
Titre du Rapport du stage

CYPRIEN PENNACHI

MENTION INTELLIGENCE ARTIFICIELLE
MASTER INFORMATIQUE

04/09/2023 - 05/04/2024

Tuteur(s) université :
MARIE TAHON
marie.tahon@univ-lemans.fr

Tuteur(s) entreprise :
GWENAËL GUEHENEUC
gwenael.GUEHENEUC@pole-
emploi.fr
JULIEN THIBAUT
julien.thibault@pole-emploi.fr



Résumé

Ce rapport se découpe en plusieurs parties, tout d'abord nous parlerons de la relation avec l'équipe métier. J'évoquerai l'importance de la communication et de l'échange interdisciplinaire dans le monde professionnel, souvent négligés dans les formations théoriques.

Dans un premier temps, j'aborderai l'organisation de l'entreprise puis la présentation du projet Autocarto.

Je continuerai ensuite sur les méthodes de travail qui sont employées dans le projet pour permettre une productivité.

Enfin, je terminerai sur les activités que j'ai réalisé dans le cadre des périodes d'alternance que j'ai faite.

Summary :

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Table des matières

1	Introduction	3
2	Présentation de l'entreprise	3
3	Présentation du projet	4
4	Déroulement du projet	5
5	Sujet de travail	6
5.1	Mise en place du poste de travail	6
5.2	mise en place de Pitest	8
5.3	Micro-service Administrateur	9
5.3.1	Création et paramétrage	9
5.3.2	Rapport sur Neo4j	10
5.3.3	Rapport de comparaison Neo4j et ELK	12
6	Annexe	13

1 Introduction

Dans le cadre de ma formation en Master Informatique Intelligence Artificielle à l'université, j'ai opté pour le parcours en alternance afin d'appliquer concrètement les compétences acquises durant mes années de Licence, ainsi que celles à acquérir lors de mon Master. De plus, ce parcours me permet de capitaliser sur une expérience professionnelle significative durant mes deux années de Master.

Au cœur de la transformation numérique des services publics en France, la Direction des Systèmes d'Information (DSI) de Pôle Emploi se positionne en tant qu'acteur essentiel dans la modernisation et l'efficacité des dispositifs d'accompagnement à l'emploi. Dans ce contexte, le projet Autocarto émerge tel un jalon crucial, visant à révolutionner la gestion et l'analyse des données géospatiales au sein de l'organisation.

En tant que collaborateur au sein de la DSI Pôle Emploi, j'ai eu le privilège d'être affecté au cœur de ce projet novateur. Au travers de cette mission, j'ai pu observer de près les enjeux, les défis et les opportunités qu'implique l'intégration d'un tel système dans un environnement aussi complexe que celui de Pôle Emploi.

Le but de ce rapport est de faire état du travail que j'ai réalisé durant ce semestre. J'y récontextualiserai mon projet, rappellerai les accomplissements du premier semestre, puis aborderai les problématiques rencontrées et les solutions envisagées pour les surmonter. Enfin, je partagerai ma vision future du projet.

2 Présentation de l'entreprise

L'entreprise Pôle Emploi/ France Travail compte pas moins de 60 000 employés dont 58 000 employés missionnés dans le cadre de l'emploi que ce soit la mise en ligne d'annonce, les relations avec les entreprises ou encore les entretiens avec les personnes sans emploi. Son pôle Systèmes d'informations, quant à lui, compte environ 1500 collaborateurs dont je fais parti.

L'organisation de la direction générale des systèmes informations de Pôle Emploi se fait sous deux angles :

- Le premier angle se fait sous forme de responsable hiérarchique.
- Le second angle se fait sous le forme de responsable de mission.

C'est-à-dire qu'un responsable hiérarchique N+1 ne travaille pas forcément sur la même mission que les collaborateurs dont il est en charge. Un responsable a une double casquette de développeur ou plus et de responsable de pôle de compétences.

positionner schéma hiérarchique et fonctionnel - voir intranet PE

Pour ma part, je suis dans l'équipe DSPC(Direction Sites & Pôles de Compétences) sous la repsonsabilité de Julien THIBAUT. Il est mon responsable hiérarchique et je gère toute la partie ressources humaines avec lui.

