

Dossier de validation

Concepteur Développeur d'Application Numérique

Nom Prénom	Étienne PICHERIT
Nom(s) Prénom(s) du ou des tuteurs	Renaud DESPIERRES
Acronyme de la certification IPI visée	CDAN
Niveau visé	II (Bac +3)
Date de la soutenance	28/11/2019 à 10h
Lieu de la soutenance	EPSI – 114 rue Lucien Faure – 33000 BORDEAUX

1 Engagement de Confidentialité



ENGAGEMENT DE CONFIDENTIALITE

Je soussignée, CORNILLE Marie-José, Directrice régionale **EPSI Bordeaux**
114 rue Lucien FAURE
33300 BORDEAUX

m'engage, ainsi que mon école, à ne pas divulguer les informations apportées par le dossier professionnel et la soutenance de l'apprenant PICHÉRIET Etienne qui fait suite à son alternance réalisée dans l'entreprise

CGI

immeuble Andromède

6 Rue des Comètes – 33187 Le Haillan

Je m'engage à soumettre avant distribution du dossier et présentation de la soutenance la liste des personnes susceptibles d'accéder à ces informations.

Réception et traitement administratif :

DUMARQUÉ Morgane
ANGLES Magali
RIVIEREZ Marie-Gladys

Personnes assistant à la soutenance :

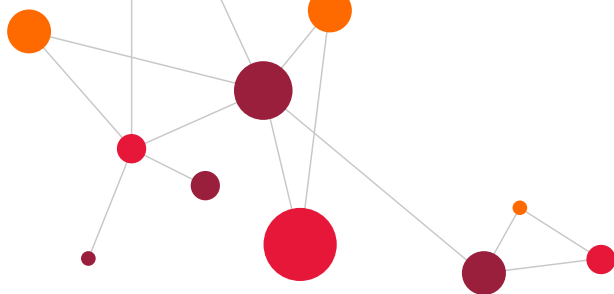
Jury de 2 professionnels

Pas de public autorisé pour cette soutenance.

Fait à BORDEAUX, le 14/05/2019

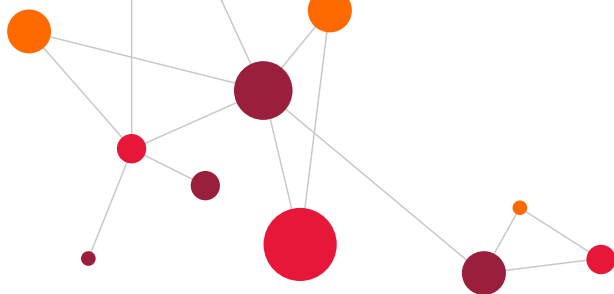
Signature de la Directrice Régionale EPSI

A handwritten signature in black ink, appearing to be 'M. Cornille', is written over the signature line.

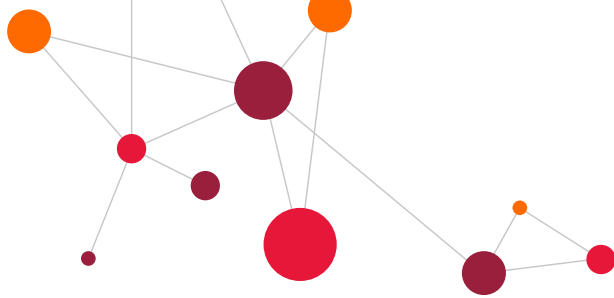


2 Table des matières

1	Engagement de Confidentialité	1
2	Table des matières	2
3	Remerciements.....	4
4	Présentation de l'entreprise, de l'établissement, du service	5
4.1	CGI EN GÉNÉRAL	5
4.2	CGI FRANCE	7
4.3	SBU OUEST ET SUD DE L'EUROPE.....	8
4.4	BU FRANCE GDC	10
4.5	SITE DE BORDEAUX.....	11
4.6	CENTRE DE PRODUCTION	12
4.7	CLIENT BNP PARIBAS.....	12
4.8	TMA.....	12
5	Présentation du poste ou du métier visé.....	15
6	SYNERGY	16
6.1	GÉNÉRALITÉS.....	16
6.1.1	INTRODUCTION.....	16
6.1.2	INFRASTRUCTURE	16
6.1.3	ARCHITECTURE	17
6.1.4	CYCLE DE VIE D'UNE ÉVOLUTION.....	22
6.2	ÉVOLUTION EXPORT HEBDOMADAIRE.....	26
6.2.1	BESOIN	26
6.2.2	ANALYSE	26
6.2.3	DU CHIFFRAGE À LA MISE EN PRODUCTION	29
6.3	ÉVOLUTION CONNECTEUR JSON	29
6.3.1	DEMANDE INITIALE	29
6.3.2	SECONDE DEMANDE.....	31
7	Gestion de matériel réseau	33
7.1	PRÉSENTATION	33
7.2	TRAVAIL EN GROUPE	34
7.2.1	OUTILS	34
7.2.2	MON RÔLE	34
7.3	APPLICATION ANDROID	35
7.3.1	INTRODUCTION.....	35
7.3.2	MAQUETTAGE	36
7.3.3	ARCHITECTURE	37
7.3.4	RÉCUPÉRATION DES DONNÉES JSON.....	38
7.3.5	DÉSÉRIALISATION DES DONNÉES JSON	39



7.3.6	OUVERTURE SUITE AU SCAN D'UN QR CODE	39
7.4	API.....	39
7.4.1	INTRODUCTION.....	39
7.4.2	ARCHITECTURE	40
7.4.3	SWAGGER.....	43
7.4.4	FONCTIONNEMENT DE L'API.....	44
7.4.5	GÉNÉRATION DU QR CODE	44
7.4.6	GESTION DES INCIDENTS	45
7.4.7	EXPORT PDF DES INCIDENTS	45
7.5	WEB	47
7.5.1	INTRODUCTION.....	47
7.5.2	ARCHITECTURE	47
8	Conclusion	50
9	Annexes	50
9.1	BLOC DE COMPÉTENCES : QUALITÉ ET SÉCURISATION DU CODE RÉALISÉ	51
9.2	BLOC DE COMPÉTENCES : AUDIT, CONCEPTION, MÉTHODE DE PROJET ...	54
9.3	BLOC DE COMPÉTENCES : RÉALISATION D'APPLICATIONS LOGICIELLES.	57
9.4	BLOC DE COMPÉTENCES : COMMUNIQUER AVEC LES ACTEURS DU PROJET	59
9.5	BLOC DE COMPÉTENCES : ADAPTER L'ENVIRONNEMENT D'EXÉCUTION, ÉCHANGER DES DONNÉES ENTRE LOGICIELS.....	61
9.6	ATTESTATION DE SUCCES A L'EXAMEN PRATIQUES ECRITES ET ORALES DE LA COMMUNICATION PROFESSIONNELLE	63
9.7	ATTESTATION DE SUCCES A L'EXAMEN BULATS	64
9.8	TOEIC BLANCS REALISES EN FORMATION	65
9.9	ATTESTATION DE SUCCES A L'EXAMEN CONCEPTION ET ADMINISTRATION DE BASES DE DONNEES	67
9.10	CURRICULUM VITAE	68
9.11	FICHE D'EVALUATION STAGIAIRE.....	69



3 Remerciements

En premier lieu, je tiens à remercier tous les membres de mon équipe à CGI pour leur accueil et leur disponibilité qui a grandement facilité mon intégration et ma collaboration à divers projets.

Je remercie particulièrement Renaud Despierres, mon chef de projet et tuteur de l’alternance pour son encadrement et sa capacité à déceler les grains de sables dans les rouages avant qu’ils ne causent de dégâts.

Mes remerciements vont également à Mathieu Vidalies, le responsable de l’application sur laquelle j’ai principalement travaillé, pour m’avoir appris les bases du fonctionnement de SYNERGY et des différents processus à suivre.

Je voudrais également remercier les enseignants de l’EPSI qui m’ont fourni les connaissances nécessaires à la réussite de cette alternance, et qui ont attisé mon goût d’apprendre au point que je souhaite poursuivre mes études.

Je tiens à remercier spécialement Michel Gillet et David Gayerie, que j’ai souvent sollicité après les cours pour éclaircir un point ou demander leur avis, et qui m’ont toujours accordé du temps pour me répondre.

Enfin, mes remerciements vont envers mes camarades Maxime Laurin et Kateryna Bouchet. Ils ont réalisé avec moi le projet Gestion Matériel, qui a été très enrichissant.

4 Présentation de l'entreprise, de l'établissement, du service

4.1 CGI en général

CGI a été fondé en 1976 par Serge Godin, au Canada à Québec. Son sigle signifie Conseiller en Gestion et Informatique. La société a démarré avec un seul client lors de sa première année, avec un chiffre d'affaire de 138 000\$. Elle est aujourd'hui parmi les plus grandes Entreprises de Service du Numérique (ESN), avec 11,5 milliards de \$ chiffre d'affaire en 2018, 77 500 salariés, plus de 5000 clients et une présence sur les 5 continents :

CGI en bref

Fondée en 1976

43 ans de croissance rentable

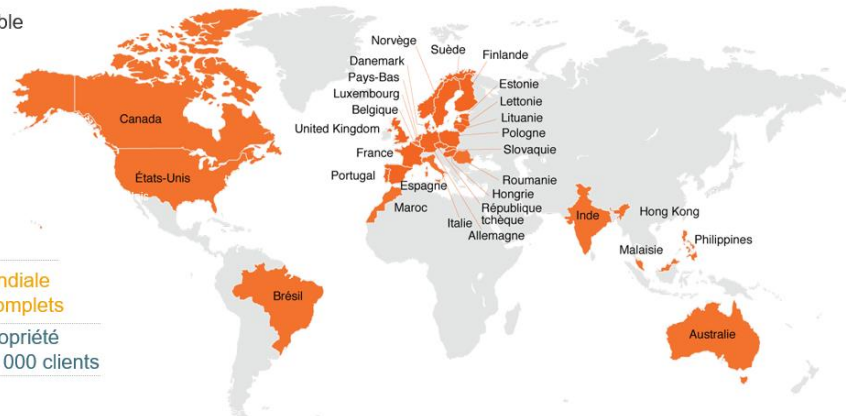
11,5 milliards \$CA de revenus

77 500 consultants

**40 pays
400 emplacements**

**5 000 clients à l'échelle mondiale
font appel à nos services complets**

Plus de 175 solutions de propriété intellectuelle desservant 30 000 clients



© 2019 CGI Inc

3

CGI fournit des services pour de nombreuses grandes entreprises de secteurs variés, ainsi qu'à des gouvernements. Voici un échantillon représentatif des clients de la société :

SECTEUR BANCAIRE	COMMUNICATIONS	GOUVERNEMENTS	SANTÉ ET SCIENCES DE LA VIE	ASSURANCE
     	          	       	    	         

SECTEUR MANUFACTURIER	PÉTROLE ET GAZ	COMMERCE DE DÉTAIL ET SERVICES AUX CONSOMMATEURS	TRANSPORT ET LOGISTIQUE	SERVICES PUBLICS
				

Pour soutenir sa croissance, la société a fait de nombreuses acquisitions au cours de son histoire, à un rythme croissant :

- 1986 : BST
- 1994 : Bell Sygma
- 2001 : IMRglobal
- 2003 : Cognicase
- 2004 : American Management System
- 2010 : Stanley Inc
- 2012 : Logica
- 2016 : Alcyane Consulting, Collaborative Consulting
- 2017 : Summa Technologies, Paragon Solutions, ECS Team, Affecto
- 2018 : Facilité Informatique, ckc AG
- 2019 : Acando AB

La fusion la plus notable est celle entre CGI et Logica, qui a permis de faire grimper les effectifs de CGI de 31 000 à 68 000.

Le fondateur, Serge Godin, est toujours à la tête de la société aujourd'hui. Sa fille, Julie Godin, est actuellement Vice-présidente exécutive du conseil, et devrait lui succéder lorsqu'il se retirera.



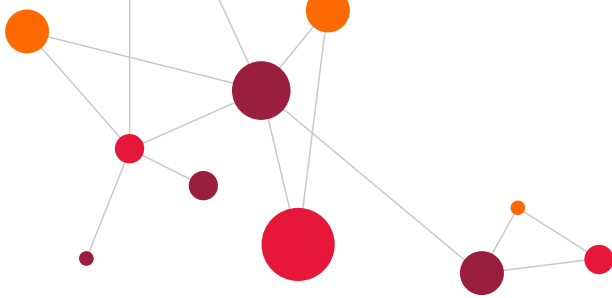
CGI est subdivisé en 9 unités d'affaires stratégiques, abrégées SBU pour Strategic Business Unit :

- Asie-Pacifique GD CoE
- Australie
- Canada
- Centre et est de l'Europe
- CGI Federal
- États-Unis CSG
- Europe du Nord
- Europe de l'Ouest et du Sud
- Royaume-Uni

Nous nous focaliserons sur la SBU Europe de l'Ouest et du Sud.

4.2 CGI France

CGI France est une filiale de CGI. Elle représentait 11 000 collaborateurs en 2018 et prévoyait un recrutement de 3000 personnes en 2019 en France. Elle représente donc aujourd'hui plus



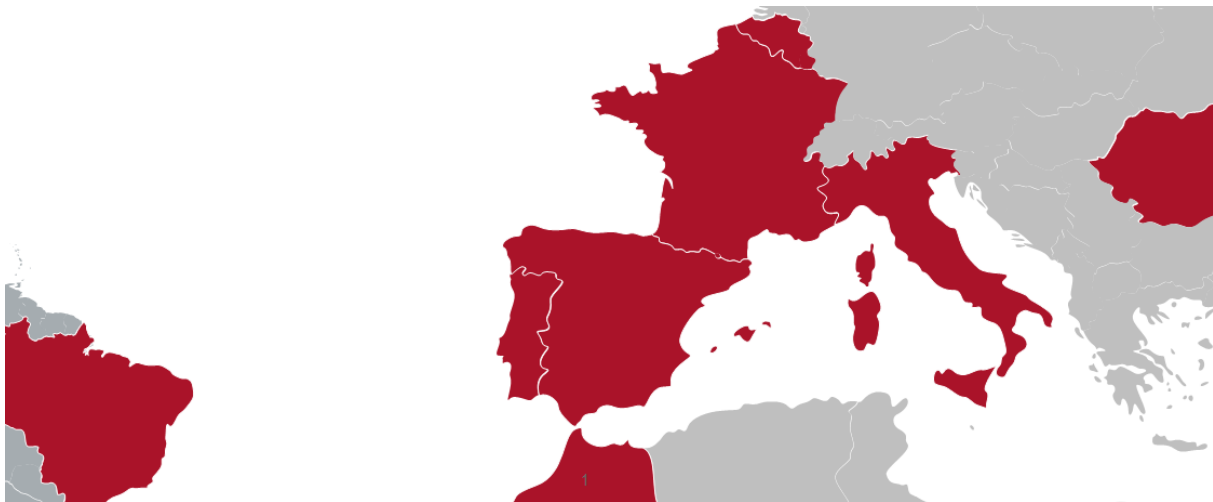
d'1/7 des effectifs de CGI, et plus des trois quarts de la SBU Ouest et sud de l'Europe. CGI France a réalisé un chiffre d'affaire de 1,13 milliards d'euros en 2018.

Toutes les BU de la SBU Ouest et sud de l'Europe sont présentes en France à l'exception de Business Consulting.

CGI France ne fait pas réellement partie du découpage organisationnelle de CGI comme le sont les SBU et les BU. Elle est dirigée par le dirigeant de sa SBU : Jean-Michel Baticle.

4.3 SBU Ouest et sud de l'Europe

La SBU Ouest et sud de l'Europe est présente en Belgique, en Espagne, en France, en Italie, au Luxembourg, au Portugal et en Roumanie. Elle inclue également le Brésil et le Maroc malgré leur éloignement géographique. Elle comporte 15 000 employés.



La SBU est dirigé par Jean-Michel Baticle :



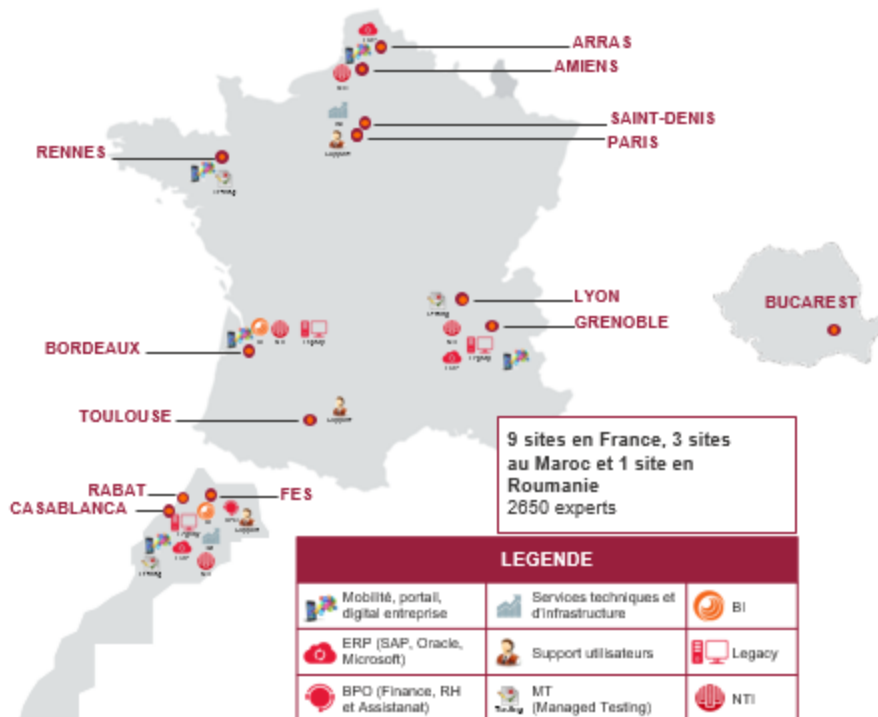
La SBU est elle-même subdivisé en plusieurs BU :

- Business Consulting
- CGP Retail and Manufacturing
- Energy Utilities Télécom Média
- Financial Services
- France GDC
- Grand Est
- Grand Ouest
- Grand Sud
- shapsha
- Nord
- TPSHR
- Europe du Sud et Brésil

J'ai travaillé au sein de la BU FGDC. Voyons la plus en détail.

4.4 BU France GDC

Global Delivery Center est un modèle de prestation de service de CGI. France Global Delivery Center (FGDC) est sa déclinaison française. FGDC comporte environ 2700 salariés et fournit plus de 350 clients. Elle est présente dans 13 sites, principalement en France mais également au Maroc et en Roumanie :



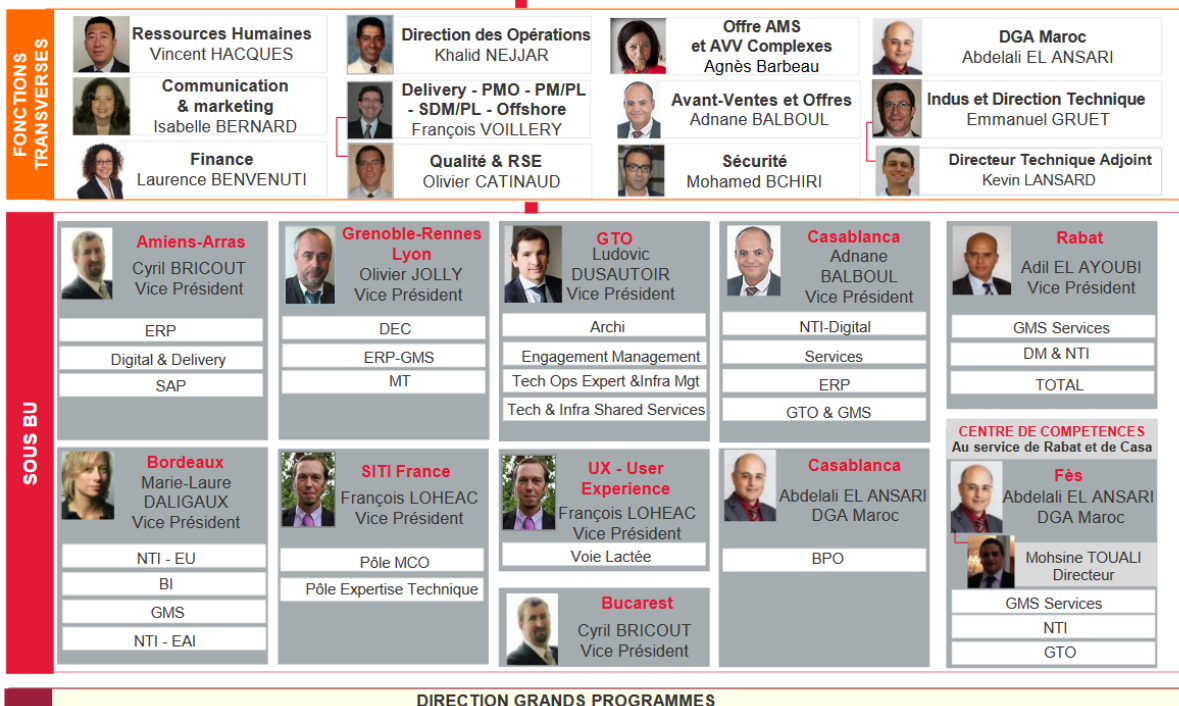
La BU FGDC est dirigé par Philippe Malhomme :



BU Leader
Michel MALHOMME



Assistante de direction
Sophie AGATHE



4.5 Site de Bordeaux



Façade du bâtiment principal du site de Bordeaux

Le site de Bordeaux comporte environ 900 membres. 3 BU y sont présentes : France GDC, Grand Ouest et TPSHR. Il se situe au 6 Rue des Comètes, à Le Haillan.

