Apprenti Développeur Fullstack à la DGFiP sur l'application PRISME

FINANCES PUBLIQUES

Rapport M2
Romain FORBIN

année 2024-2025
M2 INFORMATIQUE Filière IMAGES

MAÎTRES D'APPRENTISSAGE
Mohamed FEGHOUL & Olivier MADERN

RÉFÉRENTE UNIVERSITAIRE
Tita KYRIACOPOULOU

REMERCIEMENTS

Cette année j'ai pu continuer mon alternance au sein de la même entreprise que l'année dernière avec la section Architecture Technique et Partenaires (ATP) de la division des applications de la paye du Bureau des applications de la Comptabilité, de la Dépense de l'État et du Domaine (BSI-6) au sein de la Direction des Projets Numériques (DPN) à la Direction Générale des Finances Publiques (DGFiP).

Dans un premier temps, je tiens à remercier mes maîtres d'apprentissage Mohamed FEGHOUL et Oliver MADERN qui m'ont accompagné tout le long de cette deuxième année dans les meilleures conditions. Je tiens également à remercier l'équipe ATP, dont je fais partie depuis deux ans maintenant.

Je souhaite aussi remercier toutes les équipes présentes à BSI-6 pour leur soutien au quotidien et pour leurs encouragements tout le long de mes études.

Je voudrais également remercier l'Université Gustave Eiffel pour ses enseignements qui ont été plus qu'utiles durant mes années d'apprentissage.

SOMMAIRE

INDEX DES FIGURES PRÉSENTES DANS LE DOCUMENT

GLOSSAIRE DES ACRONYMES

Acronyme	Signification	Description
DGFiP	Direction Générale des Finances Publiques	Administration publique qui m'accueille
PRISME	Pilotage et Restitution d'Informations pour le Suivi des Maintenances Évolutives	Application sur laquelle je suis affecté
MOA	Maîtrise d'ouvrage	
MOE	Maîtrise d'œuvre	
SI	Service des Systèmes d'Informations	Un des nombreux services de la DGFiP
EIB	Expression Initiale du Besoin	
EIF	Étude d'Impact et de Faisabilité	
SFD	Spécification Fonctionnelle Détaillée	
CTG	Conception Technique Générale	
FQR	Fiche Question/Réponse	
CORE		Application du suivi des charges réalisées sur différents projets du SI
JSTL	Java standard Tag Library	
JSP	Jakarta Server Pages	
SILL	Socle Interministériel des Logiciels Libres	

INTRODUCTION

Les captures d'écran de l'application ont été retouchées pour éviter l'affichage de données sensibles.

Au cours de mes années d'études, j'ai eu l'occasion de participer à de nombreux projets proposés par mes enseignants. Ces projets m'ont permis de mettre en pratique mes compétences techniques, mais toujours en adoptant une posture principalement orientée développement. Dans le domaine de l'informatique, un projet peut être envisagé sous deux angles complémentaires :

- d'une part, la vision du développeur, centrée sur la création de fonctionnalités, la résolution de bugs, la conception technique et l'écriture de code,
- d'autre part, la vision du pilotage ou de la supervision, qui consiste à assurer un suivi global du projet, à vérifier son bon déroulement, à identifier les blocages éventuels et à garantir l'atteinte des objectifs dans les délais impartis.

C'est cette seconde approche, moins technique mais tout aussi essentielle, que j'ai pu découvrir au cours de mes années d'alternance. En effet, j'ai été affecté au projet PRISME, une application de pilotage destinée à fournir une vue d'ensemble sur l'état d'avancement d'un autre projet majeur : l'application PAYSAGE.

Créée en 2019, PRISME est une application conçue pour centraliser et synthétiser les informations liées au suivi du projet PAYSAGE. Elle offre un tableau de bord clair et structuré, à destination de l'ensemble des parties prenantes. Son objectif principal est de permettre un accès rapide à des informations clés concernant l'évolution du projet, les besoins exprimés, les difficultés rencontrées, ainsi que les actions en cours ou à venir.

L'application PRISME est utilisée par plusieurs acteurs au sein de la Direction Générale des Finances Publiques (DGFiP), notamment les agents de la MOA[1], de la MOE[2], les développeurs du projet PAYSAGE (ESI38[3]), ainsi que les équipes de BSI-6, telles que le pôle pilotage ou les analystes.

Pour assurer cette fonction de pilotage, PRISME s'appuie sur des données issues de différents outils de suivi tels que CORE[4] ou Bugzilla[5]. Elle rassemble également les informations transmises par les équipes impliquées : les demandes de maintenance émises par la MOA,

1 Maîtrise d'ouvrage.
2 Maîtrise d'œuvre.
3 Établissement de Service Informatique de Grenoble.
4 Application de suivi du réalisé des équipes.
5 Application de suivi des tickets.

les analyses et estimations de charges réalisées par la MOE (BSI-6), et les développements effectués par l'ESI38.

Un autre aspect important de PRISME réside dans le fait que les équipes qui l'utilisent sont réparties sur l'ensemble du territoire. Dans ce contexte, elle constitue un outil indispensable pour assurer une coordination efficace entre les différents intervenants, malgré la distance géographique. Grâce à PRISME, les différents acteurs peuvent suivre l'état du projet en temps réel, prioriser les demandes, et veiller au respect des délais.

Ainsi, une question se pose : en quoi l'application PRISME permet-elle d'assister efficacement l'ensemble des équipes impliquées dans le projet PAY-PAYSAGE afin d'en assurer le bon déroulement ?

Pour y répondre, je commencerai par une présentation de la DGFiP ainsi que de l'application PRISME de manière détaillée. Ensuite, je reviendrai sur plusieurs missions que j'ai réalisées au cours de cette année d'apprentissage. Enfin, je conclurai ce rapport par une réflexion professionnelle et personnelle sur cette expérience.

ENTREPRISE

Qui est la DGFiP

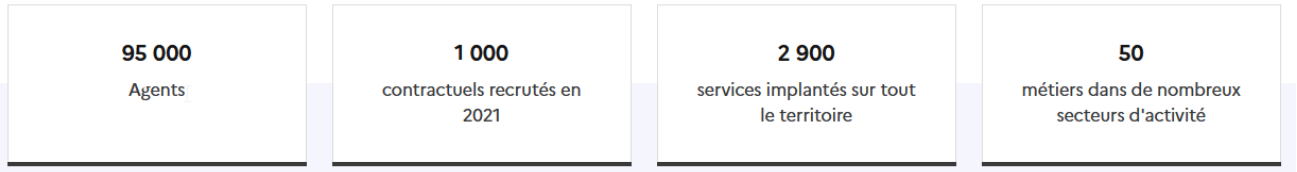
La Direction Générale des Finances Publiques (DGFiP) est une administration publique placée sous l'autorité du Ministère de l'Économie, des Finances et de la Souveraineté industrielle et numérique. Elle a été créée en 2008 à la suite de la fusion de deux grandes entités : la Direction Générale des Impôts (DGI) et la Direction Générale de la Comptabilité Publique (DGCP). Cette réorganisation avait pour objectif de moderniser et de rationaliser la gestion des finances publiques en centralisant les missions fiscales et comptables. Elle est actuellement dirigée par Amélie VERDIER depuis le 4 mars 2024 et se compose d'une administration centrale, implantée principalement à Bercy (Paris 12e arrondissement), ainsi que d'une administration décentralisée en province. Les principaux centres de services informatiques sont quant à eux localisés à Montreuil et Noisy-le-Grand en Seine-Saint-Denis (93).

Voici une présentation de la DGFiP par son ancien directeur lors d'une interview (lien vers l'interview https://youtu.be/Co4uqMQfLbM?t=24) :

« *la Direction Générale des Finances Publiques c'est une administration très vaste, près de 100 000 agents et qui est au cœur du fonctionnement financier de l'État.* » Jérôme FOURNEL, ancien Directeur Général des Finances Publiques.

Voici les chiffres concernant les acteurs de la DGFiP :

(https://choisirleservicepublic.gouv.fr/employeurs/dgfip/)



Chiffres acteurs DGFiP

Souvent perçue uniquement comme l'organisme chargé de la gestion des impôts, la DGFiP assure en réalité des missions beaucoup plus larges, telles que :

- la gestion des recettes et des dépenses publiques de l'État ;
- la tenue de la comptabilité de l'État et des collectivités locales ;
- le pilotage des retraites de l'État ;
- le contrôle fiscal et juridique ;
- la gestion de patrimoine public immobilier ;
- l'accompagnement des collectivités territoriales dans leurs fonctions budgétaires ;
- le paiement des salaires des agents publics.

La Direction Générale des Finances Publiques reste une administration assez grande. De ce fait, elle se divise en différents services (cf « Annexe 1 : Organigramme DGFiP ») :

- le service de la Sécurité Juridique et du Contrôle Fiscal ;
- le service de la Gestion Fiscale ;
- le service des Gestions Publiques Locales, des Activités Bancaires et Économiques ;
- le service de la Fonction Financière et Comptable de l'État ;
- le service des Retraites de l'État ;
- le service des Ressources Humaines ;
- le service Stratégie Pilotage Budget ;
- et le service Systèmes d'Information dirigé par Tomasz BLANC (service sur lequel je suis affecté).

Ce service est lui-même divisé en 21 structures, dont celle qui m'accueille, le bureau BSI-6 administré par Béatrice BURG (cf « Annexe 2 : Organigrame SI »).

Qui est le Bureau BSI-6

Le Bureau BSI-6 est responsable des applications de la comptabilité, de la dépense de l'État et du Domaine.

Ce bureau est composé de 73 personnes et est divisé en 5 divisions (cf « Annexe 3 : Organigramme BSI-6 ») :

- Pôle méthodologique, Technique et Infocentres ;
- Paye ;
- Domaine, Produits Divers Compta-Dépenses, Applications Transverses et de Pilotage ;
- Application des Pensions ;
- Secrétariat, Pilotage et Missions Transverses.

J'ai effectué deux années d'apprentissage dans la division PAYE dirigée par Maud-Hélène TEYSSOU. Cette division prend en charge les applications qui gèrent les payes des agents de l'État, soit environ 2,5 millions de paye chaque mois.

La division comporte 4 sections :

- la section Pilotage des Applications de la Paye (PAP) ;
- la section Paysage Application Modernisée (PAM) ;
- la section Conception des Applications de la Paye (CAP) ;
- la section Architecture Technique et Partenaires (ATP).

Je fais partie de la section ATP avec Mohamed FEGHOUL, Olivier MADERN, François ESQUERRE, Tatiana MAYOT et Laurent STANNEY.

La section ATP est une équipe transverse au sein de la division PAYE qui participe à de nombreux sujets et vient en support aux autres sections de la division au quotidien. Il s'agit d'un pôle qui est en charge des chantiers techniques, de l'outillage (dont PRISME) et de l'intégration technique de l'application PAYSAGE ainsi que de la gestion de l'architecture technique/applicative et la gestion (technique) des partenaires.

QU’EST-CE PRISME ?

Introduction

PRISME (Pilotage et Restitution d’Informations pour le Suivi des Maintenances Évolutives) est une application interne développée en 2019 au sein de la division PAYE. Son code source est porté par la forge interministérielle (une plateforme collaborative de type GitLab[6] utilisée par plusieurs administrations) et l’application est installée sur une machine virtuelle dédiée, elle-même hébergée sur un serveur mutualisé dédié aux applications d’outillage, PRISME a été conçue pour répondre aux besoins de pilotage liés à l’évolution de l’application PAY-PAYSAGE.

Elle a été créée par mon tuteur Olivier MADERN, basée sur les demandes exprimées par la responsable de l’équipe Pilotage assistée par une prestataire. L’objectif principal de PRISME est de fournir un suivi concis de l’avancement des maintenances évolutives et d’offrir une vision synthétique du périmètre fonctionnel traité. C’est donc un outil de pilotage utile et facile d’accès pour les équipes impliquées dans le projet PAY-PAYSAGE, l’application en charge de la gestion de la paye des agents de l’État.

PAY-PAYSAGE est l’application gérant la paye des agents de l’État. Elle est composée de PAY, l’application historique écrite en Cobol[7], et de PAYSAGE sa réécriture en cours en Java. L’objectif de PAYSAGE est de réaliser une conversion iso-fonctionnelle[8] et de concevoir une architecture évolutive et maintenable.

Ainsi, PRISME joue un rôle clé dans la coordination entre les différents acteurs du projet PAYSAGE (MOA, MOE, développeurs, pilotage), en permettant un accès rapide et structuré aux données de pilotage.

