



Rapport de Stage d'Ingénieur

---

# Développement d'un Outil Web de Gestion des Versions des Applications

---

*Auteur :*

M. Guoxin SUI

*Encadrants :*

M. Jean-Michel SIZUN

*Responsable d'Option :*

M. Jean-Yves MARTIN

*Référent Technique :*

M. Jérémie RECHET

23 juillet 2017

# REMERCIEMENTS

Je tiens d'abord à remercier l'entreprise VSCT de m'avoir accueilli pour ces 20 semaines de stage et de m'avoir offert la possibilité de découvrir le domaine de l'informatique avec un sujet assez intéressant.

Je souhaite particulièrement remercier mon tuteur de stage, Jean-Michel SIZUN, spécialiste d'exploitation, de m'avoir accordée ses conseils et ses soutiens qui ont mené à bien une grande partie de mes missions de stage. J'ai eu la chance de profiter de son expérience pour l'avancement de mon projet et d'avoir un rôle de développeur dans l'équipe. Grâce aussi à sa confiance, j'ai pu m'accomplir totalement dans mes missions.

Simultanément, je tiens aussi à remercier Jérémy RECHET, référent technique expérimenté attachée à notre équipe, pour le temps passé ensemble et le partage de son expertise au quotidien. Il m'a formé et m'a prêté sa main en suivant le projet en même temps. Il fut d'une aide précieuse dans les moments les plus délicats.

Pour le projet Dn-Rider, je remercie par ailleurs Jean-Marie BERCEGEAY, Jean-François le DENMAT, Emeric MARTINEAU, pour leur disponibilité de me donner des renseignements et m'accompagner lorsque j'en ai besoin.

Enfin, mes remerciements s'adressent également à l'ensemble du personnel de l'entreprise pour leur accueil, leur soutien afin de m'intégrer plus rapidement dans l'environnement de travail.

# Table des matières

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                     | <b>4</b>  |
| <b>2</b> | <b>Contexte</b>                         | <b>5</b>  |
| 2.1      | SNCF & VSC Technologies . . . . .       | 5         |
| 2.1.1    | Le Groupe SNCF . . . . .                | 5         |
| 2.1.2    | Le Groupe VSC & Rail Europe . . . . .   | 7         |
| 2.2      | Usine Logicielle & Katana . . . . .     | 11        |
| 2.2.1    | Usine Logicielle . . . . .              | 11        |
| 2.2.2    | Katana . . . . .                        | 13        |
| <b>3</b> | <b>Projet DN-Rider</b>                  | <b>15</b> |
| 3.1      | Constat & Objectif du stage . . . . .   | 15        |
| 3.2      | Contexte & Démarche . . . . .           | 17        |
| 3.2.1    | Contexte . . . . .                      | 17        |
| 3.2.2    | Démarche . . . . .                      | 17        |
| 3.3      | Application . . . . .                   | 19        |
| 3.3.1    | Architecture de l'application . . . . . | 19        |
| 3.3.2    | Choix des technologies . . . . .        | 20        |

---

|          |  |           |
|----------|--|-----------|
| 3.3.3    | IHM . . . . .                              | 21        |
| 3.3.4    | Api REST . . . . .                         | 25        |
| 3.3.5    | Intégration Continue . . . . .             | 26        |
| 3.3.6    | Test . . . . .                             | 26        |
| 3.4      | Etat fin de stage . . . . .                | 29        |
| <b>4</b> | <b>Conclusion</b>                          | <b>30</b> |
| 4.1      | Bilan personnel . . . . .                  | 30        |
| 4.2      | Bilan technique . . . . .                  | 30        |
| 4.3      | Bilan organisation et académique . . . . . | 31        |
| <b>5</b> | <b>Lexique</b>                             | <b>32</b> |
| 5.1      | Lexique Général . . . . .                  | 32        |
| 5.2      | Lexique de VSC Technologies . . . . .      | 33        |

# 1. INTRODUCTION

Dans le cadre de ma deuxième année à l'Ecole Centrale de Nantes en suivant l'option informatique, j'ai eu la chance d'effectuer un stage d'ingénieur de 20 semaines au sein de VSC Technologies.

En accompagnant de mon tuteur et d'un référent technique, j'ai développé une application web pour le projet "DN-Rider".

Le développement est organisé en méthode agiles, on essaie de pratiquer les principes de SCRUM.

L'application contient des parties IHM, API REST, documentation et intégration continue. Un série de frameworks et langages ont été utilisés : Grails, Groovy, Bootstrap, jQuery, jQueryUI, HTML, CSS et JavaScript.

Ce rapport apporte le déroulement de ce stage, les connaissances et résultats obtenus au cours de mon stage.

## 2. CONTEXTE

### 2.1 SNCF & VSC Technologies

#### 2.1.1 Le Groupe SNCF

##### Présentation Général

La Société nationale des chemins de fer français (SNCF) est l'entreprise ferroviaire publique française, officiellement créée le 1er janvier 1938 en application du décret-loi du 31 août 1937. Elle est notamment présente dans les domaines du transport de voyageurs, du transport de marchandises et réalise la gestion, l'exploitation et la maintenance du réseau ferré national dont elle est propriétaire.

La SNCF est donc une entreprise ferroviaire « intégrée » : elle exerce à la fois le métier d'exploitant (voyageurs et marchandises) et celui de gestionnaire d'infrastructure ferroviaire.

La Société nationale des chemins de fer français est devenue un établissement public à caractère industriel et commercial en 1983, alors qu'elle était auparavant une société anonyme d'économie mixte.

En 2015, le réseau ferré national propriété de SNCF Réseau compte environ 30,000 kilomètres de lignes dont 15 687 km de lignes électrifiées et 2 024 km de lignes à grande vitesse. La même année, les effectifs des trois EPICs étaient de 149 500 salariés.

Chaque jour, elle fait circuler 15 000 trains de fret et de voyageurs et transporte plus de cinq millions de voyageurs<sup>8</sup>. Par son volume d'activité et la taille de son réseau, c'est la troisième entreprise ferroviaire européenne, après la Deutsche Bahn et les chemins de fer russes.

Elle détient des participations majoritaires ou totales dans des sociétés de droit privé regroupées dans le groupe SNCF et dont la tutelle de l'État est exercée par la Direction générale des infrastructures, des transports et de la mer du ministère de l'écologie, du développement durable et de l'énergie.<sup>9</sup> Le siège social de la SNCF se trouve à La Plaine St-Denis, 2 place aux étoiles, à côté de la gare Stade de France Saint-Denis du RER D.

En 2014, la SNCF a enregistré un résultat net de 605 millions d'euros (contre une perte nette de 180 millions d'euros en 2013). L'EPIC SNCF Mobilités a réalisé un chiffre d'affaires de 27,2 milliards d'euros.

Le reste du groupe SNCF, qui a réalisé 7,1 milliards d'euros de chiffre d'affaires en novembre 2009, intervient dans les domaines suivants : logistique et transport routier de marchandises, transport routier de voyageurs (Keolis), liaison maritime (SeaFrance), ingénierie (EFFIA, INEXIA, commerce en ligne (Voyages-sncf.com), billet-tique (RITMx). Le groupe possède aussi des participations dans des sociétés ferroviaires et gestionnaires d'infrastructure portuaire partagées avec d'autres partenaires comme Eurostar, Thalys, Elipsos, Lyria et Nuovo Trasporto Viaggiatori.

