

Rapport de Stage d'Ingénieur

---

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

---

*Auteur :*

M. Guoxin SUI

*Encadrant :*

M. Jean-Michel SIZUN

*Responsable d'Option :*

M. Jean-Yves MARTIN

*Référent Technique :*

M. Jérémy RECHET

# REMERCIEMENTS

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

Je souhaite particulièrement remercier mon tuteur de stage, Jean-Michel SIZUN, intégrateur senior, de m'avoir accordé ses conseils et son soutien qui ont menés à bien une grande partie de mes objectifs de stage. J'ai eu la chance de profiter de son expérience pour l'avancement de mon projet au sein de son équipe et grâce à sa confiance, j'ai pu m'accomplir totalement et atteindre les objectifs définis dans le cadre de ce stage.

Aussi, je tiens aussi à remercier Jérémy RECHET, référent technique expérimenté attaché à notre équipe, pour le temps passé ensemble et le partage de son expertise au quotidien. Il a su se rendre disponible et m'a assister tout au long du projet et fût d'une grande aide à l'approche de nouveaux sujets.

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

Enfin, mes remerciements s'adressent également à l'ensemble du personnel de l'entreprise VSCT pour leur accueil et leur soutien à mon intégration 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 . . . . .	11
2.2.1	Usine Logicielle . . . . .	11
2.2.2	Configuration Technique et Déploiement Applicatif . . . . .	13
2.3	Devops . . . . .	14
<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 . . . . .	22
3.3.4	Api REST . . . . .	26
3.3.5	Intégration Continue . . . . .	27
3.3.6	Test . . . . .	28
3.4	Etat à la fin de stage . . . . .	30
<b>4</b>	<b>Conclusion</b>	<b>31</b>
4.1	Bilan organisation . . . . .	31
4.2	Bilan technique . . . . .	31
4.3	Bilan personnel . . . . .	32
<b>5</b>	<b>Lexique</b>	<b>33</b>
5.1	Lexique Général . . . . .	33
5.2	Lexique de VSC Technologies . . . . .	34

# 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 accompagnement de mon tuteur et d'un référent technique, j'ai développé une application web interne, appelée "DN-Rider", pour la gestion des versions des projets.

L'application contient des parties IHM, API REST, documentation et intégration continue. Un série de frameworks, langages et outils ont été utilisés : Grails, Groovy, Bootstrap, SASS, Git, Jenkins, etc.

Le développement est organisé en méthode agile avec la pratique des principes de SCRUM.

Ce rapport apporte le contexte, le déroulement et les résultats obtenus lors de ce stage.

## 2. CONTEXTE

### 2.1 SNCF & VSC Technologies

#### 2.1.1 Le Groupe SNCF

##### Présentation Générale

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.

Chaque jour, elle fait circuler 15 000 trains de fret et de voyageurs et transporte plus de cinq millions de voyageurs. 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.

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).

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.

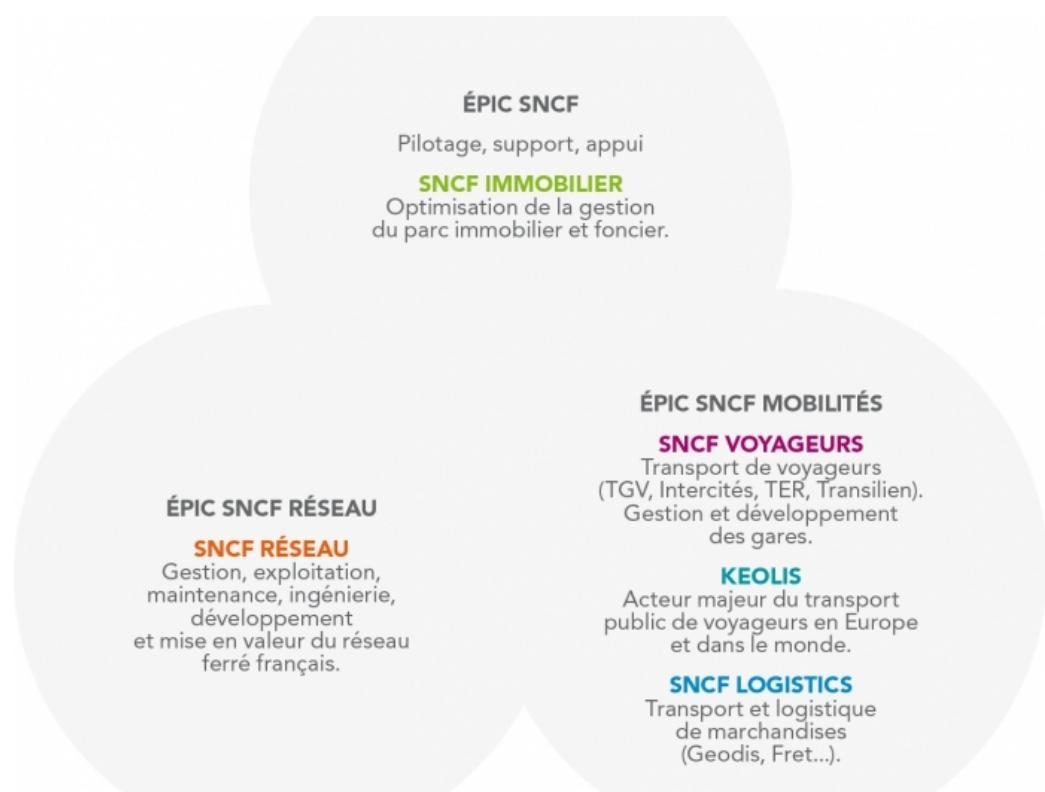


FIGURE 2.1 – Epic SNCF

### 2.1.2 Le Groupe VSC & Rail Europe

#### Présentation Générale

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 (fig.2.2) 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 l'américain Expedia et devient une agence de voyage globale. L'entreprise poursuit alors son développement en France, tout en nourrissant une ambition internationale.