La BU FGDC du site est dirigé par Marie-Laure DALIGAUX :

Organisation GDC Bordeaux



Sub-BU Leader
Marie-Laure DALIGAUX



Assistante de direction
Marie-Christine BERNET



4.6 Centre de production

À l'échelle du centre de production d'un site d'une BU, il devient plus difficile d'obtenir des informations. Les centres de productions ont récemment changé et je fais partie d'un nouveau centre qui n'est pas présent sur le schéma précédent.

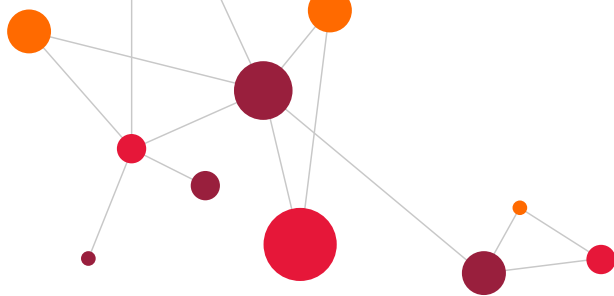
Ce pôle est lui-même subdivisé, et la sous-partie à laquelle j'appartiens est sous la responsabilité de Samir Seddik.

4.7 Client BNP Paribas

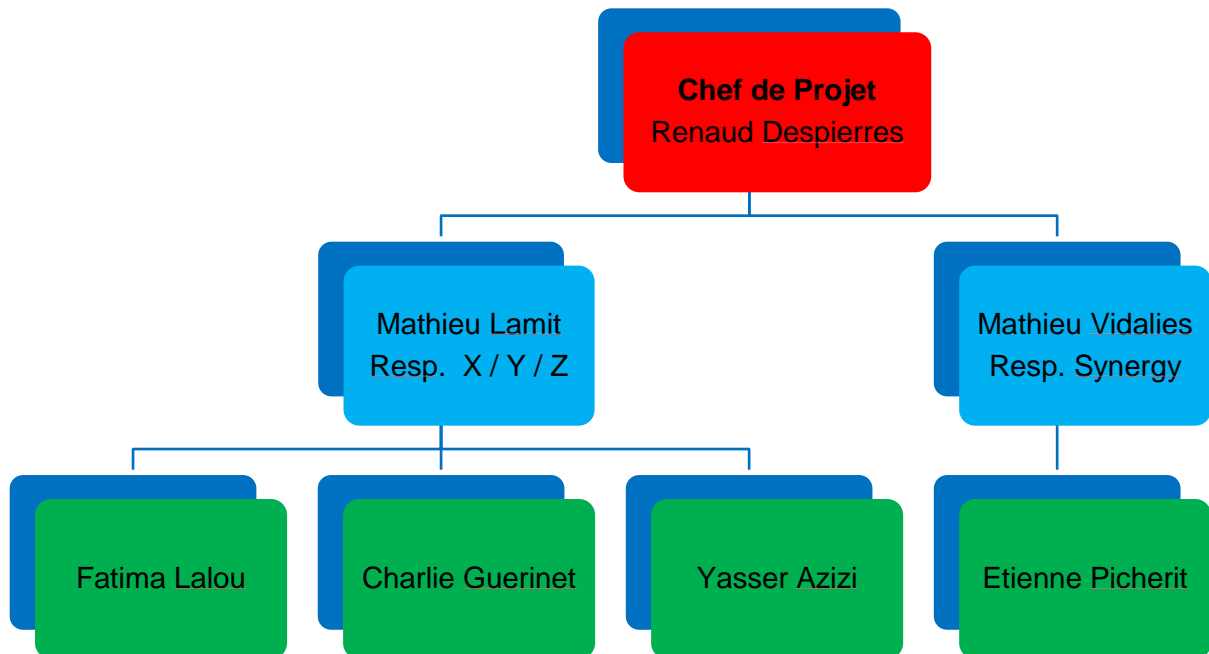
La banque BNP Paribas est la plus grande banque française en termes de chiffre d'affaire et de nombre de salariés. Avec environ 80 membres du site de Bordeaux affectés à ses projets, c'est l'un des plus gros clients de FGDC Bordeaux.

Les environnements de travail de ses projets sont sécurisés, et il est interdit d'en sortir la moindre information. C'est pourquoi il n'y aura pas d'extrait de fichier, de capture d'écran ou de lignes de codes venant de travail effectué pour ce client. De plus, les informations relatives à ce client présentent dans ce dossier ne doivent pas être divulguées, conformément à l'engagement de confidentialité inclus au début de ce dossier.

4.8 TMA



L'équipe de la TMA comporte 6 développeurs, dont je fais partis, ainsi que son chef de projet. Nous nous chargeons de la maintenance de plusieurs projets (majoritairement JEE) pour le client BNP Paribas.

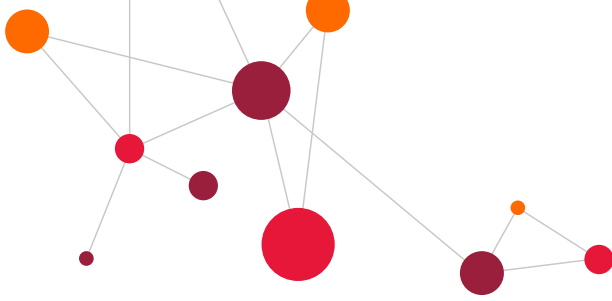


Nous travaillons dans un open-space, sur le même îlot (ensemble de 6 bureaux). Cela nous permet de facilement nous entraider ou de faire circuler une information. Nous utilisons également une boîte mail commune et Skype mais privilégions la prise de contact direct.

Nous avons à notre disposition un grand tableau inspiré de kanban qui nous permet d'indiquer les tâches sur lesquels nous travaillons et leur avancement (to do/doing/done). Nous nous réunissons deux fois par semaine devant ce tableau pour décrire de manière détaillée ce que nous avons fait les jours précédents.

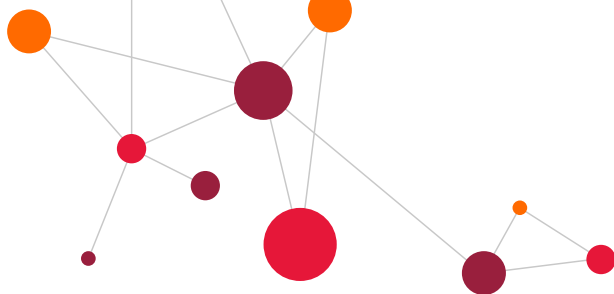
Mon collègue, Mathieu Vidalies, et moi-même passons la majorité de notre temps sur le projet SYNERGY. Nous nous chargeons :

- D'analyser, concevoir et développer les évolutions demandées par le client
- D'organiser et encadrer les livraisons de l'application
- De résoudre les incidents lorsqu'ils surviennent
- De fournir de la documentation à l'équipe IHM afin qu'elle puisse utiliser les services de l'application
- D'informer le client sur l'avancement des différents sujets sur lesquels nous travaillons
- D'assister le client lors de la recette s'il a besoin de compétences techniques.



En plus de ces missions, nous devons nous assurer que l'absence de l'un d'entre nous n'handicape pas le projet. Nous devons donc tous les deux être en mesure de poursuivre les travaux de l'autre, et devons organiser nos absences pour qu'elles ne se chevauchent pas.

Nous sommes très régulièrement en contact avec le responsable de l'application (RA) côté client. C'est ce dernier qui nous indique les tâches à réaliser sur SYNERGY et qui les priorise. Charge à nous et au chef de projet de prioriser ou non SYNERGY en fonction des besoins des autres projets.



5 Présentation du poste ou du métier visé

En 2012, je suivais un cursus scientifique au lycée. En cours de mathématique, un domaine me donna du fil à retordre : les algorithmes. Je ne parvenais pas à saisir la logique qu'il y avait derrière, et ne comprenant pas mes cours, j'ai cherché des alternatives sur internet.

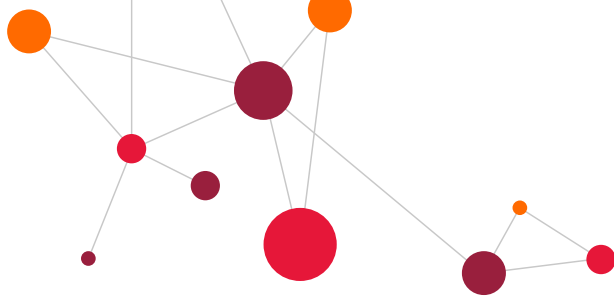
Les algorithmes que nous réalisions étaient sur des calculatrices Texas Instrument, et donc en Ti-Basic. C'est sur ce langage que j'ai trouvé un cours sur le site du zéro (aujourd'hui openclassroom). Là, ce fut le déclic. Plus que ça, c'est devenu une passion. Une activité qui stimule la créativité, que l'on peut également prendre comme une résolution d'énigme et qui en plus peut être utile.

J'ai suivi le cours au-delà du niveau demandé dans mon cursus. Et je ne me suis pas arrêté là, en lisant des cours en C, C++ puis HTML, CSS, PHP... Lorsque j'attendais les résultats du baccalauréat, j'avais réalisé et partagé un calculateur en PHP qui permettait de calculer le résultat obtenu au baccalauréat en fonction de ses notes.

Tout cela pour dire qu'en dehors du métier, je considère le développement comme une activité plaisante et satisfaisante, et je la pratiquais comme un loisir lorsque je n'avais pas l'occasion d'en faire mon travail.

Le métier de développeur revient pour moi à pratiquer cette activité au service d'un client, qu'il soit interne ou externe. Il convient donc de parfaitement saisir son besoin afin d'y répondre de manière adéquate. Il est aussi important d'être honnête, tant sur l'avancement du travail que sur notre avis concernant la demande.

Ce métier est aussi collaboratif, le travail est fait en groupe. L'entente avec ces collègues et sa hiérarchie est primordiale pour exercer dans de bonne condition. Une bonne coordination et communication est aussi nécessaire pour agir efficacement à plusieurs.



6 SYNERGY

6.1 Généralités

6.1.1 Introduction

SYNERGY permet de gérer des données clients, dont je ne peux pas révéler la nature. Dans ce document, nous appellerons ces données des dossiers.

Ces dossiers peuvent avoir différents statuts, et sont liés à de nombreux éléments de SYNERGY, comme des personnes (managers, administrateurs, conseillers...) ou des structures (agences, centre, région...).

6.1.2 Infrastructure

SYNERGY est une application faisant intervenir plusieurs tiers :

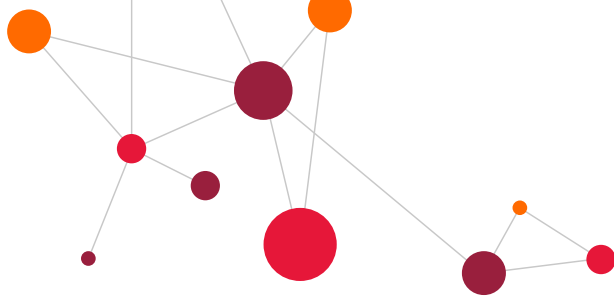
- Le tiers le plus élevé est sur un serveur mainframe. Il est développé en COBOL, et utilise une base DB2.
- Le tiers intermédiaire est sur des serveurs UNIX. Il est développé en JAVA, et utilise une base Oracle. Une partie de l'application est déployée sur le serveur d'application WebSphere.
- Enfin, le dernier tiers est l'IHM (Interface Homme Machine). Il en existe deux. La première est en java et utilise un framework jugé obsolète par le client. D'où l'existence de la seconde, en javascript qui est encore en développement, et qui succèdera à la première.

L'équipe dont je fais partie se charge du tiers intermédiaire et de l'IHM en JAVA. Nous ne développons pas les autres composants, mais nous sommes souvent en contact puisque nos composants interagissent. Nous devons nous assurer que nous appelons correctement le tiers mainframe, et que l'équipe IHM dispose des informations nécessaires pour appeler les webservices du tiers intermédiaire.

L'ensemble de ces tiers est multiplié en trois environnements. Un pour le développement, un pour la recette (qualification) et un pour la production.

Le tout est déployé sur le réseau intranet du client. De plus, des sécurités ont été mises en place à chaque niveau pour n'accepter les requêtes que si elles proviennent d'une origine précise (à la manière d'une whitelist). Par exemple, depuis l'accès au réseau BNPP que nous avons à CGI, il nous est impossible d'accéder à l'IHM ou à la base de données de qualification et de production.

Les filtres de sécurités agissent également de manières différentes selon les ports. Depuis notre accès au réseau BNPP, nous pouvons accéder aux serveurs de qualification et



production via les protocoles SSH ou SFTP, mais une requête HTTP vers leurs webservices sera bloquée.

Le code des applications est hébergé sur des serveurs SVN accessible en intranet sur le réseau du client.

6.1.3 Architecture

6.1.3.1 Introduction

Nous ne verrons que l'architecture du tiers intermédiaire dans ce document.

SYNERGY est découpé en plusieurs sous projets Maven, chacun générant un livrable (jar, ear ou archive tar). La majeure partie de ces sous projets génèrent des librairies (jar). Ces librairies sont ensuite incluses dans les composants de l'application, Batch, Business, ThirdPartyService, IHM ou Config.

Les librairies des composants Batch, Business et ThirdPartyService utilisent le framework Spring. Les connexions à la base de données sont réalisées à l'aide de l'ORM (Object-Relational Mapping) Hibernate.

6.1.3.2 Composants

6.1.3.2.1 Config

Le composant config est un projet Maven mais pas un projet java. Il rassemble divers fichiers de l'application tels que les scripts lanceur des batch ou bien les fichiers de configuration. Lors de la génération des packages, Maven regroupe ces fichiers dans une archive tar contenant de nombreux sous dossier. Nous ne la détaillerons pas dans ce document. Elle est nécessaire au fonctionnement de tous les autres composants à l'exception du composant IHM.

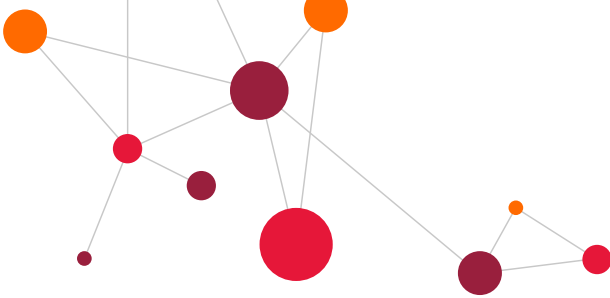
C'est le seul composant de SYNERGY qui diffère selon l'environnement sur lequel il est installé. Il en existe donc autant de versions que d'environnements sur lequel l'application doit être installée.

6.1.3.2.2 Batch

Le composant Batch est très simple. Il s'agit d'une archive tar contenant la librairie Batch, ainsi que toutes les dépendances qui lui sont nécessaires.

Ce composant est appelé par les scripts Shell du composant config afin d'automatiser divers traitements, comme une purge de la base de données, une synchronisation des données avec la partie mainframe via des imports/exports de fichier, la mise à jour du statut des dossiers s'ils ont atteint une échéance ...

Les scripts de la config sont eux même appelés par un ordonnanceur : Autosys. Nous maintenons également cet ordonnanceur. J'ai d'ailleurs effectué une mise à jour qui a



permis de fluidifier l'exécution des batchs, en utilisant des dépendances entre les batch au lieu d'horaires fixes.

6.1.3.2.3 Business

Le composant Business est une archive au format ear. Elle contient une configuration minimale ainsi qu'une archive war, elle-même contenant des librairies.

Le composant Business met à disposition de nombreux webservices, qui sont utilisé par les composants IHM et Batch.

6.1.3.2.4 ThirdPartyService

Ce composant est structuré de manière identique au composant Business. Il est appelé par le composant Business afin de réaliser d'utiliser des services externes comme les mails, les files MQ ou les appels GOAL (voir librairies Service).

6.1.3.2.5 IHM

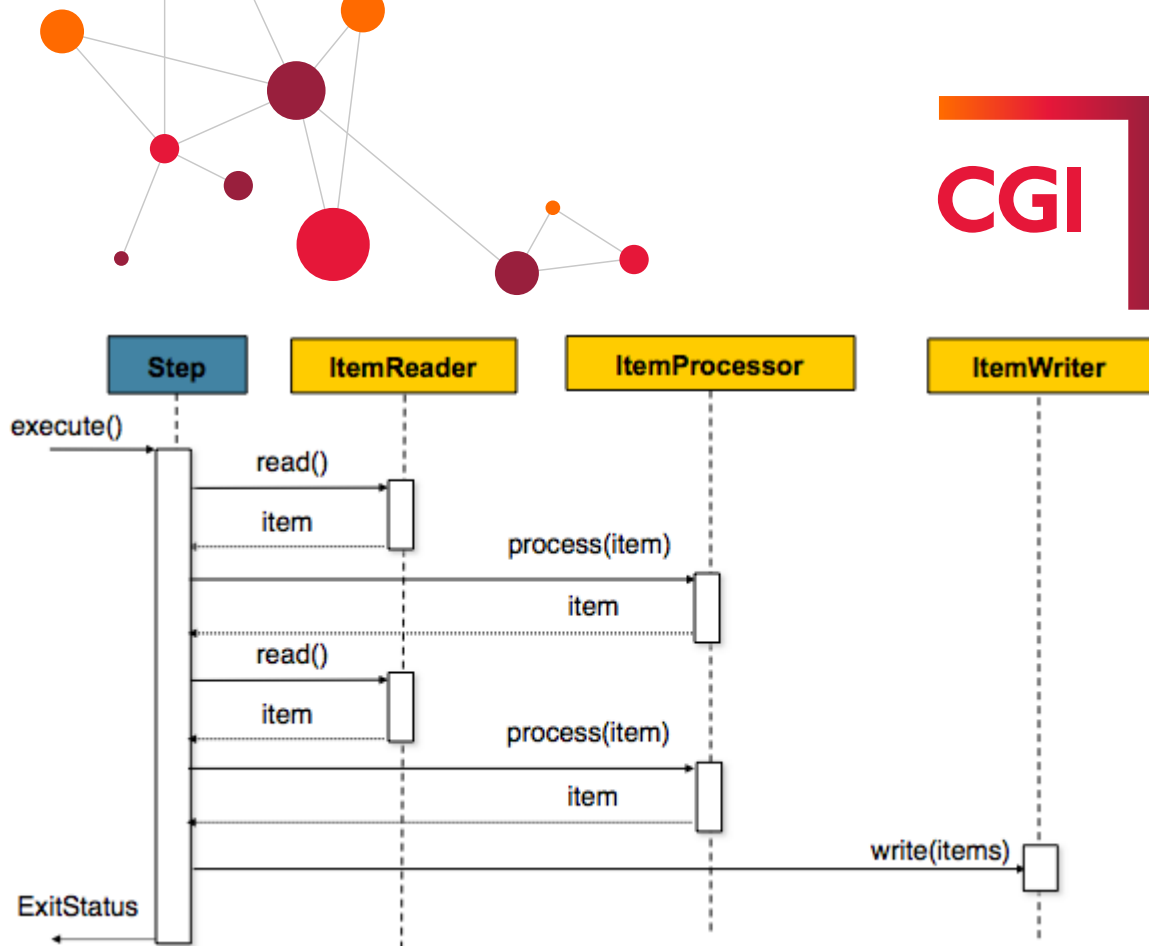
Ce composant est également une archive ear contenant un war. Cependant, il n'utilise pas le framework spring, mais un framework appartenant au client. Comme évoquée précédemment, ce framework est à présent jugé obsolète par le client, et pour cette raison, il sera remplacé par une IHM en javascript.