## Organisation

Depuis le 1er janvier 2015, la SNCF est constituée de trois établissements publics à caractère industriel et commercial (EPIC, lexique.5.1) (fig.2.1) :

- un ÉPIC SNCF, qui prend en charge le pilotage global du groupe ;
- un ÉPIC SNCF Réseau, qui gère, exploite et développe le réseau ferré français ;
- un ÉPIC SNCF Mobilités, pour le transport de voyageurs et de marchandises.

et cinq « métiers » :

- SNCF Réseau ;
- SNCF Voyageurs ;
- SNCF Logistics ;
- SNCF Immobilier ;
- SNCF Keolis.

Elle possède aussi de nombreuses filiales aussi bien de droit public que de droit privé qui forment le groupe SNCF.

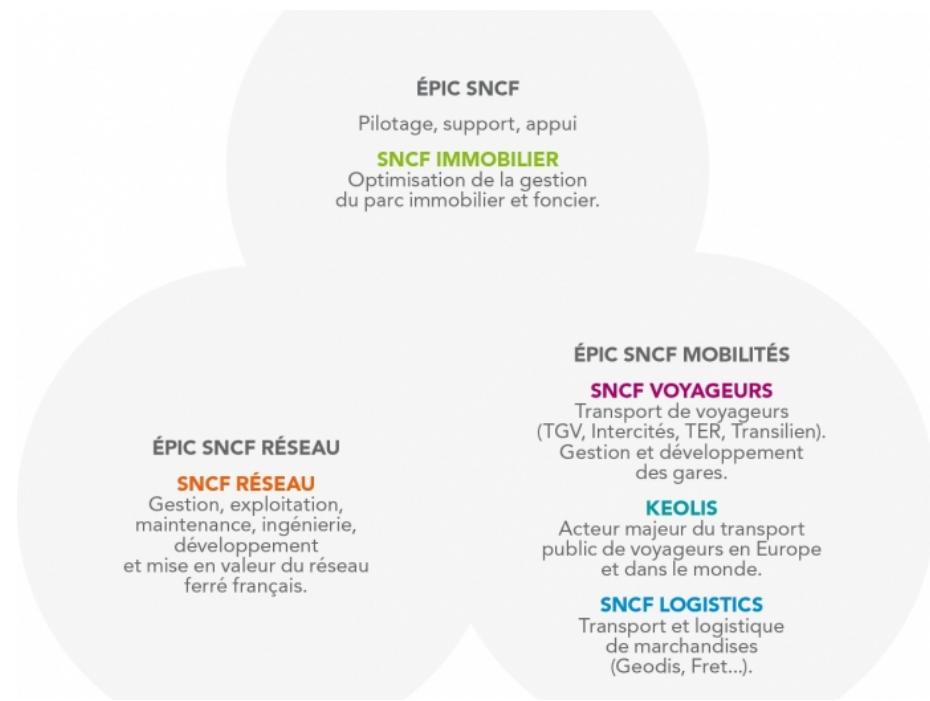


FIGURE 2.1 – Epic SNCF

### 2.1.2 Le Groupe VSC & Rail Europe

#### Présentation Général

Le Groupe VSC et Rail Europe, filiale du Groupe SNCF qui se situe dans l'EPIC "SNCF Mobilités", dirigée par Franck Gervais, est un acteur majeur du tourisme, expert de la distribution du train mais aussi de la vente des billets d'avions, de séjours, location de voitures et chambres d'hôtel, en France et en Europe. En 2015, son volume d'affaires atteint 4,1 milliards d'euros, en recul de 1,4% par rapport à 2015 en vendant 86 millions de voyages, en croissance de 4,4%. L'innovation demeure un axe central et exprime la capacité de Voyages-sncf.com à répondre aux nouveaux usages de ses clients. Aujourd'hui, en France, Voyages-sncf.com est le premier site d'e-commerce et la première agence de voyages en ligne ; le groupe rassemble 1200 collaborateurs dans le monde dont 40% à l'international (130 en Europe et 350 hors Europe).

En 2000, le site internet Voyages-sncf.com est lancé sur le périmètre France. L'entreprise est à l'époque le distributeur unique des billets de train SNCF, et a pour objectif de transformer son site en portail de voyages offrant des produits et services complémentaires au train. L'année suivante, Voyages-sncf.com forme une joint-venture avec

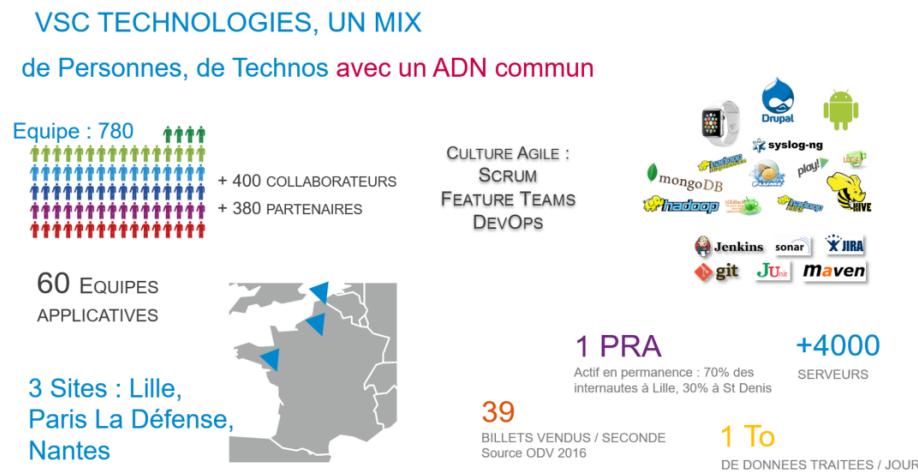


FIGURE 2.2 – VSC Technologies : Un mix

l'américain Expedia et devient une agence de voyage globale. L'entreprise poursuit alors son développement en France, tout en nourrissant une ambition internationale.

Pour répondre aux enjeux de la distribution du voyage et aux nouveaux comportements d'achats, le groupe VSC offre à ses clients mondiaux un réseau puissant, souple et adapté à leurs besoins. Il couvre plus présent dans 11 pays européens et 45 dans le reste du monde via un total de 67 sites internet et mobiles, 4 boutiques et un service de call-center. Afin de répondre aux enjeux spécifiques du marché B2B, le site Voyages-sncf.eu a été lancé en Europe en 12 langues (hors France). Le site recense plusieurs transporteurs tels que SNCF, TER, Eurostar, Thalys, TGV Lyria ; 3 compagnies de bus, 400 compagnies aériennes ; 280 000 hôtels référencés ; plus de 25 000 offres de séjours ; 30 loueurs de voitures, etc.

Le groupe va lancer la nouvelle marque OUI.sncf le 7 décembre 2017. Ce changement fera évoluer la plateforme transactionnelle.

## Historique

L'histoire de Voyages-sncf.com se confond avec celle du digital français et de l'e-commerce. Une histoire faite de révolutions technologiques, de bouleversements dans les usages, d'émergences (et de disparitions) de géants, qui font désormais partie intégrante de nos vies. Une histoire dont Voyages-sncf.com a été toujours acteur mais aussi souvent précurseur. Le fil rouge qui relie la start-up de 2000 au groupe international

d'aujourd'hui, c'est l'audace.

- Lorsque SNCF décide en 2000, contre tous les pronostics, de se lancer dans l'aventure internet, c'est de l'audace ;
- Quand Voyages-sncf.com fait le choix en 2006 d'afficher tous ses prix en TTC alors que le secteur le fait en HT, c'est de l'audace ;
- La décision de faire du mobile en 2009, c'est de l'audace ;
- Lorsque Voyages-sncf.com bascule sur 3DS pour sécuriser les paiements, alors qu'on lui promettait un effondrement de ses ventes, c'est de l'audace ;
- Lorsque Voyages-sncf.com part à la conquête du monde entier en 2013, c'est de l'audace.