En ce qui concerne mon responsable fonctionnel, il s'agit de Franck BAUMGARTEN qui est lui même sous la repsonsabilité de Jean-Michel Cabrol. Je suis donc dans le pôle fonctionnel DPIT(Direction Projets & Ingénierie Technique).

3 Présentation du projet

Ce document a pour objet de présenter les différents diagramme et organigramme réalisée dans le cadre du projet du module "Conception et Développement logiciel" dédié à la mise en oeuvre du projet alternant.

Je suis développeur logiciel pour le SI de Pôle Emploi, nouvellement France Travail. Je travaille sur le projet AutoCarto. Le projet AutoCarto est un produit IT permettant de rechercher, restituer une vision cartographique des éléments du SI.

L'enjeu est donc pour Autocarto de collecter les données depuis un maximum de sources afin de devenir le produit référent permettant une vue graphe du SI de pôle Emploi en vue du partage d'informations dans le cadre de l'arrivée de France Travail. France Travail a pour projet de partager les informations avec d'autres services pour créer une continuité dans l'historique de travail des gens.

En plus d'être le produit permettant de cartographier le SI pôle emploi, il a aussi 2 autres objectifs :

- Evaluer et contrôler la fiabilité des données ou des informations
- Analyser et identifier les éléments impactés par un incident ou à traiter la dette.

La cartographie doit servir à tous au sein de la DSI et pour se faire l'autocarto automatise au maximum celle-ci en se servant de sources diverses pour reconstituer le SI. Malheureusement on ne peut pas automatiser à 100% et une partie des sources d'informations correspond à du déclaratif manuel (G2A, ...).

Ce projet a été réalisé par une équipe de 7 personnes :

- BAUGARTEN Frank - Réf. Tech
- CHLOE Geoffrey - Développeur
- CRUZ Bastien - Développeur
- GUEHENEUC Gwenaël - Développeur
- FAURE Sébastien - PO
- SENE Fabian - Développeur
- CHATEAU Ronan - Développeur.

Les services dont sont en charge l'équipe Autocarto se représente comme ceci :

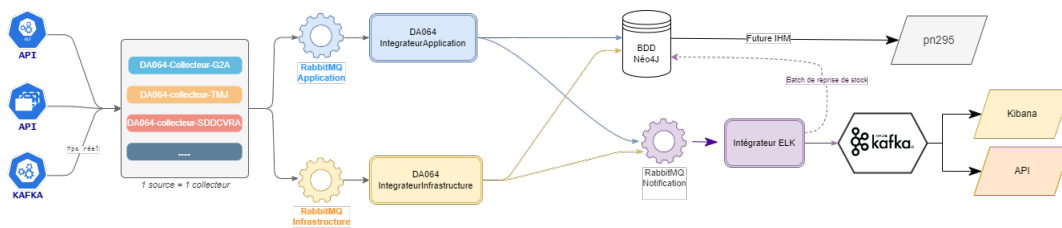


FIGURE 2 – Diagramme de fonctionnement de la collecte de données

Ce produit est basé sur une architecture logicielle dite "hexagonale" ce qui permet une isolation du noyau via des adaptateurs que nous appelons : "collecteurs" et "intégrateur" qui facilite l'ajout de fonctionnalités, leur maintenance, la traçabilité des sources. Ces informations sont ajoutés dans deux bases de données différentes : l'une sous une base de données graphe, et l'autre sur une base de données relationnel.

Le principe de l'architecture hexagonale est de ne pas lier la partie métier(JAVA pur) et les collecteurs et intégrateurs. Les données sont donc filtrés grace aux collecteurs pour en faire des objets métiers puis grâce aux intégrateur, ils sont enfin ajoutés aux BDD.

Le service permet de récupérer les informations de différentes origines. Il est ensuite filtrer par les collecteurs correspondant aux source des informations. Ces informations filtrés sont ensuite passée dans des files RabbitMQ pour les données infrastructures ou les données applicatives. Ces données sont ensuite reconverti avec les intégrateurs pour ajouter aux BDD. De mon côté, j'ai plus précisément travaillé sur le micro-service administrateur.

4 Déroulement du projet

Le projet se déroule sous forme de sprints durent 3 semaines.

Chaque début de sprint commence par une analyse et une estimation des tâches à venir.