Comme son nom l'indique, ce composant permet d'afficher une interface graphique à l'utilisateur. Il réalise des appels aux webservices du composant Business pour recevoir les données à afficher ou effectuer les actions demandées par l'utilisateur.

6.1.3.3 Librairies

6.1.3.3.1 Batch

Cette librairie contient le point d'entrée de la partie batch de l'application. Elle utilise Spring Batch pour définir les jobs de l'applications. Chaque job possède des étapes (Step), qui contiennent un Reader, un Processor, et un Writer



Source : <https://docs.spring.io/spring-batch/trunk/reference/html/configureStep.html>

6.1.3.3.2 BatchBusiness

Cette librairie contient la logique métier de la partie batch de l'application. Elle est également responsable de la connexion à la base de données oracle, et de la réalisation des appels vers les webservices du composant Business.

6.1.3.3.3 Business

Cette librairie est responsable de la logique métier du composant Business. À l'instar de BatchBusiness, elle gère la connexion à la base de données oracle, et initie les appels aux webservices du composant ThirdPartyService.

6.1.3.3.4 Pivot

Le pivot contient un ensemble de classe représentant les données d'entrées et de sorties des webservices du composants Business.

6.1.3.3.5 ThirdPartyPivot

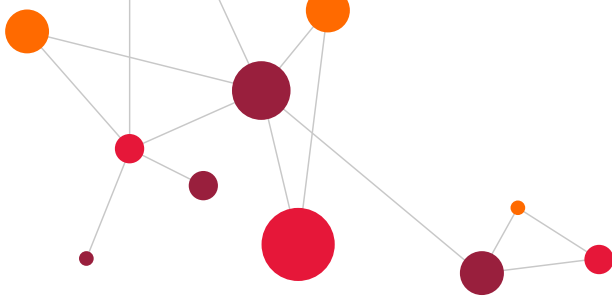
Cette librairie a le même rôle que Pivot, mais pour le composant ThirdPartyService.

6.1.3.3.6 Facade

La librairie Facade contient les points d'entrées du composant Business. Nous reviendrons dessus plus en détail plus tard lorsque nous verrons passerons à l'évolution du connecteur JSON.

6.1.3.3.7 Service

Service permet de faire des appels à des services externes. Cela inclus notamment les appels GOAL, qui permettent d'interroger le mainframe. Une librairie est générée par un outil du

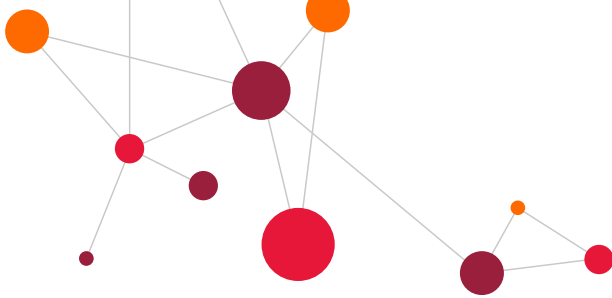


client pour chaque service mainframe de SYNERGY. Chacune de ses librairies est en dépendance du sous projet Service. Ce sont elles qui sont utilisés pour effectuer les appels GOAL.

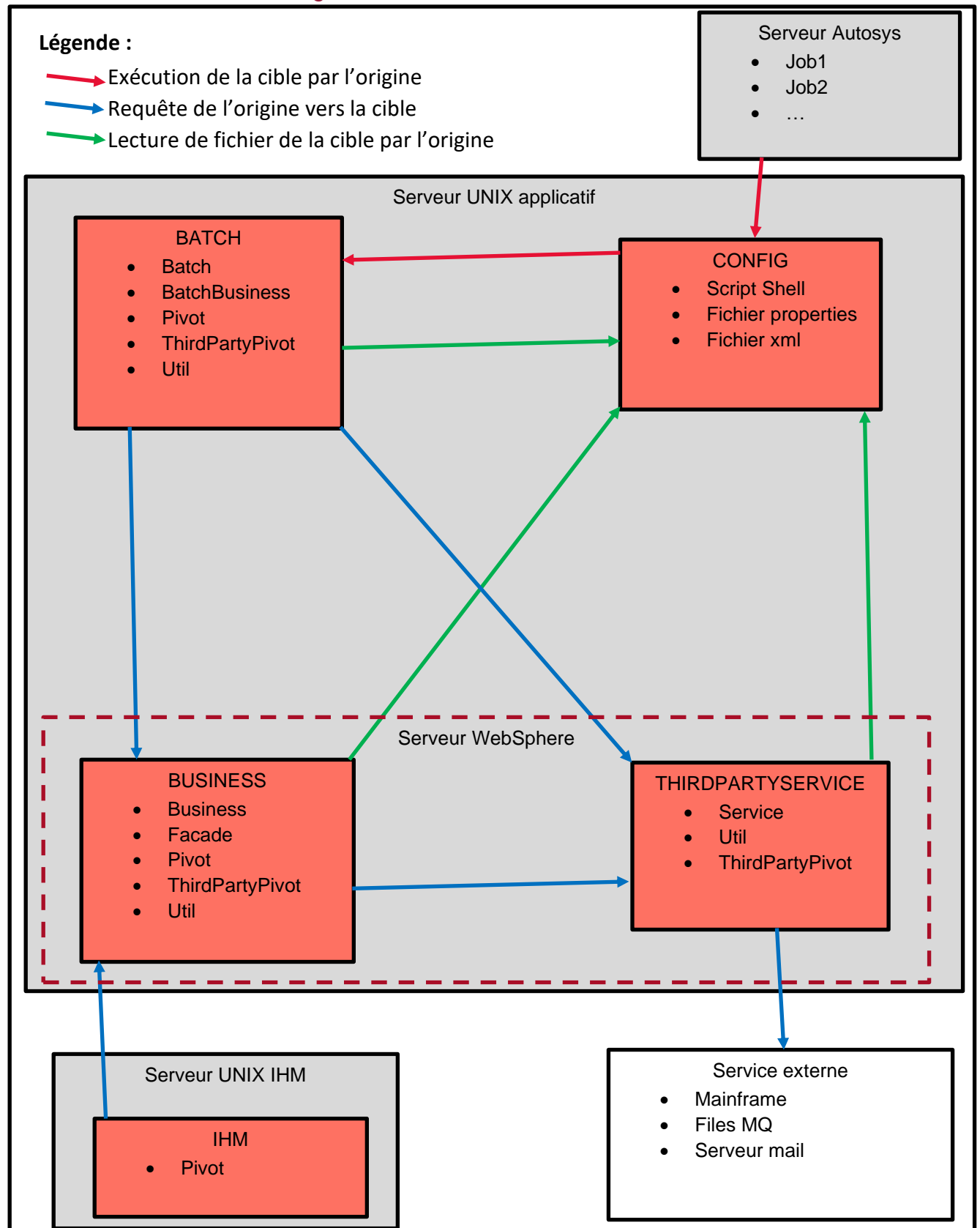
Service gère également les appels vers le serveur mail et les files MQ.

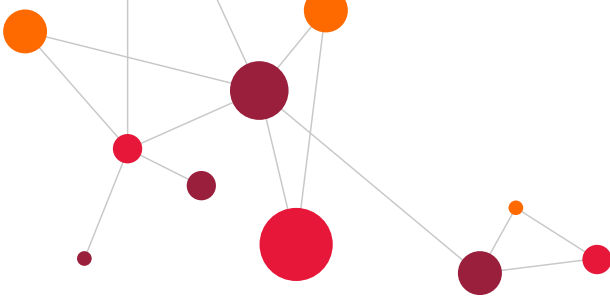
6.1.3.3.8 Util

Util est une librairie contenant quelques outils génériques comme le chargement de propriété depuis un fichier properties, le formatage de date, des comparaisons d'objet pour déterminer s'ils sont égaux, l'écriture de log dans un fichier... Elle est utilisée par tous les composants java à l'exception de l'IHM.



6.1.3.4 Résumé en image





6.1.4 Cycle de vie d'une évolution

6.1.4.1 Introduction

L'une des activités principales sur le projet SYNERGY est la réalisation d'évolution. Il s'agit d'un processus en plusieurs phases que nous allons étudier phase par phase. Nous verrons ensuite deux cas concrets d'évolutions.

6.1.4.2 Présentation

Le client organise une réunion au cours de laquelle il présente l'évolution à réaliser. Cependant, il ne s'agit pas d'une communication à sens unique. Nous nous assurons en effet d'avoir bien cerné le besoin en demandant des clarifications s'il y a le moindre doute. Nous présentons notre vision de l'évolution au client afin qu'il réagisse si elle ne correspond pas à son besoin.

6.1.4.3 Analyse

Le client nous fournit également des spécifications concernant l'évolution. La plupart du temps, nous les recevons avant la réunion afin que l'on puisse prendre connaissance du sujet. En plus de la description du besoin, elles contiennent généralement les dates de mise en qualification et mise en production souhaitée. Ces dates sont souvent impératives, car elles nécessitent presque toujours une synchronisation avec d'autres applications du client.

Ces spécifications nous servent de base pour réaliser notre analyse. Elle a deux objectifs principaux :

- Identifier les moyens à mettre en œuvre pour réaliser l'évolution
- Identifier les impacts possibles de cette évolution sur les autres fonctionnalités de l'application

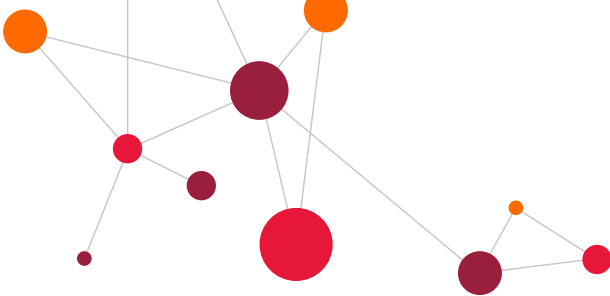
En cas d'impact, nous remontons l'information au client, afin qu'il décide si faut l'éviter ou le prendre en compte.

6.1.4.4 Chiffrage

Une fois l'analyse terminée, nous réalisons un chiffrage. Il s'agit principalement d'estimer le temps que nous prendra la mise en œuvre des actions identifiées dans l'analyse. Il faut également prendre en compte le temps nécessaire aux tests et aux livraisons.

Ce chiffrage est envoyé au client, qui peut l'accepter ou le refuser. En cas de refus, des discussions ont lieu pour en comprendre la raison (surqualité, mauvaise compréhension de l'une ou l'autre des parties...) et trouver un accord.

6.1.4.5 Test et Développement



Lorsque le chiffage est accepté, nous pouvons débiter les développements en nous appuyant sur l'analyse. Nous réalisons également des tests unitaires afin de valider le fonctionnement des développements. Ils permettront également de lever des alertes à l'avenir en échouant si un comportement de l'application est modifié.

Des tests empiriques sont également réalisés en local afin de valider le fonctionnement de l'évolution.

6.1.4.6 Livraison en développement

Une fois que nous avons une version qui nous semble satisfaisante en local, nous la livrons sur le serveur de développement. Après avoir compilé en local, nous déposons les composants sur le serveur dans un dossier au nom de la livraison. Nous supprimons d'abord les fichiers des composants batch et config de la version précédente afin de ne pas conserver de vieux fichiers. Nous extrayons ensuite le contenu des archives Batch et Config livrées. Nous mettons ensuite à jour les composants Business et ThirdPartyService sur le serveur WebSphere, et nous redémarrons ce dernier.

Si le composant IHM est impacté par l'évolution, nous devons le fournir à l'équipe IHM du client, qui a la charge de l'installer.

Nous faisons enfin quelques essais afin de valider l'installation, et nous informons le client que l'évolution est livrée en développement. Ce dernier effectue quelques tests pour valider le fonctionnement de l'évolution. En cas d'anomalies, nous repassons par une phase de test et développement.

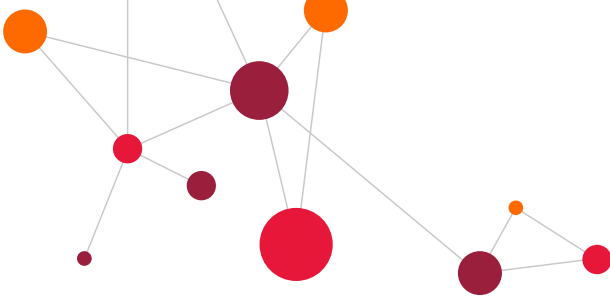
6.1.4.7 Release

Si le client valide la version livrée en développement, nous effectuons une release. Il s'agit de figer la version du logiciel, de générer les packages correspondant, et de les déposer sur le gestionnaire de dépôt Nexus. La release est également testée par SonarQube, qui analyse le code source afin de détecter d'éventuels erreurs de codage ou infractions de certaines normes.

La version est alors privée de son suffixe -SNAPSHOT, et la future version est incrémenté. Par exemple, si la version livrée en développement est la 1.0.0-SNAPSHOT, la version de la release sera 1.0.0, et la future version de développement sera la 1.0.1-SNAPSHOT. Il est bien entendu possible de choisir d'autre numéro de version s'il ne s'agit pas d'une évolution mineure.

L'ensemble de ses opérations est réalisé grâce à l'application Jenkins. Il interagit avec le SVN pour mettre à jour les versions, avec Maven pour générer les composants, et avec Nexus pour les y déposer.

Cette étape est également très importante pour détecter des erreurs. En effet, la release n'étant pas compilé en local, elle permet de s'assurer que l'ensemble des dépendances de



l'application sont présentes sur Nexus, et que tous les développements sont bien sur le SVN. De plus, Maven effectue une phase de test avant de générer les packages. En cas d'échec de l'un des tests unitaires, la release n'est pas générée.

6.1.4.8 Livraison en qualification

Nous n'avons pas les droits nécessaires pour réaliser une installation en qualification ou en production. Nous devons faire une demande de livraison aux équipes techniques du client. Elle doit notamment contenir un dossier d'installation avec toutes les instructions nécessaires à l'installation, et la date de livraison souhaitée. La première tâche est donc d'écrire le dossier d'installation, la seconde est de faire la demande.

Le jour de la livraison, nous suivons la livraison à mesure que l'équipe technique la réalise et nous assurons que tout se passe correctement. En cas de problème, nous donnons des instructions pour rétablir la situation. Il peut s'agir d'un retour arrière (livraison de la version précédente) si le problème est bloquant. Notons toutefois que, bien que nous ayons envisagé ce scénario, nous ne l'avons pas encore rencontré.

6.1.4.9 Recette

L'objectif de la recette est de tester l'évolution dans des conditions les plus proches possible de la production afin de valider l'évolution ou de détecter des anomalies.

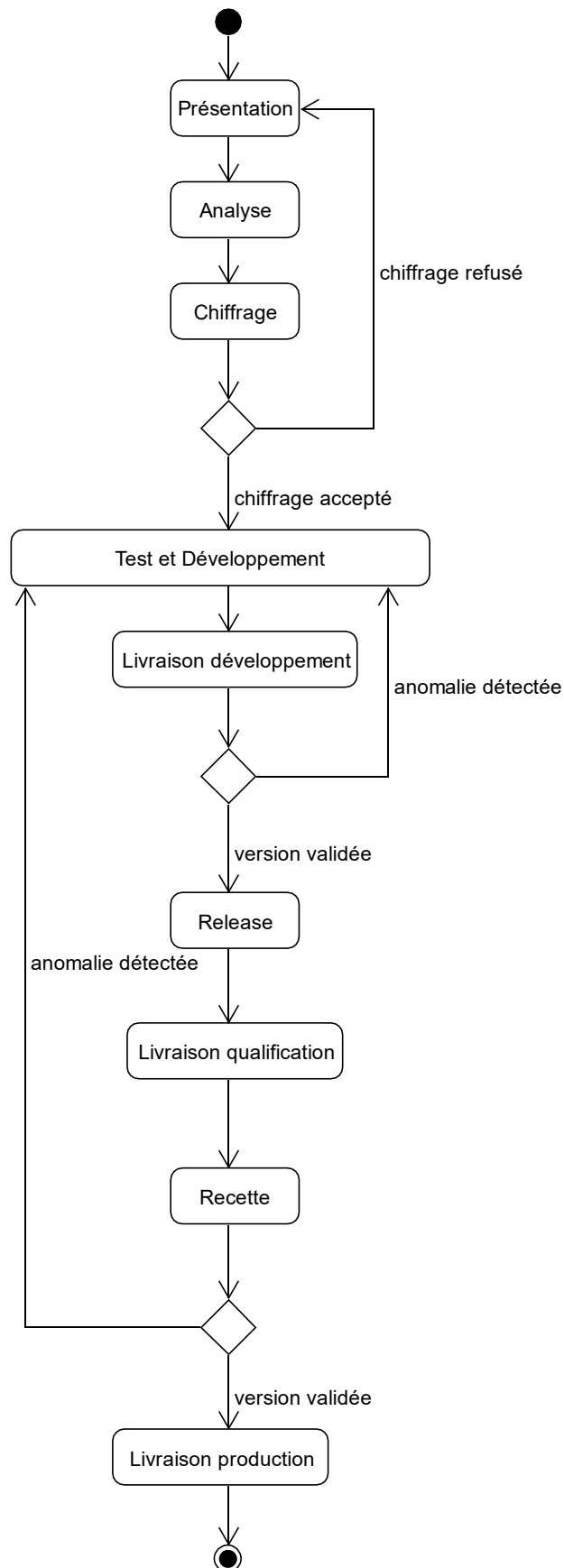
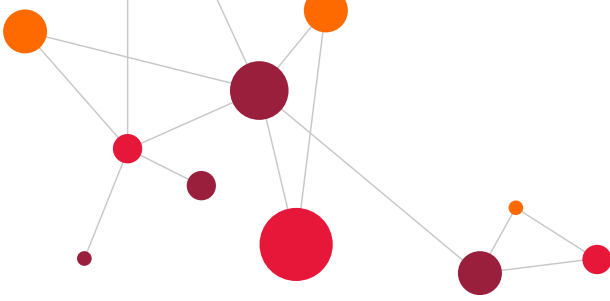
La recette est majoritairement effectuée par le client. Il nous sollicite parfois lorsqu'il souhaite effectuer des vérifications techniques. Nous pouvons par exemple fournir des requêtes SQL afin de faciliter leur validation de l'évolution ou expliquer certains comportements de l'application.

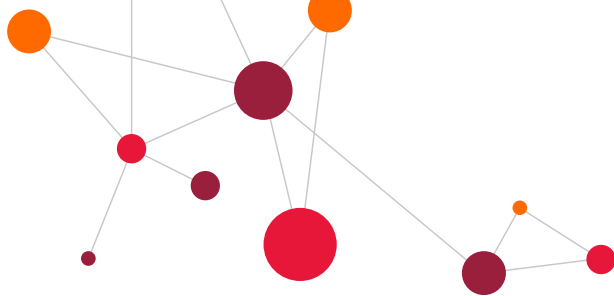
Si une anomalie est détectée, nous repassons à la phase de test et développement.

6.1.4.10 Livraison en production

Lorsqu'une recette est validée, le client nous demande de livrer en production. La procédure est identique à la livraison en qualification. À partir de ce stade, un problème sur le comportement de l'application lié à cette évolution n'est plus considéré comme une anomalie, mais comme un incident, ce qui est hors du cycle de vie de l'évolution. La livraison en production est donc la dernière étape de ce cycle. Elle induit une clôture de la demande du client dans nos outils de suivi des demandes.

6.1.4.11 Résumé avec un diagramme





6.2 Évolution Export Hebdomadaire

6.2.1 Besoin

L'un des traitements automatisés de SYNERGY consiste à exporter des dossiers clients dans des fichiers csv. Il existe 3 types d'export, l'un est quotidien et n'extrait que les dossiers modifiés au cours des dernières 24h, les deux autres sont respectivement hebdomadaire et mensuels, mais extraient tous les dossiers.

Le client souhaitait une évolution de l'export hebdomadaire. Il nous a fourni un cahier des charges et a organisé une réunion pour en discuter. Il nous a expliqué que le besoin était motivé par une évolution de l'application receveuse des fichiers d'export. Cette évolution avait une date de mise en production fixée, nous devons donc réaliser nos mises en qualification et en production aux mêmes dates que l'application receveuse.