## Organisation

L'organisation du Groupe VSC est basée sur 3 BU's : Trois BU's transport sur trois zones géographiques : France, Europe, et Overseas. L'outil industriel et nos grandes expertises sont mutualisées dans des équipes dédiées travaillant pour ces trois BU's : les Directions Produits, la Technologie et le Marketing. Enfin, des Directions corporate assistent l'ensemble.



FIGURE 2.3 – VSC : Organisation

Le Groupe VSC est la filiale de Distribution Digitale de Voyages SNCF. Voyages SNCF fait partie de Voyageurs, l'une des branches de l'EPIC SNCF Mobilités. Voyages SNCF assure le transport ferroviaire longue distance et à grande vitesse de plus de 130 millions de personnes par an dans toute l'Europe, à bord des 500 rames TGV, iDTGV, Ouigo, Eurostar, Thalys, Lyria et Ellipsos. Ouibus fait également partie de Voyages SNCF.

**VSC Technologies**

VSC Technologies, entité du Groupe Voyages-sncf.com, est en charge de la partie informatique du premier site public de e-commerce français. VSC Technologies propose des solutions informatiques dans un logique éditeur, développe des offres sur mesure pour répondre aux besoins de ses clients (SNCF, Eurostar, IDTGV...) Ils chargent de la gestion des outils front office, la maintenance et l'évolution du site Web, hébergement,

## 2.2 Usine Logicielle & Katana

### 2.2.1 Usine Logicielle

L’usine logicielle est l’ensemble de logiciels, d’outils et de procédures qui permettent de structurer et d’industrialiser les développements VSCT, ainsi que leur validation et déploiement sur l’infrastructure VSCT (hors VSCloud).

Dans sa définition actuelle : elle est la continuation de la nouvelle usine logicielle Lille et de Katana.

### USINE LOGICIELLE : DÉVELOPPEMENT

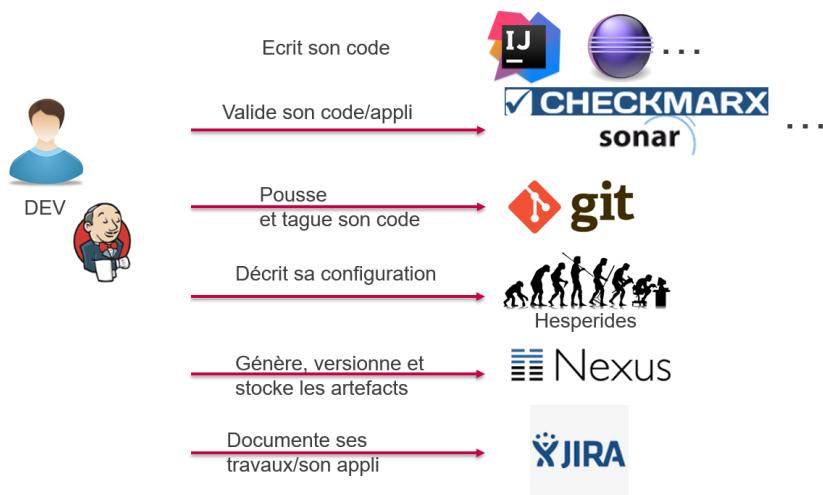


FIGURE 2.4 – Usine Logicielle : Développement

Le développeur édite ses sources sur son poste de travail et les versionne sous GIT, avec entreposage dans une forge GITLAB .

L’application fait elle-même l’objet de versions formalisées, avec les livrables individuels packagés et archivés sur un entrepôt de binaires (NEXUS).

Pour chaque application, plusieurs plateformes sont montées sur l’infrastructure pour installer l’application.

Chaque plateforme a un rôle défini. Les plateformes de production font l’objet d’une gestion spécifique, le plus souvent pilotée par les équipes d’exploitation.

Le développeur dispose d'un outillage d'intégration/déploiement continue (JENKINS) pour piloter les compilations/générations, validations, déploiements sur des plate-forme intermédiaires... Pour certaines de ces étapes, JENKINS pilote des outils-tiers (FISHEYE, CRUCIBLE, CHECKMARX, SONAR, etc).

Les serveurs (virtuels ou non) des plateformes sont accessibles en ligne de commande SSH au travers d'un frontal WALLIX, en fonction des droits de chacun.

Un frontal web (RUNDECK) permet de mettre à disposition des utilisateurs des opérations de plus haut-niveau, en self-service web sur les serveurs/plateformes (exemples : déploiement applicatif, reconfiguration applicative, arrêt/relance et autres opérations spécifiques sur les serveurs).

## USINE LOGICIELLE : OPÉRATIONS

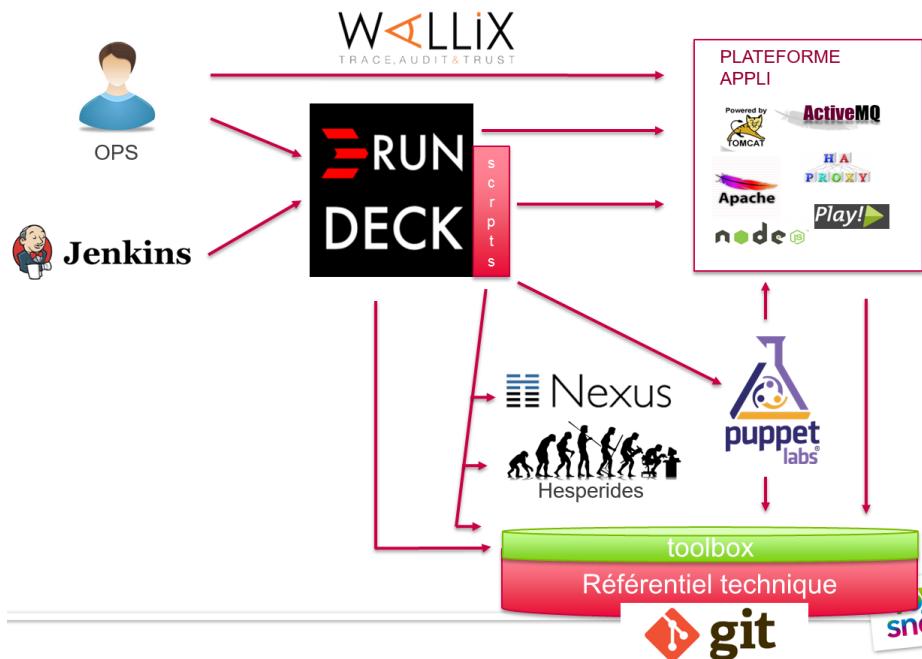


FIGURE 2.5 – Usine Logicielle : Opération

La configuration détaillée des différentes plateformes de l'application est industrialisée par :

- l'outil PUPPET (configuration technique - COTS) et un référentiel technique en infra-as-code ;
- l'outil HESPERIDES (configuration application - développement VSCT en open-source).

Sauf cas particulier, l'accès aux outils et aux plateformes se fait selon l'authentification Active Directory de l'utilisateur.

### 2.2.2 Katana

La solution Katana assure la configuration technique, le déploiement applicatif et autres tâches industrialisées sur les machines des infrastructures VSCT de Lille/St-Denis (DMZs hors-production, Assemblage, Perf, Partenaires, Rithmics, Technique, Production, SNB....), à l'exception notable des SIs du CNIT, de VSCloud et des infra Big Data.