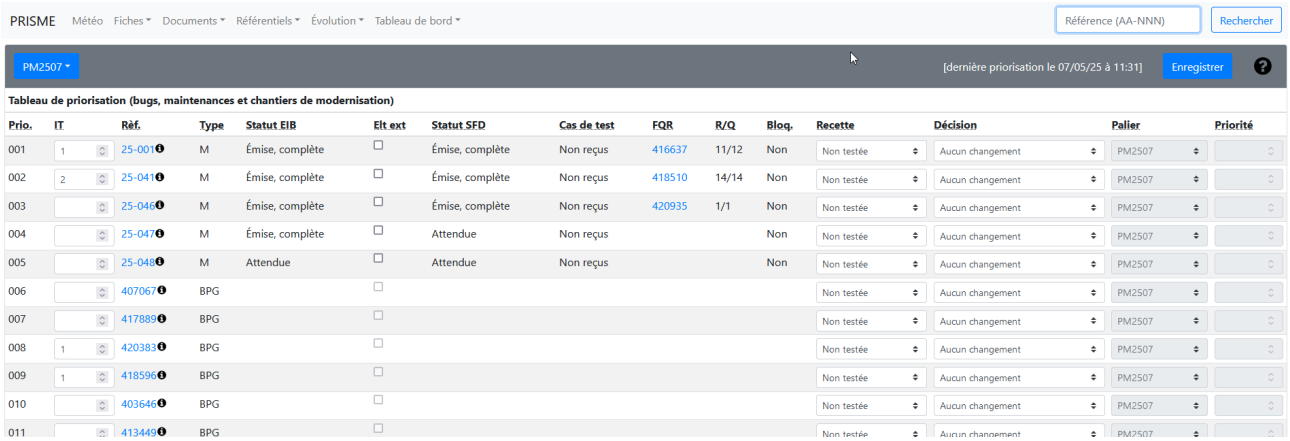
6 Forge utilisée.
7 Langage de programmation.
8 migration où les composants applicatifs et l’infrastructure sont transférés vers un environnement cible sans modification de leurs fonctionnalités.

Principe de l'application

PRISME sert au pilotage de l'application PAYSAGE en recensant les maintenances qui doivent être appliquées, les tickets répertoriant les problèmes trouvés lors de la réalisation d'une maintenance ou suite à sa mise en production mais aussi tous les documents qui leur sont liés.

Elle sert à tracer les différentes phases de chaque palier du projet PAYSAGE en suivant la bonne réception, le statut, l'état d'avancement et l'exactitude des différents livrables (EIB, EIF, SFD et CTG). Un palier est un découpage calendaire correspondant aux dates de mise en production de l'application PAY-PAYSAGE. PRISME permet aussi de faire un suivi complet du réalisé sur le projet PAYSAGE en alimentant l'application des charges de travail effectuées sur les différentes évolutions, et à avoir une indication de l'avancement de chaque phase grâce à des tableaux de bord dédiés composés de diverses restitutions graphiques (camembert, barres).

Elle répertorie aussi tous les agents qui participent ou ont participé au développement de l'application PAYSAGE et permet, notamment, à la MOA de remplir un formulaire décrivant une maintenance avec des détails comme sa priorité ou encore son palier d'application.



Prio.	IT	Réf.	Type	Statut EIB	Elt ext	Statut SFD	Cas de test	FQR	R/Q	Bloq.	Recette	Décision	Palier	Priorité
001	1	25-001	M	Émise, complète	☐	Émise, complète	Non reçus	416637	11/12	Non	Non testée	Aucun changement	PM2507	
002	2	25-041	M	Émise, complète	☐	Émise, complète	Non reçus	418510	14/14	Non	Non testée	Aucun changement	PM2507	
003		25-046	M	Émise, complète	☐	Émise, complète	Non reçus	420935	1/1	Non	Non testée	Aucun changement	PM2507	
004		25-047	M	Émise, complète	☐	Attendue	Non reçus			Non	Non testée	Aucun changement	PM2507	
005		25-048	M	Attendue	☐	Attendue	Non reçus			Non	Non testée	Aucun changement	PM2507	
006		407067	BPG		☐						Non testée	Aucun changement	PM2507	
007		417889	BPG		☐						Non testée	Aucun changement	PM2507	
008	1	420383	BPG		☐						Non testée	Aucun changement	PM2507	
009	1	418596	BPG		☐						Non testée	Aucun changement	PM2507	
010		403646	BPG		☐						Non testée	Aucun changement	PM2507	
011		413449	BPG		☐						Non testée	Aucun changement	PM2507	

Page dédiée pour la MOA

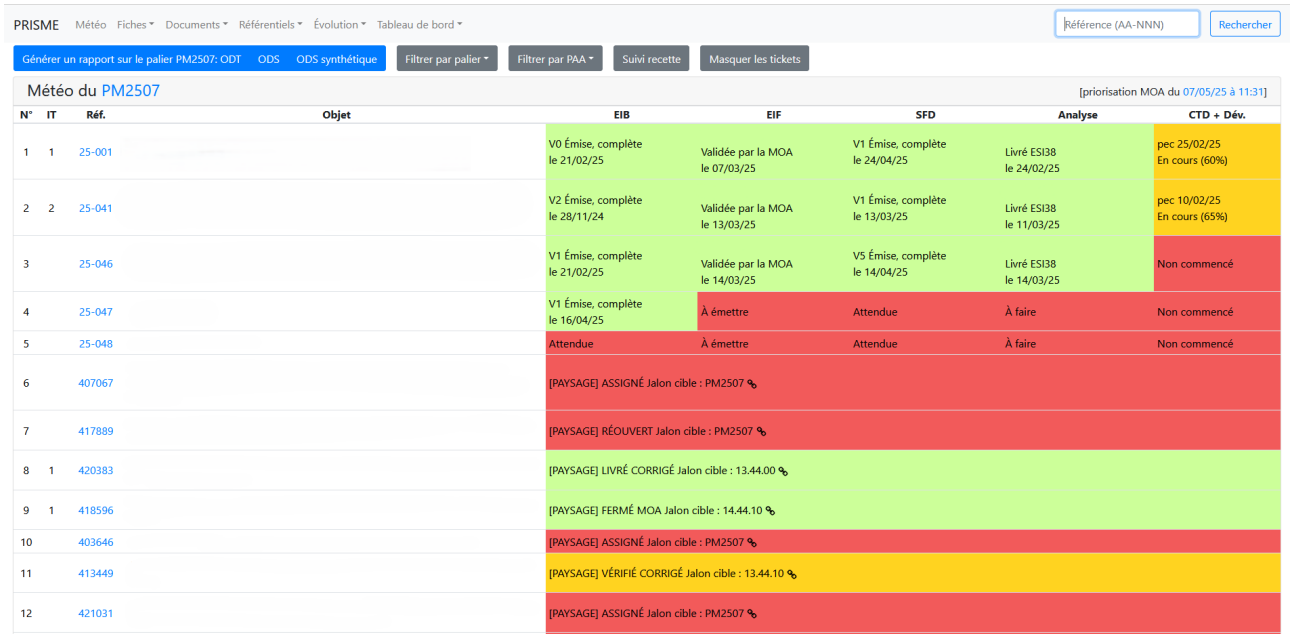
Elle permet aussi d'indiquer les tickets qui doivent être pris en charge sur le palier dont l'avancement et la date de finalisation est renseignée par l'équipe de développement via un formulaire dédié.

L'une des pages principales de l'application est la page appelée « Météo » qui consiste à afficher l'ensemble des maintenances et des tickets qui ont été priorisés sur les différents paliers. Sur chaque maintenance est indiqué les informations permettant de suivre les différentes phases :

- expression initiale du besoin (EIB) ;

- étude d'impact et de faisabilité (EIF) ;
- spécifications fonctionnelles détaillées (SFD) ;
- conception technique générale (CTG) ;
- développement (DEV).

Elle indique aussi l'avancement des développeurs sur les maintenances. Pour chaque ticket ouvert la page indique l'état d'avancement de celui-ci (nouveau, assigné, corrigé, vérifié, livré ou fermé). La page « Météo » permet également de visionner les paliers précédents et/ou suivants. Sur cette page on peut aussi exporter les informations présentes sous forme de document texte (ODT) et de tableur (ODS) détaillé ou de synthèse.



N°	IT	Réf.	Objet	EIB	EIF	SFD	Analyse	CTD + Dév.
1	1	25-001		V0 Émise, complète le 21/02/25	Validée par la MOA le 07/03/25	V1 Émise, complète le 24/04/25	Livré ESI38 le 24/02/25	pec 25/02/25 En cours (60%)
2	2	25-041		V2 Émise, complète le 28/11/24	Validée par la MOA le 13/03/25	V1 Émise, complète le 13/03/25	Livré ESI38 le 11/03/25	pec 10/02/25 En cours (65%)
3		25-046		V1 Émise, complète le 21/02/25	Validée par la MOA le 14/03/25	V5 Émise, complète le 14/04/25	Livré ESI38 le 14/03/25	Non commencé
4		25-047		V1 Émise, complète le 16/04/25	À émettre	Attendue	À faire	Non commencé
5		25-048		Attendue	À émettre	Attendue	À faire	Non commencé
6		407067		[PAYSAGE] ASSIGNÉ Jalon cible : PM2507				
7		417889		[PAYSAGE] RÉOUVERT Jalon cible : PM2507				
8	1	420383		[PAYSAGE] LIVRÉ CORRIGÉ Jalon cible : 13.44.00				
9	1	418596		[PAYSAGE] FERMÉ MOA Jalon cible : 14.44.10				
10		403646		[PAYSAGE] ASSIGNÉ Jalon cible : PM2507				
11		413449		[PAYSAGE] VÉRIFIÉ CORRIGÉ Jalon cible : 13.44.10				
12		421031		[PAYSAGE] ASSIGNÉ Jalon cible : PM2507				

Page Météo de PRISME

PRISME met à jour automatiquement les tables ticket et FQR toutes les 30 min en faisant des requêtes via l'API[9] de Bugzilla, l'outil utilisé pour le suivi des tickets où ils sont créés initialement afin de répertorier les problèmes rencontrés sur les différents projets. Cette mise à jour va actualiser la table ticket en ajoutant uniquement les tickets qui n'existent pas dans la base de données et/ou en modifiant ceux qui existent déjà dans l'application PRISME. Cette mise-à-jour peut aussi créer de nouveaux acteurs s'ils n'apparaissent pas dans l'application. PRISME va récupérer et traiter les nouvelles FQR qui ont été ajoutées à Bugzilla et modifier celles déjà existantes.

9 Interface logicielle.

Finalité de l’alternance

Le but de mon travail est de collaborer avec mon tuteur Oliver MADERN afin d’ajouter de nouvelles fonctionnalités à l’application PRISME permettant de faciliter et d’enrichir son usage pour tous ses utilisateurs. PRISME permet à ses utilisateurs de gagner beaucoup de temps en évitant la consultation de multiples outils, chacun nécessitant des accès et habilitations dédiées, et le croisement manuel de leurs informations. La DGFiP m’a engagé pour réaliser les maintenances de l’application PRISME, améliorer le modèle de données ainsi que les fonctionnalités utilisées par les utilisateurs.

Architecture de PRISME

Développée en Java, PRISME utilise des frameworks[10] modernes comme Spring Boot pour le back-end[11] et BootStrap pour la partie affichage utilisateur avec des pages en Jakarta Server Pages (JSP), lui garantissant une interface claire et un développement facile. L'application est connectée à une base de données PostgreSQL[12], dans laquelle sont centralisées les informations nécessaires au suivi des évolutions fonctionnelles. C'est une architecture 3 tiers, il y a la couche persistance (base de données), la couche applicative (traitement, service…) et la couche client (navigateur) :

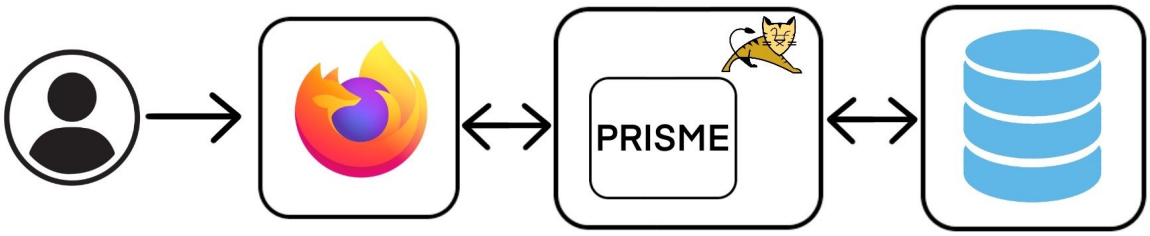


Schéma architecture 3 tiers

L'application PRISME s'appuie sur une base de données relationnelle structurée autour de 35 tables (schéma complet cf « Annexe 4 : Schéma PRISME »), organisées de manière à centraliser l'ensemble des informations nécessaires au pilotage :

- La gestion des acteurs, c'est-à-dire l'ensemble des personnes intervenant dans le projet (MOA, MOE, développeurs, analystes, etc.) ;
- Le suivi des évolutions ;
- Le référencement des différents documents de cadrage et de suivi, tels que les EIB (Expression Initiale du Besoin), les EIF (Étude d'Impact Fonctionnelle), les CTG (Conception Technique Générale), ou encore les SFD (Spécification Fonctionnelle Détaillée) ;
- Le suivi des tickets, qui permettent de tracer les anomalies ou demandes d'évolution ;
- Les FQR (Fiches Questions/Réponses) rattachées aux tickets ;
- La priorisation des travaux à effectuer, reflétant les objectifs visés par la MOA ;

10 Modules aidant au développement d'applications.
11 Partie non visible par les utilisateurs d'une application.
12 Système de base de données.

