toc428613326)

[1.1.3 Organisation 12](#_Toc428613327)

[1.1.4 En chiffres 12](#_Toc428613328)

[1.2 Mission 12](#_Toc428613329)

[1.2.1 Sujet de stage 12](#_Toc428613330)

[1.2.2 Besoin 12](#_Toc428613331)

[1.2.3 Équipe 12](#_Toc428613332)

[1.2.4 Environnement de travail 12](#_Toc428613333)

[1.2.5 Planning prévisionnel 12](#_Toc428613334)

## Société Coheris

### Présentation

Editeur de logiciels Français multinational dédiés 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.

Elle 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 1:Produit Coheris

### Organisation

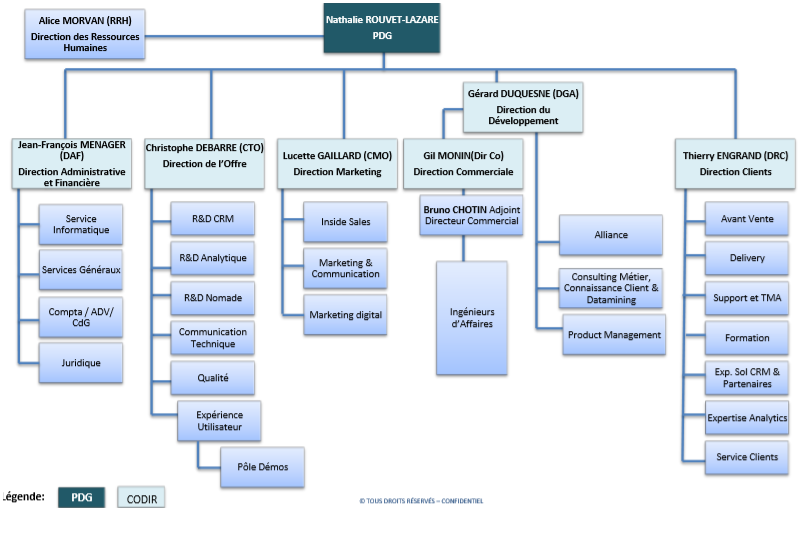


Figure 1: Organigramme de l’entreprise COHERIS

### En chiffres

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

## 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

### 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.

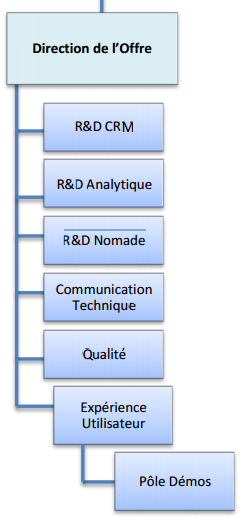


Figure 2: Organigramme de la direction de l'offre

### Environnement de travail

### Planning prévisionnel

Partie 2

# Travaux effectués

**Sommaire**

[2.1 Etude de mise en œuvre de l’approche Domain Driven disign 16](#_Toc428702324)

[2.1.1 Objectifs 16](#_Toc428702325)

[2.1.2 Etude et réalisation : 17](#_Toc428702326)

[2.1.3 Bilan 21](#_Toc428702327)

[2.2 Module Scheduler 22](#_Toc428702328)

[2.2.1 Objectifs 22](#_Toc428702329)

[2.2.2 Etude détaillée 23](#_Toc428702330)

[2.2.3 Réalisation 27](#_Toc428702331)

[2.2.4 Bilan 27](#_Toc428702332)

[2.3 Module Social et l’Intégration SPAD RealTime 27](#_Toc428702333)

[2.3.1 Objectifs 27](#_Toc428702334)

[2.3.2 Etude de l’existant 27](#_Toc428702335)

[2.3.3 Réalisation 27](#_Toc428702336)

[2.3.4 Bilan 27](#_Toc428702337)

## Etude de 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 appliqué sur les différents modules développés par la suite.

### Objectifs

Afin d’avoir une démarche unifiée de construction des modules et 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 mentionnées.

Le premier travail qui m’a été attribué est de faire la conception à partir d’un module exemplaire en appliquant l’approche DDD et en prenant 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 doient être plus flexible en terme de déploiement et de mise en œuvre. En effet, les modes adoptées de 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 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 d’une durée de 2 semaines est multiple : la documentation de la conception de déférents types de déploiement et bien comprendre l’approche Design Driven Developpment, la mise en œuvre du design pattern « CQRS‐Event Sourcing » en respectant les modes de déploiement.