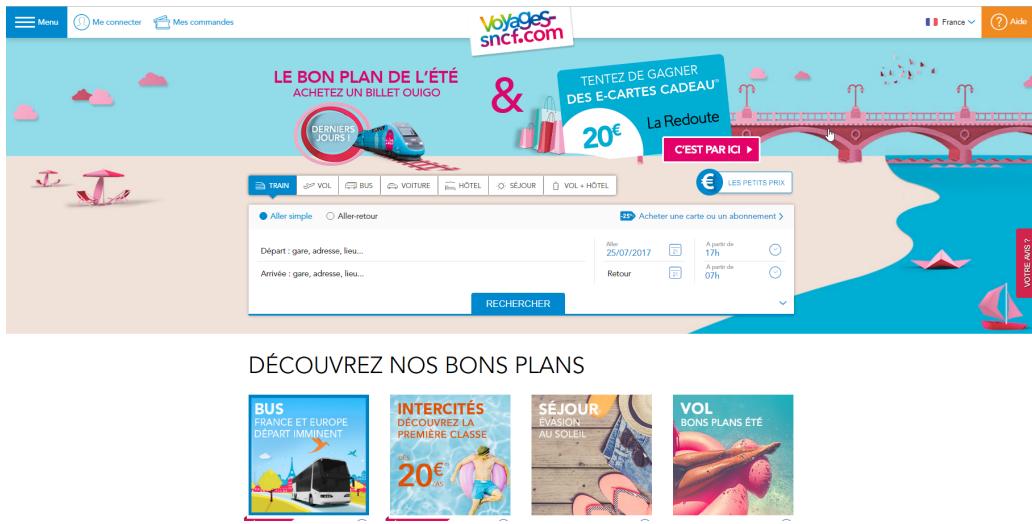


FIGURE 2.2 – Le site de Voyages Sncf

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.

## 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 :



FIGURE 2.3 – VSC : Organisation

les Directions Produits, la Technologie et le Marketing. Enfin, des Directions corporate assistent l'ensemble.

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 (VSCT)

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 une logique éditeur, développe des offres sur mesure pour répondre aux besoins de ses clients (SNCF, Eurostar, IDTGV...)

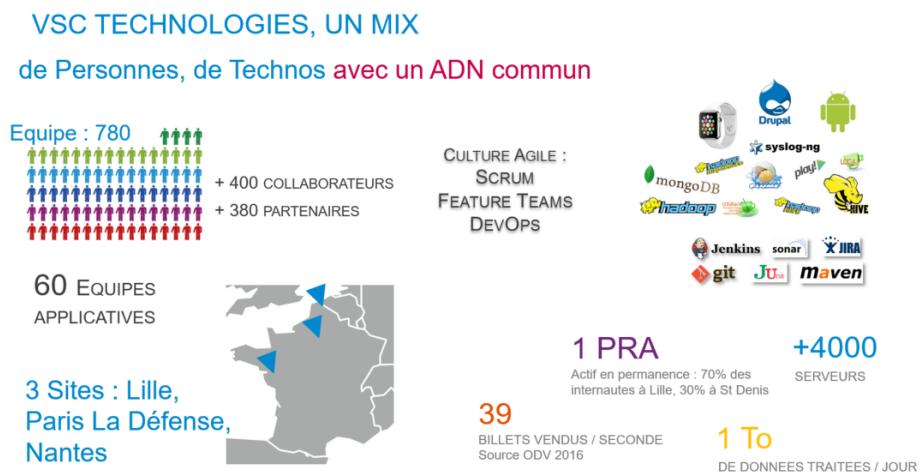


FIGURE 2.4 – VSC Technologies : Un mix

## 2.2 Usine Logicielle

### 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 classique (hors cloud privé).

Dans sa définition actuelle : elle est la continuation d’autres initiatives internes :

- la nouvelle usine logicielle Lille ;
- Katana.

#### Outillage pour le développement

### USINE LOGICIELLE : DÉVELOPPEMENT

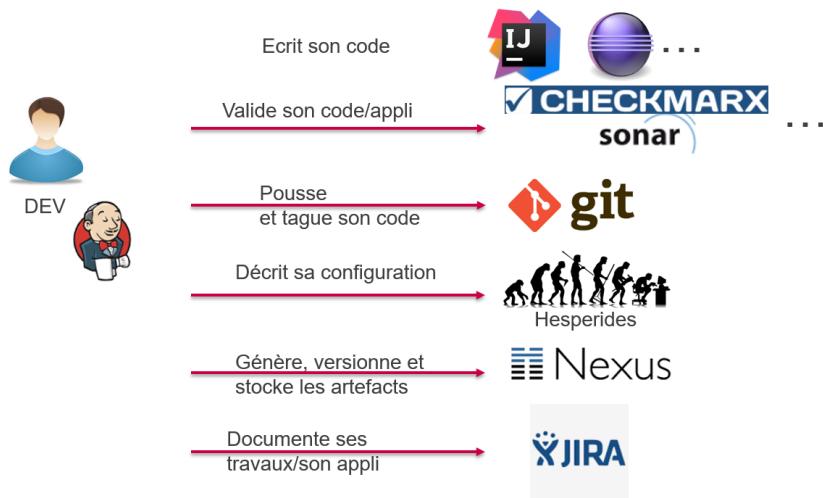


FIGURE 2.5 – Usine Logicielle : Développement

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 plateforme intermédiaires... Pour certaines de ces étapes, JENKINS pilote des outils-tiers (FISHEYE, CRUCIBLE, CHECKMARX, SONAR, etc).