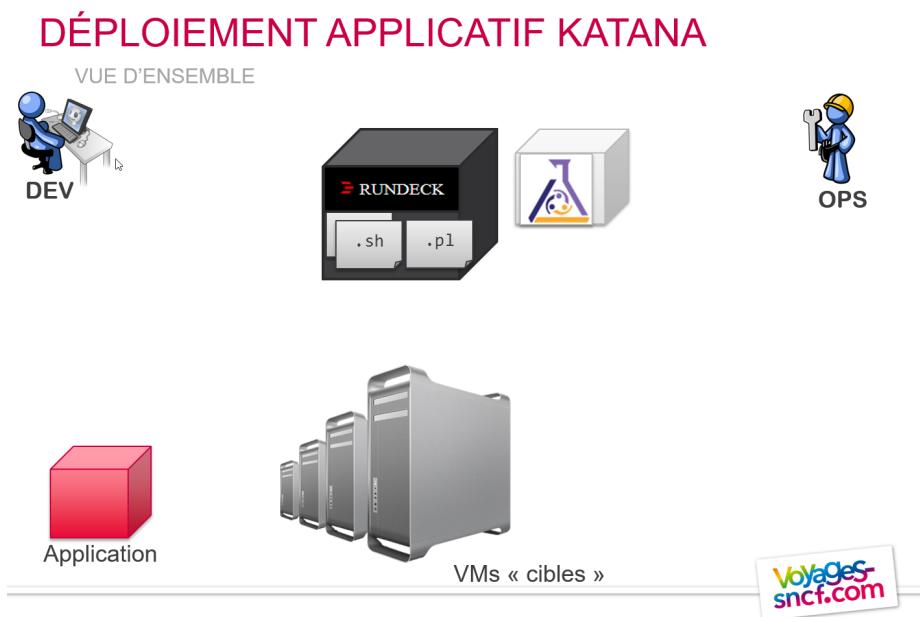


FIGURE 2.6 – Déploiement Applicatif Katana

Elle est composée de :

- un référentiel de données techniques sur les plateformes/machines sous forme infraAsCode (fichiers env Yaml pour les plateformes, fichiers de configuration Puppet) ;
- un référentiel des configurations applicatives : Hesperides (géré par sa propre communauté) ;
- un catalogue de scripts (bash/perl/groovy/...), issu des projets de déploiement VSCT, implantant les opérations de déploiement/contrôle/supervision/etc ;
- un orchestrateur de tâches en self-service, Rundeck, qui pilote les tâches du Puppet et du catalogue de script (en central, ou sur les machines). Il peut être contrôlé soit manuellement (par son IHM web), soit par API REST.

Katana s'inscrit dans l'Usine Logicielle VSCT. La solution s'intègre en particulier avec les outils :

- Jenkins : peut piloter l'orchestrateur de tâches dans le cadre des processus d'intégration ou de déploiement continus
- Nexus : pour le stockage des livrables applicatifs et autres artefacts (en particulier la note de livraison(lexique. 5.2), le catalogue de scripts, les projets rundeck...)
- GIT : pour le stockage et le versionnement des fichiers (code du catalogue de scripts, référentiels de données en Yaml, fichiers de configuration Puppet...)

# 3. PROJET DN-RIDER

## 3.1 Constat & Objectif du stage

Au sein de la direction Delivery, les équipes de développement travaillent avec les équipes d'exploitation selon des principes devops. Pour cela, elles disposent d'un outillage self-service (Katana) pour, entre autres, livrer leurs applications et les déployer sur les différentes plateformes (usine, recette, pré-production, production). A date, cet outillage supporte les activités de plus de 150 applications, sur plus de 1500 plateformes.

Pour utiliser cet outillage, les équipes de développement génèrent pour chaque version de leurs applications une note de livraison (lexique.5.2) qui porte l'ensemble des informations utiles associées (identifiant et emplacement des livrables applicatifs, moyens de tester le bon fonctionnement, pré-requis et instructions d'installation, etc). Ce document prend la forme d'un fichier structuré (JSON).

Objectif de ce projet est de créer une application web (IHM + API REST) pour manipuler les notes de livraison Katana, qui permet de :

- gérer l'objet NDL de manière plus simple qu'avec Nexus/filesystème ;
- éviter les tableaux de suivi, type notes d'installation, tableaux de dépendances... qui sont édités manuellement ;
- fédérer certaines fonctionnalités (extraction d'information, identification des packages à installer sur une plateforme...) par rapport aux scripts bash/groovy/perl/ruby....
- outiller le suivi du cycle de vie des versions par rapports aux informations remontées par les outils de l'usine logicielle et Katana.

L'application devra être légère, dynamique et facilement évolutive, et non-constraignante pour les équipes.

Le but est de fiabiliser le processus de livraison et de mise en production des appli-

cations.

Le stage contient, dans une démarche itérative :

- le recueil des besoins auprès des utilisateurs (passé un cahier des charges initial) ;
- La conception de l'application en accord avec le responsable d'équipe :
  - Conception des pages, des services REST et de l'application dynamique sous-jacente ;
  - Modélisation des données nécessaires à l'application, le cas échéant.
- Le développement de l'application ;
- La documentation de l'application ;
- Le rapport des activités au responsable de l'équipe.

## 3.2 Contexte & Démarche

### 3.2.1 Contexte

Le projet se déroule dans une équipe d'intégration. Le projet est cadré par mon encadrant de stage qui désigne les objectifs du projet. Pour le réaliser, je suis accompagné d'un référent technique.

On a une structure transverse, le travail est autonomie, mais j'ai le support de toute l'équipe.

### 3.2.2 Démarche

Au sein de la VSCT, on met beaucoup de points sur la méthode agile et la discipline "Devops" (lexique. 5.1). Dans cette équipe, on applique un "Kanban" (lexique. 5.1) pour l'organisation des travaux. Comme sur la figure (fig. 3.1), chaque membre dispose d'une ligne sur le tableau et chaque ligne est découpé en trois parties : "TODO", "ENCOUR" et "DONE" qui correspondent aux tâches à faire, tâche en train de faire et tâches réalisés. Les espaces laissés à gauche sont pour des idées qu'on va peut-être planifier un jour.

On applique la "post-it theory" (lexique. 5.1). Les post-its sont classés par leurs couleurs, les plus profonds correspondants aux tâches plus importants. Certaines cartes sont marquées "TS" (Technic Story) dessus, c'est-à-dire que c'est un petit soucis technique qui n'est pas urgent à être résolu, on a une grande liberté de l'organiser selon notre convenance.

Normalement on fait deux fois par semaine la revue du Kanban pour garder le rythme. Chaque fois mes tuteurs valident ce que j'ai réalisé et m'aident à planifier les tâches suivantes.

Les codes sources sont gérés en git et stockés sur Gitlab. Au bout d'un mois et demi, on a la première version de l'application et on commence à faire intégration continue en utilisant Jenkins pipeline. Chaque fois qu'il y a un commit sur la branche "master", l'application sera déployée toute seule.



FIGURE 3.1 – Kanban

### 3.3 Application

#### 3.3.1 Architecture de l'application

**Schéma :** L'application agit en frontal de Nexus via Lucène (lexique.5.1), l'api fournie par Nexus. Sur la figure (fig.3.2) on voit les flux éventuels depuis une application dans le navigateur et ceux depuis une application dans le serveur.