### Etude et réalisation :

L’architecture du module est en multicouche (N‐Layered). Elle assureles 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 encapsuler dans un « .jar » à part.

#### Les differents couches du module :

L’architecture du module et ses difféentes couches  est d´détaillée en annexes (Annexe B, page 80). Il faut cependant savoir que des refontes ont était faite 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 basées sur le Framework MVC standard de  de la spécification JEE7  « JavaServer Faces 2.2 », exploite les services exposées 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 (Fig. 2.4).

Figure

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 gére 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 choisie.  On a rencontré des difficultés pendant cette phase du à la séparation des encapsulations des différentes couche 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 définit dans un script de Build Gradle.

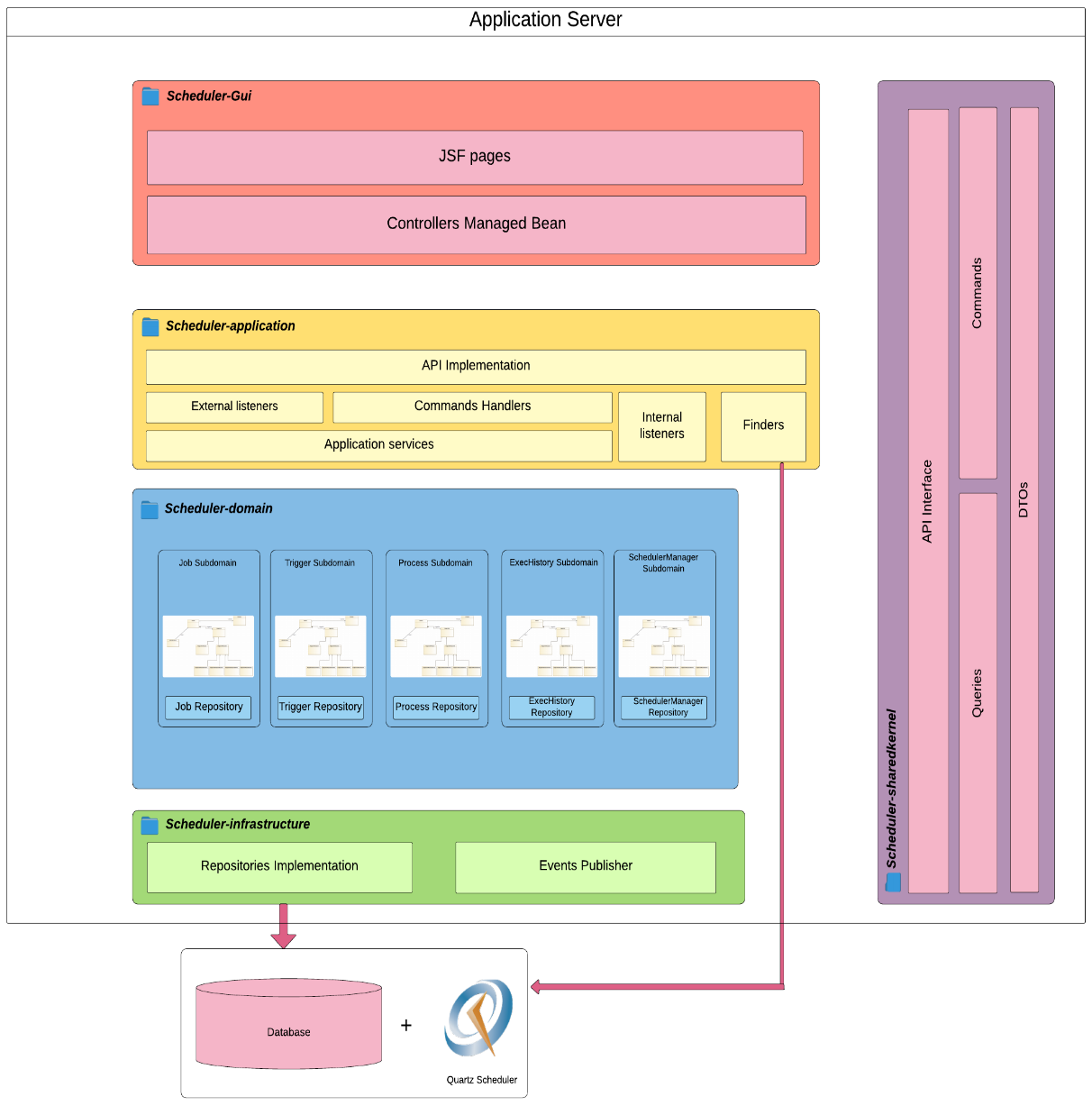
* **Mode1**:

Figure :Mode1

* **Mode 2**

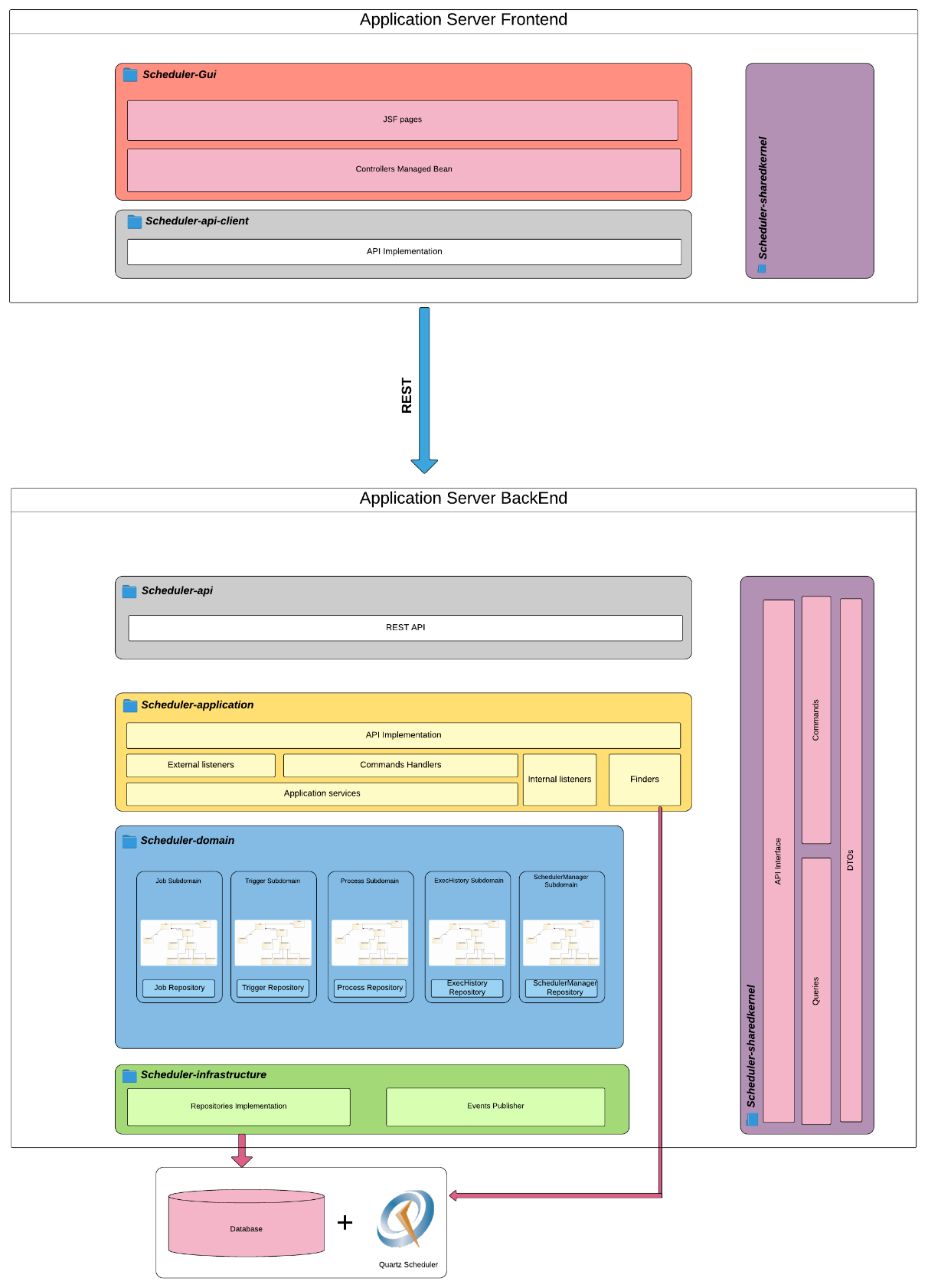


Figure :Mode 2

* **Mode 3 :**

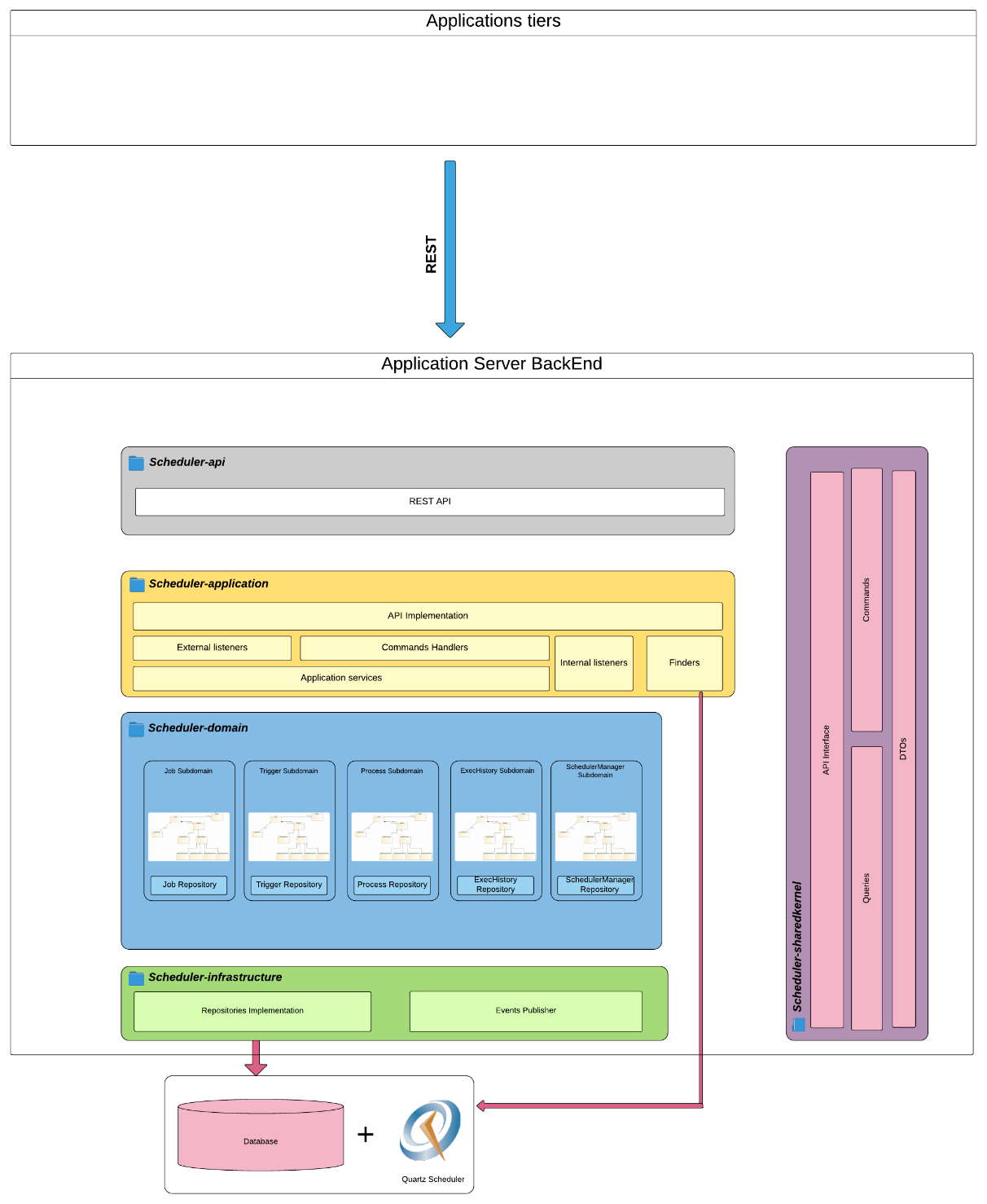


Figure : Mode 3

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

### Bilan

L’objectif de cette première étape a ´été tenu et a constitué une excellente introduction au sujet principal de mon stage. La conception qui a été réalisé va me servir à organiser le développement selon les différentes couches pour chaque module. En effet, afin de réaliser dans de bonnes conditions les développements des modules  avec la nouvelle architecture, il était indispensable que je sois formé sur l’outil de build gradle pour écrire taches de build avec un langage de scripts Groovy.