Les tâches sont estimées grâce a une suite de Fibonacci.

Chaque nombre correspond à une charge de travail bien précise. Par exemple, le nombre '1' correspond à une charge d'une journée, '2' correspond à environ 2/3 jours. '5' correspond 2 semaines. Ces charges sont réparties selon les connaissances de chacun. Certaines tâches pour lequel personne ne connaît l'estimation sont reportés si elles ne sont pas urgentes en revanche si elles le sont elles sont assignés pour faire avancer la tache et pouvoir donner une estimation du travail.

Certaines tâches sont assignées directement à certains membres de l'équipe, en général avec une personne pour les seconder pour partager les connaissances sur le sujet en question.

Chaque matin, une mêlée est organisé ou l'équipe parle chacun son tour. Chaque

membre de l'équipe décrit les tâches réalisés le jour précédent en précisant les difficultés rencontrées et les tâches à faire pour la journée qui vient.

Enfin, à chaque fin de sprint, une réunion de débrief est organisée pour parler des sprints backlogs.

voir comment sont faites les estimations sous forme de taches

Du côté des idées proposées, on les ajoute sous forme de carte qui seront possiblement ajoutée ou non au prochain sprint. Ces cartes sont ajoutées sur Jira.

Il existe aussi une page Confluence sur tout ce qui concerne la documentation du projet. Cette documentation contient notamment quelques schémas UML qui montre l'organisation du projet sous la forme de diagramme de classe 8 ou encore de diagramme fonctionnel.

Le projet Autocarto utilise aussi GitLab dans le cadre du travail en équipe. Chacun travaille soit seul sur un sujet soit avec un collègue pour faire un partage de connaissance grâce au pair-programming. Dans mon cas, je fais beaucoup de pair-programming avec mon tuteur Gwenaël qui m'explique et m'accompagne sur les différentes tâches que j'ai à réaliser.

5 Sujet de travail

Le projet que j'avais à mettre en place depuis Septembre est un micro-service administrateur interne à Autocarto.

Il permet de relever des potentielles anomalies sur les différents autres micro-services pour prévenir et régler les problèmes avant que ces derniers partent en production. Il permet aussi d'analyser selon les besoins, les données pour savoir si les autres micro-services font le bon traitement en collecte comme en intégration dans les bases de données. Les données sont intégrés dans deux types de bases de données différentes ; l'une est une base de données graphe (Neo4j) et l'autre une base de données relationnelles (ELK). Je n'ai pas eu de ligne directrice au début au projet mais plutôt des tâches à réaliser au cours de chaque période d'alternance.

Ajouter Gantt à postériori des activités réaliser

5.1 Mise en place du poste de travail

Lors de la première période d'alternance, j'ai pu m'acclimater à la vie en entreprise et sur Nantes pendant les deux premières semaines. En effet, les premiers jours ont été dictés par la visite du site de la DSI Pôle Emploi de Nantes-Tardieu ainsi que la mise en place de mon poste de travail et d'explications du fonctionnement de

l'entreprise Pôle Emploi.

J'ai fait une formation pour les nouveaux arrivants qui se nomme Ariane. Cette formation a été faite fin octobre sur Montreuil où se situe la majorité des responsables de la DSI de France Travail. J'ai pu voir ou revoir comment s'organise les systèmes d'informations ainsi que l'organisation administrative des agents. Une plus large explication du règlement intérieure a été faite autour des horaires, du salaire ou des avantages de l'entreprise.

Ces deux jours ont été rythmé par les intervenants qui nous présentait tout les outils administratif de Pôle Emploi tel que le site de déclarations d'heures, le site de déclarations de jours de congés mais aussi les demandes de formations.

En ce qui concerne les formations, les intervenants ont beaucoup insistés sur le fait que nous devons faire des formations. En effet, la plupart des agents de la DSI de France Travail ne pensent pas à faire des jours de formation.

Nous avons eu un intervenant qui nous a parlé de la sécurité. C'était centré sur les cyberattaques ainsi que les mesures de sécurité au sein du site. Nous avons eu une aparté sur les mesures de 1er secours et d'évacuation en cas d'incendie.