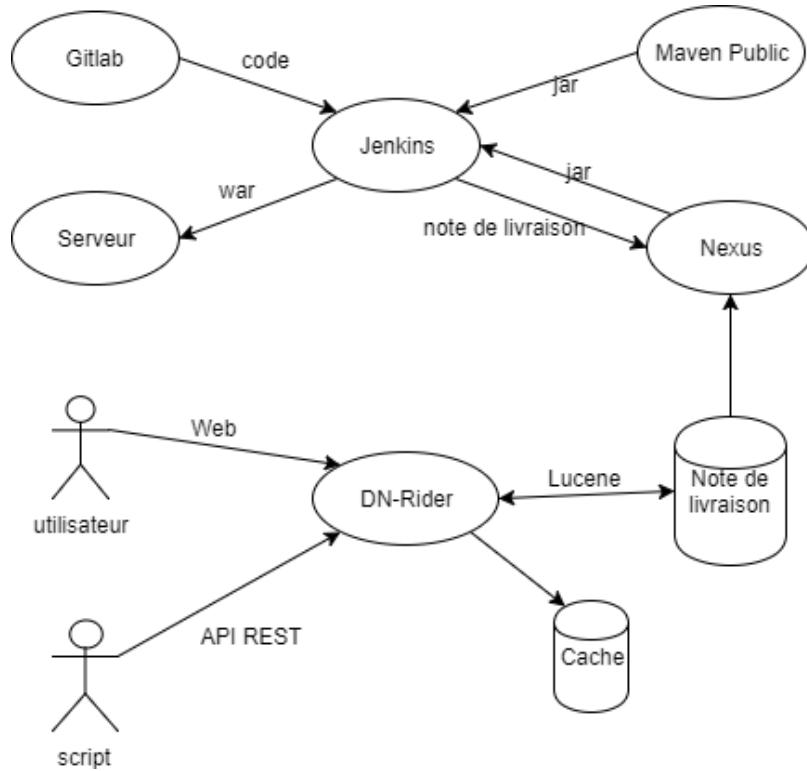


FIGURE 3.2 – Schéma

L'application pourra évoluer pour supporter un autre logiciel 'entrepôt' que Nexus.

**Modèle MVC :** L'application suit le motif Modèle-Vue-Contrôleur. Il est composé de trois types de modules ayant trois responsabilités différentes :

- Modèle : Il n'y pas de base de données dans cette application. Ça correspond aux données de Nexus et à la cache.
- Vue : La présentation de l'interface graphique. Elle est réalisée en utilisant Bootstrap et jQuery dans cette application.
- Contrôleur : La logique concernant les actions effectuées par l'utilisateur. Réalisé en Grails dans cette application.

**La couche service :** On extrait les codes techniques des contrôleurs et former des services. Après, on peut les appeler depuis les contrôleurs. Ça aide à éviter les codes répétitifs et rendre les codes métiers propre. Par exemple, on crée un service "NexusConsumer" qui contient toutes les opérations de Nexus.

### 3.3.2 Choix des technologies

**Grails :** Grails est un framework open source de développement agile d'applications web basé sur le langage Groovy et sur le patron de conception Modèle-Vue-Contrôleur. Il convient à la taille de ce projet. De plus il y a référent compétent qui connaît bien ce framework.

**Groovy :** Groovy est le nom d'un langage de programmation orienté objet destiné à la plate-forme Java. Groovy utilise une syntaxe très proche de Java, par rapport à Java, il est moins verbeux et plus efficace.

**Bootstrap :** Boostrap est un framwork front-end qui contient une collection d'outils utiles à la création du design de sites. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plate-forme de gestion de développement GitHub. Ca aide à créer l'IHM de site facilement.

**jQuery :** jQuery est une bibliothèque JavaScript libre et multi-plateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web. Etant donné la complexité de l'IHM de l'application, jQuery est suffisant pour réaliser les animations et les ajax.

**Spock :** Spock est un framework de test Java capable de gérer le cycle de vie complet d'une application logicielle.

**Backlog :** Backlog (fig.3.3) se réfère à une accumulation d'œuvres en attente d'exécution ou aux commandes à remplir. Cette méthode convient à la méthode qu'on a appliquée au cours du projet.

```

## Domaines:
IC : pipeline de déploiement continue
API: API REST
WEB: IHM Web

## Elements
|
- [ ] [DOC] présenter les fonctionnalités implantées dans README.md
- [ ] [IC] pipeline de déploiement continu sur "epidural" vers "crisdorgames"
- [ ] [IC] tests automatiques de non-regressions
- [ ] [WEB] accès aux écrans par URL
- [ ] [WEB] Consolidation page de Comparaison (ex: sélection d'un intervalle de versions, toutes les versions)
- [ ] [WEB] Editeur de NDL (avec validation et stockage)
- [ ] [WEB/API] swagger
- [ ] [API] Récupérer une note de livraison au format json (GET /api/deliveryNotes/*APP*/*VERSION*)
- [ ] [API] Récupérer la liste des note de livraison
  - GET /api/deliveryNotes/*APP*
  - GET /api/deliveryNotes/*APP*/releases (seulement les releases)
  - GET /api/deliveryNotes/*APP*/snapshots (seulement les snapshots)
  - format JSON ou Textuel selon paramètre format (?format=json (default) OU ?format=text)
- [ ] [API] Récupérer la liste des applications avec note de livraison (GET /api/applications)
  - format JSON ou Textuel selon paramètre format (?format=json (default) OU ?format=text)
- [ ] [API] Stocker une note de livraison
  - POST /api/deliveryNotes/*APP*/releases?version=*VERSION* (erreur si la version cible est une release déjà existante)
  - POST /api/deliveryNotes/*APP*/snapshots?version=*VERSION* (erreur si la version cible est une release déjà existante)
  - PUT /api/deliveryNotes/*APP*/*VERSION* (erreur si la version cible est une release déjà existante)
- [ ] [API] Supprimer une note de livraison (DELETE /api/deliveryNotes/*APP*/*VERSION*)

```

FIGURE 3.3 – BACKLOG

### 3.3.3 IHM

IHM de l'application est réalisé en Bootstrap, jQuery et GSP (le mécanismes de grails pour rendre les vues). Il contient tous un "navbar" qui permet de naviguer entre les pages différentes. Le "navbar" contient des éléments suivants :

- Nom de l'application, nom de version et le numéro de déploiement sur Jenkins ;
- Quatre buttons vers les quatre pages ;
- Un dropdown menu qui permet de switcher les langages (Français, Anglais, Chinois). Par défaut la langue est configurée selon localisation. La fonctionnalité d'internationalization est réalisée en utilisant plugin "i18n", nous avons changé le format et l'encodage par défaut pour ajouter une langue non-latino.

L'application contient principalement les cinq écrans suivant :

#### La page d'accueil (fig.3.4)

Cette page contient des éléments suivants :

- Un bar de recherche qui ramène à la page recherche ;
- Un modal (fig.3.5) pour la configuration personnalisée ;
- Des buttons comme accès rapide vers la page recherche ;
- Un footer qui ramène vers la documentation, les codes sources de l'application et la documentation de l'api REST.