### Outillage pour l'opération des machines

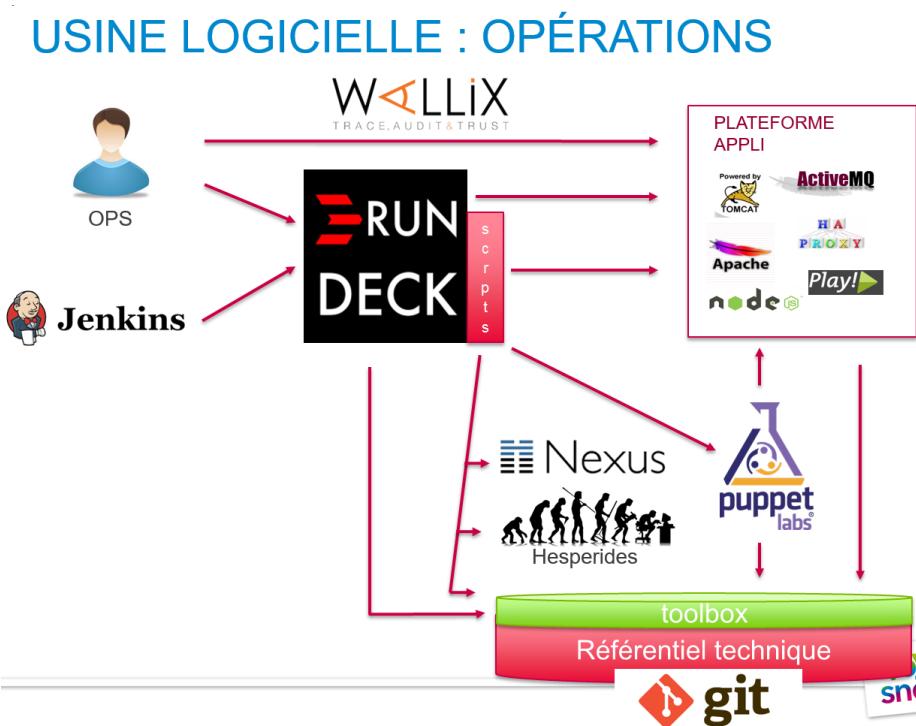


FIGURE 2.6 – Usine Logicielle : Opération

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).

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 Configuration Technique et Déploiement Applicatif

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.

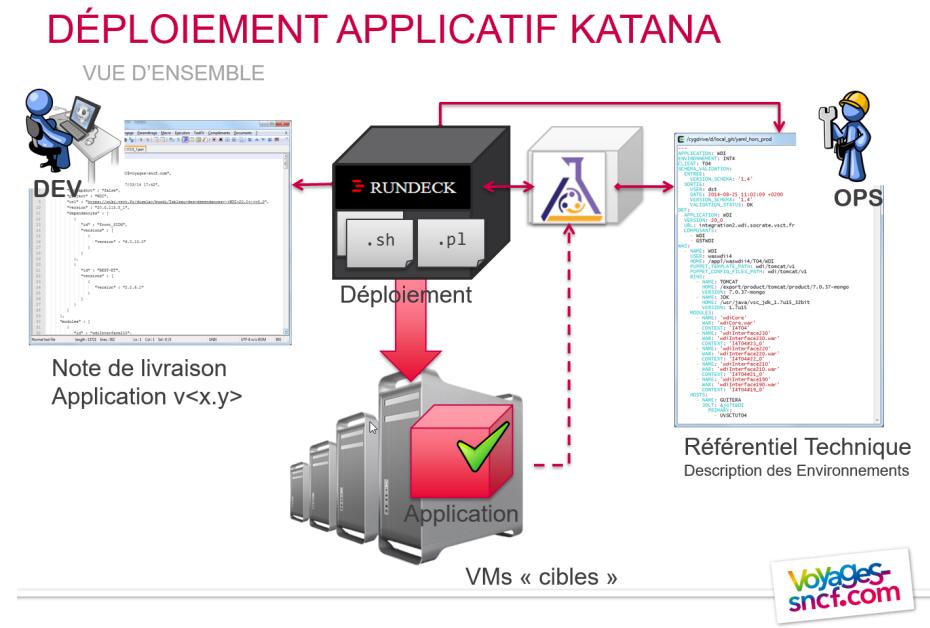


FIGURE 2.7 – Déploiement Applicatif Katana

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...)

## 2.3 Devops

Au sein de la VSCT, on met beaucoup de points sur la méthode agile et les principes "Devops" (lexique. 5.1).

Le devops est un mouvement visant à l'alignement de l'ensemble des équipes du système d'information sur un objectif commun. Les équipes de développement travaillent sous méthodologie SCRUM (lexique.5.1). Ils possèdent de plus en plus de compétences sur l'intégration grâce à l'automatisation des processus. Ça simplifie les communications entre équipes et économise du temps.

# 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 1800 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).

A ce stade, la note de livraison est soit éditée à la main, soit générée automatiquement dans le traitement d'intégration continue (exemple : par MAVEN), et archivé dans l'entrepôt de livrable.