- Diverses tables de jointure, assurant la cohérence des relations entre les acteurs, les documents, les maintenances et les tickets.

Les versions des composants utilisées dans l'application PRISME ne sont pas les dernières sorties, car la DGFiP doit approuver une version avant de l'autoriser pour la production. Pour valider une version, il faut :

- qu'elle ne contienne pas d'erreurs contraignantes ;
- qu'elle ne possède pas de failles de sécurité connues ;
- qu'elle soit stable et maintenue dans le temps.

De ce fait la version de Java utilisée est OpenJDK 11, celle de PostgreSQL est PostgreSQL 14. L'application utilise des frameworks afin de faciliter son développement : pour le côté back-end, le framework utilisé est Spring Boot avec sa version de 2.1.7.RELEASE et pour le côté WEB un des frameworks utilisé est BootStrap avec sa version 4.3.1.

TECHNOLOGIES

Outils de développement et de déploiement

Outil	Version
Eclipse	2020-06 (4.16.0) (version DGFiP)
Maven	3.9.9
Forge interministériel (Git)	v17.11.3
PgAdmin	9.1

Matrice technologique

Nom	Version	Fonction
Ubuntu	24.04	Système
Tomcat	9.0.22	Serveur d'application
PostgreSQL	14.13	Base de données
OpenJDK	11	
Spring Boot	2.1.7.RELEASE	
BootStrap	4.3.1	
ODF Toolkit	1.0.0-BETA1	
Hibernate	5.3.10	
Chartjs	2.9.3	
Chartjs-plugin-Datalabels	4.3.0	
Jodconverter-local	4.3.0	
Java Standard Tag Library (JSTL)	1.2	
Javax.persistence-api	2.2	
Validation-api	2.0.1	

Base de Données

PostgreSQL est un système de base de données relationnelle orienté objet[13], open source[14], puissant et très utilisé. Il possède de nombreuses fonctionnalités aidant les développeurs à créer leur application. Il permet de gérer des données de manière fiable qu'elle que soit leur volumétrie.

PostgreSQL a été choisi, parmi les systèmes de base de données proposés par le SILL[15], pour sa facilité d'utilisation et de mise en place dans une application Java. Il correspond parfaitement au cas de PRISME étant une application interne visible uniquement par des agents de la DGFiP. Il est compatible avec beaucoup de framework Java notamment Spring Boot, qui est utilisé sur l'application.

La version PostgreSQL 14 est celle utilisée pour PRISME. Cela permet d'avoir accès à beaucoup de fonctionnalités sur les requêtes SQL et permet aussi d'avoir une bonne performance sur certaines requêtes réalisées. Elle est aussi toujours supportée et maintenue.

Au sein de la DGFiP, l'outil PgAdmin est utilisé pour interagir avec les bases de données PostgreSQL pour administrer de multiples serveurs et leurs bases de données. Elle permet toutes les interactions avec les bases mais en version graphique (faire des dumps[16] ou encore pouvoir les importer rapidement). Elle permet aussi de voir la structure des tables et leurs informations. PgAdmin permet aussi de créer des profils utilisateurs pouvant exécuter des scripts facilement selon le profil voulu.

13 Programme s'articulant autour de concepts.
14 Source accessible gratuitement et par tous.
15 Socle Interministériel des Logiciels Libres
16 Script comportant des informations pour recréer une base de données.

Back-End

PRISME est une application en Java sous OpenJDK 11. Pour faciliter son développement, l'application utilise le framework Spring Boot.

Spring Boot est un framework open source basé sur Spring. Il permet une création simple et rapide de projet en configurant automatiquement certains paramètres de Spring ne laissant que peu à remplir. Il est compatible avec beaucoup de modules tels que les bases de données PostgreSQL et la partie WEB en BootStrap. Ce framework est très pratique et simple d'utilisation avec une bonne documentation. Il possède aussi de nombreux guides pour aider les développeurs qui voudraient commencer une application de manière autonome.

Il possède également de nombreuses fonctionnalités de bases permettant de démarrer rapidement la création d'une application :

- un simple ajout de classe « Entity » crée automatiquement la table correspondante dans la base de données : langage de définition des données (DDL) ;
- des requêtes SQL générées automatiquement par l'application : langage de requête de données (DQL) et langage de manipulation des données (DML) ;
- une structure de données explicite pour faciliter la compréhension de lecture de code ;
- une documentation claire et précise.

Néanmoins Spring Boot possède quelques défauts comme des messages d'erreurs peu précis mais pouvant être corrigés, car le framework jouit d'une forte popularité, et une recherche internet permet de trouver rapidement une réponse.

Comme cité précédemment, il s'agit d'un framework qui a besoin de peu de configuration manuelle. Facile d'utilisation, bien documenté avec beaucoup d'aide, ce sont les raisons pour lesquelles l'initiateur du projet, Olivier MADERN, l'a choisi.

En plus d'être compatible avec PostgreSQL, il est aussi compatible et simple d'utilisation avec BootStrap qui est un framework pour le côté WEB et celui utilisé sur PRISME.

Partie WEB

Pour la partie statique, l'application utilise le framework BootStrap. BootStrap est l'un des frameworks open source les plus populaires et puissant du monde. Il possède beaucoup de guides pour aider au développement des applications. C'est un framework facile d'accès et facile d'utilisation étant donné sa renommée. Il permet une mise en place rapide de pages WEB pour une application avec du CSS[17], des scripts JavaScript et une utilisation simple avec Spring Boot. Il a aussi de nombreuses fonctionnalités déjà implémentées permettant l'interaction avec différents modules d'application.

BootStrap possède de nombreux avantages comme :

- un large choix de personnalisation d'architecture ;
- une construction rapide ;
- des variables CSS permettant une personnalisation visuelle de l'application plus large avec en plus des accès à des icônes personnalisées fournies par BootStrap ;
- mise en place d'une API utilitaire ;
- de puissants plugins[18] JavaScript.

Les pages WEB de PRISME sont des pages en Jakarta Server Pages (JSP). Ce sont des pages WEB dynamiques qui permettent d'introduire du code Java dans des balises d'une page HTML. L'avantage du JSP est qu'il est basé sur le langage Java et donc permet de l'implémenter dans une application Java plutôt facilement, principalement via l'utilisation des bibliothèques JSTL[19]. JSP permet aussi une utilisation simple du framework Spring Boot directement sur la page en question grâce à des balises de type « <spring> » (voir JSTL dans la partie suivante).

Par contre, pour fonctionner, la partie dynamique est dépendante d'un serveur d'application comme Tomcat pour compiler et s'exécuter. Tomcat est un conteneur WEB pour des applications.

17 Langage de personnalisation graphique pour les sites WEB.
18 Programme additionnel.
19 Java Standard Tag Library.

Dépendances Supplémentaires

PRISME possède des dépendances en plus des principales citée précédemment :

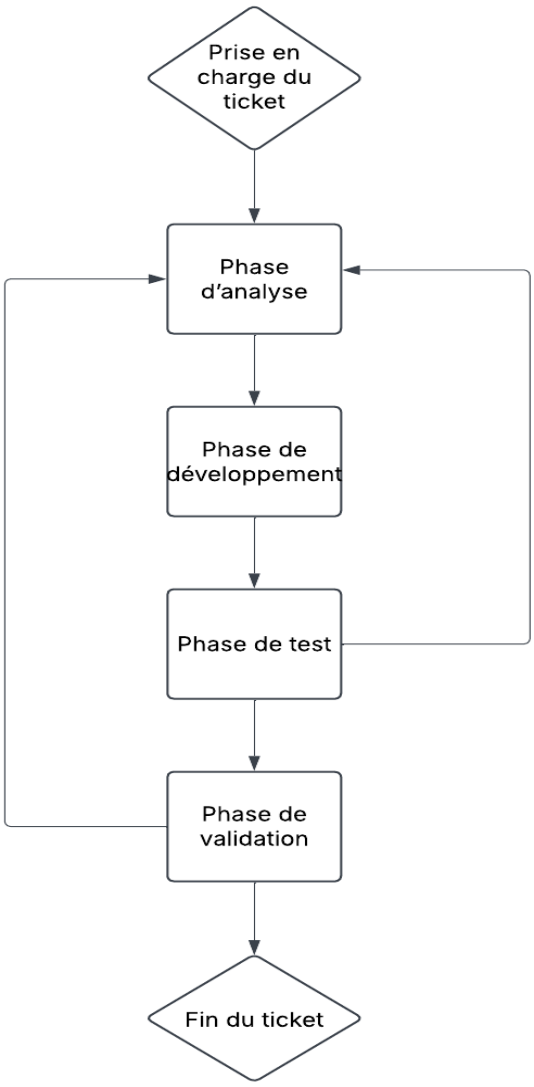
- ODF Toolkit : ensemble de modules Java fournissant des fonctions pour créer, lire et écrire des fichiers open document (star office/libre office/open office) ;
- Hibernate : framework Java permettant de relier les classes Java aux tables de base de données et de simplifier la gestion de la persistance et des requêtes relationnelles agnostique au niveau SGBD ;
- Chartjs : bibliothèque JavaScript permettant de créer facilement des graphiques interactifs en utilisant l'élément HTML5 canvas [20];
- Chartjs-plugin-Datalabels : plugin pour Chart.js qui permet d'afficher des étiquettes de données directement sur le graphique ;
- Jodconverter-local : librairie Java permettant d'automatiser la conversion de documents en utilisant LibreOffice en mode local ;
- Tomcat : serveur Web, écrit en Java, servant d'environnement d'exécution pour les servlets[21] et implémente les spécifications Jakarta Servlet[22] et JSP, permettant de gérer des applications Web dynamiques ;
- JSTL : dépendance Maven qui fournit la Java Standard Tag Library (JSTL), une bibliothèque de balises pour faciliter la création de pages JSP ;
- Javax.persistence-api : API Java fournissant la spécification standard pour la gestion de la persistance des données en utilisant JPA (Java Persistence API), permettant de relier des objets Java à des bases de données relationnelles ;
- Validation-api : API Java utilisée pour la validation des contraintes, fournissant des interfaces telles que Validator pour les appliquer.

20 Conteneur HTML permettant de créer des applications WEB dynamiques.
21 Composant qui permet de créer dynamiquement des données sur un serveur HTTP.
22 Composant côté serveur qui gère les requêtes HTTP.

PLAN D'ACTION

Durant cette deuxième année, mon plan d'action pour réaliser les travaux qui me sont confiés n'a pas changé et est composé de 4 phases :

- phase d'analyse ;
- phase de développement ;
- phase de test ;
- phase de validation.



Cycle de vie d'un ticket

Phase d’analyse

La phase d’analyse commence lorsque je reçois un ticket sur le dépôt du projet, pouvant être de différentes natures. Le ticket peut être une refonte de la base de données, une correction, une amélioration et/ou un ajout/modification d’une fonctionnalité déjà présente. Toutes mes phases d’analyse commencent de la même manière : je commence par poser le problème (le comprendre, savoir quelle partie de l’application devra être modifiée, les effets de bords que cela peut engendrer…).

Durant cette phase, en cas de doute ou d’incompréhension sur certains points, j’échange de vive voix avec la personne ayant ouvert le ticket ou par les commentaires présents sur la forge.

C’est une phase qui a évolué depuis mon année précédente en d’alternance. Plus fort de connaissances face à des demandes de moins en moins détaillées, je traite le ticket en autonomie totale ce qui me demande plus de temps pour l’analyser. Les interactions avec les acteurs sont plus importantes.

Une fois cette phase terminée, je passe à l’étape suivante.

Phase de développement

La plupart du temps, la phase de développement est traitée à mon poste de travail, sur les applications Eclipse et PgAdmin. Il se peut que durant cette phase, je rencontre des problèmes de code que je résous en m’appuyant sur de la documentation. Si, malgré mes recherches, le problème de code persiste, alors je m’adresse à un de mes tuteurs pour des explications.