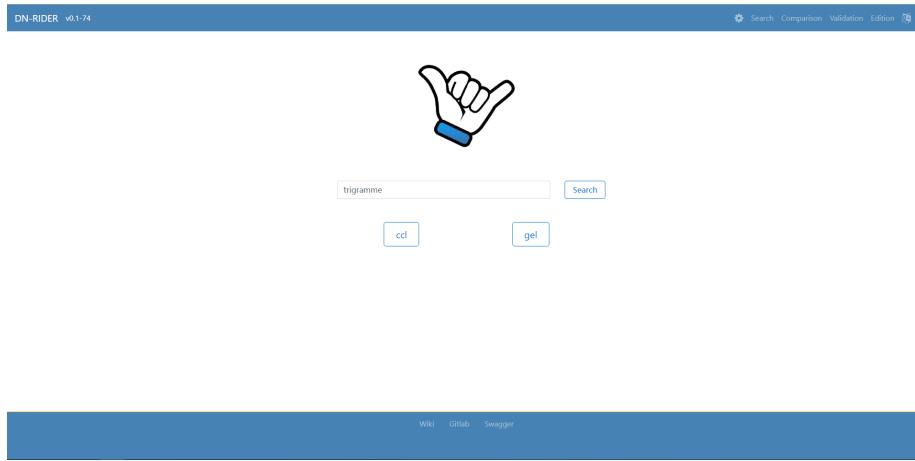


FIGURE 3.4 – La page d'accueil

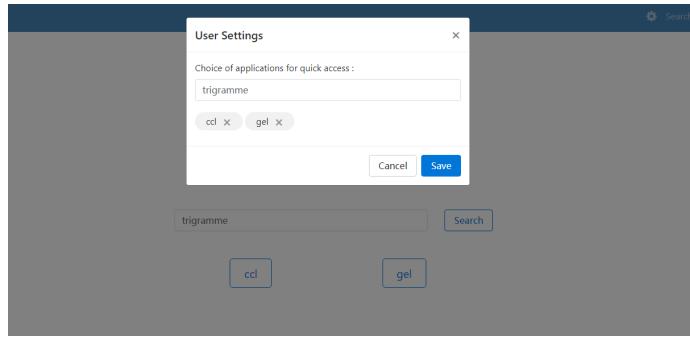


FIGURE 3.5 – Le modal configuration

### La page recherche (fig.3.6)

Cette page permet de choisir une notes de livraison et l'afficher.

Elle contient des éléments suivants :

- Un sidebar collapsible qui contient un formulaire pour choisir les notes de livraisons. On peut filtre par le type de release et la grammaire de l'expression régulière (regex) ;
- Une note de livraison affichée au format json manipulative. L'affichage de json est réalisé avec un plugin jQuery ;
- Des buttons qui permettent de switcher le format d'affichage et des liens vers la page validation et l'application Nexus.

The screenshot shows the DN-RIDER v0.1.74 interface with the search term 'ccl' entered. The results pane displays 167 results, mostly for versions 75.00-SNAPSHOT and 74.00-SNAPSHOT. The results are presented in JSON format, showing details like package name, version, and dependency information.

```

{
  "results": [
    {
      "version": "75.00-SNAPSHOT",
      "dependency": "com.vsc.spaceclient:spaceclient-war:75.00-SNAPSHOT"
    },
    {
      "version": "74.00-SNAPSHOT",
      "dependency": "com.vsc.spaceclient:spaceclient-war:74.00-SNAPSHOT"
    }
  ]
}
  
```

FIGURE 3.6 – La page recherche

### La page comparaison (fig.3.7)

Cette page permet de comparer les différentes versions de notes de livraisons d'une trigramme (lexique.5.2).

Elle contient des éléments suivants :

- Un sidebar collapsible qui contient un formulaire pour choisir les notes de livraisons ;
- Un tableau qui compare les packages des notes de livraisons, le contenu des packages se présentent dans un popover sous format json manipulatif.

The screenshot shows the DN-RIDER v0.1.74 interface in comparison mode for the 'wdl' application. The left sidebar lists various versions of packages, and the main area displays a table comparing their contents across different releases. The table includes columns for Module, Name, and multiple release versions (e.g., 20.0.173.0.2, 20.0.173.1.1, etc.).

| Module          | Name                           | 20.0.173.0.2 | 20.0.173.1.1 | 20.0.173.1.2 | 20.0.173.2.1 | 20.0.173.2.2 | 20.0.174.0.1 | 20.0.175.0.1 | 20.0.175.0.2 |
|-----------------|--------------------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| svctplf         | svctplf-tux                    | 3.298        | 3.298        | 3.298        | 3.298        | 3.298        | 3.298        | 3.299        | 3.299        |
| svctar          | svctar-tux                     | 2.1.0        | 2.1.0        | 2.1.0        | 2.1.0        | 2.1.0        | 2.1.0        | 2.1.0        | 2.1.0        |
| wdlAdmin        | wdlAdmin-war                   | 20.0.173.0   | 20.0.173.1   | 20.0.173.1   | 20.0.173.2   | 20.0.173.2   | 20.0.174.0   | 20.0.175.0   | 20.0.175.0   |
| wdlAdmin        | wdlAdmin-libext                | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         |
| wdlCore         | groovyBatches-batch            | 3.10.0       | 3.0.9        | 3.10.0       | 3.10.0       | 3.10.0       | 3.10.0       | 3.10.0       | 3.10.0       |
| wdlCore         | wdlCore-war                    | 20.0.173.0   | 20.0.173.1   | 20.0.173.1   | 20.0.173.2   | 20.0.173.2   | 20.0.174.0   | 20.0.175.0   | 20.0.175.0   |
| wdlCore         | wdlCoreConfig-configuration    | 1.1          | 1.1          | 1.1          | 1.1          | 1.1          | 1.1          | 1.1          | 1.1          |
| wdlCore         | wdlExt-libext                  | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         |
| wdlInterface270 | wdlInterface270-propertiesLink | file         |
| wdlInterface270 | wdlInterface270-war            | 20.0.173.0   | 20.0.173.1   | 20.0.173.1   | 20.0.173.2   | 20.0.173.2   | 20.0.174.0   | 20.0.175.0   | 20.0.175.0   |
| wdlInterface270 | wdlLibext-libext               | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         |
| wdlInterface280 | wdlInterface280-propertiesLink | file         |
| wdlInterface280 | wdlInterface280-war            | 20.0.173.0   | 20.0.173.1   | 20.0.173.1   | 20.0.173.2   | 20.0.173.2   | 20.0.174.0   | 20.0.175.0   | 20.0.175.0   |
| wdlInterface280 | wdlLibext-libext               | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         | 20.3         |
| wdlInterface290 | wdlInterface290-propertiesLink | file         |
| wdlInterface290 | wdlInterface290-war            | 20.0.173.0   | 20.0.173.1   | 20.0.173.1   | 20.0.173.2   | 20.0.173.2   | 20.0.174.0   | 20.0.175.0   | 20.0.175.0   |

FIGURE 3.7 – La page comparaison

### La page validation (fig.3.8)

Cette page permet de vérifier si une note de livraison est validée selon un json schéma.

Elle contient des éléments suivants :

- Une zone qui permet d'édition une note de livraison ;
- Un bouton qui ramène vers une page qui affiche le schéma ;
- Une fenêtre qui affiche les résultats de validation. Il y a trois types de résultats possibles :
  - Json non valide : Afficher la position de l'erreur et un lien vers l'erreur dans la zone d'édition ;
  - Schéma non valide : Afficher les informations erreurs détaillées au format json ;
  - Valide : Message de réussite.