L'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 vise à être légère, dynamique et facilement évolutive, et non-constrainingante pour les équipes.

Le but est de fiabiliser le processus de livraison et de mise en production des applications.

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.

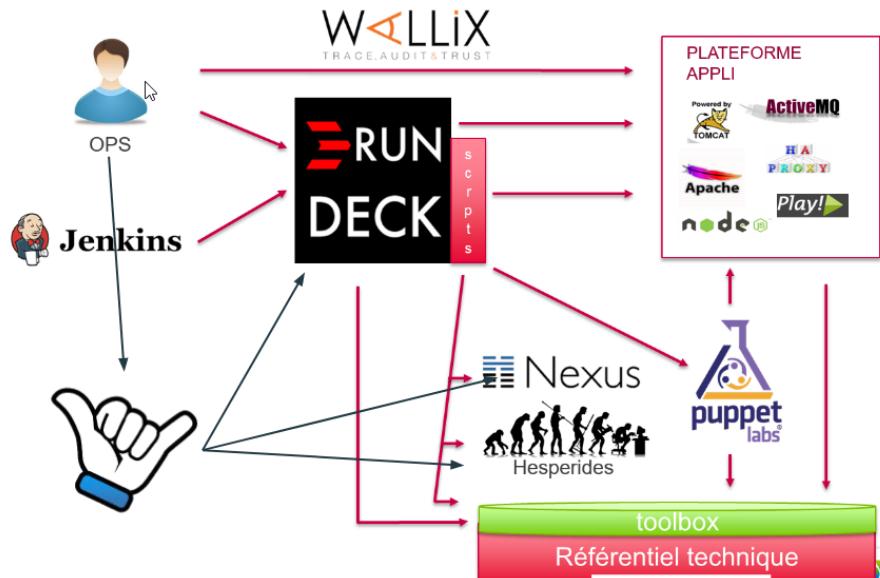


FIGURE 3.1 – Dn Rider

L'application est développée de manière conforme aux normes de l'entreprise aux niveaux des tests automatiques et de l'intégration continue.

## 3.2 Contexte & Démarche

### 3.2.1 Contexte

Le projet se déroule dans une équipe historiquement d'intégration. Ce n'est pas une équipe de développement. La fonction de l'équipe évolue suite au mouvement "de-vops", maintenant ils soutiennent des équipes de développement sur l'intégration. On a une structure transverse, les membres sont sur des projets différents.

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. Le travail se fait en autonomie, mais j'ai l'appui de toute l'équipe.

### 3.2.2 Démarche

Dans cette équipe, on applique un "Kanban"(lexique. 5.1) pour l'organisation des travaux. Comme sur la figure (fig. 3.2), chaque membre dispose d'une ligne sur le tableau et chaque ligne est découpé en trois parties : "TODO", "EN-COUR" et "DONE" qui correspondant 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.

Les tâches sont représentées par des "post-it" et des éléments du backlog (fig.3.3), 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.

Le backlog contient une liste de fonctions ou tâches techniques que l'équipe maintient. Backlog se réfère à une accumulation d'œuvres en attente d'exécution ou aux commandes à remplir. Par rapport au "post-it", le backlog contient des informations plus générales.

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.