Le besoin concernait le traitement hebdomadaire. Il fallait qu'il n'extrait que les dossiers de la semaine en cours au lieu d'extraire toutes les données. De plus, contrairement à ce qui était fait jusqu'à présent dans les trois traitements, il ne fallait pas inclure tous les dossiers ayant été mis à jour la semaine en cours, mais seulement ceux ayant changés de statut.

6.2.2 Analyse

6.2.2.1 Introduction

Une fois le besoin compris suite à la lecture du cahier de charge et une réunion avec le client, je suis passé à l'analyse. Elle consiste à lire et comprendre le code actuel, et à visualiser les changements à effectuer pour mettre en œuvre l'évolution.

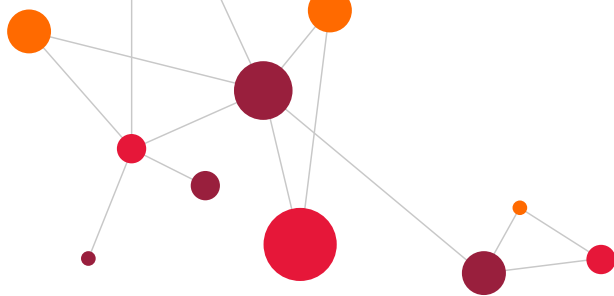
Cette étape est nécessaire pour avoir une vue d'ensemble de la tâche à réaliser. Comme nous l'avons vu précédemment dans le cycle de vie d'une évolution, elle permet également d'estimer le temps nécessaire au développement pour la phase de chiffage.

Les traitements automatiques sont initialisés par Autosys, lorsque certaines conditions horaires sont remplies. Dans le cas de l'extraction hebdomadaire, il s'agit des dimanches à minuit. À ce moment-là, un script Shell du composant CONFIG est lancé. C'est donc par ce script qu'a démarré mon analyse.

6.2.2.2 Script lanceur

Le script Shell comporte plusieurs étapes. Tout d'abord, il récupère la date du jour, et la stocke dans une variable. Il démarre ensuite le composant Batch dans un nouveau processus à l'aide de la commande `java -jar`. En plus des arguments génériques, notamment celui permettant d'identifier le job, deux arguments lui sont fournis : le mode de lancement et la date du jour.

À ce stade du script, le processus Shell attend que le processus fils effectuant le traitement java se termine. Laissons de côté la partie java pour le moment. Une fois le traitement java



terminé, une extraction au format csv a été réalisée. Elle est renommée par le script pour correspondre au type d'extraction (hebdomadaire, quotidienne ou mensuelle). Une copie du fichier est ensuite envoyée sur le réseau intranet pour être récupérable par d'autres applications. L'extraction est enfin renommée avec la date du jour puis déplacée dans un autre dossier pour être archivée.

6.2.2.3 Job spring batch

Le script Shell démarre le composant Batch en lui fournissant plusieurs paramètres. L'un de ces paramètres permet d'identifier le job à lancer, à l'aide de l'id de son bean dans le contexte spring. La première étape que fait le job d'export est de vérifier la cohérence des paramètres fournis. Il en vérifie 3, la date de début, la date de fin, et le mode de lancement.

Le mode de lancement doit être un entier contenu dans l'énumération des modes de lancement : avant, après, intervalle. Dans le cas contraire une exception est levée, provoquant l'échec du job. Les contrôles sur les dates dépendent du type de lancement :

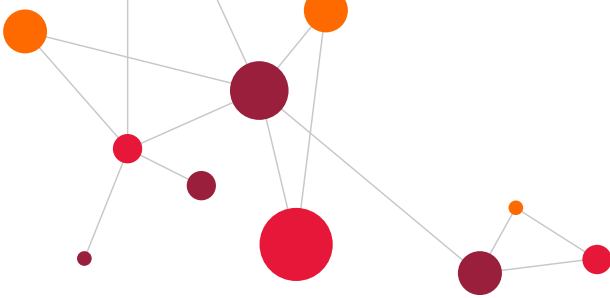
- Dans le cas d'un lancement *avant*, la date de début est fixée à la date minimale autorisée par l'objet calendrier. La date de fin doit être définie et inférieure ou égale à la date actuelle.
- Dans le cas d'un lancement *après*, le traitement est très similaire. La date de début doit être définie et inférieure ou égale à la date actuelle, et la date de fin est fixée à la date actuelle.
- Dans le cas d'un lancement *intervalle*, la date de début et de fin doivent être définies et inférieures ou égales à la date actuelle.
- Dans tous les cas, la date de début doit être inférieure ou égale à la date de fin.

Comme pour le contrôle du type de lancement, une exception est levée et provoque l'échec du job si un des contrôles de date est non valide.

Une fois ces dates contrôlées et définies, une nouvelle étape est lancée. Elle consiste à récupérer dans la base de données les dossiers ayant été mis à jour à une date comprise entre la date de début et la date de fin, et de les stocker en mémoire sous forme d'objet. La récupération est réalisée en une seule requête. Le mapping pour convertir son résultat en objet est réalisé par Hibernate.

Les dossiers sont enfin fournis à la dernière étape : la conversion des objets en chaîne de caractères au format csv, et leur écriture dans un fichier. Il s'agit du fichier que nous avons évoqué précédemment, celui qui est renommé par le script Shell et envoyé sur le réseau à la suite du traitement java.

6.2.2.4 Modifications à apporter



Le traitement hebdomadaire est lancé en mode *avant*. Il ne fournit que la date de fin, qui est la date d'exécution afin de tout récupérer. Afin de ne récupérer que les dossiers de la semaine en cours, il faut le lancer en mode *intervalle*. Cependant, ce mode (ainsi que tous les autres) ne récupère pas les dossiers ayant changé de statut dans un intervalle, mais les dossiers ayant été mis à jour, même s'ils n'ont pas changé de statut. Un nouveau mode ou un nouveau paramètre est donc nécessaire pour spécifier au job qu'il doit effectuer une récupération des dossiers différentes.

J'ai opté pour un nouveau paramètre. Le risque d'impact sur les autres traitements d'export est plus grand, mais cela permet plus de souplesse. En effet, si un nouveau mode avait été ajouté, il aurait été très semblable au mode *intervalle* à la différence près que la récupération des dossiers aurait été effectué par une autre requête. En ajoutant un paramètre spécifiant le type de récupération (mise à jour simple ou mis à jour statut seulement), il est possible de combiner n'importe lequel des 3 types de lancement avec le type de récupération souhaitée. Cela faciliterait grandement un éventuel nouveau besoin d'export concernant la mise à jour de statut.

Il nous est à présent possible de lister les actions à réaliser :

Une énumération avec le type de récupération est donc à ajouter. Afin de ne pas impacter les autres exports, si le type de récupération n'est pas renseigné, c'est la récupération sur la date de mise à jour du dossier qui est choisie.

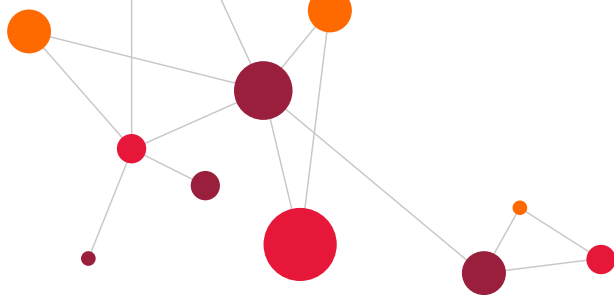
Un contrôle doit être ajouté à la première étape pour vérifier que si le type de récupération est défini, il doit être référencé par l'énumération.

Un test sur cette énumération est également à faire avant de récupérer les dossiers. Ce n'est pas la même méthode que l'on appellera pour les récupérer selon le type de récupération.

Enfin, il faut écrire la méthode de récupération des dossiers par leur date de dernière mise à jour de statut. Cette donnée est déjà présente dans la base de données et mis à jour par les différents processus de l'application. Il ne s'agit donc pas d'ajouter un nouvel indicateur à intégrer à de multiples endroits. Il suffit d'utiliser cette colonne pour filtrer les dates et récupérer les dossiers voulus.

Du côté du script Shell, les paramètres fournis lors de l'appel java change. Le mode de traitement n'est plus *avant*, mais *intervalle*. Il faut récupérer les dossiers de la semaine seulement. Comme ce job doit tourner les dimanches, je propose de fixer sa date de fin au précédent dimanche (ou la date d'exécution si on est un dimanche), à minuit. Cela permet de pouvoir relancer ce job au cours de la semaine si un incident a empêché son exécution à la date prévue ce qui serait impossible en utilisant simplement la date d'exécution. La date de début n'était pas renseignée. Elle doit être fixée à 7 jours avant la date de fin, pour bien récupérer les dossiers de la semaine précédant l'exécution du job.

6.2.2.5 Impacts



J'avais prévu de ne modifier que des fichiers spécifiques au traitement d'export. Aucun impact n'était donc possible sur d'autres traitements. Il y a toutefois 3 traitements d'exports (quotidien, hebdomadaire et mensuel). L'ajout du paramètre "type de récupération" pouvait poser problème s'il n'était pas facultatif. Mais puisque le mode de récupération classique est utilisé par défaut, ce paramètre ne devrait causer aucun impact aux autres traitements d'exports.

Aucun impact ne semblait donc possible, mais il était prévu de tester également les deux autres traitements d'exports par mesure de précaution.

6.2.3 Du chiffage à la mise en production

Suite à cette analyse, j'ai réalisé le chiffage. Après validation par mon chef de projet, je l'ai envoyé au client. Ce dernier l'a accepté et j'ai réalisé les développements. Les étapes du cycle de vie de l'évolution s'étant déroulées sans encombre je ne m'attarderai pas dessus.

Cette évolution n'était pas d'un grand niveau de complexité, mais c'est la première que j'ai réalisée intégralement. C'est pourquoi je l'ai incluse dans ce dossier.

6.3 Évolution Connecteur JSON

6.3.1 Demande initiale

6.3.1.1 Besoin

Le composant IHM doit être remplacé, en raison de l'obsolescence de son framework java. Une nouvelle IHM doit être développée, dans un framework javascript appartenant au client. Or, les données transmises par les requêtes de l'IHM et les réponses du composant Business sont dans un format propre au framework java.

Le client nous demande donc de créer un point d'entrée permettant de recevoir et envoyer des flux dans un format compréhensible par le framework javascript : JSON. Il s'agit bien d'ajouter et non de remplacer, car les deux IHM doivent cohabiter le temps que la nouvelle soit entièrement développée validée et déployée.

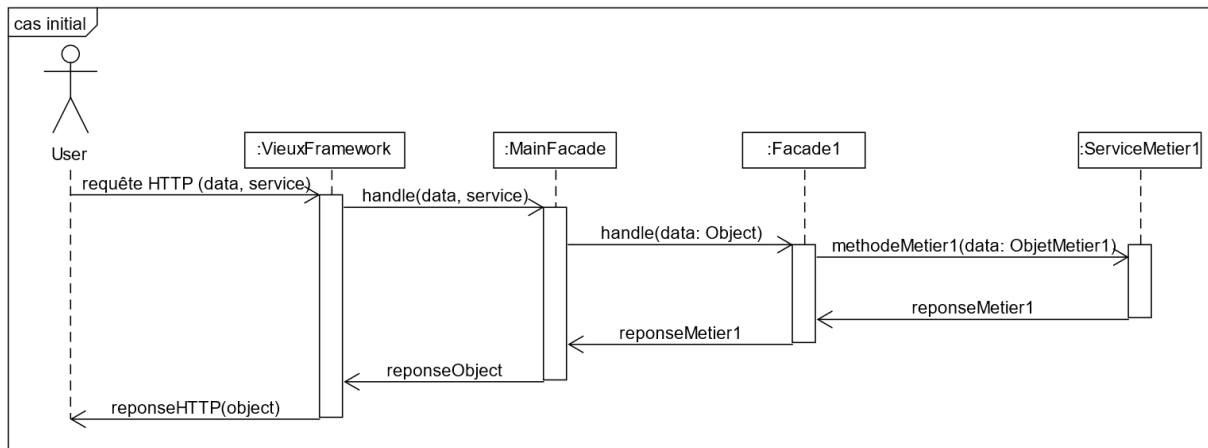
6.3.1.2 Analyse

Avant d'ajouter un point d'entrée, voyons comment fonctionne le point d'entrée actuel. Celui-ci est situé dans le composant Business, dans la librairie Façade. Il est appelé par le framework obsolète et prend deux paramètres : data et service. Data va contenir l'ensemble des données métiers envoyés, tandis que service permet d'identifier quelle façade de l'application est appelé.

Les façades contiennent toutes une méthode handle. Elle prend en paramètre un objet de type Object, qui est ensuite casté en un objet métier. Il est ensuite transmis à la couche business, qui effectue les traitements, et renvoie un objet de sortie. Cet objet est ensuite à

nouveau retourné par la méthode handle de la façade. Le point d'entrée le récupère dans une variable de type Object, et le retourne au framework.

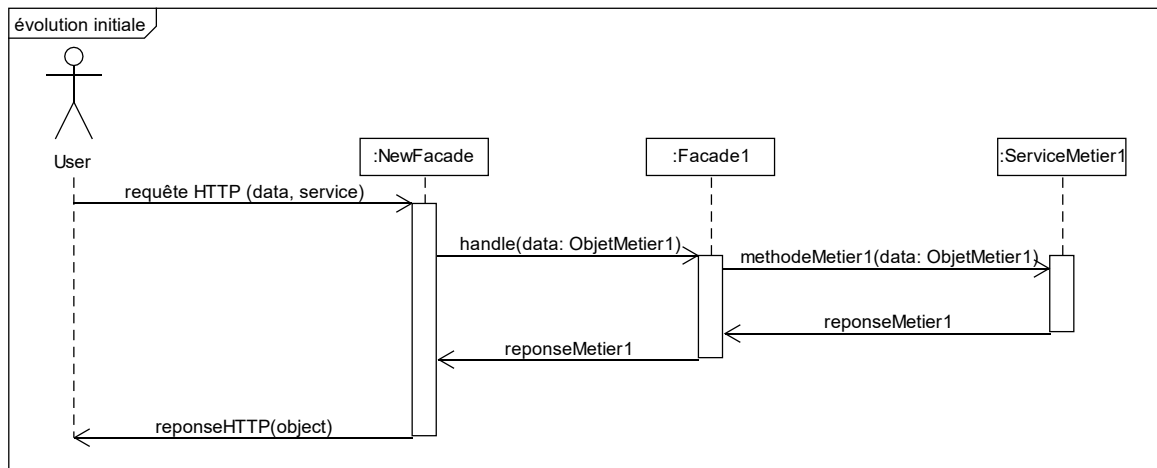
Pour plus de clarté, résumons toutes ces étapes avec un diagramme de séquence :



Pour créer un nouveau point d'entrée il faut donc une alternative au framework obsolète. Il faut également transformer le JSON en objet métier et vice versa. Or, cette conversion pose problème dans le typage très générique qu'utilise la façade actuelle :

- En entrée, il faut choisir en quelle classe le JSON doit être désérialisé. Cela revient à faire des traitements non génériques alors que la prochaine étape (la façade) attend un type générique (Object). Identifier en quelle classe le JSON doit être désérialisé ne peut se faire qu'en effectuant des traitements spécifiques selon le paramètre service reçu.
- En sortie, la sérialisation est difficilement réalisable si nous recevons un objet de type Object. En effet, il faudrait d'abord tester s'il s'agit de l'instance de chacun des cinquante différents types possibles d'objet de sortie, puis le caster en ce type.

Dans les deux cas, il est plus logique de se substituer également à la façade. Nous sommes obligés de réaliser un test sur le paramètre d'entrée service afin de savoir comment désérialiser le JSON. Mais ce test étant déjà écrit dans l'ancienne façade, cela ne représente que très peu de charge de travail. En ce qui concerne le retour, se substituer à l'ancienne façade principal signifie appeler directement les façades, et donc obtenir un retour typé. Ce typage nous permet de savoir comment sérialiser.



6.3.1.3 Du chiffage aux tests sur le serveur de développement

Le chiffage a été réalisé par mon collègue, validé par le chef de projet et envoyé au client. Une fois accepté, nous avons réalisé les développements. Ils se sont déroulés sans problème particulier. La majeure partie du travail a consisté à mettre à jour tous les objets contenus dans la librairie pivot afin qu'ils puissent être sérialisés en JSON par la librairie jackson.

Les tests en local étant concluants, le connecteur a été livré sur le serveur de développement.

6.3.2 Seconde demande

6.3.2.1 Besoin

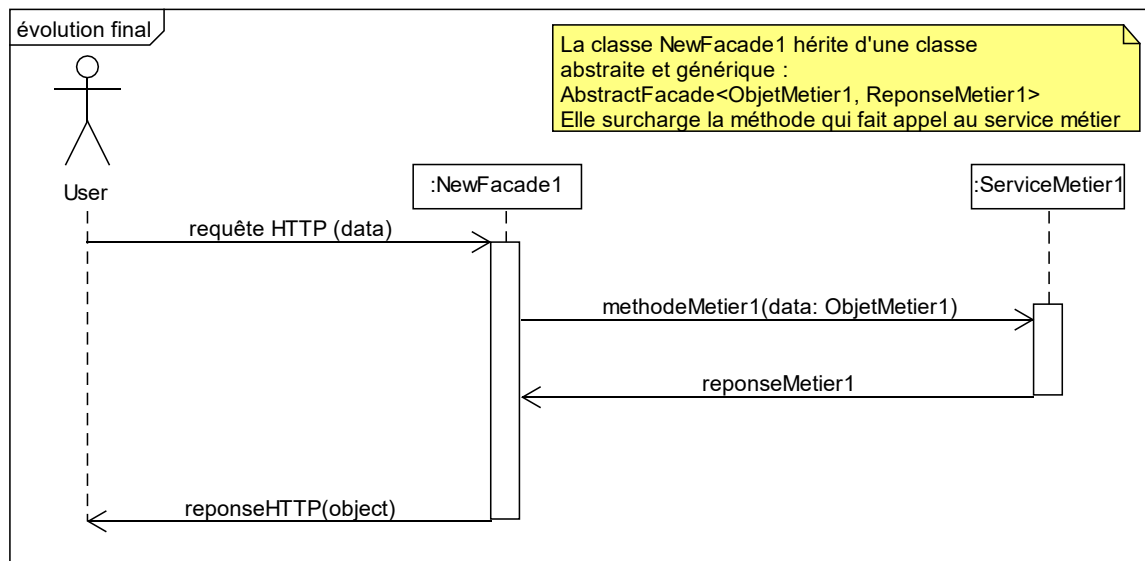
Après que nous ayons déployé le connecteur JSON sur le serveur de développement, l'équipe IHM a démarré ces tests. Rapidement, elle a remonté un problème bloquant. Il y avait une incompatibilité technique entre le connecteur livré et le framework javascript utilisé. L'équipe IHM a en effet déclaré qu'il leur était impossible d'appeler tous les services avec la même url, et de spécifier le service voulu en paramètre.

Nous avons remonté ce problème au responsable d'application. Il nous a alors demandé de résoudre ce problème en faisant autant de point d'entrée que de service.

6.3.2.2 Analyse

Puisqu'il nous faut créer autant de point d'entrée que de service, il n'y a plus besoin d'une classe qui dispatche vers les différentes façades. De plus, pour éviter de complexifier l'application en ajoutant une nouvelle couche à l'existant, nous n'utiliserons plus les anciennes façades. Nous effectuerons les appels à la couche business dans les nouvelles.

Comme les façades se ressemblent énormément, nous créerons une façade abstraite générique dont toutes les façades hériteront.



6.3.2.3 Du chiffage aux tests sur le serveur de développement

Il n'y a rien de particulier à noter dans le cycle de vie de cette évolution jusqu'aux tests en développement. À cette étape, nous avons assisté à de nombreuses reprises l'équipe en charge du développement de la nouvelle IHM. Une demande annexe à celle-ci a été réalisée, où nous avons créé une rétro documentation de chacun des services de l'application et fournit des exemples de données d'entrées et de sorties.