Enfin, dans le cadre de l'engagement de France Travail pour l'égalité homme-femme, nous avons eu droit à une formation sur les gestes et conversation qui peuvent poser ou non. Personnellement, j'ai trouvé très intéressante cette formation et m'a permis de m'intégrer plus vite.

J'ai pu apprendre à utiliser les différents outils utilisés pour la documentation notamment Confluence ou toute la documentation fonctionnelle ainsi que technique est renseignée. J'ai aussi fait la découverte de Jira, qui permet un suivi des tâches réalisées par l'équipe.

D'autres outils comme SonarQube ou Concourse qui permettent respectivement de vérifier les bugs, la couverture de tests ainsi que la taux de code dupliqué tandis que l'autre permet de vérifier l'état de création de chaque micro-service pour la mise en fabrication, puis pré-production et enfin production.

Du point de vue du développement logiciel, mon tuteur Gewnaël m'a chaudement conseillé Eclipse comme framework. J'avais aussi la possibilité de programmer sur IntelliJ mais la connaissance d'Eclipse de mon tuteur m'a permis de m'intégrer au travail plus vite.

Enfin, j'utilise Bruno pour les requêtes qui sont à réaliser sur la base de données ELK. Pour ce qui est de Neo4j, un environnement et une interface propre à la base qui est en ligne. En parrallele, j'ai revu avec mon tuteur les bonnes pratiques de programmation ainsi que des concepts de programmation. Les principes qu'il m'a expliqué et que j'ai appris ou revu sont :

- IOC,
- approche hexagonale,
- SOLID,
- DRY,

— KISS

5.2 mise en place de Pitest

Ma première mission au sein du projet est de mettre en place un module Pitest. Pitest est un module de modification de code. Il vérifie la solidité des tests en créant une version mutée. Il change alors les tests, les méthodes, les instances pour cela. Les tests, les méthodes et les variables d'instances sont changés selon les types de tests choisies. Le module pitest permet de vérifier la robustesse des tests unitaires. Pour le faire fonctionner, on ajoute la dépendance pitest dans le fichier de configuration "pom.xml" du micro service.

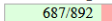
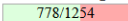
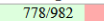
```
<dependency>
  <groupId>org.pitest</groupId>
  <artifactId>pitest-parent</artifactId>
  <version>1.15.0</version>
  <type>pom</type>
  <scope>test</scope>
</dependency>
```

FIGURE 3 – dépendance du module Pitest

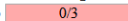
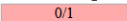
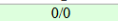
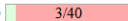
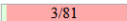
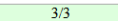
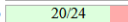
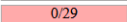
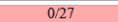
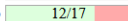
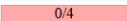
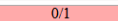
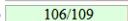
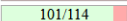
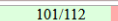
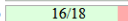
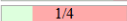
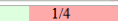
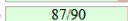
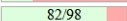
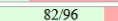
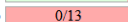
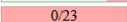
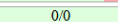
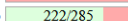
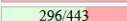
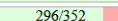
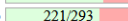
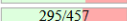
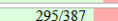
Pour faire en sorte que Pitest ne se lance à chaque fois que le projet compile, un profil a été créé pour pouvoir faire fonctionner Pitest. fichier de configuration "pom.xml" du micro service⁷. La création du pipeline Pitest sur un microservice se fait sur une branche GitLab nommée *concourse* qui permet de paramétrer les micro-services ainsi que les pipelines que chacun contient. Un fichier YML permet d'ajouter le pipeline Pitest qui est directement lié au module. Ce pipeline a d'abord été créé sur un micro-service avant d'être copié sur les autres.

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
26	77% 	62% 	79% 

Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage	Test Strength
fr.pe.gssi.service.da064.collecteur.aae	1	0% 	0% 	100% 
fr.pe.gssi.service.da064.collecteur.aae.administration	2	8% 	4% 	100% 
fr.pe.gssi.service.da064.collecteur.aae.config	1	83% 	0% 	0% 
fr.pe.gssi.service.da064.collecteur.aae.donnee.fichier	2	71% 	0% 	0% 
fr.pe.gssi.service.da064.collecteur.aae.donnee.fichier.deserializer	5	97% 	89% 	90% 
fr.pe.gssi.service.da064.collecteur.aae.donnee.message	2	89% 	25% 	25% 
fr.pe.gssi.service.da064.collecteur.aae.donnee.message.deserializer	4	97% 	84% 	85% 
fr.pe.gssi.service.da064.collecteur.aae.persistance.neo4j	1	0% 	0% 	100% 
fr.pe.gssi.service.da064.collecteur.aae.service.rabbitmq.fichier	4	78% 	67% 	84% 
fr.pe.gssi.service.da064.collecteur.aae.service.rabbitmq.message	4	75% 	65% 	76% 