FIGURE 3.2 – Kanban

```
## Domains:  
  
IC : pipeline de déploiement continue  
  
API: API REST  
  
WEB: IHM Web  
  
## Elements  
|  
- [ ] [DOC] présenter les fonctionnalités implémentées 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 (défaut) 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 (défaut) 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 de Dn-Rider

## 3.3 Application

### 3.3.1 Architecture de l'application

**Schéma :** L'application agit en frontal de l'api de Nexus. Les recherches se font par Lucène (lexique.5.1). Sur la figure (fig.3.4) on voit les flux éventuels depuis une application dans le navigateur et ceux depuis une application dans le serveur.

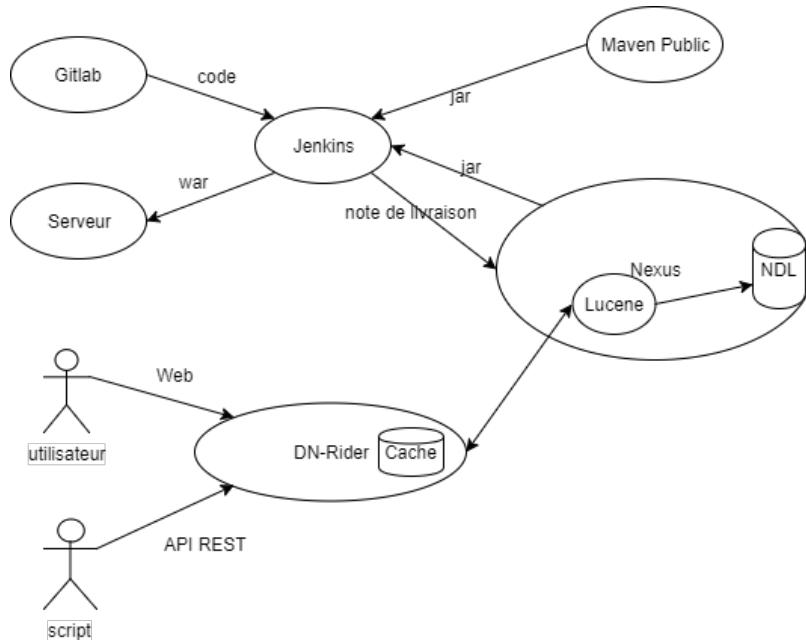


FIGURE 3.4 – Schéma

**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. La persistance de données est assuré par Nexus et cache intégré.
- 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. L'application pourra donc évoluer pour supporter un autre logiciel 'entrepôt' que Nexus.

### 3.3.2 Choix des technologies

Par principe, on choisit les technologies open-source qui conforment à l'objectif et au contexte de ce projet.

#### Coté Serveur

**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.

**RESTful API :** REST (representational state transfer) est un style d'architecture pour les systèmes hypermédia distribués. Les APIs(application programming interface) de services web qui adhèrent aux contraintes de l'architecture REST sont appelés RESTful APIs

**Swagger :** Swagger est une spécification pour les fichiers d'interface lisibles par machine pour décrire, la production, la consommation et la visualisation de Service Web REST.

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



FIGURE 3.5 – Choix des Technologies : Coté Serveur

### Coté Client

**Bootstrap :** Bootstrap est un framework 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. Cela permet de créer l'IHM d'un 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 (lexique.5.1).

**jQuery UI :** jQuery UI est une collection de widgets, effets visuels et thèmes implantés avec jQuery, des feuilles de style en cascade et du HTML.

**HTML 5, CSS 3, ECMAScript 6 :** On essaie toujours d'utiliser les technologies récents.

**SASS :** SASS (Syntactically Awesome Stylesheets) est un langage de génération de feuilles de style (CSS).

### Gestion du Code et Intégration Continue

**Git :** Git est un logiciel de gestion de versions décentralisé. Les codes sources sont générés en git et stockés sur Gitlab. L'authentification se fait par SSH.



FIGURE 3.6 – Choix des Technologies : Coté Client

**Jenkins :** Jenkins est un outil open source d'intégration continue. Jenkins Pipeline est une suite de plug-ins qui prend en charge la mise en œuvre et l'intégration des pipelines de distribution continue dans Jenkins. 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 se déploie toute seule.



FIGURE 3.7 – Choix des Technologies : Gestion du Code et Intégration Continue

### 3.3.3 IHM

L'interface homme-machine (IHM) de l'application est réalisé en Bootstrap, jQuery et GSP (le mécanismes de grails pour rendre les vues). Chaque page contient une entête qui permet de naviguer entre les différentes pages. L'entête contient des éléments suivants :

- Nom de l'application, identifiant de version et de déploiement ;
- Quatre boutons vers les quatre pages ;
- Un menu déroulant qui permet de basculer entre les langages (Français, Anglais, Chinois). Par défaut la langue est configurée selon localisation. La fonctionnalité d'internationalisation 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 suivants :

- Page d'accueil ;
- Page recherche ;
- Page comparison ;
- Page validation ;
- Page edition.

### Paged'accueil (fig.3.8)

Cette page contient des éléments suivants :

- Un champ de recherche qui renvoie à la page recherche ;
- Une fenêtre modale (fig.3.9) pour la configuration personnalisée ;
- Des boutons comme accès rapide vers la page recherche ;
- Un pied de page qui renvoie vers la documentation, les codes sources de l'application et la documentation de l'api REST.

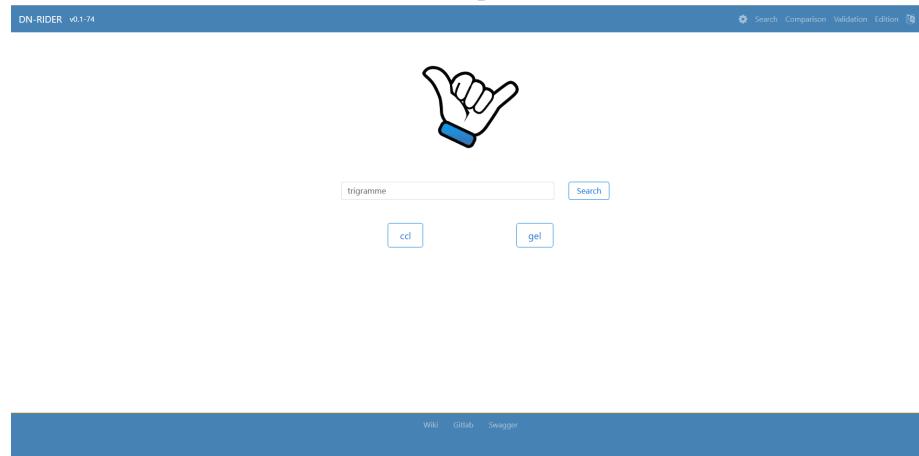


FIGURE 3.8 – La page d'accueil

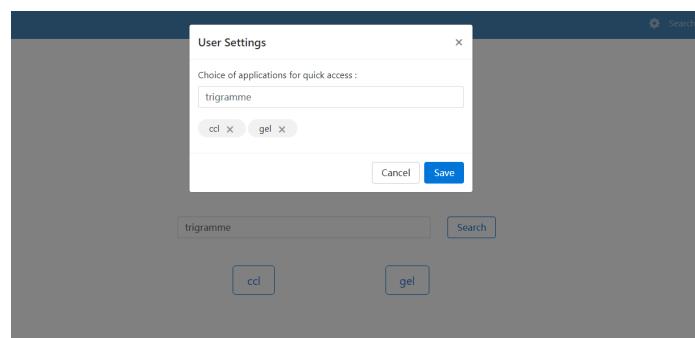


FIGURE 3.9 – La fenêtre modale de configuration

### Page recherche (fig.3.10)

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

Elle contient des éléments suivants :

- Un barre latérale pliant qui contient un formulaire pour choisir les notes de livraisons. On peut filter par le type de release et une expression régulière (regex) ;
- Une note de livraison affichée au format json (affichage brut ou arborescent). L'affichage de json est réalisé avec un plugin jQuery ;
- Des boutons qui permettent de basculer le format d'affichage et des liens vers la page validation et l'application Nexus.