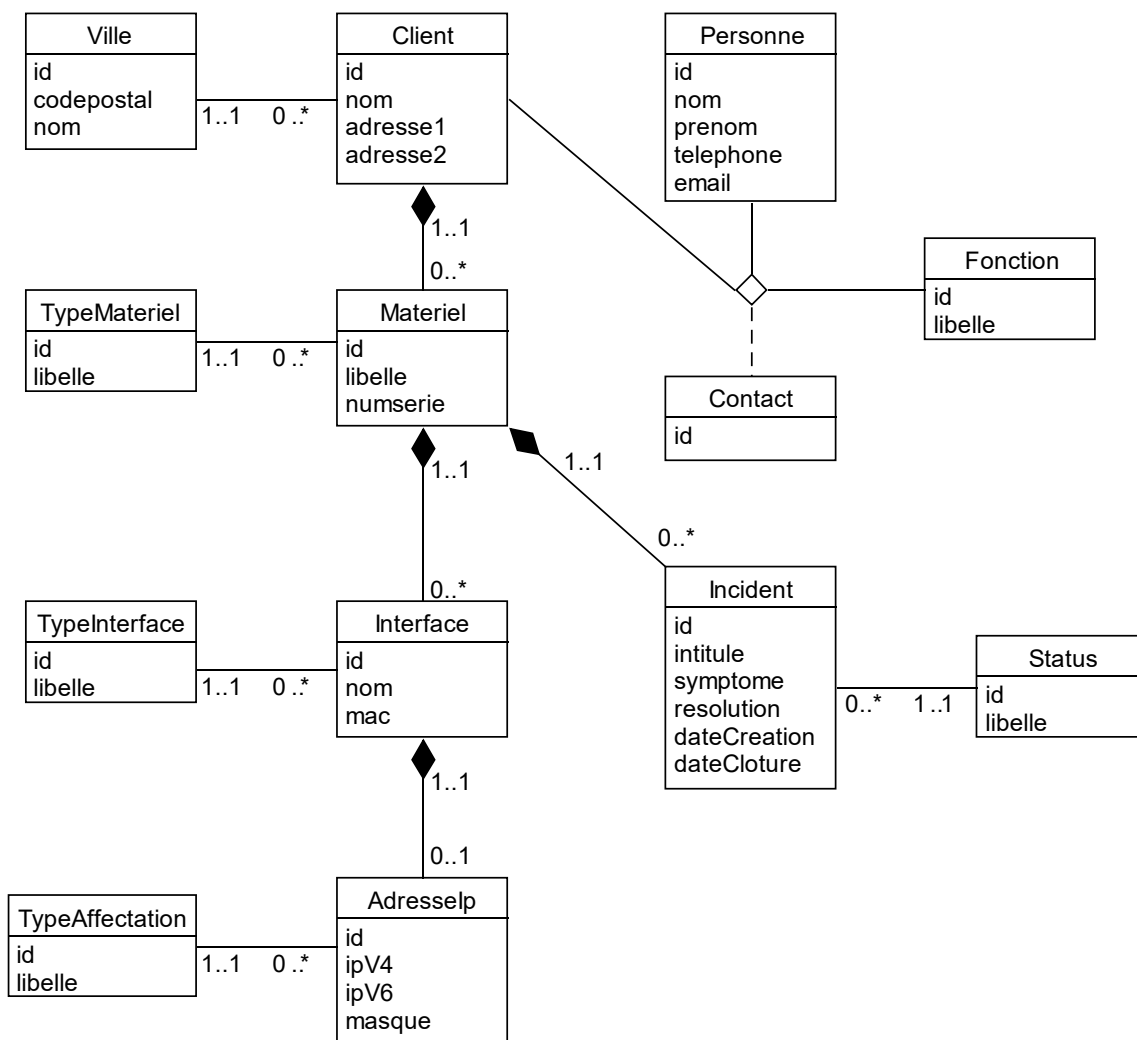
À l'heure où j'écris ses lignes, le nouveau composant IHM est toujours en développement.

7 Gestion de matériel réseau

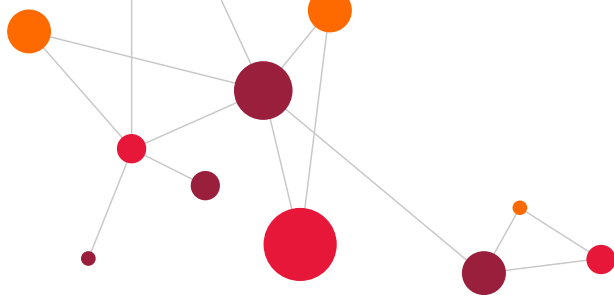
7.1 Présentation

Ce projet a été réalisé en formation mais également et surtout sur mon temps libre. C'est un travail de groupe, réalisé avec Maxime Laurin et Kateryna Bouchet. Il nous a été demandé par l'EPSI afin de pouvoir évaluer notre niveau, et notre capacité à travailler ensemble.

Ce projet gère les clients d'une entreprise de location de matériel informatique, qui ont tous la particularité d'être connectable en réseau. Voici un schéma présentant les différents objets métier de l'application.



Ce projet devait être composé de trois parties (application Android, API, web), chacune ayant des spécifications. La suite de ce document consistera à présenter rapidement chacune de ces parties et quelques-uns de leurs points marquants ou fonctionnalités.



7.2 Travail en groupe

7.2.1 Outils

L'outil le plus important que nous utilisons est Git. Il s'agit d'un logiciel de gestion de version, qui nous permet donc de partager et recevoir les modifications effectuées sur le projet. Les 3 IDE que nous avons utilisées étaient capables de le gérer (Eclipse Spring Tool Suite, Android Studio et Visual Code), mais nous avons également utilisé les lignes de commandes de git bash.

Sur chaque dépôt, nous utilisons deux branches : dev et master. Lorsque nous avons une version stable en dev, nous la fusionnons vers la branche master avec les commandes suivantes :

```
git checkout dev
git pull
git checkout master
git pull
git merge dev
git push
```

Nous avons également une branche security sur le dépôt de l'API. Il s'agit de tests de ma part pour ajouter une authentification à l'API. L'authentification était fonctionnelle, et fonctionne grâce à un token JSON. Toutefois, je ne suis pas allé jusqu'au bout du travail en créant les rôles, pour ne pas complexifier le projet. Les professeurs nous ont expliqué qu'ils préféraient que l'on se concentre sur les fonctionnalités des applications.

Les dépôts git ont été créés sur GitHub pour chacune des trois parties du projet :

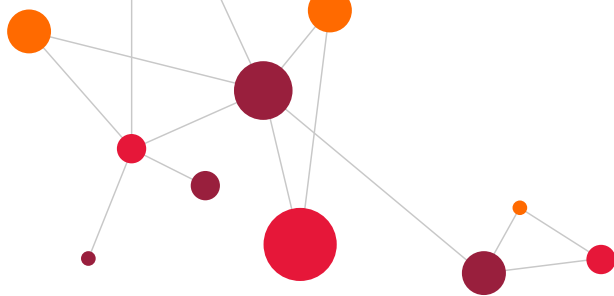
- API : <https://github.com/maxime173/ApiGestionMateriel>
- WEB : <https://github.com/maxime173/WebGestionMateriel>
- APP : <https://github.com/maxime173/GestionMateriel>

Si vous souhaitez les consulter, le plus simple est d'aller sur le front web, car des liens vers les 3 dépôts et le swagger de l'API y sont présents : <https://web-gestion-materiel.herokuapp.com/>

Le chargement initial peut être un petit peu long car lorsque les applications ne sont plus utilisées depuis un certain temps, heroku les met en sommeil. Le temps de relance ne devrait pas durer plus de 30 secondes.

7.2.2 Mon rôle

J'occupais une place importante dans notre groupe.



Étant le plus expérimenté, je me suis chargé de développer les traitements les plus techniques (sérialisation/désérialisation, envoie/réception de requête/réponse HTTP, conversion en PDF, génération d'image...).

J'effectuais également des recherches sur les framework que nous allions utiliser (Spring Boot et Angular) afin de pouvoir expliquer comment procéder au reste du groupe.

Je suis aussi celui qui était chargé de résoudre les problèmes sur lesquels mes camarades bloquaient. J'effectuais également les déploiements en ligne de l'API et de la partie web.

Enfin, j'ai été chef de projet en quelques sorte. En effet, bien que je n'aie pas suivi de méthodologie particulière, je fixais les priorités sur certaines tâches, et je m'efforçai de maintenir à jour la liste des tâches effectués sur le serveur discord que nous utilisions.

Pour l'API, j'ai créé un tableau de bord présentant la totalité des fonctionnalités de l'API (au moment du tableau). L'entité est à gauche, viennent ensuite les fonctionnalités de cette entité puis le développeur qui doit les réaliser.

Le code couleur est simple :

- Rouge signifiait qu'il ne faut pas développer la fonctionnalité
- Vert signifie qu'elle est développée
- Blanc signifie que la fonctionnalité est à faire. Il n'y en a plus sur le tableau car le développement est terminé.

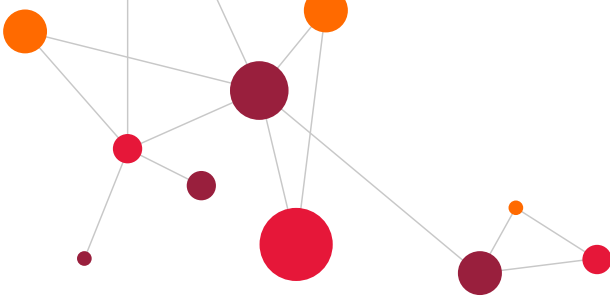
	Récupérer	Récupérer+	Créer	Modifier	Supprimer	Développeur
Personne	Membre	Non	Membre	Membre	Non	Etienne
Fonction	Membre	Non	Non	Non	Non	Etienne
Contact	Membre	Membre	Membre	Membre	Membre	Etienne
Ville	Membre	Non	Membre	Non	Non	Maxime
Client	Membre	Non	Technicien	Membre	Technicien	Maxime
Materiel	Membre	Membre	Membre	Technicien	Membre	Maxime
TypeMateriel	Membre	Non	Non	Non	Non	Maxime
Interface	Membre	Membre	Technicien	Technicien	Membre	Katya
TypeInterface	Membre	Non	Non	Non	Non	Katya
Adresselp	Membre	Membre	Technicien	Technicien	Membre	Katya
TypeAffectation	Membre	Non	Non	Non	Non	Katya

Tableau de bord du suivi des fonctionnalités de l'API

L'indication Membre ou Technicien correspondait au type d'utilisateur qui devait avoir accès à ces fonctionnalités, mais comme expliqué précédemment, nous n'avons pas développé la connexion à l'API.

7.3 Application Android

7.3.1 Introduction



L'application mobile est la première partie qu'il nous a été demandé de faire. L'objectif était que l'on découvre le développement d'application Android par nos propres moyens, et que l'on puisse présenter l'application à la fin de la semaine. Il nous était imposé d'utiliser un langage natif d'Android pour développer l'application : java ou kotlin. Aucun membre de notre groupe n'était familiarisé avec le développement d'application Android. Nous avons donc opté pour java, langage avec lequel nous avons déjà travaillé, afin de ne pas risquer de se noyer dans un océan de nouveauté.

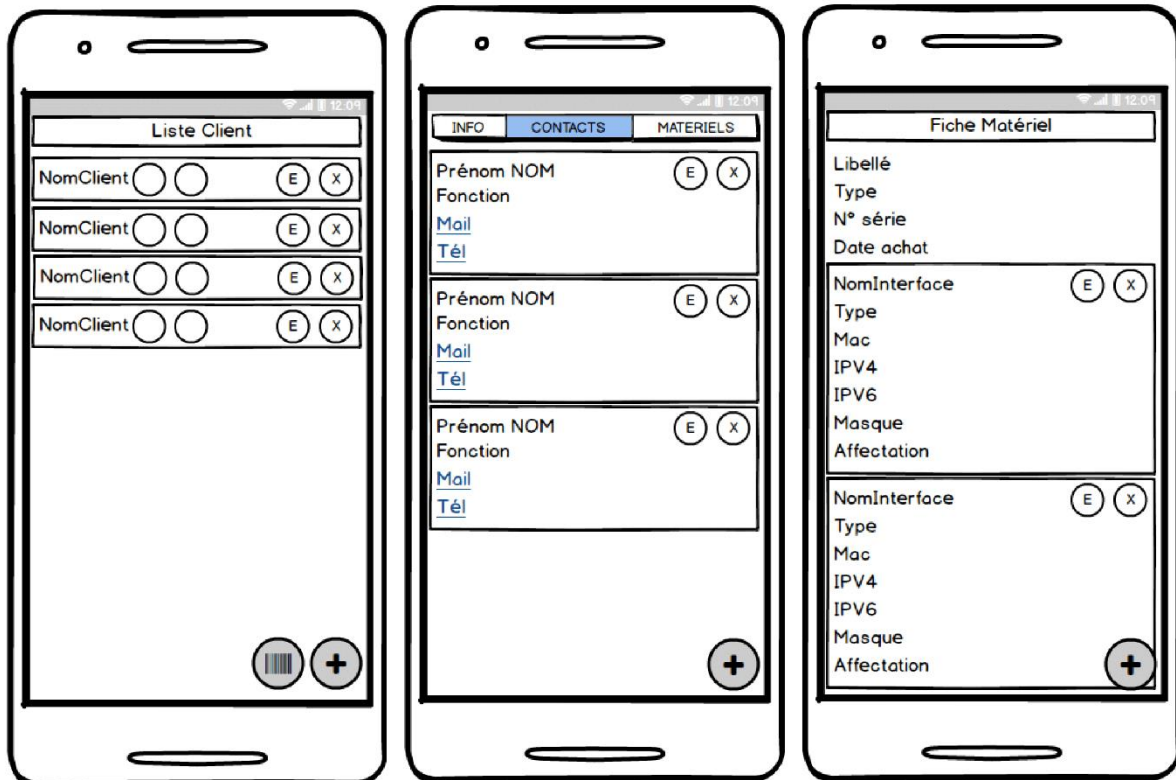
Plusieurs objectifs étaient à réaliser avant la fin de la semaine où nous avons travaillé sur l'application :

- Réaliser les différentes vues de l'application permettant d'afficher les données métiers
- Être capable d'ouvrir l'application SMS ou le client mail du téléphone à la bonne adresse depuis les contacts d'un client
- Récupérer le contenu d'un fichier json hébergé en ligne par les professeurs, et le désérialiser en objet métier
- Ouvrir l'application directement à une fiche matérielle spécifié par un QR code

7.3.2 Maquettage

La première question que l'on s'est posé est « Quoi » ou en reformulant : « Que doit faire notre application ». La réponse vient assez rapidement en consultant la liste des objectifs ci-dessus. Vient alors une question plus épineuse : « Comment ? ».

Nous avons décidé de commencer par y répondre en nous soustrayant des questions techniques. Pour représenter les différentes vues, nous avons créé des maquettes.



Exemples de maquettes réalisées

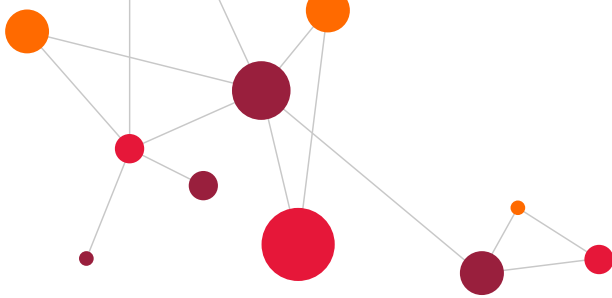
Puis, nous avons recherché comment faire ces vues techniquement. Nous avons compris que les vues devaient hériter des classes Activity, comment utiliser des layout personnalisés pour répéter un design avec des données différentes dans les listes, comment utiliser les Fragment pour faire des onglets...

7.3.3 Architecture

Une fois ces connaissances découvertes, nous avons créé les bases du projet. Nous avons décidé d'adopter l'architecture 3 tiers : présentation, métier (que nous appelons service) et accès aux données (que nous appelons repositories). Toutefois, nous ne l'avons pas respecté par manque de temps dans la semaine alloué à cette application. Nous avons en effet appelé directement la couche repositories dans la couche présentation. Nous n'avons malheureusement presque pas retravaillé cette application par la suite, et ce problème n'a donc jamais été corrigé.

L'API utilisant également cette architecture, et de manière correcte, je ne m'attarderai pas sur l'architecture de l'application Android. Résumons simplement :

- Les templates (layout) sont dans les ressources de l'application (/app/src/main/res/layout).



- La couche présentation contient les classes java manipulant les vues : les Activity, les Fragments, les ViewHolder et les Adapter (qui permettent la création de liste de données).
- La couche service devrait contenir à minima des classes avec des méthodes passe-plats pour lier la couche présentation à la couche repositories. Mais comme expliqué précédemment, cela n'a pas été fait.
- La couche repositories réalise les requêtes http pour récupérer du JSON, et le sérialise.
- Nous avons également un package utilisable par toutes les couches : entite. Il contient les entités de l'application, les objets métiers.

7.3.4 Récupération des données JSON

Un fichier JSON contenant les données métiers de l'application a été mis en ligne. Il fallait donc un moyen de les récupérer : effectuer une requête HTTP. Pour ce faire, nous avons utilisé la librairie Volley.

Pour ajouter la librairie, nous avons ajouté la dépendance suivante dans le fichier build.gradle :

```
implementation 'com.android.volley:volley:1.1.1'
```

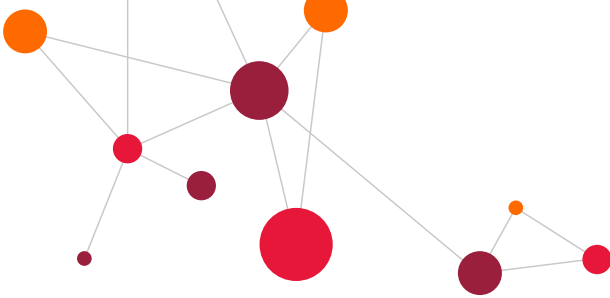
La création de requête HTTP avec cette librairie est simple. Il suffit de créer un objet de type Request, et de l'ajouter à une RequestQueue. Il y a cependant deux petites subtilités :

- Les requêtes sont réalisées de manière asynchrone. Ce qui est préférable, afin d'éviter de figer l'application jusqu'à avoir une réponse. Pour réagir de manière asynchrone, il faut à la requête un Response.Listener et un Response.ErrorListener. Les traitements inclus dans les listeners seront effectués une fois la réponse reçue.
- La création d'une RequestQueue étant légèrement coûteuse en ressource, la documentation android recommande de ne la créer qu'une fois, et de faire un singleton pour la gérer. Nous avons suivi cette recommandation.

Dans un second temps, lorsque j'ai déployé l'API sur heroku, j'ai changé le lien de la requête HTTP, pour qu'il pointe vers l'API.

Une action supplémentaire était nécessaire pour effectuer des requêtes, et nous avons perdu beaucoup de temps à le découvrir en cherchant la cause autre part : avoir la permission du téléphone sur laquelle l'application est installée d'utiliser internet. Il suffit d'ajouter une ligne dans le manifeste pour déclencher la demande d'autorisation :

```
<uses-permission android:name="android.permission.INTERNET" />
```

Une fois tout ceci effectué, la requête fonctionnait et du texte au format json était bien récupéré. Il restait alors à le désérialiser.

7.3.5 Désérialisation des données JSON

En java, plusieurs librairies fiables peuvent permettre de désérialiser du json en objet. Ayant déjà utilisé jackson lorsque j'ai travaillé sur le connecteur JSON (voir Évolution Connecteur JSON), j'ai opté pour cette dernière.

Pour désérialiser du JSON avec cette librairie, il nous a fallu annoter les différentes entités de l'application, afin que la librairie sache qu'elle attribut correspond à quel champ en JSON. Il est également obligatoire que chaque entité possède un constructeur vide. Une fois ces deux choses faites, une simple ligne permet la conversion :

```
this.clients = new ObjectMapper().readValue(json, new  
TypeReference<List<Client>>() {});
```

La seule complexité ici vient du fait que le json contient une liste de client. Il faut donc un type qui correspondent cette liste de client. C'est ce qui est créé avec une classe appartenant à la librairie : TypeReference.

7.3.6 Ouverture suite au scan d'un QR code

Une application android peut-être appelé par une autre grâce aux Intent. Lorsque l'on installe une application android sur un téléphone, celui-ci enregistre son ou ses intent-filter si elle en possède. Il est possible d'utiliser des filtres standard, comme « sms » afin que l'application puisse être appelé lors d'action standard.