Report generated by [PIT](#) 1.15.0

Enhanced functionality available at [arcmutate.com](#)

FIGURE 4 – Rapport de robustesse Pitest

Enfin, j'ai pu faire une présentation à toute l'équipe AutoCarto en visio-conférence sur le sujet . Cette présentation a été appréciée et comprise par tous. Néanmoins, et c'est une remarque de mon tuteur, une répétition avant la présentation aurait été apprécié de mon côté car je ne m'étais pas bien remémoré les notions vues précédemment.

5.3 Micro-service Administrateur

5.3.1 Création et paramétrage

Durant le mois de Octobre, j'ai pu mettre en place accompagné de l'expertise de Gwenaël, le micro-service administrateur. J'ai pu apprendre et manipuler les fichiers de paramétrage d'un micro-service.

Le pipeline sur concourse a été créé pour permettre le bon fonctionnement du micro-service dans sa phase de fabrication. J'ai donc plus ou moins copié les fichiers d'un micro-services en fonctionnement pour ajouter ou retirer les modules dont on ne se sert pas.

Du côté de la mise en place coté Eclipse, les projets sont créés à partir de Maven, le MS (micro-service) Administrateur ne fait donc pas exception. Il a fallu ajouter un fichier XML avec les dépendances nécessaire pour son bon fonctionnement.

Une fois qu'il est configuré comme il faut, j'ai pu apprendre le mode de programmation hexagonal. Ce mode de programmation permet de séparer le code Java pur qu'on appelle métier, de la partie API, mais aussi de la partie "appel aux base de données". L'avantage de ce genre de pratique de code est que lorsqu'on modifie le code du côté Rest ou du côté API, cela n'affecte pas toute la chaîne.

Voici un schéma d'un appel API au micro-service administrateur :

Mettre le schéma avec Appel API qui appelle créateurRapport qui lui appelle accès externe qui appelle une BDD qui retourne des infos qui sont mappés qui sont retournés au créateurRapport qui sont affichés sur une page Web ou non selon le mode.

5.3.2 Rapport sur Neo4j

En ce qui concerne les rapports, il y a des modes de reception. L'un est le mode JSON qui est l'un des plus classiques. Il renvoie les informations sous la même forme que les objets Java qu'il est censés renvoyer. Il existe d'autres mode de renvoi des informations :

- le mode 'screen' qui permet de renvoyer les informations sur la page
- le mode 'file' qui renvoie le rapport sur un fichier CSV.

Le mode par défaut est généralement le mode Json. Le premier rapport que j'ai fait a été le rapport 'NC'. Ce rapport a été créé pour répondre à une demande du Product Owner, Sébastien. Le but de ce rapport est de vérifier que sur les Instances de Composant Techniques(ICT), les id ne contiennent pas 'NC'. Cette demande a été faite suite à un soucis remarqué par le PO sur la base Neo4j. Certains identifiants contiennent 'NC' sans raison connu au préalable. Le fait est que la difficulté de ce rapport était de tous les trouver car certains identifiants contiennent 'NC' pendant que d'autres contiennent 'nc'. Certains identifiants ont le code 'nc' dans la source pendant que d'autres l'ont dans l'identifiant métier. Les id sont formatés de telle manière, Source-nom-métier.

Nous avons donc modifier la requête pour que soit pris en compte cet spécificité. J'ai donc pu utiliser et faire des requêtes sur la base Neo4j grâce au langage Cypher. Voici la requête Cypher pour récupérer les éléments de la base Neo4j :

```
@Query("""
MATCH (n)
WHERE n.idMetier=~"(?i)nc.*" OR n.idSource=~"(?i)nc.*" OR n.id=~"(?i)nc.*" AND NOT(n.id contains("NCP"))
RETURN
  {properties: {}} AS noeud,
  n.id AS id_nc,
  labels(n)[0] AS type,
  n.origine AS origine,
  n.idSource AS idSource,
  n.idMetier AS idMetier,
  n.horodatageCreation as horodatageCreation,
  n.horodatageDernierAcces as horodatageDernierAcces,
  n.horodatageModification as horodatageModification """)
List<NoeudNeo4j> listerNC();
```

FIGURE 5 – requête Neo4j

Plusieurs rapports ont été fait comme le rapport racine sans contenu, le rapport

orphelin ou les rapports infoBase qui repertorient tout les types de noeuds.