```

DN-RIDER v6.1.74
APP: ccl
ReleaseType: All
Version: 75.00-SNAPSHOT
Filter: regex
Search [x]

167 results
• 75.00-SNAPSHOT
• 74.10-SNAPSHOT
• 74.002-0
• 74.001-1
• 74.001-0
• 74.00-0
• 74.00-SNAPSHOT
• 73.003-0
• 73.00-0
• 73.00-SNAPSHOT
• 73.002-0
• 72.002-0
• 72.001-0
• 72.000-0
• 72.00-SNAPSHOT
• 71.003-0
• 71.002-0
• 71.001-0
• 71.00-0
• 71.00-SNAPSHOT
• 70.003-0
• 70.002-0
• 70.001-0
• 70.00-0
  
```

JSON

```

{
  "NDI_pour_nudeck": {
    "dependencies": [],
    "packages": [
      {
        "extension": "war",
        "packageId": "http://nexus/service/local/artifact/maven/content?r=public&g=com.vsc.vsc.espaceclient&a=espaceclient-war&v=75.00-SNAPSHOT&ext=war",
        "module": "espaceclient",
        "name": "espaceclient-war-75.00-SNAPSHOT",
        "targetTechnologies": "WAS",
        "type": "war",
        "isSnapshot": true,
        "version": "75.00-SNAPSHOT",
        "checkedForRelease": [
          {
            "expectedHttpResponse": 200,
            "expectedPattern": "75.00-SNAPSHOT",
            "urlToCheck": "/admin/version"
          }
        ],
        "info": "espaceclient-war",
        "hesperideVersion": "75.00-SNAPSHOT"
      },
      {
        "extension": "war",
        "packageId": "http://nexus/service/local/artifact/maven/content?r=public&g=com.vsc.vsc.espaceclient&a=espaceclient-wars-ft1&v=75.00-SNAPSHOT&ext=war",
        "module": "espaceclient",
        "name": "espaceclient-wars-ft1-75.00-SNAPSHOT",
        "targetTechnologies": "WAS",
        "type": "ft1",
        "isSnapshot": true,
        "version": "75.00-SNAPSHOT",
        "checkedForRelease": [
          {
            "expectedHttpResponse": 200,
            "expectedPattern": "75.00-SNAPSHOT",
            "urlToCheck": "/espaceclient/wars-ft1"
          }
        ],
        "info": "espaceclient-wars-ft1"
      },
      {
        "filename": "ccl-110n-configuration-75.00-SNAPSHOT.tar.gz",
        "packageId": "http://nexus/service/local/artifact/maven/content?r=public&g=com.vsc.vsc.espaceclient&a=ccl-110n-configuration&v=75.00-SNAPSHOT&ext=tar.gz",
        "module": "espaceclient",
        "name": "ccl-110n-configuration-75.00-SNAPSHOT",
        "targetTechnologies": "WAS",
        "type": "tar",
        "isSnapshot": true,
        "version": "75.00-SNAPSHOT",
        "checkedForRelease": []
      }
    ]
  }
}
  
```

FIGURE 3.10 – La page recherche

### Page comparaison (fig.3.11)

Cette page permet de comparer les différentes versions de notes de livraisons d'une même application, qui est identifiée par un trigramme (lexique.5.2).

Elle contient des éléments suivants :

- Un barre latérale pliant 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".

The screenshot shows a comparison table titled "Comparison" in the top right corner. The table has columns for "Module" and "Name" followed by seven columns representing version numbers: 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, and 20.0.175.0.2. The table contains data for various modules such as svctpf, svctstar, wdiAdmin, wdiCore, and wdiInterface, with values ranging from 2.0.0 to 3.10.0.

FIGURE 3.11 – La page comparaison

### Page validation (fig.3.12)

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

Elle contient des éléments suivants :

- Une zone qui permet d'éditer une note de livraison ;
- Un bouton qui renvoie vers une page qui affiche le schéma ;