Phase de test

Puis vient la phase de tests où je passe la majorité de mon temps à essayer diverses fonctionnalités de l’application. Je m’assure alors que les changements effectués ont bien été ajoutés sans altérer les autres fonctionnalités et résultats déjà présents dans l’application. Si le résultat n’est pas celui attendu, je redémarre un cycle complet d’analyse.

Phase de validation par mon tuteur

Enfin vient la phase de validation de mon travail. Une fois le ticket corrigé sur ma branche de travail dédiée au ticket, j'attends le retour de mon tuteur Oliver MADERN qui valide ou non mon travail. En cas de non validation, des commentaires sont ajoutés sur la requête de fusion afin de le documenter, puis je reprends le ticket pour de le corriger. Dans le cas où tout est bon et qu'il n'y a pas de problème, alors mon travail est fusionné sur la branche principale du projet.

Puis je prends en charge un autre ticket pour faire un nouveau cycle de travail.

MISSIONS

Cette année, j'ai eu accès à l'environnement de production afin de pouvoir déployer moi-même les modifications et d'exécuter des scripts sur la base de données. J'ai eu beaucoup plus d'autonomie sur le projet en commençant par la possibilité de travailler seul sur les modifications mineures, principalement sur des corrections d'anomalies et de dysfonctionnement. Pour des tickets plus importants, j'attendais la validation de mon tuteur avant de le déployer.

J'ai aussi pu participer à des réunions ponctuelles d'un autre projet, Paye Zéro Papier (POP), sur lequel mon tuteur m'a demandé mon avis sur le modèle de données de l'application POP. Elles étaient hors périmètre de ma mission mais intéressantes professionnellement.

Les missions que je détaille dans ce rapport concernent PRISME : si un problème survient sur l'application ou que cette dernière a besoin de modifications, une demande est faite sur le dépôt du projet présent sur la forge interministérielle. Ces demandes sont représentées par des tickets et leur traitement peut être plus ou moins long selon la tâche demandée. La majeure partie des tickets que j'ai pris en charge cette année sont des tickets d'amélioration et des tickets de corrections (cf « Annexe 5 : Tickets résolus sur les 3 derniers mois »).

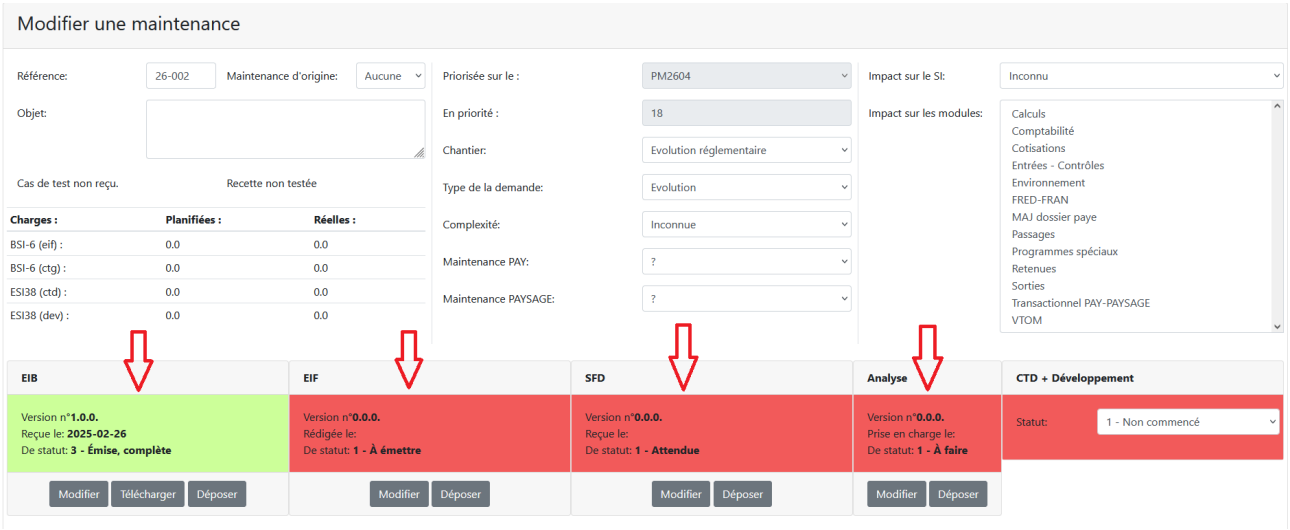
Missions d'améliorations

Durant cette année d'alternance, j'ai réalisé de nombreuses missions d'améliorations. Ce type de missions consiste à améliorer aussi bien le visuel que l'interaction avec certaines parties de l'application. Pour effectuer ces améliorations je devais faire des changements au niveau du back-end et/ou du la partie WEB. C'est le type de missions que j'ai le plus réalisées durant cette année en alternance. Ce sont des missions assez courtes dans le temps.

La première mission que je vais décrire est l'interaction entre PRISME et CORE. Cette amélioration consiste à modifier la manière dont PRISME récupère les informations depuis l'application CORE pour mettre à jour les charges effectuées par les acteurs, sur PRISME. J'ai dû effectuer ce changement afin d'acheminer le projet sur le cloud interministériel. Ce changement a aussi permis d'ajouter une historisation des mises à jour des charges sur PRISME. Les documents déposés sont téléchargeables depuis la page des fichiers.

Page d'historique des dépôts CORE

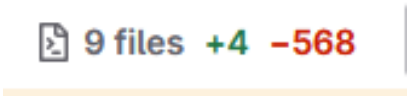
Autre amélioration que j'ai apportée sur l'application : la création et le renseignement automatique des différentes phases (EIB, EIF, SFD et CTG) associées à une nouvelle maintenance qui permet un gain du temps pour les utilisateurs. Elle limite des potentiels erreurs ou oublis et évite la création manuelle et répétée d'élément techniques ne portant pas de données fonctionnelles.



Page d'une maintenance avec l'information des documents liés

Un ticket que j'ai également résolu pour ce type de mission consiste à changer la manière dont les fichiers sont déposés sur PRISME. Avant il était possible de déposer un fichier directement via la page « liste fichier » de l'application. Ce dépôt enregistrait le document dans un dossier « documents_sans_maintenance » qui était inutile et n'avait pas vraiment de sens au niveau fonctionnel. Désormais il est donc impossible de déposer un fichier depuis cette page, si un fichier doit être déposé, il doit se faire sur la page du type de documents dédiés afin d'enregistrer proprement sur la machine hôte de l'application.

Une autre mission effectuée concernait la simplification du code visant à améliorer l'application et de regarder toutes les fonctionnalités obsolètes pour les retirer en vérifiant que toutes les pages fonctionnent. Cette mission m'a permis de faire une vérification sur toutes les pages présentes sur PRISME, de vérifier si tout fonctionnait correctement et d'échanger avec mon tuteur afin de savoir si les fonctionnalités que j'ai identifiées comme obsolètes l'étaient réellement. Souvent j'essaie de simplifier le code en même temps que je développe mais pour ce ticket plusieurs fonctions existaient avant mon arrivée sur le projet. Cela a permis de retirer plus de cinq cent cinquante lignes de code.



Exemple de simplification de code

Il y a aussi eu un ticket concernant l'optimisation, car, suite à une évolution récente, l'application PRISME prenait 5 à 6 secondes à afficher la page « Météo » qui fait partie des pages qui chargent le plus vite (moins d'une seconde pour l'afficher en temps normal). Pour corriger cela j'ai, dans un premier temps, regardé les différentes fonctions qui pouvaient être appelées afin d'analyser leur complexité, puis, je me suis rendu compte que le problème venait que la base de données était trop sollicitée. Ce problème était causé par le framework Spring Boot. Sur Spring Boot, il est possible d'avoir les informations des tables qui référencent une autre table en faisant des liaisons inverses. Malheureusement si ces liaisons sont d'un certain type (« OneToOne »[23]), cela génère des requêtes en trop. L'annotation JPA « OneToOne » a deux fonctionnements :

- par défaut : l'entité qui possède la clé étrangère dans la base de données ;
- inversé (via le paramètre « mappedBy ») : l'autre entité qui référence la relation, elle ne possède pas la clé étrangère en base de données mais la référence sur l'application.

Ce deuxième point crée des requêtes SQL pour charger les tables qui n'étaient pas référencées sur les requêtes d'une page. Ce ticket est encore ouvert, car il est toujours possible d'améliorer l'efficacité de certaines requêtes mais la modification décrite au-dessus était une demande pour laquelle il a été spécifié de régler cela le plus rapidement possible pour ne pas ralentir le travail.

```
8x : select * from public.maintenance maintenanc0_ where maintenanc0_.id_evolution=?
8x : select * from public.modernisation modernisat0_ where modernisat0_.id_evolution=?
8x : select * from public.ticket ticket0_ where ticket0_.id_evolution=?
```

Exemple de requêtes qui s'exécutaient en trop

Une des missions que j'ai pris en charge mais qui va demander beaucoup de temps à finaliser est la montée de version de toutes les dépendances de PRISME. Les dépendances utilisées sont soit dans des versions trop anciennes, soit en version bêta qui sont maintenant dépassées. Pour les frameworks majeurs telles que Spring Boot, la montée de version implique des changements notamment la version de Java utilisée, ainsi que les méthodes utilisées pour interagir avec le framework.

Cette mission est clairement complexe avec une phase d'analyse beaucoup plus complète et qui risque de prendre beaucoup de temps. La première étape est la lecture de documentation pour faire les montées de version. Puis la prise de note de tous les changements que cela pourrait impliquer. *Par exemple PRISME utilise la version de Spring Boot 2.1.7.RELEASE. La bascule vers 3.4.4 répondrait à la demande mais cela nécessite au*

23 Annotation utilisée pour modéliser une relation un-à-un entre deux entités dans une application.

préalable de passer dans la dernière version de 2.7.X. Entre la version actuelle et la version intermédiaire (2.7.X) il y a déjà de nombreuses modifications à faire au niveau du code. Ensuite pour passer en version 3.4.4, ce changement impacte la version de Java utilisée puisqu'il faut au moins Java 17 avant de pouvoir faire la transition et de même que pour la montée de version précédente il y aura beaucoup de changements au niveau du code.

Les principaux problèmes rencontrés ici sont de comprendre la demande du mieux possible et de rendre le travail le plus propre possible, aussi bien au niveau du code, qu'au niveau visuel pour l'utilisateur. Pour cela, je communique souvent avec les créateurs des tickets pour avoir toutes les informations nécessaires à la résolution de ces tickets.

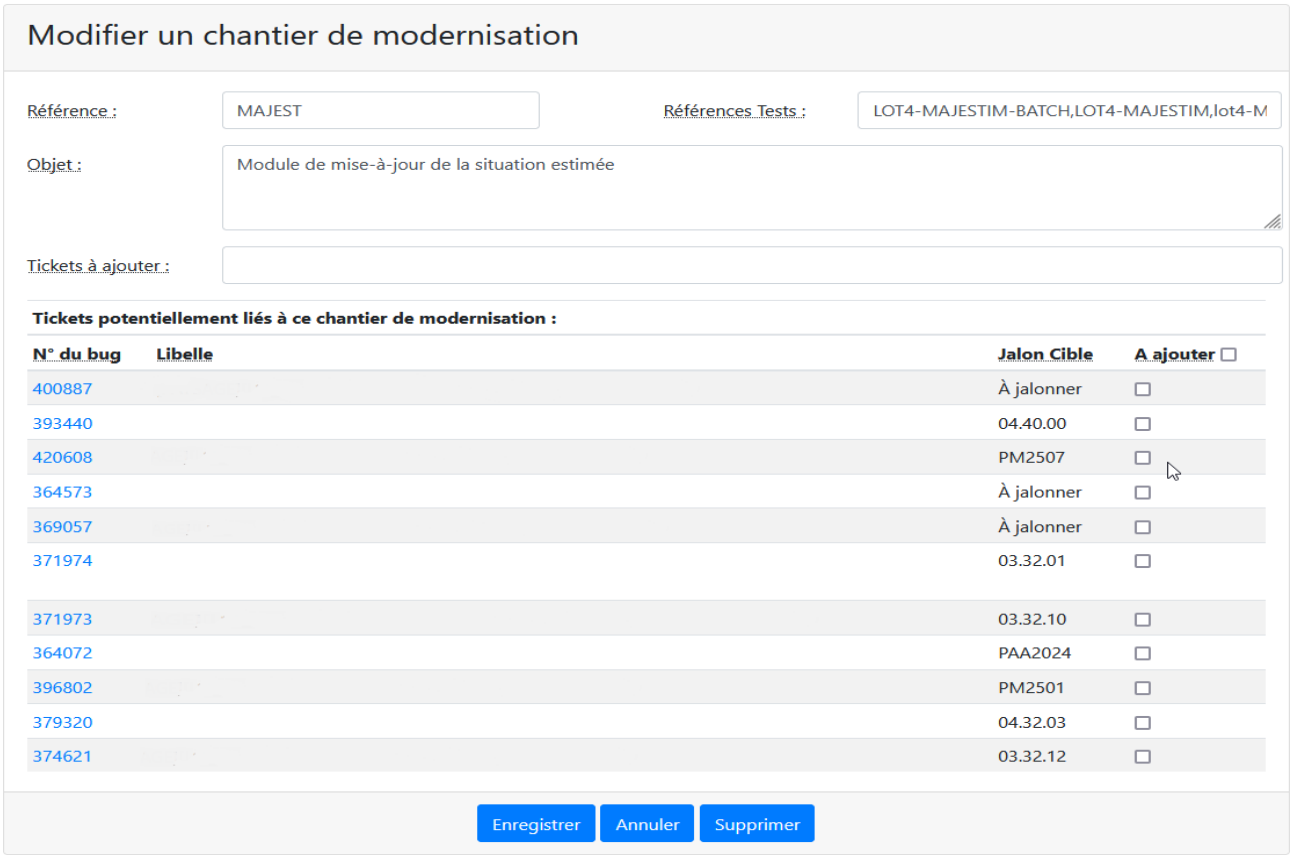
Missions de fonctionnalités

La majorité des tickets décrivant mes missions possèdent l'étiquette « fonctionnalité » et « amélioration » car si une fonctionnalité est ajoutée, l'application est améliorée. Ici je fais la distinction entre les deux entités car le type « fonctionnalité » correspond plus à une amélioration fonctionnelle alors que « amélioration » répond à une optimisation technique.