## Module Scheduler

### Objectifs

Afin de planifiés des traitements souvent longs  dans les différentes modules de la nouvelle offre, on a besoin d’un module Scheduler. C’est un gestionnaire de taches, paramétrable et intégrable selon le besoin. En effet, le module social a besoin de planifié des taches de fond qui collecte les posts et les commentaires Facebook dans les différentes Fanpage pour les qualifié après.

L’objectif de ce travail planifié sur 2 mois 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.

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. L’étude a été réalisée sur une semaine, l’installation et la manipulation de la nouvelle architecture a également duré une semaine, enfin, l’intégration du framework  et  le développement d’un prototype a été accompli en deux étapes durant trois semaines.

### 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 spécifier des annotations pour définir une tache avec l’expression cron []. En revanche, Il existe plusieurs schedulers disponibles la plateforme J2EE qui sont plus évolué et qui sont open source. Quartz est un projet terracota qui qui propose des composants orientés entreprise JAVA. Mon rôle a donc a été de réaliser une veille technologique afin d’acquérir des informations techniques et d’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 qui a été exécuter 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 :

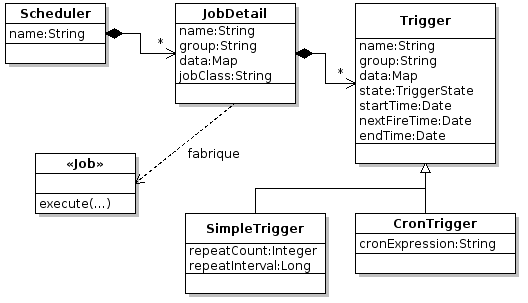


Figure 6: Concepts Quartz

* **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 nous permet une manipulation libre et flexible du framework. Selon les spécifications fonctionnelles, le module doit nous permettre de

* Planifier des taches : quotidienne, hebdomadaire, mensuelle et annuelle.
* Définir un traitement qui pourra être exécuté par plusieurs taches planifiées.
* Consulter l’historique de chaque tache exécuté et son état d’exécution.

Cela ramène à définir 5 domains 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'entitité 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 et 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 composants qui assure le dialecte avec des ressources externe exp : 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 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 faite sur les couches : shared-kernel, application, domain, infrastructure, api et api-client.

Chaque modèle de conception, selon DDD, a son propre package dans chacun de ces couche.

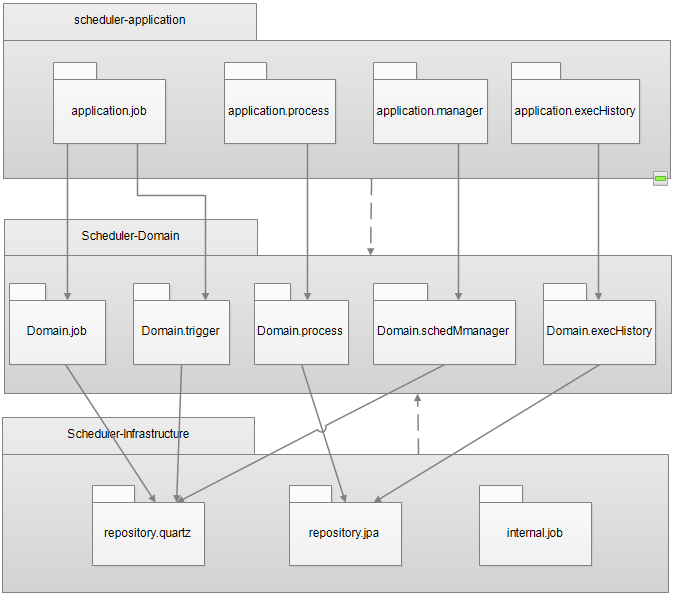


Figure 7: Scheduler

La configuration de quartz, comme été mentionner dans la partie étude, doit prendre ces paramètres de démarrage à partir les configurations standard du serveur d’application. Mais il était été 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 dévlopper dans le module. Avant de commencer les travaux IHM, les maquettes ont été validées par le Product Manager. Il m’a été demandé de se baser d’une part sur les Framework :

* Bootstrap pour que les composant soient résponsives,
* Primefaces qui propose une bibliothèque riche de composants et des exemples qui se base sur JSF facile à intégrer.