L'un des derniers rapports que j'ai fait à la demande du product Owner est le rapport doublon.

Ce rapport consiste à trouver dans la base neo4j les noeuds qui ont le même idSource. Ce rapport fait suite au fait que le PO ait trouver des doublons dans les noeuds de "fluxContenu" dans la base neo4j. L'objectif a donc été de résoudre ce soucis pour voir ce qui pose problème dans les collecteurs d'AutoCarto, ou alors les intégrateurs.

Nous avons donc commencé nos recherches avec Gwenaël dans ce sens. Nous avons commencé par faire des requetes sur la base pour trouver plusieurs éléments qui ont le même "idSource". Une fois trouver, nous avons restreint les possibilités car sur l'environnement de fabrication les identifiants qui contiennent "autocartoTest" sont nombreux car il s'agit de l'environnement de la base ou l'on fait les tests d'intégration. Pour ce faire, nous avons donc fait une requete qui ne contient pas dans son "idSource" les éléments de Test.

Nous avons ensuite remarqué qu'entre un noeud FluxContenu et un "ICTContenu", il était possible que l'idSource soit le même. Nous avons modifié la requete pour aller dans ce sens.

Nous sommes finalement tombé sur une requete qui :

- ne doit pas contenir "autocartotest" dans son idSource
- doit comparer deux éléments qui ont le même type
- doivent avoir le même idSource

.

requete neo4j

Finalement, nous avons les noeuds qui sont considéré comme doublons. Le traitement est simple, les rendre unique pour voir en tout combien de noeuds sont le double d'un autre noeud. Nous sommes tombés sur 4000 éléments en double au premier abord ce qui est énorme. J'ai ensuite fait un différent traitement lors de la comparaison. J'avais d'abord utilisé une HashMap qui permet de regrouper les doubles entre eux mais ils n'étaient pas unique. J'ai donc changé cela suite au conseil de mon tuteur qui a suggéré d'utiliser un HashSet.

Le HashSet a le même principe de fonctionnement que la HashMap mais ne prend qu'un exemplaire d'un élément ce qui rend le traitement bien plus facile. J'ai donc ensuite refait cela pour tomber à environ 3000 éléments en double sur tout les Contenu que j'avais à disposition.

Enfin, après avoir montré au PO le résultat de ce rapport, il m'a demandé de trier les noeuds doublons selon leur collecteurs d'origine pour potentiellement en tirer des conclusions. Cela a été fait par la suite.

5.3.3 Rapport de comparaison Neo4j et ELK

La complexité des rapports est montée d'un cran ensuite avec le développement de rapport de comparaison entre base neo4j et elk. Le principe est donc de comparer les informations entre les deux bases pour vérifier si les informations concordent. Sur demande du PO, le premier rapport a été le rapport de comparaison des instances de composant techniques. Celui-ci a pour but de comparer en préproduction les ICT entre elles car il y avait des incohérences. Sur la période des vacances de fin d'année, nous avons mis en place le rapport en comparant :

- l'idSource
- l'idMetier
- et le nom du noeud

Après cela, nous avons encore des problèmes car certains noeuds n'avaient pas le même Id alors que les éléments de comparaison étaient les mêmes. En effet, certains identifiants ont une option telle que 'tp' ou 'td' (je ne sais pas pourquoi) qui les rend différents (surement le fait qu'ils soient créés en début, milieu et fin du traitement dans la base).

idSource	nom du noeud	idMetier1	identifiant
source1	flux1	metier1	source1-flux1-metier1
source1	flux1	metier1	source1-flux1-tp-metier1

FIGURE 6 – comparaison Neo4j/ ELK

Suite à mon retour sur Nantes, Gwenaél ainsi que Sébastien(PO) ont parlé et la structure d'intégration aux bases a changé. Un champ Id a été créé dans la base ELK pour faciliter la comparaison entre les bases.