La première mission que je vais citer ici a plusieurs étiquettes, car en plus d'avoir celle indiquant une mission de fonctionnalité elle a aussi celle indiquant que c'est une amélioration. Elle consiste à suivre l'avancement de chantiers de modernisation sur PRISME. Les chantiers de modernisation sont des améliorations apportées sur l'application PAYSAGE distinctes des tickets ou des maintenances évolutives. J'ai développé une première partie de ce ticket qui est actuellement sur l'environnement de production. Le ticket a été divisé en trois parties, suite à une présentation faites aux acteurs concernés et à leurs retours, afin de faire les modifications et l'intégration progressivement.

Cette première partie consistait à rajouter un type d'évolution à suivre sur PRISME, les chantiers de modernisation. Pour cela j'ai dû modifier la base de données en rajoutant des tables, et adapter les interactions de PRISME. Actuellement, il existe des chantiers de modernisation sur lesquels des tickets peuvent être liés, et qui peuvent être priorisés sur un palier. Cet ajout change certaines actions qui sont déjà présentes sur PRISME, tel que l'ajout de ticket car si un chantier de modernisation possède un ticket, il n'est pas possible que ce dernier soit priorisé sur un palier pour être suivi.

La deuxième partie fait majoritairement des ajustements visuels suite à la présentation de ce qui a été ajouté. Le premier changement est l'affichage des tickets liés à un chantier de modernisation, qui sont maintenant séparés par palier sur l'affichage de la page de ce dernier. L'ajout d'un chantier a aussi été amélioré en évitant de marquer son nom sur la fiche MOA, mais en le choisissant dans une liste déroulante avec une indication du nombre de tickets liés qui ont pour cible ce palier. Suite à cet ajout, la page « Météo » peut maintenant dérouler un chantier de modernisation afin d'afficher tous les tickets liés à ce palier. Ces modifications sont principalement là pour faciliter la lecture et l'interaction avec ces chantiers.



Page d'un chantier de modernisation

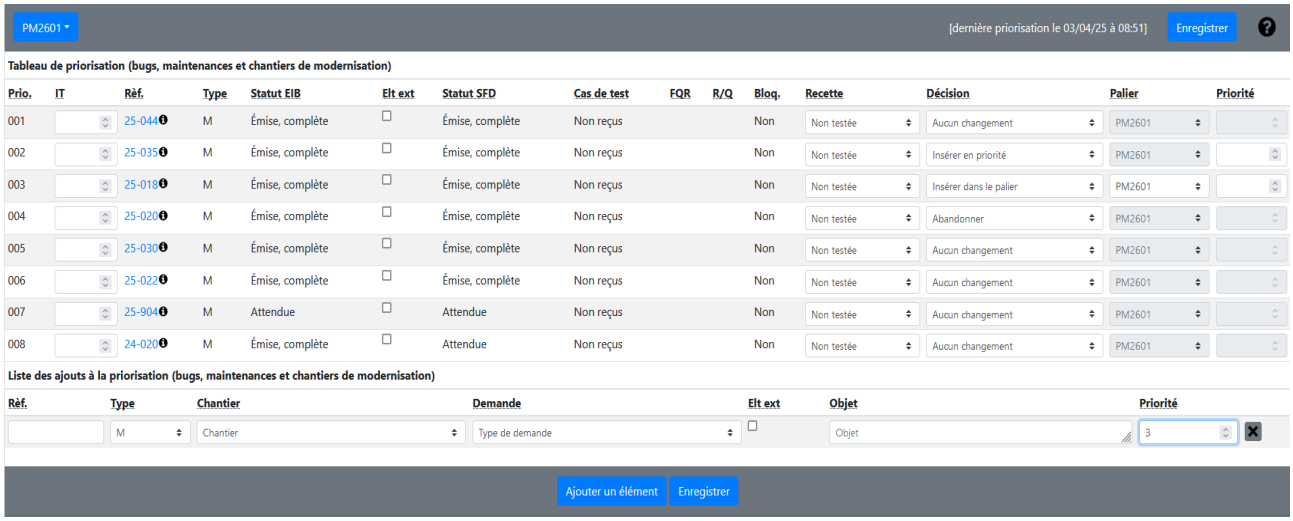
Cette mission reste encore ouverte à des changements, dans le cas où une amélioration légère peut être intégrée sans faire de modifications fonctionnelles majeures. Par exemple pour ce ticket il m'a été demandé de mettre en avant des informations complémentaires en enrichissant les pages avec des liens et de nouvelles données.

J'ai aussi codé, suite à une demande, un changement d'affichage sur la page « Météo » pour les évolutions qui sont en itération 9. Les itérations correspondent aux différentes versions applicatives qui vont être livrées et la 9 est une itération fictive, donc elle sert pour mettre à l'écart des demandes qui ne pourront pas être réalisées. Pour ce ticket, j'ai dû souvent échanger avec le demandeur afin de se mettre d'accord sur l'affichage voulu et savoir s'il existait une possibilité que ce changement d'affichage dérange d'autres acteurs. Ce ticket est assez court puisqu'il ne fait qu'un changement de couleur sur l'affichage de la page, et le back-end lui n'est pas impacté par ce changement.

Une autre mission de fonctionnalité que j'ai réalisée, mais qui est aussi étiquetée comme mission d'amélioration, consiste à améliorer la lecture automatique des bordereaux de livraison. Un bordereau de livraison est un document qui indique les évolutions (maintenances évolutives et tickets) qui sont incluses dans la livraison visée.

Sur cette mission j'ai dû modifier la base de données afin d'ajouter des informations concernant les livraisons. J'ai aussi fait des modifications d'affichage afin que toutes les informations puissent être affichées sur la page décrivant une livraison. Les modifications apportées permettent de suivre plus de modules[24] (seulement le module applicatif et le module de données étaient tracés avant) et de modifier les données d'une livraison plus facilement. J'ai aussi codé la lecture des bordereaux de livraison afin d'y extraire les informations voulues avec la bibliothèque ODF-TOOLKIT. Dans l'annexe, il y a une capture d'écran d'une page d'une livraison montrant tout ce qui peut être tracé maintenant sur PRISME (cf « Annexe 6 : Page d'une livraison »).

Sur PRISME, j'ai changé la méthode d'ajout d'évolution sur la fiche MOA en simplifiant son utilisation, pour ajouter une évolution, il était nécessaire de la prioriser en fin de palier, puis de la changer de priorité une fois qu'elle a été ajoutée. J'ai mutualisé ces fonctions pour les regrouper en une seule qui permet d'ajouter une évolution directement à la priorité souhaitée. J'ai aussi limité les erreurs possibles lors d'une modification sur la fiche MOA, en ne donnant accès qu'à certains champs seulement si le bon changement à appliquer est sélectionné, cela permet de ne plus faire de faute d'inattention dans le cas où la mauvaise ligne est modifiée, ou même d'indiquer les champs qui doivent être modifiés selon le changement choisi.



Fiche MOA avec décision (champs grisés non modifiables)

24 Partie de l'application PAY-PAYSAGE.

Pour ce type de ticket, les principaux problèmes rencontrés sont le temps de développement. Étant en alternance, je ne suis qu'une semaine sur deux en entreprise, ce qui rallonge les temps de développement de certains tickets, mais c'est plus rapide qu'avec le rythme de l'année dernière (deux jours en entreprise / trois jours à l'université). Ces tickets peuvent aussi inclure des présentations aux différents acteurs concernés afin de savoir si le travail que j'ai réalisé correspond à la demande, s'il y a d'autres modifications ou améliorations à apporter, ou si certaines parties sont inutiles.

Missions de corrections

Les missions de corrections consistent à résoudre un problème qui a été trouvé pendant l'utilisation de l'application. Ce sont généralement des missions courtes avec un objectif simple. Elles mettent souvent peu de temps à se résoudre, car ce sont souvent des erreurs sur une situation précise, généralement facile à reproduire et donc rapides à comprendre et à corriger. Ce type de missions est pris en charge en priorité afin de permettre aux acteurs de reprendre leurs travaux en cours sur PRISME le plus rapidement possible.

Un ticket que j'ai dû corriger est sur la lecture d'une FQR qui ne se mettait pas à jour sur PRISME alors qu'elle était modifiée côté Bugzilla. Le problème ici venait du fait qu'une FQR est un tableau où il y a les questions et les réponses, sauf que là, il y avait un autre tableau dans la réponse et PRISME ne lisait pas les FQR qui étaient composées de plus d'un tableau. Pour cela j'ai adapté l'existant pour permettre de prendre en compte toutes les FQR, peu importe leur format, et de lire directement le tableau qui nous intéressait.

Une autre mission de correction m'a été transmise concernant un affichage incorrect : quand un acteur n'est plus actif, il n'est plus du tout affiché même si c'est lui l'acteur responsable d'une modification. Normalement le but du champ actif sur un acteur était de ne plus les proposer dans les listes pour simplifier les modifications à faire les documents du type EIB, EIF, SFD et CTG. Pour régler ce problème, j'ai décidé de fournir directement cette information à la partie WEB avec une variable.

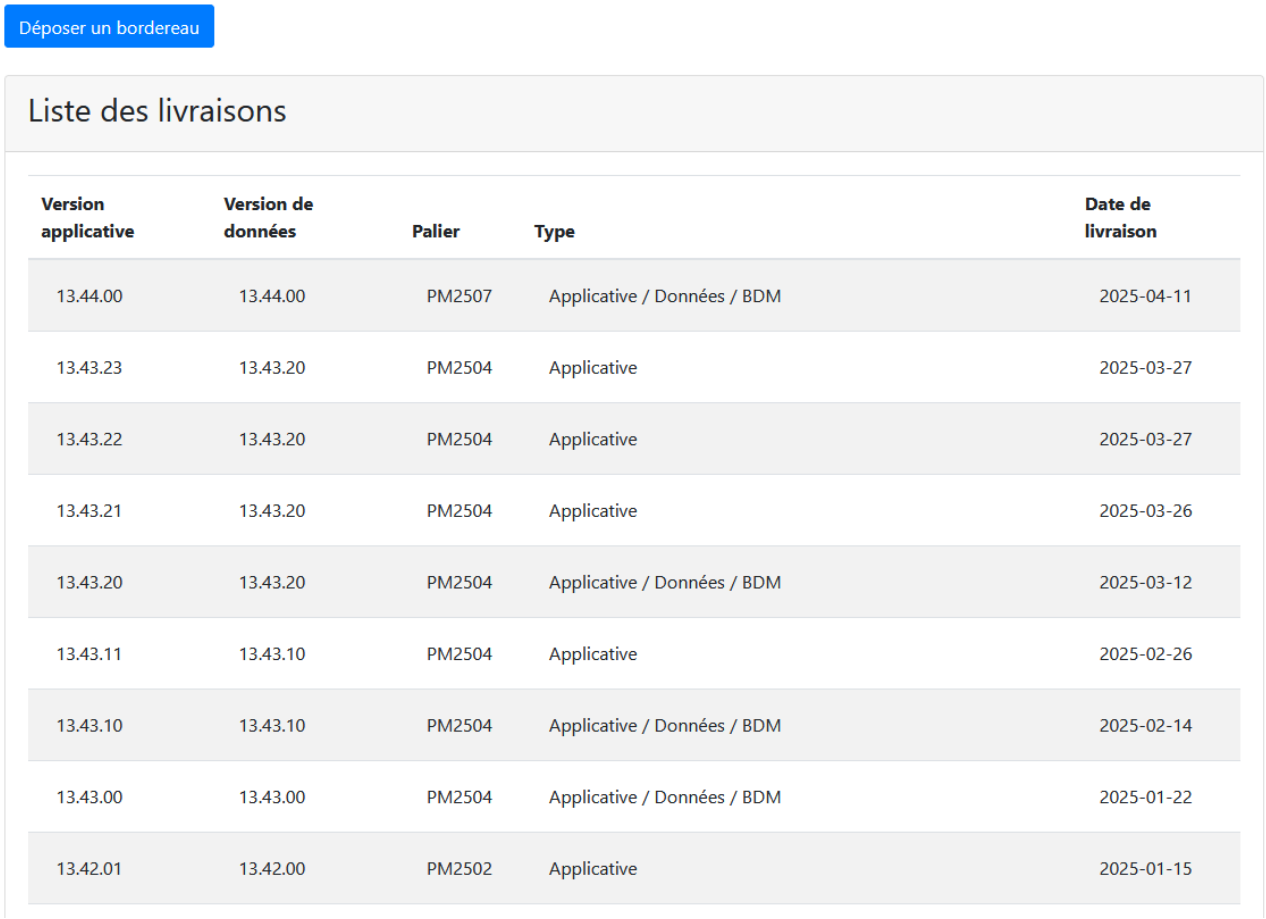
Un autre ticket que j'ai corrigé était une anomalie présente sur PRISME qui bloquait le changement de palier d'une évolution alors qu'il aurait dû passer tous les contrôles. Le problème venait d'un mauvais contrôle du côté du back-end à cause d'une imprécision sur une requête SQL. Avec l'aide de gestionnaires de traces, il a été résolu rapidement : tous les contrôles pour la fiche MOA sont faits sur une classe dédiée.