- Une fenêtre qui affiche les résultats de validation. Il y a trois types de résultats possibles :
  - Donnée mal formée (syntaxe json incorrect) : Afficher la position de l'erreur et un lien vers l'erreur dans la zone d'édition ;
  - Donnée invalide (syntaxe json corret, main contenu non conforme par rapport à un schéma) : Afficher les informations erreurs détaillées au format json ;
  - Valide : Message de réussite.

### Page edition (fig.3.13)

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

Elle contient des éléments suivants :

- Un barre latérale pliant qui contient un formulaire pour choisir les notes de livraisons ;
- Une zone qui permet d'éditer une note de livraison ;
- Une fenêtre qui affiche les résultats de validation ;

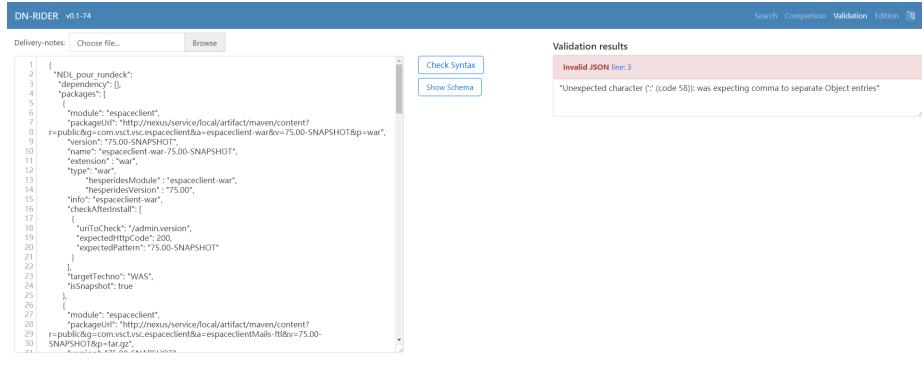


FIGURE 3.12 – La page validation

- Une fenêtre modale (fig.3.14) qui permet de stocker la note de livraison.

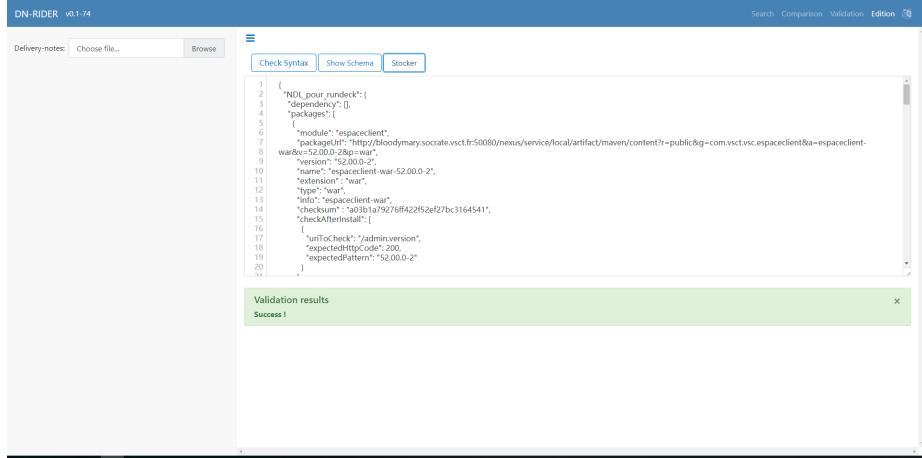


FIGURE 3.13 – La page edition

### 3.3.4 Api REST

L’application fourni 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.3.15) réalisé en Swagger pour la documentation de l’api. Il est réalisé en utilisant un plugin et en ajoutant des

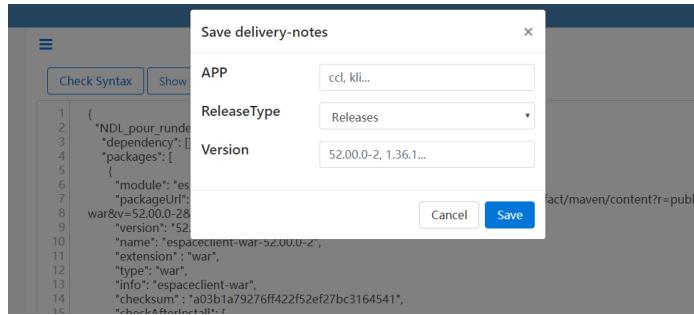


FIGURE 3.14 – La fenêtre modale de save

annotations dans les contrôleurs.

The screenshot shows the Swagger UI for the DN-RIDER API. It displays the "DeliveryNotes" controller with two methods: "get /api/deliveryNotes/{app}/{releaseType}" and "get /api/deliveryNotes/{app}/{version}". The "get /api/deliveryNotes/{app}/{releaseType}" method is highlighted. Below the methods, there is a table for "Parameters" and a "Try it out" button. At the bottom, there is a list of available operations: "POST /api/deliveryNotes/{app}/{releaseType}" (Stocker une note de livraison), "DELETE /api/deliveryNotes/{app}/{version}" (Supprimer une note de livraison), "GET /api/deliveryNotes/{app}/{version}" (Récupérer une note de livraison), and "PUT /api/deliveryNotes/{app}/{version}" (Mettre à jour une note de livraison).

FIGURE 3.15 – 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 suit les principes de l'intégration continue. L'intégration continue 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.3.16) 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'application ;
- "deploy" : Copier le war executable sur le serveur de production ;
- "run" : Exécuter l'application.

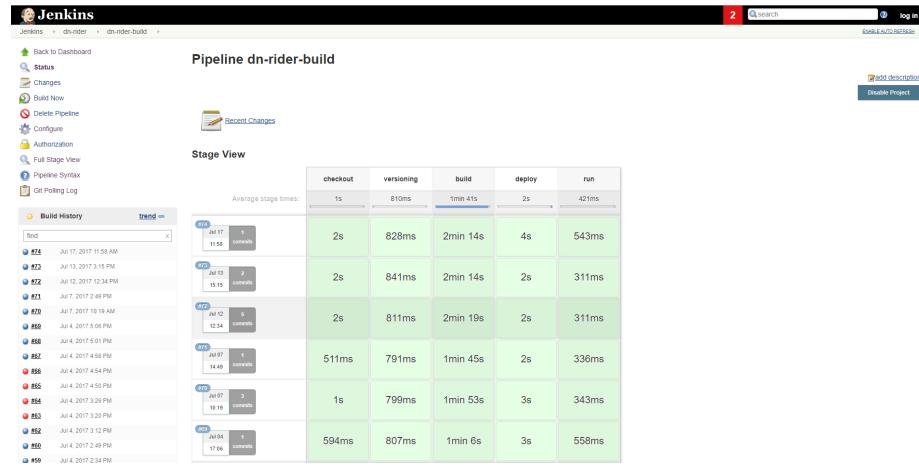


FIGURE 3.16 – Jenkins

### 3.3.6 Test

Les tests sont faits pour s'assurer le bon fonctionnement et de l'absence de regressions en cas d'évolution. Ils sont à lancer avant chaque confirmation de modification de code et chaque déploiement. (fig.3.17)

Les tests contiennent des parties suivantes :

- les tests unitaires : Le test unitaire permet de vérifier le bon fonctionnement d'une partie précise d'une application. Cette partie est couverte par les tests d'intégration dans cette application ;
- les tests d'intégration : Dans le test d'intégration, chaque module indépendant du logiciel est assemblé et testé dans l'ensemble. Ils sont réalisés en utilisant le framework "Spock" et ils couvrent l'ensemble des fonctionnalités de l'api REST ;
- les tests fonctionnels : Les tests fonctionnels sont pour objectif de vérifier que les fonctionnalités demandées sont bien supportées. Ils sont prévu pour la partie IHM.

L'automatisation de test sur Jenkins est prévue.