J'ai donc modifié ma requête en ne comparant plus que le identifiant des noeuds entre eux. Le souci est donc réglé pour ce rapport. Ce fut la même chose pour les rapports de comparaison de flux entre les deux bases.

Un dernier rapport a lui été créé pour comparer les noeuds Flux qui sont dans l'environnement de préproduction et celui de production dans la base ELK exclusivement. Je le mets dans cette partie car il s'agit du même type de traitement que les deux autres rapports précédemment cités.

6 Annexe

```
<profile>
  <id>pitest</id>
  <build>
    <plugins>
      <plugin>
        <groupId>org.pitest</groupId>
        <artifactId>pitest-maven</artifactId>
        <version>1.15.0</version>
        <configuration>
          <targetClasses>
            <param>fr.pe.gssi.service.da064.collecteur.aae*</param>
          </targetClasses>
          <targetTests>
            <param>fr.pe.gssi.service.da064.collecteur.aae*</param>
          </targetTests>
          <mutators>
            <mutator>STRONGER</mutator>
            <mutator>NON_VOID_METHOD_CALLS</mutator>
            <mutator>INLINE_CONSTS</mutator>
            <mutator>REMOVE_CONDITIONALS </mutator>
          </mutators>
        </configuration>
      </plugin>
    </plugins>
  </build>
</profile>
```

FIGURE 7 – Profil Pitest

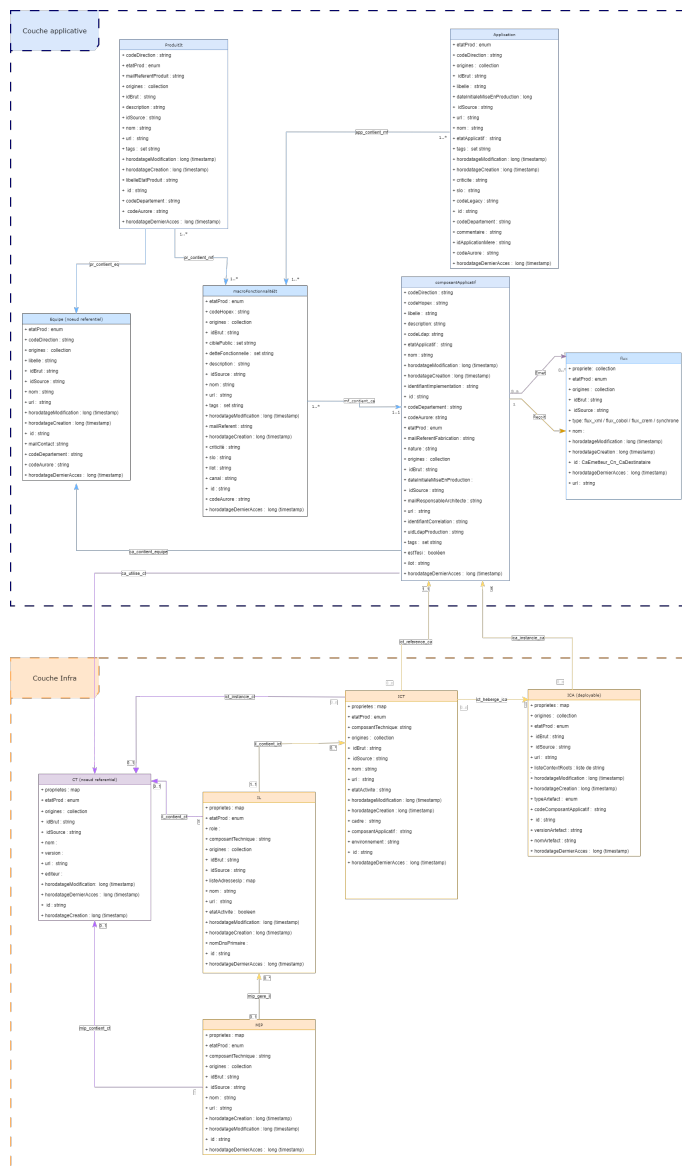


FIGURE 8 – Diagramme de classe des bases