The screenshot shows the DN-RIDER v0.1.74 interface. On the left, there is a code editor window titled 'Delivery-notes' containing a JSON document. The JSON content is as follows:

```

1  {
2   "NDL_pour_rundeck": [
3     {
4       "dependency": [],
5       "packages": [
6         {
7           "name": "espaeclent",
8           "packageUri": "http://nexus/service/local/artifact/maven/content?r=public&q=c.com.vvct.vsc.espaeclient&a=espaeclient-war&v=75.00-SNAPSHOT&p=war",
9           "version": "75.00-SNAPSHOT",
10          "module": "espaeclient-war-75.00-SNAPSHOT",
11          "extension": "war",
12          "type": "war",
13          "headersModule": "espaeclient-war",
14          "expectedVersion": "75.00",
15          "info": "espaeclient-war",
16          "checkAfterInstall": [
17            {
18              "urlToCheck": "/admin/version",
19              "expectedHttpCode": 200,
20              "expectedContent": "75.00-SNAPSHOT"
21            }
22          ],
23          "targetTechno": "WAS",
24          "isSnapshot": true
25        },
26        {
27          "module": "espaeclient",
28          "packageUri": "http://nexus/service/local/artifact/maven/content?r=public&q=c.com.vvct.vsc.espaeclient&a=espaeclientMails-ftt&v=75.00-SNAPSHOT&p=jar.gz"
29        }
30      ]
31    }
32  }
  
```

Below the code editor are two buttons: 'Check Syntax' and 'Show Schema'. To the right, a 'Validation results' window is open. It has a red header bar with the text 'Invalid JSON line: 3'. The main area contains the message: '\*Unexpected character ('}' (code 5B)) was expecting comma to separate Object entries\*'.

FIGURE 3.8 – La page validation

### La page édition (fig.3.9)

Cette page permet d'édition, valider et stocker une note de livraison.

Elle contient des éléments suivants :

- Un sidebar collapsible qui contient un formulaire pour choisir les notes de livraisons ;
- Une zone qui permet d'édition une note de livraison ;
- Une fenêtre qui affiche les résultats de validation ;
- Un modal (fig.3.10) qui permet de stocker la note de livraison.

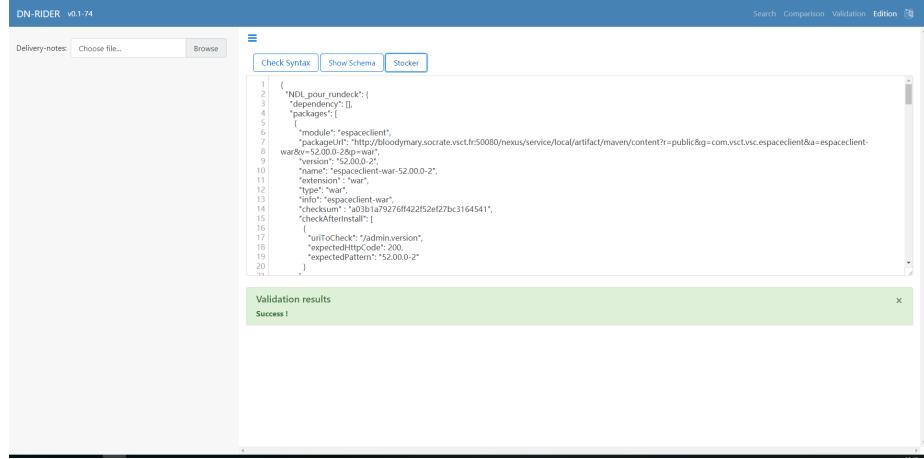


FIGURE 3.9 – La page édition

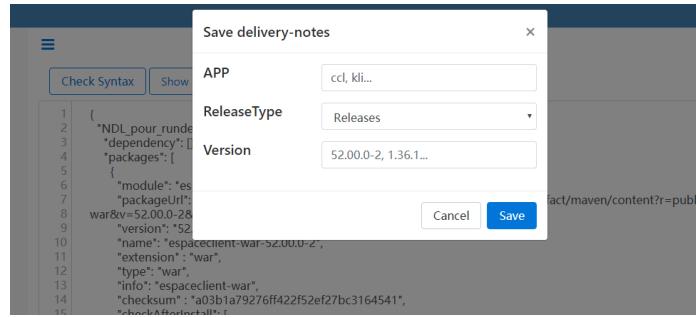


FIGURE 3.10 – Le modal configuration

### 3.3.4 Api REST

L’application fournie une interface api REST. Comme dans l’IHM , l’api permet de chercher, valider, stocker des notes de livraison. En plus, elle permet de supprimer des notes de livraison. Ce qui est différent de l’IHM, on ne peut pas comparer les différents versions d’une trigramme, mais on peut obtenir le contenu d’un package d’une note de livraison.

L’application fournie aussi une interface graphique (fig.3.11) réalisé en Swagger pour la documentation de l’api. Il est réalisé en utilisant un plugin et en ajoutant des annotations dans les contrôleurs.

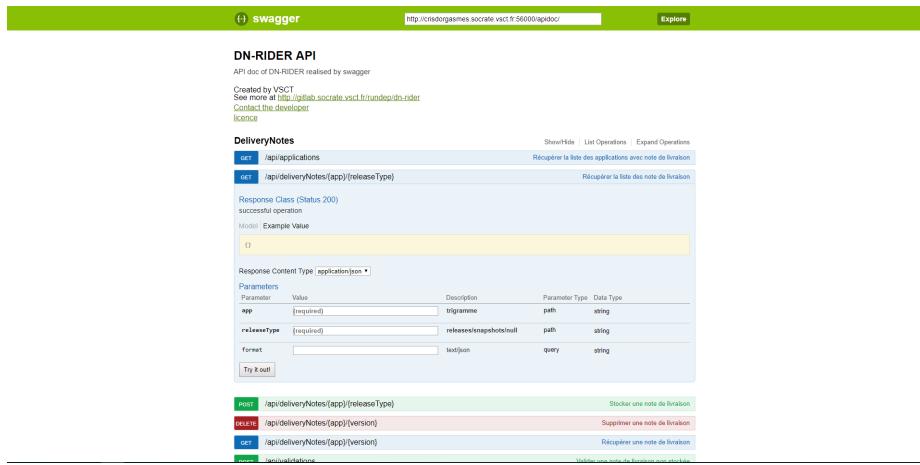


FIGURE 3.11 – La page swagger

### 3.3.5 Intégration Continue

Afin d'améliorer la qualité du code et du produit final et faciliter le déploiement de l'application, on pratique les principes de l'intégratioin continue. L'intégration continu de l'application se réalise en utilisant git et jenkins pipeline.

Les codes sources sont gérés en git et stocké dans Gitlab. Au début on a défini des actions à réaliser obligatoirement avant de faire "git push" :

- Faire relire le code par un tiers ;
- Faire valider par un tiers que la feature développée fonctionne comme attendu ;
- Valider la nouvelle fonction par un ou des tests ;
- Exécuter grails command "test-app".

Le pipeline (fig.3.12) contient les étapes suivantes :

- "checkout" : Récupérer les codes de Gitlab ;
- "versioning" : Modifier la version de l'application selon le numéro de déploiement sur Jenkins ;
- "build" : Builder l'appcation ;
- "deploy" : Copier le war executable sur le serveur de production ;
- "run" : Exécuter l'application.