Durant cette année, j'ai aussi corrigé un problème mineur qui n'était pas présent à la création de l'application PRISME : la taille insuffisante des messages de commentaire que l'on peut mettre. Au départ, cette taille était de 200 caractères sauf que pour la première fois un commentaire avait besoin d'être plus long. J'ai modifié le nombre maximum de caractères sur la base de données, puis j'ai adapté le code afin qu'il corresponde au changement.

Le problème que j'ai pu rencontrer pendant les missions de corrections est la reproduction des erreurs sur mon environnement de développement afin d'y ajouter des traces me permettant de les comprendre et de les corriger. Le code à modifier est rarement un problème pendant ce type de mission. En général, elles se résolvent rapidement dans le temps sauf si un bug arrive sur une modification spécifique pour un élément en particulier. Travaillant sur un environnement de développement, ma base de données n'est pas tout le temps synchronisée avec celle de la production, donc cela se peut que je n'arrive pas à reproduire l'anomalie jusqu'à que je décide de synchroniser mes données avec celles de la production.

Mission d’alimentation de la base de données

Cette année en alternance j’ai pu réaliser une mission d’alimentation des données de l’application en production. Grâce à l’amélioration de la lecture automatique des bordereaux de livraisons, maintenant, il est possible de tracer facilement le contenu des livraisons (tickets corrigés, maintenances réalisées, modules livrés) afin qu’il puisse être indiqué sur PRISME. Plus de deux cent cinquante bordereaux de livraison ont été déposés, traités et vérifiés sur l’environnement de production pour que ces données puissent servir dans de futures améliorations. Cette mission m’a pris du temps, parce que chaque dépôt devait être contrôlé afin de vérifier les informations récupérées, car les documents déposés avaient des formats différents. La capture d’écran ci-dessous affiche quelques bordereaux de livraison que j’ai ajoutés sur PRISME depuis la mission de fonctionnalité concernant la lecture des bordereaux.



Page liste livraison

Pour l’instant c’est la seule mission de ce type-là que j’ai réalisée, puisqu’étant développeur, les seules fois où j’allais sur l’application en production, c’était pour vérifier que le redémarrage de l’application s’était fait correctement après un déploiement. Je n’ai pas de nouveaux tickets de ce type sur la forge mais, avec cette mission, cela m’a permis de travailler avec les données directement à jour. Sur mon environnement de développement, je ne récupère pas systématiquement les données de production, car la plupart du temps je n’en vois pas l’utilité puisque les modifications que j’apporte ne sont pas directement reliées à des données spécifiques.

Je n’ai pas rencontré de problème en effectuant cette mission. Elle a pris du temps car je devais vérifier chaque bordereau de livraison après son dépôt pour contrôler s’il ne manquait aucune information dessus. Il y a eu aussi une attention accrue car tout se déroulait sur l’environnement de production et je ne voulais pas commettre d’erreurs.

Mission actuelle

Ma mission actuelle est d'améliorer et de documenter l'application PRISME pour rendre l'application plus simple pour les utilisateurs et pour les développeurs en faisant du code propre. Pour cela j'ai fait la documentation pour les utilisateurs de la plupart des pages présentes sur l'outil PRISME. J'ai également codé de manière à ce que le code soit compréhensible pour les futurs développeurs.

Une présentation de l'application devra être faite afin de pouvoir expliquer le fonctionnement de l'application aux acteurs qui vont reprendre son développement, car en septembre mon tuteur Olivier MADERN, initiateur du projet, et moi-même ne pourront plus assurer son support (pour cause de fin de mission pour moi et de formation pour mon tuteur).

Actuellement, 17 tickets sont encore en attente sur PRISME, certains sont déjà commencés et ont besoin de compléments, d'autres ont besoin d'être spécifiés et les autres restent à analyser (cf « Annexe 7 : Page ticket de la forge »).

Conclusion

En somme, toutes les modifications que je code et que je déploie sur PRISME ont un but commun : rendre accessible et compréhensible l'application pour tous les acteurs qui utilisent l'application. PRISME est quotidiennement utilisé par l'ensemble des acteurs soit en simple consultation, soit pour effectuer la mise à jour des données présentes. Durant cette année d'alternance, il est déjà arrivé que PRISME ne soit plus accessible ou dysfonctionne sur une page et en général, soit les acteurs se déplacent dans le bureau, soit un mail est envoyé à mon tuteur Olivier MADERN et moi-même dans l'heure qui suit pour que l'application soit relancée. C'est dans ces moments que je me rends compte à quel point l'application est utilisée et importante. Je ne consulte pas la version de production tous les jours étant donné que la majeure partie de mes missions ne concerne que du développement, donc ces mails et les tickets présents sur la forge me montrent que l'application est réellement utilisée.

Compatibilité avec le M2

Les sujets abordés lors de l'alternance ne correspondent pas directement à la filière que j'ai choisie puisque PRISME est une application écrite en Java comportant une base de données, un back-end et une partie WEB, qui sont des notions qui ne sont pas traitées dans le Master 2 IMAGES. Le master qui correspondait le mieux pour cette alternance est le Master 2 Logiciel et ingénierie des données, mais la voie que j'ai prise est d'aller dans le Master 2 IMAGES pour les notions qu'il présente.

Néanmoins, même si mon alternance ne correspondait pas à la filière que j'ai choisie, cela m'a permis de continuer à coder en Java. Au final, ces années d'alternance au sein de la DGFiP m'ont permis de monter en compétences sur des langages de programmation et sur des technologies qui sont très utilisés dans le monde du travail.

Comme cette année d'alternance est la continuité de mon alternance de Master 1, j'étais directement prêt à travailler et à reprendre les travaux que j'avais déjà commencés l'an passé. Durant cette deuxième année j'ai moins eu besoin de consulter de documentation pour réaliser le travail souhaité. Néanmoins, je pouvais consulter la documentation en cas de besoin et dans le cas où je ne trouvais pas de réponse, je demandais à mes tuteurs.

Cette année n'a fait que conforter la raison pour laquelle je souhaite devenir un ingénieur informatique. Développer me correspond et, avec cette année d'alternance en plus, j'ai pris conscience de l'importance de la communication avec les différents utilisateurs pour mener à bien les demandes déposées.

BILAN DE L'ALTERNANCE

Cette année, le rythme d'alternance était d'une semaine en entreprise et d'une semaine à l'université ce qui, de mon point de vue, était plus agréable que l'année passée. Cela permet une transition plus douce entre les deux et facilite la reprise de la mission en cours avec un peu moins de temps de travail perdu. L'an passé, reprendre un sujet pouvait prendre une demi-journée, le temps de relire toutes les modifications et se remettre à niveau, ce qui ne laisse qu'une journée et demie de travail complet, alors que cette année, le temps pour reprendre une mission est le même sauf que cette fois il reste quatre jours et demi pour travailler et avancer.

Cette année en alternance m'a permis de mettre en pratique toutes les connaissances apprises durant mes années à l'université et cela m'a permis également d'entretenir mon niveau en Java, SQL ou encore HTML/JavaScript.

J'ai acquis de l'expérience dans l'écriture en langage Java, ainsi que sur les technologies PostgreSQL, Spring Boot et BootStrap.

Pouvoir contribuer à un projet comme PRISME m'a permis de voir certaines contraintes que je n'avais pas encore eues sur mes projets universitaires, comme le fait que l'application puisse convenir à des utilisateurs ayant plusieurs rôles et communiquer fréquemment avec les utilisateurs pour répondre à leurs besoins.

J'ai aussi pu voir la difficulté de prendre un projet en cours. Dans un projet déjà commencé, on débute forcément par une lecture de sa documentation et du code pour comprendre son fonctionnement en détail permettant d'être plus efficace lors de modifications. Cela m'a permis de voir d'autres structures de code et de manières de programmer. J'ai réalisé l'importance de lire de la documentation pour prendre en main les différentes technologies utilisées sur le projet.

En définitive, cette année d'alternance à la DGFiP était très instructive et m'a permis de gagner en autonomie, en communication et en rapidité de développement.

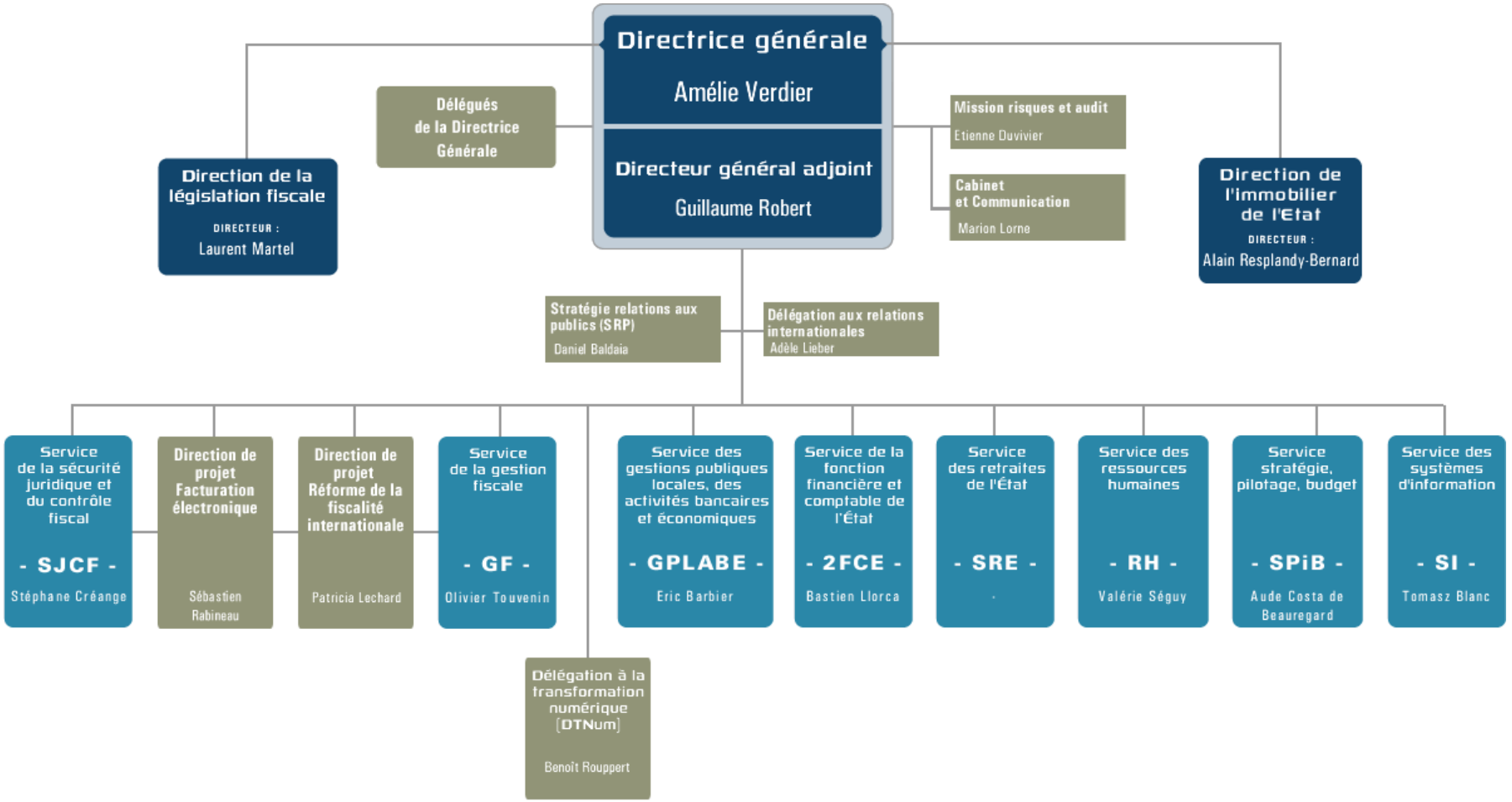
SOURCES

Sources qui m'ont servi pour rédiger ce rapport :