Mais dans notre cas, nous souhaitons que l'application s'ouvre à la bonne fiche matérielle. Il s'agit d'un cas spécifique, allons donc utiliser un filtre spécifique. Voici le filtre ajouté :

```
<data android:scheme="GestionMateriel"/>
```

Il nous a ensuite fallu générer un QR code qui appelle ce filtre et qui fournisse les paramètres nécessaires pour retrouver la fiche. Il en faut deux : l'id du matériel dont on veut consulter la fiche, et l'id du client qui le possède. Nous avons opté pour la syntaxe suivante :

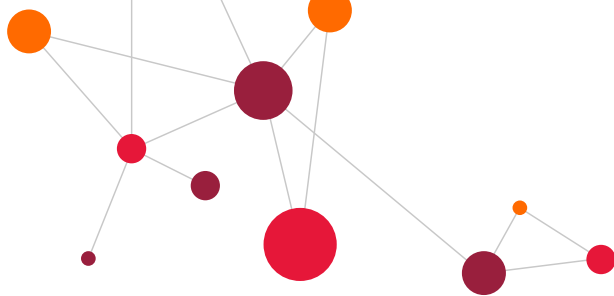
```
GestionMateriel://clients/idClient/materiels/idMateriel
```

Le scan d'un QR code contenant ce texte ouvrait bien l'application. Il restait alors à faire en sorte qu'elle navigue jusqu'à la fiche matérielle voulu.

Pour se faire, au démarrage de l'application, nous vérifions si un paramètre a été transmis via une Intent. Si c'est le cas, il faut les récupérer. Il suffit ensuite de fournir ces paramètres à l'activité qui affiche la fiche matérielle pour qu'elle obtienne et affiche les bonnes données.

7.4 API

7.4.1 Introduction



Une fois la semaine dédiée à la réalisation de l'application Android écoulee, nous devons faire l'API, qui serait par la suite appelée par l'application Android, et la partie web.

Plusieurs contraintes étaient posées :

- L'API devait être en Java
- Elle devait renvoyer du JSON
- Elle doit également générer le QR code à scanner par l'application Android
- En plus des objets métiers précédemment vu, il devait être possible de créer des incidents
- Il faut pouvoir réaliser un export en PDF des incidents d'un client pour une période donnée

Nous avons vu en cours une manière de faire des API avec la librairie JAX-RS. Toutefois, nous avons remarqué qu'à CGI, la grande majorité des projets Java utilisait le framework Spring. Nous trouvions plus enrichissant d'apprendre à utiliser ce framework pour réaliser l'API.

Ayant des bases avec le framework puisque j'ai travaillé avec à CGI, j'ai décidé de faire des recherches et un petit projet de test pour découvrir, m'exercer et montrer aux autres membres de mon groupe comment procéder avec Spring. Après avoir lu plusieurs documentation et guide, je suis parvenu à un résultat qui me semblait satisfaisant sur le projet de test. J'ai opté pour le Spring Boot, qui permet de se passer de configuration xml en définissant une configuration par défaut. La configuration se fait alors par des annotations et des fichiers properties.

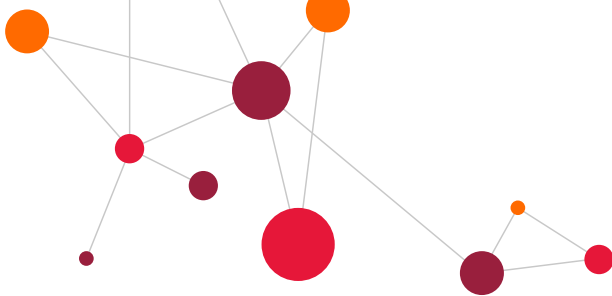
L'API devait recevoir des requêtes des autres parties (front) de l'application. Il fallait donc que l'API soit déployée sur un réseau. Notre groupe travaillant sur son temps libre, et donc pas depuis le même réseau local, j'ai préféré la déployer en ligne. J'ai choisi Heroku pour sa simplicité et sa gratuité. Ce déploiement a toutefois ajouté un problème d'encodage des caractères spéciaux dans les réponses HTTP de l'API. Nous n'avons pas consacré de temps à la résolution de ce problème, que nous ne rencontrons pas en local.

L'API est déployée à cette url (il faut ajouter un suffixe pour accéder à un service de l'API) :
<https://api-gestion-materiel.herokuapp.com/>

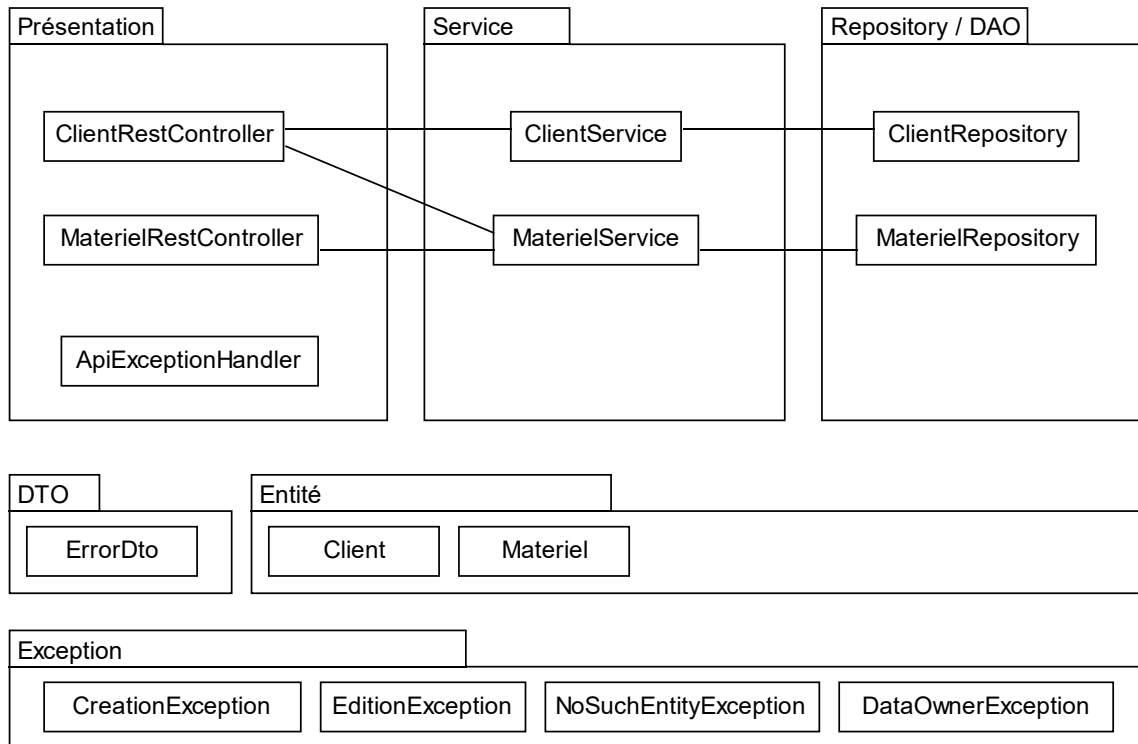
7.4.2 Architecture

7.4.2.1 Introduction

L'API réutilise le socle que j'ai créé pour mon projet de test. Ayant choisis Spring Boot, ce dernier est très succinct. Il est en effet composé du fichier pom.xml pour gérer les dépendances Maven, et du fichier application.properties qui définit les accès à la base de données, les paramètres de JPA, le port d'écoute, l'encodage en UTF-8 et la durée des timeouts.



L'architecture utilisée est une architecture trois tiers :



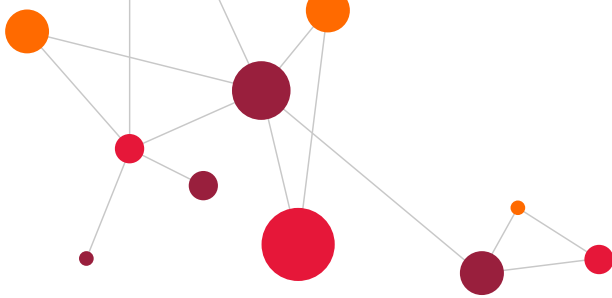
7.4.2.2 Entités

Les entités ne sont pas une couche à proprement parler, car toutes les couches l'utilisent. Il s'agit plutôt d'un package. Il contient les objets métiers, qui sont annoté pour pouvoir être utilisable par JPA (ici implémenté par Hibernate).

Nous avons également un DTO (Data Transfer Object). Il s'agit d'objet qui ne sont pas utilisé dans les traitements, mais qui permette de reformater les données métiers avant de les envoyer. Nous n'utilisons pas réellement de DTO dans l'application. Il n'en existe en fait qu'un seul : ErrorDto, qui est envoyé si une exception survient. Plus de détail seront fourni ci-dessous dans la partie dédiée à la couche présentation.

7.4.2.3 Repository

La couche repositories contient les accès aux données. Grâce à Spring Data, cette couche ne contient que très peu de code. Nous ne définissons en effet que des interfaces qui implémente `JpaRepository<>`. Elles contiennent des méthodes ayant un nommage spécifique, qui permet à Spring d'implémenter l'interface à notre place. Pour certains cas particuliers, la requête à exécuter est renseignée à l'aide de l'annotation `@Query`. L'accès à la base, la création de la requête et la conversion du résultat de la requête en entité est entièrement réalisé par Spring Data. Dans notre API, il existe un repository par entité.



7.4.2.4 Service

La couche service contient toute la logique métier de l'application. Nous avons un service par entité, et un repository est injecté par spring dans chaque service. Nous avons défini 4 interfaces générique :

- `IGettableService<>`
- `ICreatableService<>`
- `IEditableService<>`
- `IDeletableService<>`

Les différents services de l'application implémentent toujours une interface qui leur est propre, lié à leur entité, et une ou plusieurs des interfaces listées ci-dessus. Cela nous permet de nous assurer que tous nos services ont un fonctionnement similaire. Un deuxième avantage est que nous sommes obligés d'implémenter les méthodes de l'interface. Il n'est donc pas possible d'accidentellement ajouter une fonctionnalité de modification générique qui ne respecteraient pas certaines règles de gestion de l'entité.

7.4.2.5 Présentation

La couche présentation contient les contrôleurs de l'application. Ces derniers contiennent les points d'entrées de l'application. Il y en a un par méthode.

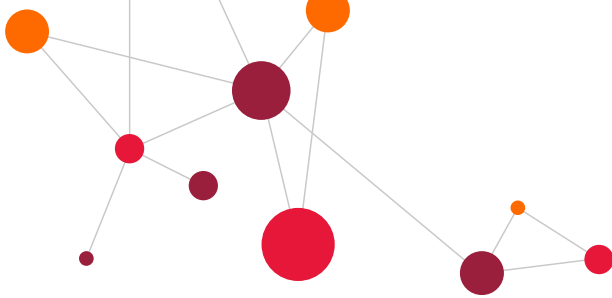
Chaque méthode se veut très succincte, car la logique métier est de la responsabilité de la couche service. Les contrôleurs se contentent de convertir les données reçues, de les fournir à la couche service, et de retourner les données envoyées par la couche service, en spécifiant le code HTTP si nécessaire.

Spring se charge de la désérialisation et sérialisation. Il utilise pour cela la librairie jackson, mais de manière totalement transparente. La seule indication nécessaire à Spring est une annotation qui définit le format de sortie voulu (JSON).

Il y a un contrôleur par entité, mais une entité pouvant en gérer d'autres, un contrôleur peut se faire injecter les services de plusieurs entités. Le meilleur exemple de ce projet est le client. Il possède des contacts et du matériel. Lorsqu'un contact est créé, la requête est effectuée sur le client à qui il appartient. C'est donc le contrôleur de l'entité client qui gère la création de contact, et qui utilise pour cela le service des contacts.

La couche présentation contient également une classe `ApiExceptionHandler` portant l'annotation `@RestControllerAdvice`. Si une exception survient, cette classe est appelée. Elle se charge de renvoyer une réponse HTTP à la place du contrôleur.

Cette réponse est uniquement composée du message d'erreur de l'exception et du code HTTP. Ces deux informations sont stockées dans le DTO `ErrorDto`. Le code http est défini en fonction du type d'exception. Admettons par exemple que l'exception est `CreationException`



soit levé. Cette exception a été créée par nos soins et n'est levée que lorsqu'il est impossible de créer une entité en raison de ses mauvais paramètres. Nous savons donc que lorsqu'elle survient, l'erreur vient des données envoyées par l'utilisateur, et le code 400 est renvoyé. En revanche, dans le cas d'une `NullPointerException`, il s'agirait d'une erreur du serveur, et dans ce cas, c'est le code 500 qui est retourné. La classe peut retourner un dernier code : 404. Celui-ci n'est sélectionné que lorsque l'exception personnalisée `NoSuchEntityException` est retournée.

7.4.3 Swagger

Notre API inclut la librairie `springfox`, qui permet d'intégrer `swagger` dans `spring`. Toutefois, nous n'avons pas du tout utilisé toute la puissance de la librairie. Nous nous sommes contenté d'utiliser `swagger ui`, qui nous a beaucoup servi pour tester les différents webservices de l'API. Il est disponible à cette url : <https://api-gestion-materiel.herokuapp.com/swagger-ui.html> Ce lien est également présent dans la barre de navigation de la partie web du projet.

client-rest-controller Client Rest Controller

GET /api/v1/clients getAll

POST /api/v1/clients create

GET /api/v1/clients/{id} getById

PUT /api/v1/clients/{id} edit

DELETE /api/v1/clients/{id} deleteClient

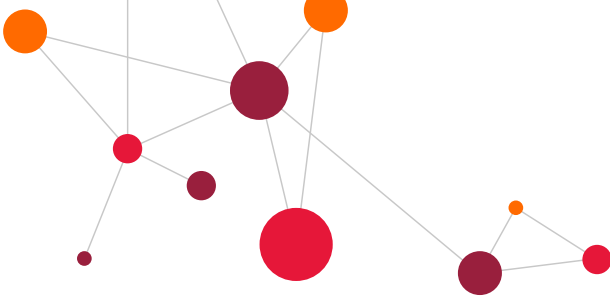
Parameters Cancel

Name	Description
id * required integer(\$int64) (path)	id <input type="text" value="id - id"/>

Execute

Responses Response content type: application/json; charset=UTF-8

Un aperçu de Swagger UI



7.4.4 Fonctionnement de l'API

L'API reçoit des requêtes HTTP. Elle doit faire correspondre cette requête avec l'un de ces traitements. Nous avons établi quelques règles pour cela. Par soucis de simplicité, l'entité gérée dans ses règles sera appelée X :

- Un contrôleur responsable de l'entité X interceptera les requêtes dont l'uri commence par `api/v1/X`. Dans l'uri, X doit être au pluriel.
- Une requête GET vers l'uri `api/v1/X` retournera l'ensemble des entités X
- Une requête GET vers l'uri `api/v1/X/Y` retournera l'entité X ayant l'id Y.
- Une requête POST vers l'uri `api/v1/X` créera une entité X avec les données fournies dans le corps de la requête si elles sont valides.
- Une requête PUT vers l'uri `api/v1/X/Y` modifiera l'entité X ayant l'id Y avec les données fournies dans le corps de la requête si elles sont valides.
- Une requête DELETE vers l'uri `api/v1/X/Y` supprimera l'entité X ayant l'id Y si cela n'enfreint pas de règle de gestion.
- Une requête vers l'uri `api/v1/X/Y/Z`, où Z est le nom d'une entité appartenant à X suivra les mêmes règles que précédemment. Toutefois, nous ne montons pas de plus d'un niveau pour ne pas surcharger les contrôleurs.

7.4.5 Génération du QR code

La génération de QR code peut sembler complexe. Et elle le serait probablement s'il n'y avait pas de librairie permettant de le faire. Une librairie est très connue en java pour créer les QR code et code barre : Zxing. Cependant, la librairie n'est plus vraiment maintenue. J'ai opté pour QRGen, qui ajoute une surcouche à Zxing et qui est très simple d'utilisation. Voyez plutôt :

```
QRCode.from(uriApp).withSize(200, 200).to(ImageType.PNG)
```

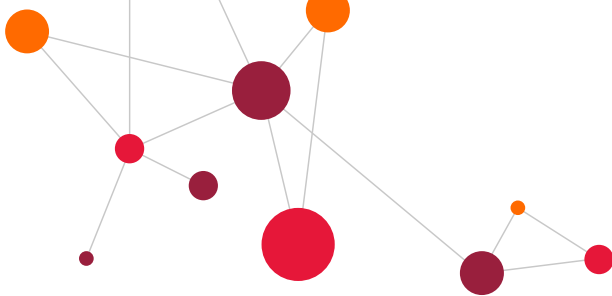
En une ligne, nous avons généré un QR code contenant l'adresse d'une fiche matérielle pour l'app Android, sous forme d'une image png. Un problème peut alors se poser, comment faire en sorte qu'une API renvoie des données sous forme d'image ? Une image n'étant finalement que du binaire, il suffit :

- de la retourner sous forme de tableau de byte

```
QRCode.from(uriApp).withSize(200,200).to(ImageType.PNG).  
stream().toByteArray();
```

- de préciser dans le mime type de la réponse HTTP qu'il s'agit d'une image en PNG

```
@GetMapping(value="/{id}/materiels/{idMateriel}/qrcode",  
produces=MediaType.IMAGE_PNG_VALUE)
```



7.4.6 Gestion des incidents

Les incidents sont des entités qui ont été ajoutés dans un second temps, à la demande des professeurs. Ils ont donné lieu à une modification de la base de données. De plus, comme les incidents sont liés à un matériel, les flux json retournés lorsqu'ils contiennent un matériel étaient également différents. L'application Android ne connaissant pas les incidents, j'y ai ajouté l'entité afin que la désérialisation fonctionne à nouveau.

Les incidents possèdent les champs suivants :

- un intitulé
- un champ symptôme, qui est la description de l'incident par le demandeur
- un champ résolution, qui correspond au retour du technicien lorsqu'il résout l'incident
- une date de création
- une date de clôture
- un statut (ouvert ou fermé)

Dans notre application, les incidents sont des entités moins génériques que les autres. J'ai créé des règles de gestion qui encadrent la modification de ces champs :

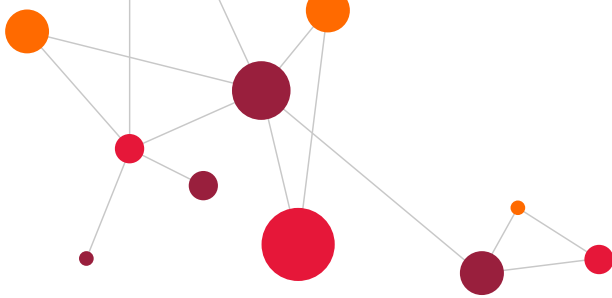
- À la création, la date de création est définie à la date actuelle, la date de clôture et la résolution sont nulles et le statut est ouvert.
- Lors de la modification, seuls les champs intitulés, symptôme et résolution sont modifiables, les autres seront ignorés.
- Il est impossible de clore un incident ayant un champ résolution vide.
- Il est impossible de modifier un incident clôt.

7.4.7 Export PDF des incidents

7.4.7.1 Récupération des incidents

Il nous est demandé de réaliser l'export des incidents pour un client précis entre deux dates. Il convient tout d'abord de se demander quel incident peut être considéré comme entre les deux dates. Est-ce un incident démarré dans l'intervalle ? Ou bien clôturé dans l'intervalle ? En condition réelle, j'aurais appelé le client pour obtenir d'avantage d'informations. Mais dans le cas présent, le travail était réalisé sur mon temps libre et d'avantage orienté vers l'évaluation de mes compétences que la complétion d'un besoin. J'ai donc considéré que les incidents à exporter seraient ceux ayant été à un état différent de « clôturé » durant l'intervalle fourni.

Il s'agit là d'un cas trop complexe pour être exprimé via des conventions de nommage pour que Spring Data réalise la requête pour nous. Je lui ai donc fourni :



```
@Query( "select i " +
        "from Incident i " +
        "inner join Matériel m on m.id = i.materiel.id " +
        "inner join Client c on m.client.id = c.id " +
        "where " +
        "(i.dateCloture is null or i.dateCloture >=
:dateDebut ) " +
        "and i.dateCreation <= :dateFin " +
        "and c.id = :idClient ")
    public List<Incident> getAllByClientBetween(Date
dateDebut, Date dateFin, Long idClient);
```

Notez que le langage de cette requête n'est pas SQL, mais HQL (Hibernate Query Language). C'est pour cela que l'on peut voir plusieurs "points" à la suite dans les jointures, HQL étant orienté objet.

7.4.7.2 Conversion en PDF

Nous avons appris en cours à générer des PDF en java à l'aide de la librairie pdfbox. Il fallait au préalable créer un template en PDF à l'aide de LibreOffice, puis le modifier avec la librairie en écrivant du texte à des positions déterminées sur le document. Cette méthode me semblait extrêmement rigide, et j'ai tenté d'en trouver une autre pour respecter l'objectif fixé par les professeurs.