### 3.3.6 Test

Les tests sont faits pour assurer le bon fonctionnement. Ils sont à lancer avant chaque confirmation de modification de code et chaque déploiement. (fig.3.13)

Les tests contientent des parties suivantes :

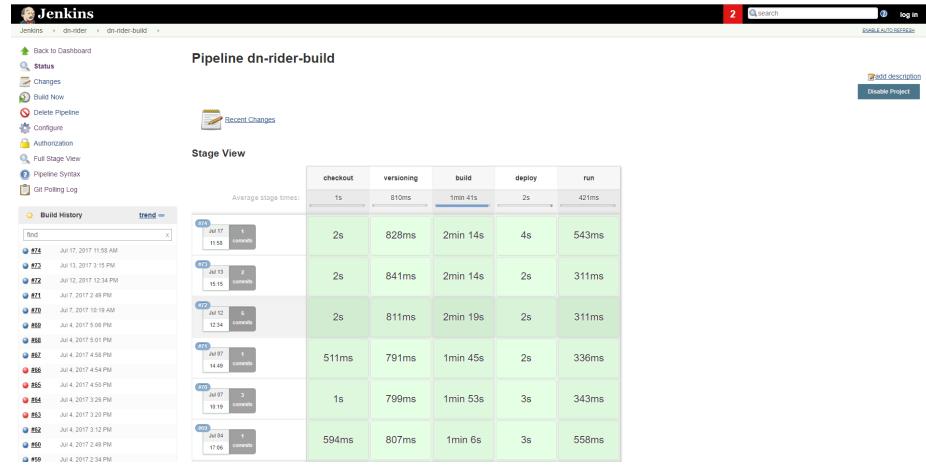


FIGURE 3.12 – Jenkins

- les tests intégrations : Ils sont réalisés en utilisant le framework "Spock", ça couvre l'ensemble des fonctionnalités de l'api REST ;
  - les tests fonctionnels : Ils sont prévu pour la partie IHM.
- L'automatisation de test sur Jenkins est prévu.

```
grails> test-app
:compileJava UP-TO-DATE
:compileGroovy UP-TO-DATE
:buildProperties UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:compileTestJava UP-TO-DATE
:compileTestGroovy UP-TO-DATE
:processTestResources UP-TO-DATE
:testClasses UP-TO-DATE
:test UP-TO-DATE
:compileIntegrationTestJava UP-TO-DATE
:compileIntegrationTestGroovy UP-TO-DATE
:processIntegrationTestResources UP-TO-DATE
:integrationTestClasses UP-TO-DATE
:integrationTest
:mergeTestReports

BUILD SUCCESSFUL

Total time: 22.35 secs
| Tests PASSED
```

FIGURE 3.13 – Test

### 3.4 Etat fin de stage

**Fonctionnalité** La première version de l’application opérationnelle est sur un serveur de production. Just’qu’à présent, l’application a été déployée il y a plus de 70 fois. Après mon stage, cette application va devenir un projet communautaire, les développeurs de VSCT peuvent proposer des contributions, les tests automatiques et déploiement automatique seront appliqués pour assurer le bon déploiement.

**Démo** Un démo au sein de l’équipe des intégrateurs est déjà réalisé. Ce n’était pas encore une version complète, mais les intégrateurs ont compris le but de ce projet et ont proposé leurs idées d’amélioration.

Un démo global était prévu au début juillet pour recueillir de nouveau besoins. Il est retardé et replanifié au début août.

**Bilan** Par rapport à l’objectif, le travail réalisé a couvert l’ensemble des objectifs, mais il reste des éléments suivants :

- Des tests fonctionnels et l’automatisation de test ;
- Des configurations précises de cache ;
- Le traitement des cas exceptionnels dans Nexus.

# 4. CONCLUSION

## 4.1 Bilan personnel

C'était ma première expérience de travailler dans une grande entreprise informatique. Ce stage a été très enrichissant pour moi, car il m'a permis de découvrir les aspects d'une entreprise informatique et expérimenter le processus de développement.

## 4.2 Bilan technique

Grâce à ce projet, j'ai pu avoir ma première expérience en tant que développeur full-stack.

J'ai pu appliquer les différents frameworks de côté serveur et de côté client (Grails 3, Bootstrap 4, jQuery, jQueryUI). J'ai pu pratiquer les langages que je connais (HTML, CSS, JavaScript, Java) et j'ai eu la chance d'utiliser les langages que je ne connais pas (Bash, SCSS, Groovy) et les normes récentes (ECMAScript 6). J'ai pu expérimenter l'environnement open-source.

Au niveau de gestion des codes, j'ai pris une basique connaissance de git et les méthodes d'organisation des branches et des merge requestes.

J'ai pu participer au déploiement manuel de l'application et ensuite à la réalisation de l'intégration continue. J'ai pu observer la façon de travailler des spécialistes et ensuite créé un petit bout de scripts comme ce qu'ils ont fait.

Ce qui est le plus important, je me forme progressivement l'habitude de chercher des informations dans les documentations officielles des frameworks, des plugins et des langages, je commence à comprendre comment chercher mes questions précises dans les moteurs de recherche comme "Google" et les communautés comme "Stack

Overflow", je me sens plus capable qu'avant de trouver les problèmes et les résoudre.

### **4.3 Bilan organisation et académique**

J'ai eu la chance de travailler en autonomie et d'être aidé par une équipe. Le contexte était contraignant au niveau de temps, cahier des charges et contrat, j'ai eu la liberté d'organiser mon travail. J'ai bénéficié l'organisation transverse de l'équipe et j'ai pu pratiquer la méthode agile du développement.

# 5. LEXIQUE

## 5.1 Lexique Général

**EPIC :** Établissement public à caractère industriel et commercial.

**Nexus :** Nexus est un entrepôt de binaires.

**Lucène :** Nexus est une interface api fournie par Nexus.

**Kanban :** Kanban est une méthode de gestion des connaissances relatives au travail, qui met l'accent sur une organisation de type Juste-à-temps en fournissant l'information ponctuellement aux membres de l'équipe afin de ne pas les surcharger. Dans cette approche, le processus complet de l'analyse des tâches jusqu'à leur livraison au client est consultable par tous les participants, chacun prenant ses tâches depuis une file d'attente.

**Devops :** Le devops est un mouvement visant à l'alignement de l'ensemble des équipes du système d'information sur un objectif commun, à commencer par les équipes de dev ou dev engineers chargés de faire évoluer le système d'information et les ops ou ops engineers responsables des infrastructures (exploitants, administrateurs système, réseau, bases de données,...). Ce qui peut être résumé par : travailler ensemble pour produire de la valeur pour l'entreprise.

**Post-it theory :** C'est une méthode « hyper procédurière » de conduite de projet agile. « Hyper procédurière » car elle s'appuie sur un ensemble de rituels tels que le daily meeting, la démonstration, le sprint planning etc.

## 5.2 Lexique de VSC Technologies

**Note de livraison :** Une note de livraison est un fichier qui fait le lien entre la vue technique et la vue applicative.

**Trigramme :** Un trigramme est une application qui a une abréviation de trois lettres.

# Table des figures

|      |  |    |
|------|--|----|
| 2.1  | Epic SNCF . . . . .                        | 7  |
| 2.2  | VSC Technologies : Un mix . . . . .        | 8  |
| 2.3  | VSC : Organisation . . . . .               | 9  |
| 2.4  | Usine Logicielle : Développement . . . . . | 11 |
| 2.5  | Usine Logicielle : Opération . . . . .     | 12 |
| 2.6  | Déploiement Applicatif Katana . . . . .    | 13 |
| 3.1  | Kanban . . . . .                           | 18 |
| 3.2  | Schéma . . . . .                           | 19 |
| 3.3  | BACKLOG . . . . .                          | 21 |
| 3.4  | La page d'accueil . . . . .                | 22 |
| 3.5  | Le modal configuration . . . . .           | 22 |
| 3.6  | La page recherche . . . . .                | 23 |
| 3.7  | La page comparaison . . . . .              | 23 |
| 3.8  | La page validation . . . . .               | 24 |
| 3.9  | La page edition . . . . .                  | 25 |
| 3.10 | Le modal configuration . . . . .           | 25 |

|                                |    |
|--------------------------------|----|
| 3.11 La page swagger . . . . . | 26 |
| 3.12 Jenkins . . . . .         | 27 |
| 3.13 Test . . . . .            | 28 |