Tableau de bord : où on peut trouver d’une part, les taches en cours d’exécution et son état de chaque d’autre part une vue global sur l’état du scheduler par des statistiques faite par un diagramme.

Figure

Gestionnaire de taches : C’est dans cet vue on peut ajouter des taches selon un formulaire dynamique. Et Aussi on trouve plus des détails sur chaque tache.

figure

Gestionnaire des traitements : C’est dans cet vue on peut ajouter des traitements selon un formulaire dynamique.

figure

Gestionnaire d’Historiques : où se trouve l’historique de tous les taches qui ont été exécuté.

figure

### Bilan

Les principaux fonctionnalités, comme l’ajout d’une tâche quoi que ce soit elle se déclenche une seul fois, quotidiennement, hebdomadaire ou mensuelle ont ´été test ´es 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 n´nécessaire pour la partie gui a été relativement faible. Dans cette partie de travail, il m’a été nécessaire de me faire une formation sur le framework bootstrap afin de bien organiser les composants pour qu’ils soient parfaitement visualisable sur des écrans de différentes tailles. En fin, le module scheduler était un bon exercice pour entamer les développements sur le module social qui le but de mon stage 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 de certaines évolutions. L’intégration du textmining dans le module était une des primordiales évolutions qui a été planifié sur le module social à travers l’appel aux fonctionnalités de la solution SPAD RealTime.

La part de mes travaux dans cette partie, n’était pas la migration du connecteur social en module DDD. Elle était dans une première partie l'initiation des développements sur le module par la création de la structure nécessaire basée sur l’architecture DDD. Et Dans une second partie, elle était de trouver une approche d’intégrer les model 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 fonctionnalités 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-y 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 qui ont été réalisées en parallèle dans mon stage, le serveur a été 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.