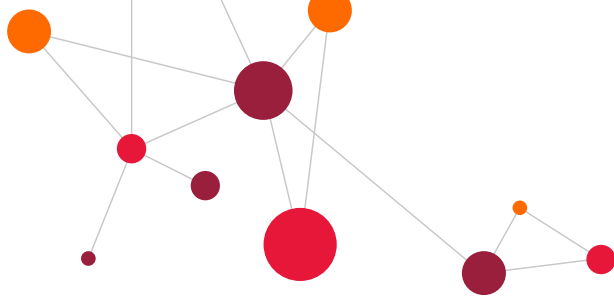
J'avais en tête qu'il était très probablement possible de générer du code html et de le convertir en PDF. Cela permettrait également à l'API de fournir un format supplémentaire pour cet export. Après quelques recherches, il s'est avéré que c'était en effet possible avec la librairie Flying Saucer. Cette dernière est sous licence AGPL, qui exige la mise à disposition du code source des applications qui l'utilisent. Notre API étant déjà Open Source, ce n'est pas un problème pour nous.

J'ai donc réalisé une première conversion d'une liste d'incident en chaîne de caractère au format HTML. Pour ce faire, j'ai créé un template très simple, que j'ai stocké dans une chaîne de caractère. Ce dernier contient 4 sous-chaînes qui seront remplacées dans les traitements :

- {DEBUT}, {FIN} et {CLIENT} pour les indiquer dans le titre de la page
- {LIGNES} qui doit contenir des lignes de tableau au format HTML représentant les incidents

Une fois cette conversion en HTML réalisée, une nouvelle conversion en PDF est réalisable :

```
ByteArrayOutputStream pdfStream = new ByteArrayOutputStream();
ITextRenderer renderer = new ITextRenderer();
```

```
renderer.setDocumentFromString(html);  
renderer.layout();  
renderer.createPDF(pdfStream);  
pdfStream.close();
```

Nous souhaitons que l'API retourne le pdf en binaire, comme pour l'image du QR code. C'est pourquoi j'ai choisi `ByteArrayOutputStream` pour stocker le pdf. Cette classe contient en effet une méthode permettant sa conversion en tableau de byte : `toByteArray()`.

```
ResponseEntity.ok().headers(headers).  
body(pdfStream.toByteArray());
```

7.5 Web

7.5.1 Introduction

La dernière partie de ce projet est la partie web. Nous n'avons aucune contrainte technique la concernant. Le plus facile aurait été d'ajouter une couche présentation à l'API qui aurait pu faire appel à la couche service. Nous trouvions toutefois dommage de ne pas l'utiliser, et nous avons plutôt souhaité nous orienter vers un framework javascript pour cette partie. Aucun d'entre nous n'avait de bonnes connaissances sur ces framework, mais ce fut justement l'occasion de les approfondir.

Nous avons énormément hésité entre les framework les plus populaires : Angular, React et Vue. Nous avons finalement opté pour Angular, car nous avons eu une initiation à ce framework et qu'il semble être le plus utilisé des trois par CGI.

7.5.2 Architecture

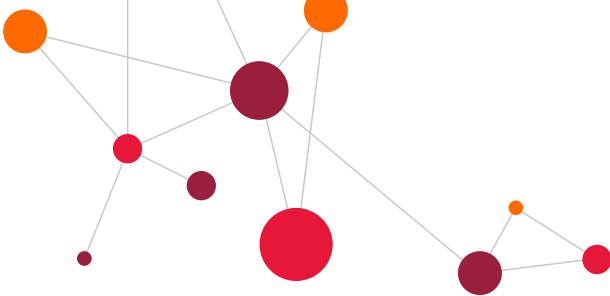
L'architecture de la partie web de notre projet est une architecture standard d'un projet Angular. Nous n'avons fait qu'un seul module.

7.5.2.1 Entité

Comme pour les autres projets, nous avons un dossier contenant l'ensemble des objets métiers à manipulé. La conversion du JSON envoyé par l'API en ces objets est réalisée par le framework.

7.5.2.2 Service

Il existe un service par entité. Les services sont chargés d'effectuer des traitements (récupération, ajout, modification, suppression) sur les entités. Comme la partie web ne contient pas de stockage de données, les services effectuent des requêtes http vers l'API pour réaliser ces traitements.



Tous les services héritent d'une classe générique abstraite que j'ai écrite. Elle contient l'url de l'api, et une en-tête http qui définit Accept et Content-Type à application/json, ce qui convient à la quasi-totalité des requêtes.

7.5.2.3 Composants

Chacun de nos composants contient une classe principale, qui gère une vue. Ces classes ne sont pas appelées contrôleur en Angular, mais y ressemblent fortement puisqu'elles gèrent une vue et font appels à un service.

À la création d'un composant, Angular injecte les différents éléments présents dans son constructeur. C'est à ce moment-là que le service est fourni au composant. Il appelle ensuite sa méthode `ngOnInit`. Dans cette méthode, nous appelons l'API grâce au service afin d'obtenir les données à afficher dans la vue. Le retour des données est écouté de manière asynchrone afin de ne pas figer l'application.

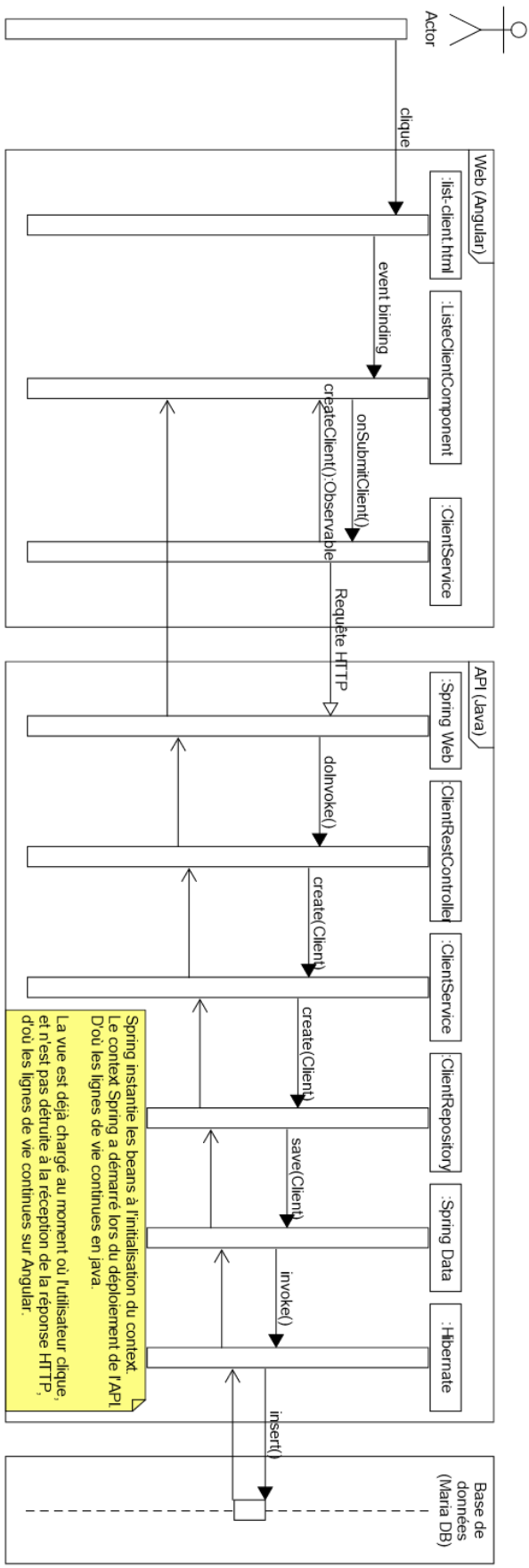
Sur demande de la vue (event binding), le composant peut être amené à effectuer divers traitements. Le plus souvent, il s'agit de récupérer les données d'un formulaire et de les envoyer à l'API pour créer ou modifier une entité. Mais il peut également s'agir d'affichage. Par exemple, le composant peut changer la valeur d'un booléen, qui est lié (property binding ou directive) à l'affichage ou non d'une modale.

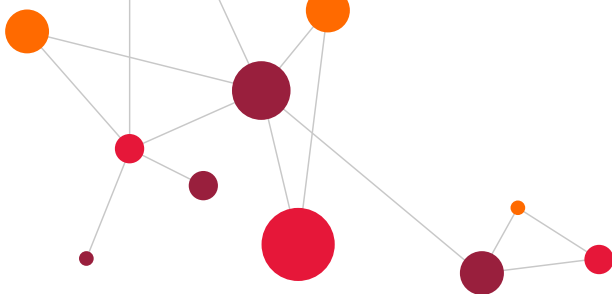
7.5.2.4 Vue

Les vues sont des parties de pages au format HTML. Elles contiennent toutefois des directives tels que `*ngIf` ou `*ngFor` qui permettent de dynamiquement changer son affichage. Comme nous l'avons évoqué, son contenu peut également être modifié via le property binding. Les sollicitations envers le composant sont effectuées via l'event binding, en général suite à un clic sur un élément.

7.5.2.5 Schéma

Voyons sur la page suivante un diagramme de séquence présentant la création d'un client, du clic d'un utilisateur sur la partie web, à la requête en base de données.





8 Conclusion

Cette année d'alternance m'a permis d'approfondir mes connaissances dans de nombreux domaines, et d'en découvrir de nouveaux.

Je suis notamment très satisfait d'avoir saisi les principes du développement en couche, et de l'architecture trois tiers. La découverte des serveurs UNIX et du Shell était également très intéressante. Je considère aujourd'hui que mon ignorance à ces sujets était une véritable lacune.

Cette année fut aussi l'occasion de pratiquer le métier et de comprendre réellement ce qu'être un développeur implique et signifie.

Après cette alternance, que le diplôme soit obtenu ou non, je continuerai à travailler au sein de CGI. Toutefois, cette année a ravivé ma soif d'apprendre, quelque peu émoussée par les cours très théoriques de la licence informatique que je suivais au CNAM. Je trouve mes connaissances incomplètes, et j'aimerais les approfondir. Je souhaite pour cela reprendre mes études en Septembre 2019 pour un master. Ma hiérarchie a déjà donné son accord de principe sur la question.

À long terme, je ne me vois pas progresser les échelons pour grimper dans la hiérarchie, car cela m'éloignerait du cœur du métier que j'affectionne tant. En revanche, je me vois bien m'orienter vers les métiers d'architecte ou d'expert technique.

9 Annexes

9.1 Bloc de compétences : Qualité et sécurisation du code réalisé

Compétences ou capacités qui seront évaluées	Critères d'évaluation	Exemples d'activités et tâches	Activités pratiquées	Origine de l'acquisition	Preuves apportées & ref. annexe
Formaliser, identifier les résultats attendus.	La liste de contrôle des attendus fonctionnels est paraphée.	Étude de l'existant.	Analyse de l'existant et des modifications à réaliser pour faire une évolution.	E(2 S)	6.2.2
Respecter des contraintes.	Un plan d'assurance qualité est observé.	Rédaction du cahier des spécifications fonctionnelles.	Rétro-documentation des webservices pour l'équipe IHM.	E(3 S)	6.3.2.3
Respecter les recommandations qualité de la norme en vigueur pour l'architecture des logiciels.	L'application est organisée en couches indépendantes. Les règles métier sont encapsulées dans des services logiciels. L'accès aux données est réalisé par des services logiciels indépendants du mode de stockage. L'exécution de l'application est répartie entre un	Conception/architecture d'applications logicielles. Conception de services métiers. Conception de services d'accès aux données. Détermination du nombre de tiers de l'application.	Conception et architecture de plusieurs logiciels. Le plus représentatif étant l'API présenté dans ce document.	E(3 M) F(2 M)	7.4.2

	nombre d'ordinateurs adapté au contexte.				
Anticiper les évolutions.			Réalisation d'évolution de manière à rendre possible facilement de nouveaux traitements.	E(1 S)	6.2.2.4
Qualifier les risques	Un formulaire d'estimation des risques est rempli.	Estimation, qualification des risques sécurité.	Réalisation d'études d'impact.	E(1 S)	6.2.2.5
Respecter une norme de présentation des écrans et documents de sortie.	Une norme de présentation des données est respectée. Les interfaces Homme/Machine sont validées.	Réalisation d'une interface homme/machine (IHM) Réalisation des maquettes de sorties interactives. Réalisation des maquettes de sortie imprimée.	Réalisation de maquettes avant de créer l'interface d'une application Android.	F(0.2 S)	7.3.2
Concevoir des programmes avec une orientation objets.	Une programmation orientée objets est utilisée. Le taux de réutilisation du code utile est > 80 %. Des gabarits sont utilisés. Une charte de nommage est utilisée.	Programmation de logiciels.	Réalisation de nombreux projets durant la formation et en entreprise, tous orientés objet.	F(1 A)	7.1

	Le taux de documentation interne du code est > 8 % et < 15 %.				
Garantir un accès sécurisé aux données.	Les anomalies d'accès aux données ne génèrent pas d'interruption de l'exécution et sont répertoriées.	Programmation de l'accès aux données de l'entreprise.	Réalisation d'une API accédant à une base de données. Les identifiants de connexion ne sont pas dans le code, qui est open source, mais dans des variables d'environnement du serveur applicatif.	F(1 M) E(3 S)	7.4.2.3 7.5.2.5
Livrer le logiciel déverminé.	Des outils de contrôle automatique du code sont utilisés. Aucun défaut visible ne persiste.	Tests unitaires. Préparation des jeux de tests. Contrôles de l'existence d'anomalies.	Réalisation de tests unitaires. Exécution des tests unitaires lors de la release et annulation de la génération de packages en cas de d'échec.	E(2 S)	6.1.4.5 6.1.4.7

Livrer le logiciel conforme aux attentes.	Les contraintes spécifiques au projet sont respectées. Un manuel d'assurance qualité est respecté. Une méthode de recettage est utilisée. L'étape du projet est validée.	Recettage du logiciel. Validation d'une étape du projet.	Les évolutions réalisées en entreprise passent obligatoirement par une phase de validation puis une phase de recette.	E(3 S)	6.1.4.9 6.1.4.6 6.1.4.11
Clôturer une mission.	Le PV de réception du logiciel est validé.	Mise en exploitation.	Les évolutions réalisées en entreprise sont livrées en production, ce qui clôture leur demande.	E(1 S)	6.1.4.10

9.2 Bloc de compétences : Audit, conception, méthode de projet

Compétences ou capacités qui seront évaluées	Critères d'évaluation	Exemples d'activités et tâches	Activités pratiquées	Origine de l'acquisition	Preuves apportées & ref. annexe
Formaliser des processus, les règles de gestion et d'organisation des données de l'entreprise.	La procédure du service utilisateur est formalisée et validée. La procédure du service utilisateur est conforme aux règles du système de	Étude de l'existant. Identification des procédures en place. Contrôle de la conformité des procédures utilisées avec la gouvernance de	Réalisation d'une procédure de livraison d'application sur un serveur d'application WebSphere.	E(2 S)	6.1.4.8

	<p>management des services de l'entreprise.</p> <p>La circulation du document résultat du traitement prévu est matérialisée dans un diagramme de workflow.</p> <p>La proposition de reconstruction de la procédure est validée.</p> <p>La base de données est modélisée.</p>	<p>l'entreprise.</p> <p>Recensement des documents utilisés, identification de leur circulation et des acteurs concernés.</p> <p>Reconfiguration de procédure.</p> <p>Conception d'une base de données.</p>	<p>Rédaction de nombreux dossier d'installations.</p>		
Concevoir des éléments logiciels réutilisables.	Une méthode de conception par objets est utilisée.	Conception de l'architecture applicative.	Réalisation de classes abstraites contenant du code générique utiles à de nombreuses classes filles.	E(1 M) F(2 S)	7.5.2.2 6.3.2.2
Produire du logiciel en équipe.	Une méthode AGILE est utilisée.	Programmation en équipe. Écriture de code.	Les deux projets présentés dans ce mémoire sont réalisés en équipe.	E(1 A) F(3 M)	7.2

Remonter les alertes au(x) décideur(s).	Absence de signaux d'alertes au point de contrôle du projet.	Coordination de l'avancement.	Prise de contact régulière avec le responsable d'application pour l'informer de l'avancement d'incidents.	E(2 S)	4.8
Estimer des délais.	Les étapes du projet sont planifiées.	Planification des tâches du projet.	Les évolutions réalisées en entreprise sont triées par priorité. Leurs dates de mise en production est impérative car dépendante d'autres mises en production d'applications. Le temps nécessaire à leur développement est déterminé lors du chiffrage.	E(1 S)	6.1.4.4
Concevoir une solution logicielle.	Le projet est conforme au schéma directeur de l'entreprise et respecte les principes d'urbanisation du S.I.	Conception de la solution logicielle.	Les frameworks utilisés ainsi que les normes de développements sont imposées par le client, et contrôlés par des règles	E(1 A)	6.1.4.7

	Les spécifications fonctionnelles produites respectent le cahier des charges fourni.		de SonarQube à chaque release.		
Anticiper des répercussions.	L'impact de modification est acceptable.		Réalisation d'études d'impact.	E(1 S)	6.2.2.5

9.3 Bloc de compétences : Réalisation d'applications logicielles

Compétences ou capacités qui seront évaluées	Critères d'évaluation	Exemples d'activités et tâches	Activités pratiquées	Origine de l'acquisition	Preuves apportées & ref. annexe
Encapsuler des solutions logicielles spécifiques dans des services logiciels génériques.	Le service d'accès aux données est opérationnel.	Programmation. Investigations documentaires fonctionnelles ou techniques complémentaires. Transcription des spécifications fonctionnelles en algorithmes. Transcription des algorithmes en code source. Compilation, déverminage du code source.	Développements de services métier variés dans une API, injecté par Spring dans la couche contrôleur via leur interface, et non leur classe.	F(1 M)	7.4.2.4 7.4.2.5

Produire du logiciel générique réutilisable.	Des services logiciels internes sont réutilisables.		Réalisation d'un socle Spring Boot lors de mes recherches sur le framework, réutilisé pour l'API. Réalisation d'un socle JEE en cours réutilisable pour plusieurs projets.	F(0.2 S)	7.4.2.1
Produire du logiciel partageable.	Des services logiciels sont partageables en local. Des services logiciels sont partageables à distance.		Le projet Gestion Matériel est open source. Les webservices de l'API sont utilisés par les deux interfaces graphique et utilisables par n'importe quelle technologie pouvant effectuer une requête HTTP sur internet et parser du JSON.	F(2 M)	7.2.1
Intégrer des éléments logiciels hétérogènes et produire des exécutables livrables.	Le logiciel est livrable, prêt pour la mise en production.	Agglomération des différents éléments logiciels en unités	Maven agglomère de nombreuses librairies et génère un package livrable si les tests	E(2 S) F(1 S)	6.1.4.7

		de traitement, réalisation des tests unitaires.	unitaires sont tous valides.		
Modifier un algorithme sans générer de dysfonctionnements.	La modification n'entraîne pas de régression fonctionnelle.		La réalisation d'évolution en entreprise est précédée d'une analyse et d'une étude d'impact qui permet d'éviter les régressions.	E(6 M)	6.1.4.3
Contrôler des délais.	Le compte-rendu d'activité est renseigné, les écarts sont constatés.	Mise à jour du planning de réalisation.	Le responsable d'application est averti en cas de problème susceptible de retarder la livraison de l'évolution.	E(0.2 S)	6.3.2.1

9.4 Bloc de compétences : Communiquer avec les acteurs du projet

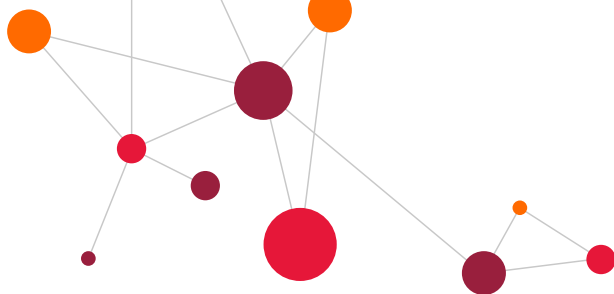
Compétences ou capacités qui seront évaluées	Critères d'évaluation	Exemples d'activités et tâches	Activités pratiquées	Origine de l'acquisition	Preuves apportées & ref. annexe
User d'une communication professionnelle tant en français qu'en anglais.	Le compte-rendu de la réunion est validé.	Élaboration et rédaction de documents techniques, commerciaux ou internes à destination, des utilisateurs,	Échanges écrits et oraux avec les clients	E(1 M) F(1 M)	4.8 7.2 9.6