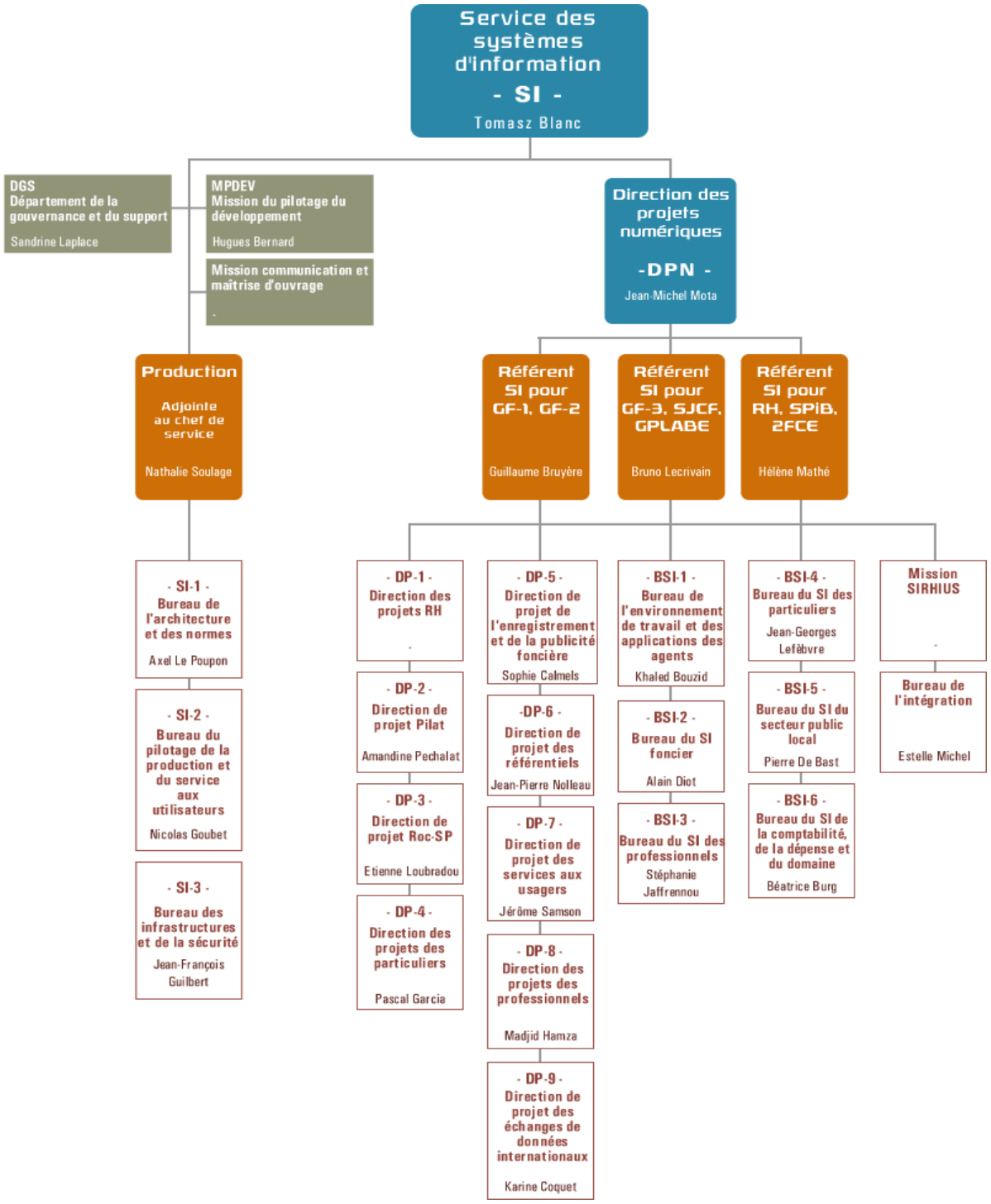
- PostgreSQL : https://www.postgresql.org/about/
- Spring Boot : https://spring.io/projects/spring-boot
- BootStrap : https://getbootstrap.com/
- BootStrap page about : https://getbootstrap.com/docs/4.1/about/overview/
- Tomcat : https://tomcat.apache.org/
- site de la DGFiP : https://www.economie.gouv.fr/dgfip
- site de recrutement DGFiP : https://choisirleservicepublic.gouv.fr/employeurs/dgfip/
- certains sites internes à la DGFiP

ANNEXES

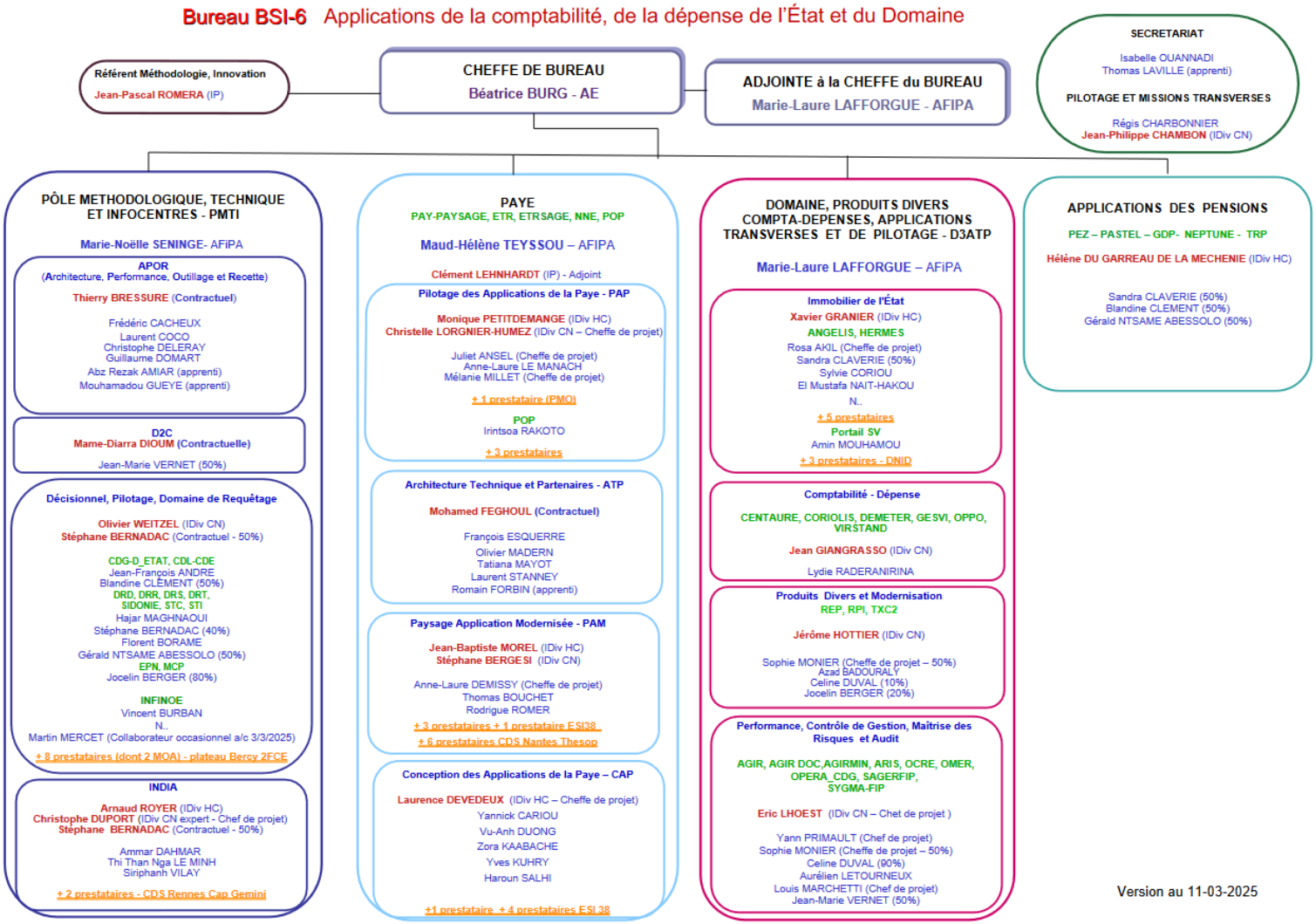
Annexe 1 : Organigramme DGFiP



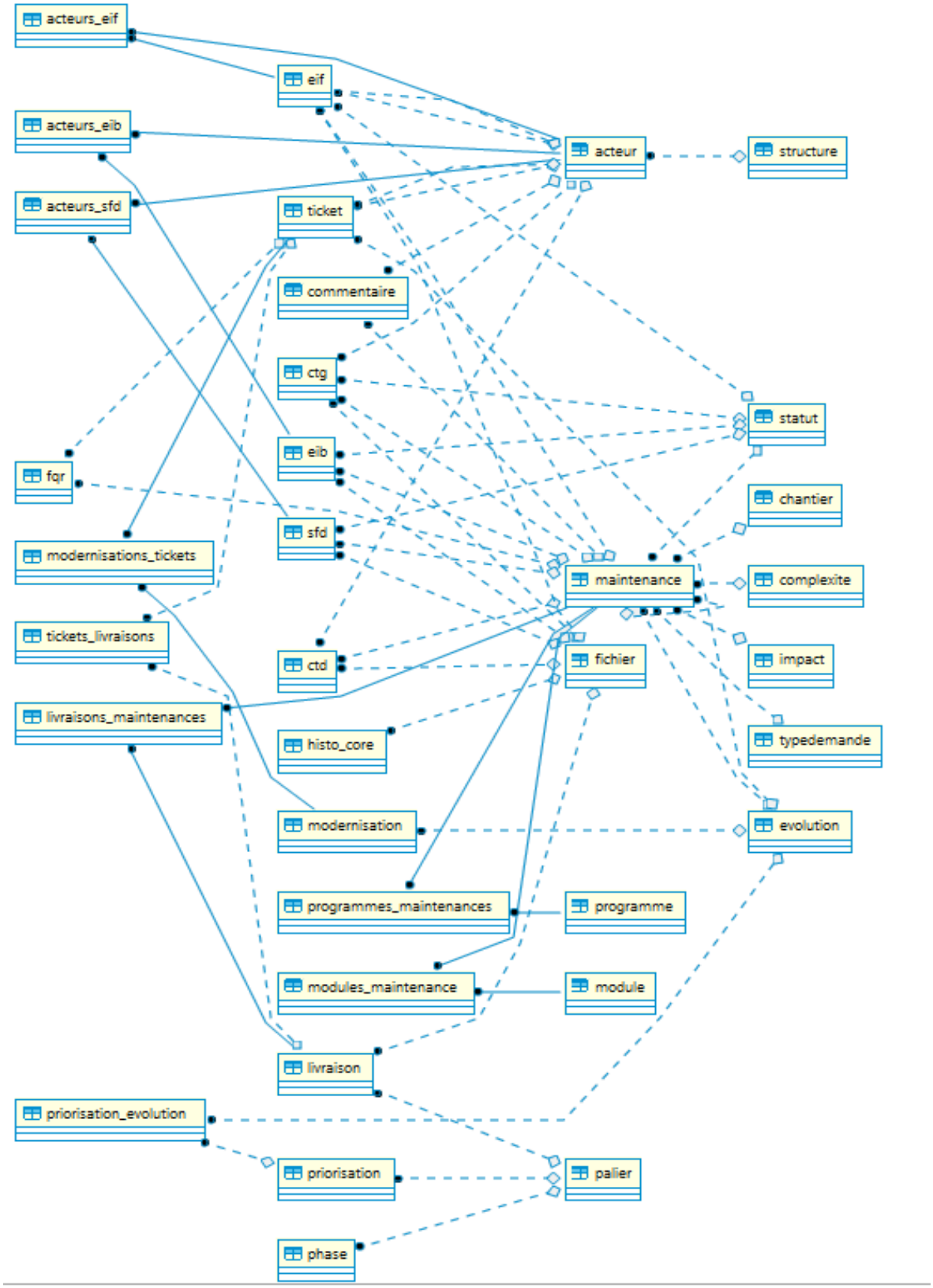
Annexe 2 : Organigramme SI



Annexe 3 : Organigramme BSI-6



Annexe 4 : Schéma PRISME



Annexe 5 : Tickets résolus sur les 3 derniers mois

Resolve "Modification de l'interaction avec CORE"
!53 · created 1 week ago by Romain Forbin
Amélioration
Merged
updated 1 week ago