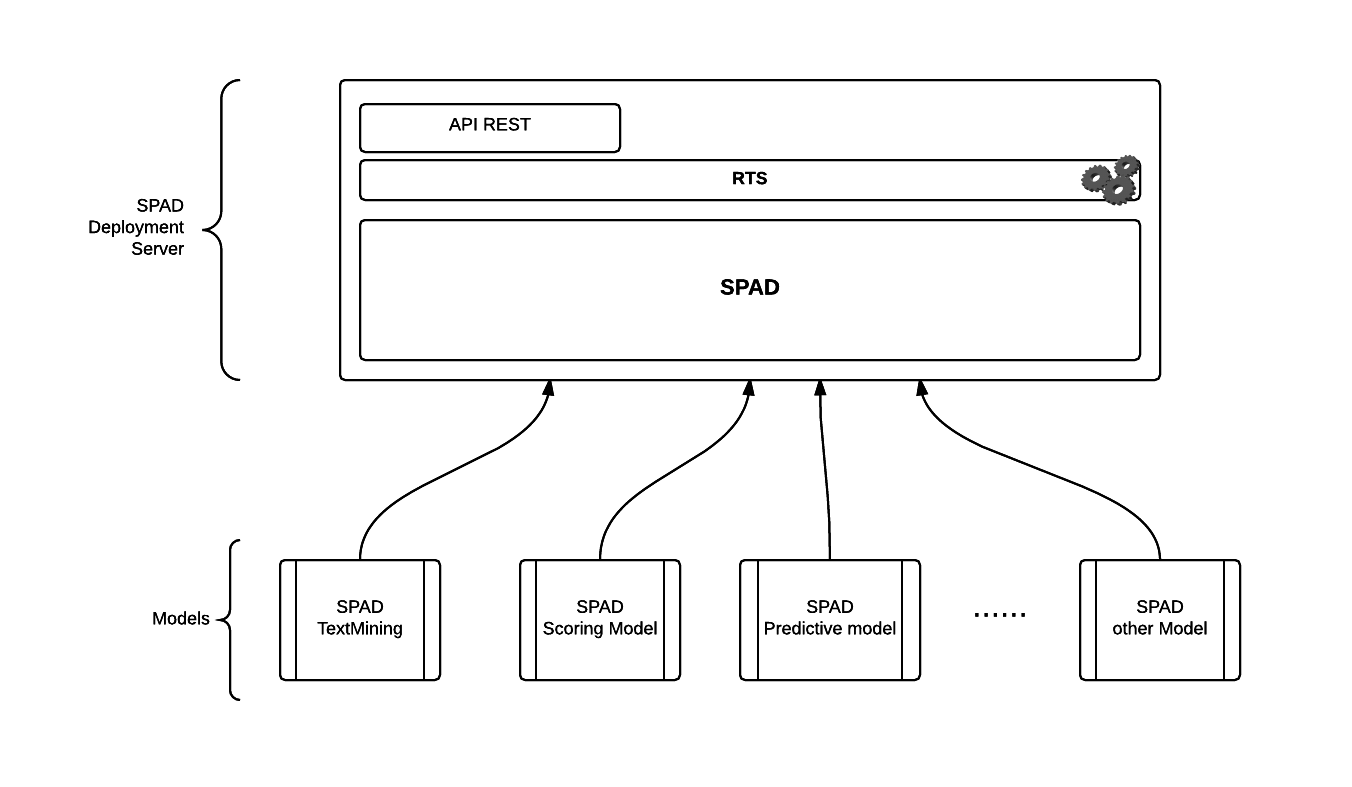


Figure 8:Architecture de serveur de déploiement SPAD RealTime

#### 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

|  |  |  |  |
| --- | --- | --- | --- |
| 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 son administrateur. 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 commencent 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’une fanpage publique, j’avais besoin d’étendre l’api du connecteur social CRM avec des nouvelles fonctionnalités.

Image

Tous ces appels ajoutés, elle prend le nom de la fanpage comme un paramètre de recherche dans le graph API facebook. De plus, il était primordiale de migré 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 tache scheduler qui fait la mise à jour périodiquement 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 cotés 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 qu’il nous a fourni le serveur pour une qualification d’un message. En effet, la liste comme on le constate contient la décision du serveur à propos du contenu de message *intervention* suivi de pourcentage de chaque Target dans le même ordre qui a été écrit dans le fichier du model. Dans un premier temps, et suivant la spécification fonctionnelle du module, on a besoin juste 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 du le module social, j’ai étais amené à réaliser une IHM qui affiche les messages provenant du post ou du commentaire avec leur qualification selon le champ *BusinessMotif*.

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

### Bilan

L’intégration de SPAD Real Time dans le module social DDD a été un travail extrêmement complet et 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 ont ´été 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**

[1. Résultats obtenus 15](#_Toc428305910)

[1.1. Mise en œuvre de l’approche Domain-driven design 15](#_Toc428305911)

[1.2. Planificateur de tache de hautes disponibilités 15](#_Toc428305912)

[1.3. Refonte de l’IHM du module social existant 15](#_Toc428305913)

[1.4. Intégration de la Textmining dans le module social 15](#_Toc428305914)

[2. Difficultés rencontrées 15](#_Toc428305915)

[2.1. Compilation des modules découplés 15](#_Toc428305916)

[2.2. Tester l’efficacité de l’algorithme de Textmining 15](#_Toc428305917)

[3. Perspectives 15](#_Toc428305918)

[3.1. Intégration de SPAD Real Time dans le module mail 15](#_Toc428305919)

[3.2. Portabilité des composants IHM 15](#_Toc428305920)

## Résultats obtenus

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

### Planificateur de tache de hautes disponibilités

### Refonte de l’IHM du module social existant

### Intégration de la Textmining dans le module social

## Difficultés rencontrées

### Compilation des modules découplés

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

## Perspectives

### Intégration de SPAD Real Time dans le module mail

### Portabilité des composants IHM

# Conclusion

# Ressources

# Glossaire

# Annexes

**Sommaire**

[A. Planning 20](#_Toc428306459)

[B. Architecture d’un module avec Domain Driven Disign, Event Sourcing et CQRS 20](#_Toc428306460)

## Planning

## Architecture d’un module avec Domain Driven Disign, Event Sourcing et CQRS