```
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
:integrationTestClassses UP-TO-DATE
:integrationTest
:mergeTestReports

BUILD SUCCESSFUL

Total time: 22.35 secs
| Tests PASSED
```

FIGURE 3.17 – Test

### 3.4 Etat à la fin de stage

**Fonctionnalité** La première version de l'application opérationnelle est sur un serveur de production. Jusqu'à présent, l'application a été déployée 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** Une 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.

Une démo global était prévue au début juillet pour recueillir de nouveau besoins. Elle a été retardé et replanifié au début août pour synchroniser avec un autre projet de stage.

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

- Des tests fonctionnels pour compléter l'automatisation de test ;
- Un perimétrage plus précis des caches ;
- Le traitement de cas exceptionnel lié à certaines applications dans Nexus ;
- D'autres APIs.

# 4. CONCLUSION

## 4.1 Bilan organisation

J'ai eu la chance de travailler en autonomie et d'être soutenu par une équipe qui a su me laisser la liberté d'organiser mon travail selon mon rythme, sans fortes contraintes de temps ni de cahier des charges ou de contrat. J'ai donc pu bénéficier de l'organisation transverse de l'équipe et ainsi pratiquer les méthodes de développement agile.

## 4.2 Bilan technique

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

J'ai pu utiliser les différents frameworks du côté serveur et du côté client (Grails 3, Bootstrap 4, jQuery, jQueryUI). Cela m'a permis de mettre en application mes compétences sur des langages tels que HTML, CSS, JavaScript ou encore Java, mais aussi de découvrir de nouvelles technologies telles que Bash, SCSS et Groovy.

Au niveau de la gestion du code, j'ai su monter en compétences sur git au travers de méthodes d'organisation sur les branches ou encore les "merge requests".

Egalement, j'ai pu découvrir les différents processus utilisés en entreprises. Cela m'a permis de participer au déploiement manuel de l'application puis à la mise en place de son intégration continue.

Enfin, j'ai su gagner en confiance en apprenant à répondre à des problématiques au travers de recherches dans les documentations officielles des langages, des frameworks, des plugins ou encore des forums.

### 4.3 Bilan personnel

Ce stage a été ma première expérience de travail dans une grande entreprise informatique. Ce fut très enrichissant car il m'a permis de découvrir les aspects de cette entreprise et d'expérimenter les processus de développement et de déploiement. J'ai ainsi pu mettre en pratique mes connaissances acquises au cours de ma formation mais surtout j'ai pu les enrichir.

Grâce à cette expérience, je saurai mieux appréhender les projets informatiques et mettre en application tout ce que j'ai pu y découvrir. Cela me servira dans le cadre de la poursuite de mes études.

# 5. LEXIQUE

## 5.1 Lexique Général

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

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

**Apache Lucène :** Apache Lucène est un moteur d’indexation textuel. Il est utilisé 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.

**SCRUM :** 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.

**Ajax :** Ajax (Asynchronous JavaScript And Xml) est un ensemble de techniques de développement Web en utilisant de nombreuses technologies Web sur le côté client pour créer des applications Web asynchrones.

## 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 l'identifiant d'une application. Historiquement, cet identifiant se ramène à un code sur trois caractères.

# Table des figures

2.1	Epic SNCF . . . . .	7
2.2	Le site de Voyages Sncf . . . . .	8
2.3	VSC : Organisation . . . . .	9
2.4	VSC Technologies : Un mix . . . . .	10
2.5	Usine Logicielle : Développement . . . . .	11
2.6	Usine Logicielle : Opération . . . . .	12
2.7	Déploiement Applicatif Katana . . . . .	13
3.1	Dn Rider . . . . .	16
3.2	Kanban . . . . .	18
3.3	Backlog de Dn-Rider . . . . .	18
3.4	Schéma . . . . .	19
3.5	Choix des Technologies : Coté Serveur . . . . .	21
3.6	Choix des Technologies : Coté Client . . . . .	22
3.7	Choix des Technologies : Gestion du Code et Intégration Continue . .	22
3.8	La page d'accueil . . . . .	23
3.9	La fenêtre modale de configuration . . . . .	23

3.10 La page recherche . . . . .	24
3.11 La page comparaison . . . . .	25
3.12 La page validation . . . . .	26
3.13 La page edition . . . . .	26
3.14 La fenêtre modale de save . . . . .	27
3.15 La page swagger . . . . .	27
3.16 Jenkins . . . . .	28
3.17 Test . . . . .	29