Resolve "Modification de l'interaction avec CORE"
!52 · created 2 weeks ago by Romain Forbin
Amélioration
Merged
updated 1 week ago

Resolve "CORE : MàJ KO"
!51 · created 2 weeks ago by Olivier Madern
Correctif
Merged
updated 2 weeks ago

Resolve "CORE : MàJ KO"
!50 · created 2 weeks ago by Olivier Madern
Correctif
Merged
updated 2 weeks ago

Resolve "Livraisons : Reprise et amélioration - 2"
!49 · created 3 weeks ago by Romain Forbin
Amélioration Fonctionnalité
Merged
updated 2 weeks ago

Resolve "Amélioration pour mettre un palier inconnu à une livraison"
!48 · created 3 weeks ago by Romain Forbin
Amélioration
Merged
updated 3 weeks ago

Resolve "Problème d'affichage des maintenances"
!47 · created 3 weeks ago by Romain Forbin
Correctif
Merged
updated 3 weeks ago

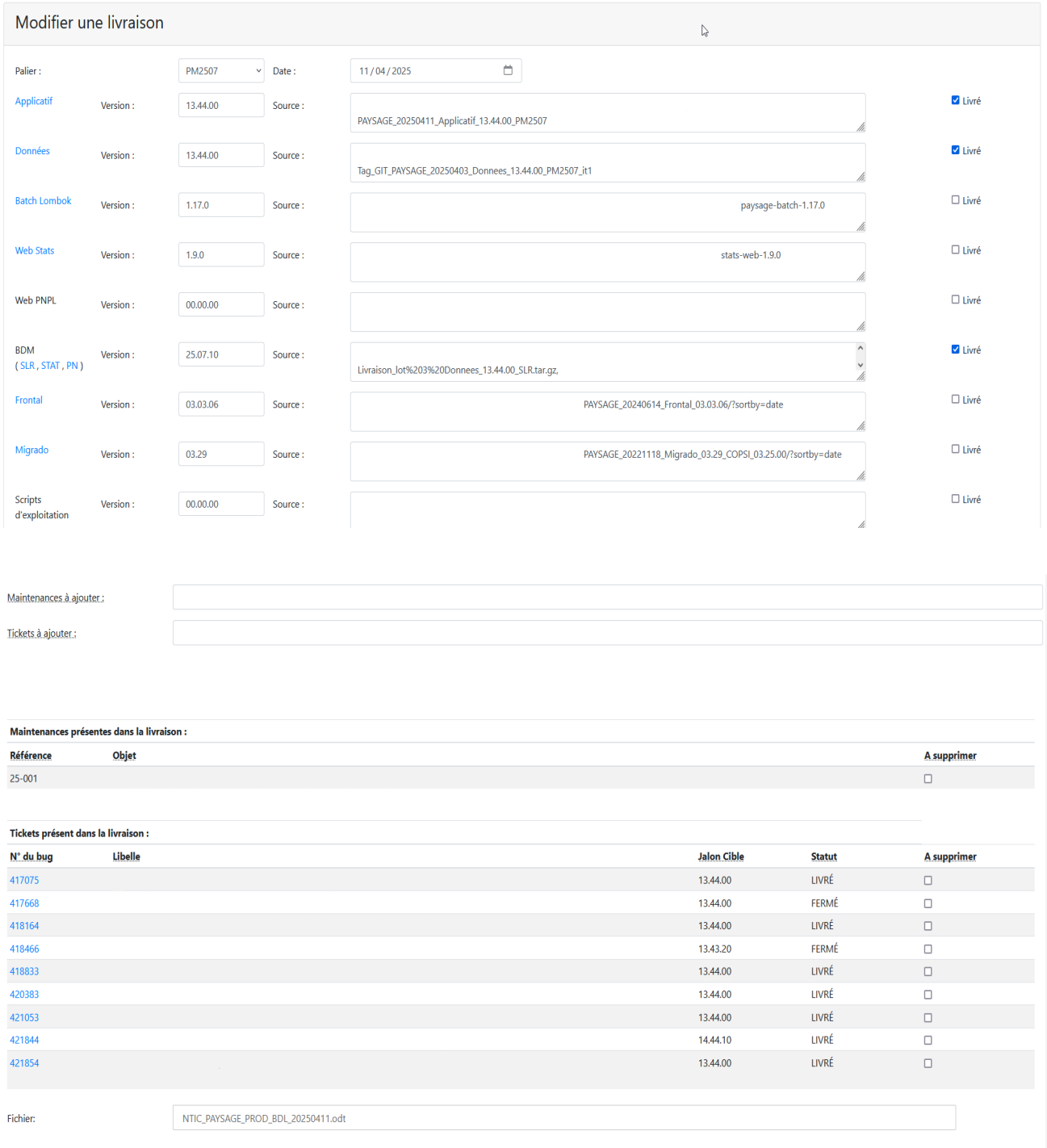
Resolve "Changement d'affichage pour les évolution en itération 9"
!46 · created 2 months ago by Romain Forbin
Fonctionnalité
Merged
updated 2 months ago

Resolve "Problème de FQR pas mis-à-jour"
!45 · created 2 months ago by Romain Forbin
Correctif
Merged
updated 2 months ago

Resolve "Ajout d'élément null dans les listes acteursMOEC et acteursMOA des EIF"
!44 · created 3 months ago by Romain Forbin
Correctif
Merged
updated 3 months ago

Resolve "Bilan recette : demande ajout colonne Priorité en 1ère colonne avec un tri correspondant"
!43 · created 3 months ago by Romain Forbin
Amélioration
Merged Approved
updated 3 months ago

Annexe 6 : Page d'une livraison



Modifier une livraison

Palier : PM2507 Date : 11 / 04 / 2025

Applicatif	Version :	13.44.00	Source :	PAYSAGE_20250411_Applicatif_13.44.00_PM2507	Livré
Données	Version :	13.44.00	Source :	Tag_GIT_PAYSAGE_20250403_Donnees_13.44.00_PM2507_it1	Livré
Batch Lombok	Version :	1.17.0	Source :	paysage-batch-1.17.0	Livré
Web Stats	Version :	1.9.0	Source :	stats-web-1.9.0	Livré
Web PNPL	Version :	00.00.00	Source :		Livré
BDM (SLR , STAT , PN)	Version :	25.07.10	Source :	Livraison_lot%203%20Donnees_13.44.00_SLR.tar.gz,	Livré
Frontal	Version :	03.03.06	Source :	PAYSAGE_20240614_Frontal_03.03.06/?sortby=date	Livré
Migrado	Version :	03.29	Source :	PAYSAGE_20221118_Migrado_03.29_COPSI_03.25.00/?sortby=date	Livré
Scripts d'exploitation	Version :	00.00.00	Source :		Livré

Maintenances à ajouter :

Tickets à ajouter :

Maintenances présentes dans la livraison :

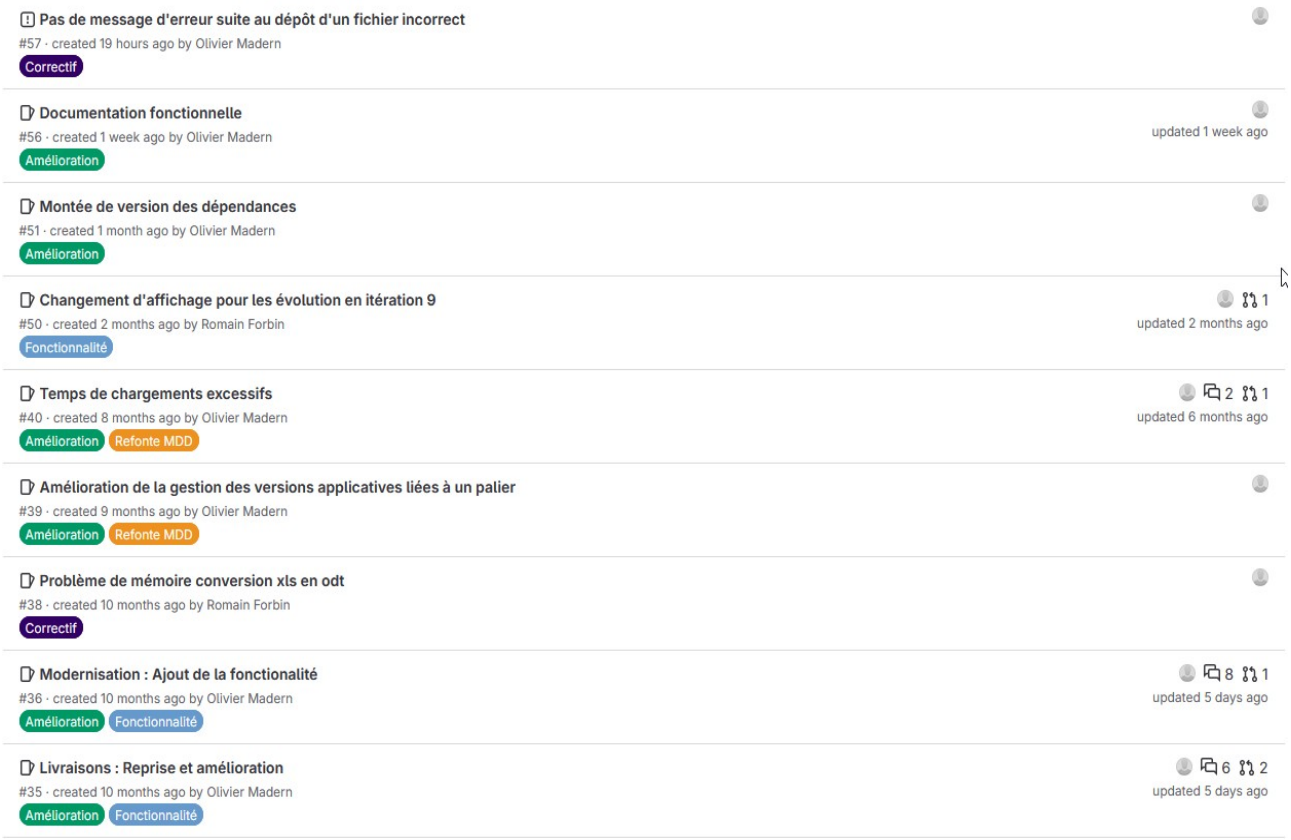
Référence	Objet	A supprimer
25-001		☐

Tickets présent dans la livraison :

N° du bug	Libelle	Jalon Cible	Statut	A supprimer
417075		13.44.00	LIVRÉ	☐
417668		13.44.00	FERMÉ	☐
418164		13.44.00	LIVRÉ	☐
418466		13.43.20	FERMÉ	☐
418833		13.44.00	LIVRÉ	☐
420383		13.44.00	LIVRÉ	☐
421053		13.44.00	LIVRÉ	☐
421844		14.44.10	LIVRÉ	☐
421854		13.44.00	LIVRÉ	☐

Fichier: NTIC_PAYSAGE_PROD_BDL_20250411.odt

Annexe 7 : Page ticket de la forge



RÉSUMÉ

Français

Les suivis de projet sont souvent complexes, car il faut combiner des informations qui proviennent de différentes sources comme des sites ou des différents acteurs. C'est pour cela que PRISME a été développée, car cette application réunit toutes les informations nécessaires pour suivre l'avancement du projet PAYSAGE. Elle permet aussi d'avoir un historique clair et rapide sur ce qui a été fait ou de ce qu'il reste à faire. Et également de générer des restitutions qui peuvent être présentés lors de réunions, ou encore mettre en avant des problèmes présents sur l'application en indiquant s'ils sont résolus, livrés ou bloquants. Elle offre une interaction indirecte entre les différents acteurs par le biais des fiches pilotes, MOA, analystes et développeurs.

English

Project monitoring is often complicated because it's necessary to regroup information that come from different place like WEB sites or people. That's why PRISME has been developed, this application regroup all necessary information to follow the progress of the PAYSAGE project. It makes it possible to have a clear history and a fast access to what has been done or what remains to do. It also allows to generate reports that could be shown during meetings, or highlights actual problems on the application indicating whether they are solved, delivered or blocking. It leads to create an indirect interaction between actors with pages dedicated to pilots, MOA, analysts and developers.