	Le score du TOEIC est > 749	des clients ou des collaborateurs, ...	et les membres de mon équipe. Validation d'un module de formation en communication au cours d'une licence informatique au CNAM.		<u>9.7</u> <u>9.8</u>
Interagir efficacement dans un environnement de travail collaboratif.	Le document collectant l'expression des besoins des utilisateurs est validé. L'aide du logiciel est rédigée. La documentation du livrable est diffusée. La présentation est appréciée. Les utilisateurs sont opérationnels, le transfert des nouvelles compétences est validé.	Rédaction des spécifications fonctionnelles de la solution informatique. Écriture des interfaces homme/machine. Relations avec les clients. Animation de réunions de travail et interviews d'utilisateurs. Démonstrations, recettage de livrables. Formation des utilisateurs au logiciel.	Communications très fréquentes avec le client, et avec les membres de l'équipe (Morning, retour, alerte...).	E (1 M)	<u>4.8</u>

9.5 Bloc de compétences : Adapter l'environnement d'exécution, échanger des données entre logiciels

Compétences ou capacités qui seront évaluées	Critères d'évaluation	Exemples d'activités et tâches	Activités pratiquées	Origine de l'acquisition	Preuves apportées & ref. annexe
Réaliser des échanges de données informatisés (EDI).	<p>Les données sont consolidées.</p> <p>La base de données tierce est accédée.</p> <p>L'interface d'échange de données est opérationnelle.</p>	<p>Réalisation d'un procédé d'échange de données informatisés.</p> <p>Rétro-documentation de logiciels et de bases de données.</p> <p>Consolidation, agrégation de données.</p> <p>Programmation de l'interface d'échange de données.</p>	<p>Développement de l'évolution connecteur JSON.</p> <p>Développement d'une API récupérant ou modifiant des données en bases, et retournant le résultat dans un format standard (JSON).</p>	<p>E (1 M)</p> <p>F (1 M)</p>	<p>6.3</p> <p>7.4</p>
Automatiser des traitements.	L'environnement de tests est opérationnel.	<p>Réalisation d'un environnement de tests.</p> <p>Création, configuration de machines virtuelles.</p> <p>Installation, configuration de serveurs d'applications, Web et base de données.</p>	<p>Travail sur un ordonnanceur qui lance les batchs selon certaines conditions (horaire, succès du batch précédent...)</p>	E (1 M)	6.1.3.2.2

Programmer des scripts systèmes.		Écriture de scripts systèmes pour adapter l'environnement d'exécution.	Mis à jour et création de scripts Shell lanceur des batchs.	E(3 S)	6.2.2.2 6.2.2.4 (fin)
----------------------------------	--	--	---	--------	--



9.6 Attestation de succès à l'examen Pratiques écrites et orales de la communication professionnelle

République française
Ministère de l'Éducation nationale,
de l'Enseignement supérieur et de la Recherche

Conservatoire national des arts et métiers
Centre régional en Nouvelle-Aquitaine
16 cours de la Marne - 33800 Bordeaux



MINISTÈRE
DE L'ÉDUCATION NATIONALE,
DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE



le cnam
Nouvelle-Aquitaine

M. PICHERIT-STEINBRUCKER Etienne
2 BIS AVENUE DU 11 NOVEMBRE 1918
33320 LE TAILLAN MEDOC

En cas de changement d'adresse, prévenir votre centre d'enseignement.

Attestation de succès à l'examen

Original - Aucun duplicata ne sera délivré.

Code :	CCE105	Crédits : 4 ECTS
Intitulé :	Pratiques écrites et orales de la communication professionnelle	
Délivrée à	N° d'élève :	NAQ414521
	Nom :	PICHERIT-STEINBRUCKER
	Prénom :	Etienne
	Né le :	11/03/1994 à Courcouronnes (91)
Année universitaire :	2018-2019	Note obtenue : 11 / 20
Premier semestre		

Fait à Talence, le 21/03/2019
Pour l'administrateur général et par délégation

Le Directeur,

Jean-Sébastien CHANTÔME

Cette attestation sanctionne le succès à un examen final ou aux épreuves de contrôle organisées en cours d'année (note égale ou supérieure à 10 sur 20). Le code de l'unité d'enseignement (3 lettres et 3 chiffres) correspond à la nomenclature utilisée par le Cnam dans le cadre de la constitution de ses filières et diplômes. Chaque unité d'enseignement (UE) est affectée d'une valeur en crédits. Les crédits représentent l'unité de mesure du travail de l'élève pour l'UE concernée. Les crédits sont capitalisables dans les formations d'une même filière du Cnam. Ils sont aussi transférables dans un autre établissement d'enseignement supérieur, en France ou dans un autre pays de l'espace européen.

Cette attestation, pour être valable, ne doit être ni surchargée, ni grattée.

9.7 Attestation de succès à l'examen BULATS

BULATS

English

CAMBRIDGE ENGLISH
Language Assessment
Part of the University of Cambridge

Candidate Test Report

Candidate Number: 6

Family Name: Picherit-Steinbrucker

First Name(s): Etienne

Test: English - Reading and Listening

Company/Organisation: CNAM

Test Date: 24/11/2018

Language: English

Overall Band: CEFR Level: C1

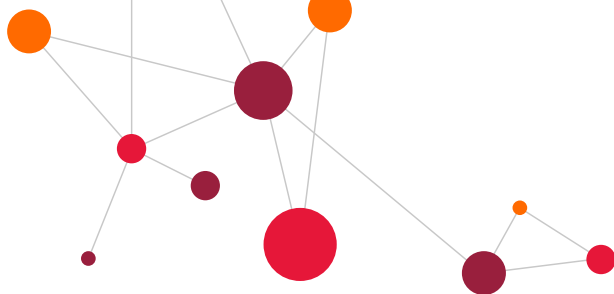
Profile:

Overall Score	81	C1
Listening Score	87	C1
Reading/Language Knowledge Score	77	C1

The scores are given on a standard scale out of 100.

Centre de Passation
Cambridge English
BULATS
Garonne Nouvelle - Aquitaine
Bordeaux
Nathalie
Cambridge English
Language Assessment
Authorised BULATS Agent
Agent BULATS agréé

FR85
FR855



9.8 TOEIC blancs réalisés en formation

Deux TOEIC blanc ont été réalisé en formation. Le premier avant les cours d'anglais (910), le second après (935).

Etienne PICHERIT

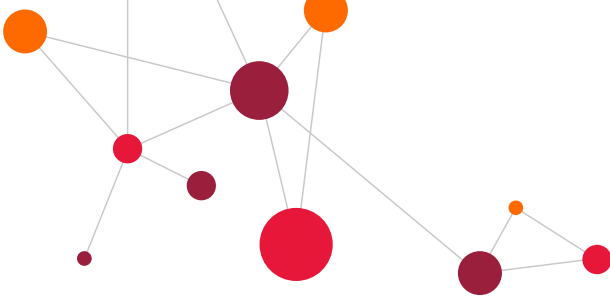
LISTENING SECTION																			
1	A	B	C	D	26	A	B	C	D	51	A	B	C	D	76	A	B	C	D
2	A	B	C	D	27	A	B	C	D	52	A	B	C	D	77	A	B	C	D
3	A	B	C	D	28	A	B	C	D	53	A	B	C	D	78	A	B	C	D
4	A	B	C	D	29	A	B	C	D	54	A	B	C	D	79	A	B	C	D
5	A	B	C	D	30	A	B	C	D	55	A	B	C	D	80	A	B	C	D
6	A	B	C	D	31	A	B	C	D	56	A	B	C	D	81	A	B	C	D
7	A	B	C	D	32	A	B	C	D	57	A	B	C	D	82	A	B	C	D
8	A	B	C	D	33	A	B	C	D	58	A	B	C	D	83	A	B	C	D
9	A	B	C	D	34	A	B	C	D	59	A	B	C	D	84	A	B	C	D
10	A	B	C	D	35	A	B	C	D	60	A	B	C	D	85	A	B	C	D
11	A	B	C	D	36	A	B	C	D	61	A	B	C	D	86	A	B	C	D
12	A	B	C	D	37	A	B	C	D	62	A	B	C	D	87	A	B	C	D
13	A	B	C	D	38	A	B	C	D	63	A	B	C	D	88	A	B	C	D
14	A	B	C	D	39	A	B	C	D	64	A	B	C	D	89	A	B	C	D
15	A	B	C	D	40	A	B	C	D	65	A	B	C	D	90	A	B	C	D
16	A	B	C	D	41	A	B	C	D	66	A	B	C	D	91	A	B	C	D
17	A	B	C	D	42	A	B	C	D	67	A	B	C	D	92	A	B	C	D
18	A	B	C	D	43	A	B	C	D	68	A	B	C	D	93	A	B	C	D
19	A	B	C	D	44	A	B	C	D	69	A	B	C	D	94	A	B	C	D
20	A	B	C	D	45	A	B	C	D	70	A	B	C	D	95	A	B	C	D
21	A	B	C	D	46	A	B	C	D	71	A	B	C	D	96	A	B	C	D
22	A	B	C	D	47	A	B	C	D	72	A	B	C	D	97	A	B	C	D
23	A	B	C	D	48	A	B	C	D	73	A	B	C	D	98	A	B	C	D
24	A	B	C	D	49	A	B	C	D	74	A	B	C	D	99	A	B	C	D
25	A	B	C	D	50	A	B	C	D	75	A	B	C	D	100	A	B	C	D

10

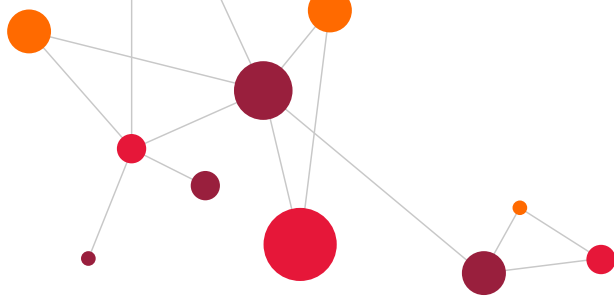
READING SECTION																			
101	A	B	C	D	126	A	B	C	D	151	A	B	C	D	176	A	B	C	D
102	A	B	C	D	127	A	B	C	D	152	A	B	C	D	177	A	B	C	D
103	A	B	C	D	128	A	B	C	D	153	A	B	C	D	178	A	B	C	D
104	A	B	C	D	129	A	B	C	D	154	A	B	C	D	179	A	B	C	D
105	A	B	C	D	130	A	B	C	D	155	A	B	C	D	180	A	B	C	D
106	A	B	C	D	131	A	B	C	D	156	A	B	C	D	181	A	B	C	D
107	A	B	C	D	132	A	B	C	D	157	A	B	C	D	182	A	B	C	D
108	A	B	C	D	133	A	B	C	D	158	A	B	C	D	183	A	B	C	D
109	A	B	C	D	134	A	B	C	D	159	A	B	C	D	184	A	B	C	D
110	A	B	C	D	135	A	B	C	D	160	A	B	C	D	185	A	B	C	D
111	A	B	C	D	136	A	B	C	D	161	A	B	C	D	186	A	B	C	D
112	A	B	C	D	137	A	B	C	D	162	A	B	C	D	187	A	B	C	D
113	A	B	C	D	138	A	B	C	D	163	A	B	C	D	188	A	B	C	D
114	A	B	C	D	139	A	B	C	D	164	A	B	C	D	189	A	B	C	D
115	A	B	C	D	140	A	B	C	D	165	A	B	C	D	190	A	B	C	D
116	A	B	C	D	141	A	B	C	D	166	A	B	C	D	191	A	B	C	D
117	A	B	C	D	142	A	B	C	D	167	A	B	C	D	192	A	B	C	D
118	A	B	C	D	143	A	B	C	D	168	A	B	C	D	193	A	B	C	D
119	A	B	C	D	144	A	B	C	D	169	A	B	C	D	194	A	B	C	D
120	A	B	C	D	145	A	B	C	D	170	A	B	C	D	195	A	B	C	D
121	A	B	C	D	146	A	B	C	D	171	A	B	C	D	196	A	B	C	D
122	A	B	C	D	147	A	B	C	D	172	A	B	C	D	197	A	B	C	D
123	A	B	C	D	148	A	B	C	D	173	A	B	C	D	198	A	B	C	D
124	A	B	C	D	149	A	B	C	D	174	A	B	C	D	199	A	B	C	D
125	A	B	C	D	150	A	B	C	D	175	A	B	C	D	200	A	B	C	D

8

910



Anonymisation	ABS			
Anonymisation	840			
Anonymisation	575			
Anonymisation	750			
Anonymisation	965			
Anonymisation	650			
Anonymisation	445			
Anonymisation	925			
Anonymisation	780			
Anonymisation	960			
Anonymisation	745			
Anonymisation	ABS			
Anonymisation	835			
Anonymisation	875			
Anonymisation	ABS			
Anonymisation	840			
Anonymisation	ABS			
Anonymisation	ABS			
Anonymisation	795			
Anonymisation	ABS			
Anonymisation	910			
Anonymisation	705			
Anonymisation	620			
Anonymisation	950			
Anonymisation	960			
Anonymisation	795			
PICHERIT	935			
Anonymisation	780			
Anonymisation	615			



9.9 Attestation de succès à l'examen Conception et administration de bases de données

le cnam
Hauts-de-France

ANNEE UNIVERSITAIRE : 2017-2018
Amiens, le 15/10/2018

ATTESTATION DE SUCCES A L'EXAMEN

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE
CONSERVATOIRE NATIONAL DES ARTS ET METIERS
Hauts-de-France

M. PICHERIT-STEINBRUCKER Etienne
2 BIS AVENUE DU 11 NOVEMBRE 1918
33320 LE TAILLAN MEDOC


Numéro et intitulé de l'UE : NFE113 Conception et administration de bases de données
Crédits (ECTS) : 6
Second semestre

Délivrée à : N° : HDF427383
Nom : PICHERIT-STEINBRUCKER
Prénom : Etienne
Né le : 11/03/1994 à Courcouronnes (91)

NOTE OBTENUE : 13 / 20

POUR L'ADMINISTRATION GENERALE DU CONSERVATOIRE NATIONAL
DES ARTS ET METIERS ET PAR DELEGATION

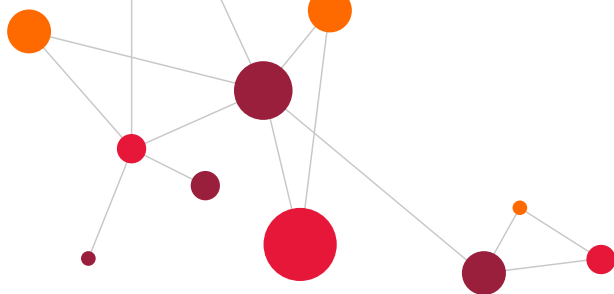
Le Directeur régional,
Claude VERGER



Cette attestation sanctionne le succès à un examen final ou aux épreuves de contrôle organisées en cours d'année (note égale ou supérieure à 10 / 20).

Le code de l'unité d'enseignement (3 lettres et 3 chiffres) correspond à la nomenclature utilisée par le Cnam dans le cadre de la constitution de ses filières et diplômes.

Chaque unité d'enseignement (UE) est affectée d'une valeur en crédits. Les crédits représentent l'unité de mesure du travail de l'auditeur pour l'UE concernée. Les crédits sont capitalisables dans les formations d'une même filière du Cnam. Ils sont aussi transférables dans un autre établissement d'enseignement supérieur, en France ou dans un autre pays de l'espace européen.



9.10 Curriculum Vitae



Étienne
Picherit-Steinbrucker
Développeur Informatique

2 bis, rue du 11 novembre 1918
33320 Le Taillan-Médoc
06.61.73.87.90
etienne.picherit@gmail.com

EXPÉRIENCES PROFESSIONNELLES

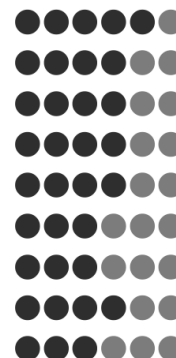
- 2018-2019**
Développeur Junior
À CGI, Le Haillan (33)
- Sept. 2017**
Magasinier
À Avi Orn Industries, Blanquefort (33)
- 2015-2017**
Apprentis Logisticien
À Avi Orn Industries, Blanquefort (33)
- Oct. 2015**
Ouvrier de montage
À Avi Orn Industries, Blanquefort (33)
- Été 2015**
Stagiaire en Développement Web
À Intespace (filiale Airbus), Toulouse (31)
- Sept. 2014**
Ouvrier de montage
À Avi Orn Industries, Blanquefort (33)

FORMATIONS


- 2018-2019**
Concepteur d'Application Numérique
À l'EPPI, Bordeaux (33)
- 2017-2018**
Licence informatique (Bac +3)
Au CNAM, Bordeaux (33)
- 2015-2017**
TSMEL (Bac +2)
À l'AFTRAL, Artigues-près-Bordeaux (33)
- 2015**
Développeur Logiciel (Bac +2)
À l'AFPA, Toulouse (31)
- 2012-2014**
Deux 1^{ère} année Médecine Vétérinaire
Université de Liège, Liège (Belgique)
- 2012**
Baccalauréat Scientifique
Lycée Gustave Eiffel, Dijon (21)

COMPÉTENCES

Java (JEE, Maven, Spring, Hibernate, Swing, Gradle, Android)
JavaScript/TypeScript (jQuery, Angular)
C# (ASP.NET MVC, Entity Framework, WinForm, ADO)
PHP (Symfony, PDO)
SQL (MariaDB/MySQL, Oracle, SQL Server, SQLite)
HTML/CSS (Bootstrap)
UML/Merise
Anglais technique
Management



9.11 Fiche d'évaluation stagiaire



**CERTIFICATION PROFESSIONNELLE
EVALUATION ENTREPRISE**

STAGIAIRE

Nom : PICHERET
Prénom : ETIENNE
Certification visée : C DAN

ENTREPRISE

Nom de la société :
Tuteur(trice) :
Courriel :

«tuteur Civilite»,
Afin de parfaire la validation de la certification professionnelle de votre stagiaire, nous vous remercions de remplir exhaustivement cet **original** et nous le retourner dans les *meilleurs délais*. Ce document sera examiné par le jury en vue de l'attribution de la certification professionnelle de votre stagiaire.


L'intégration dans l'équipe de travail est-elle satisfaisante ?	<input checked="" type="radio"/> Oui / <input type="radio"/> Non
Le comportement en situation professionnelle est-il adapté à la culture de l'entreprise ?	<input checked="" type="radio"/> Oui / <input type="radio"/> Non
Le comportement relationnel est-il adapté au métier visé ?	<input checked="" type="radio"/> Oui / <input type="radio"/> Non
La capacité de travail fournie correspond-elle au niveau d'un professionnel du métier ?	<input checked="" type="radio"/> Oui / <input type="radio"/> Non
Le niveau de responsabilités attendu est-il satisfait ?	<input checked="" type="radio"/> Oui / <input type="radio"/> Non
Des questions sont-elles posées à bon escient ?	<input checked="" type="radio"/> Oui / <input type="radio"/> Non
Des difficultés en relation avec le niveau de responsabilités sont-elles surmontées ?	<input checked="" type="radio"/> Oui / <input type="radio"/> Non
Le niveau d'obligation de réserve demandé est-il pris en compte ?	<input checked="" type="radio"/> Oui / <input type="radio"/> Non
Les activités effectuées concourent-elles à la couverture du champ de compétences ?	<input checked="" type="radio"/> Oui / <input type="radio"/> Non
Le métier associé à la certification visée est-il compris ?	<input checked="" type="radio"/> Oui / <input type="radio"/> Non
Le stagiaire est-il apte à occuper la fonction visée même comme débutant ?	<input checked="" type="radio"/> Oui / <input type="radio"/> Non

Appréciation générale suite au stage en entreprise :

Etienne a réussi son intégration à l'équipe par ses qualités humaines et professionnelles. C'est une personne de confiance, toujours force de proposition et capable pour prendre en charge un travail. Il est très agréable de travailler avec lui.

Fait à Boulogne le 19/09/2019

Signature du tuteur
et cachet de l'entreprise



Signature du stagiaire

Association ADP regie
par la loi du 1^{er} juillet 1901
Siret : 409 801 677 00017 Code
APE : 85.59 A
N° reglement : 11 75 27 97 775

Centre de formation
44 bis, quai de Babouville de 137 813 885 d
Paris
Siret social : 12, rue Alexandre Parodi 75010 Paris

CGI France
8 rue des Comètes
Bâtiment Andréas - CS 10028
92187 Le Haillan Cedex
Tél : +33 1 87 78 87 00
Fax : +33 1 87 78 87 01
www.cgi.fr

GROUPE IGS

© Groupe IGS 2019