Apprenti Développeur Fullstack à la DGFiP

Rapport M1
Romain FORBIN

année 2023-2024
M1 INFORMATIQUE

MAÎTRES D'APPRENTISSAGE
Mohamed FEGHOUL & Olivier MADERN

TUTEUR UNIVERSITAIRE
Tita KYRIACOPOULOU

REMERCIEMENTS

J’ai eu l’opportunité d’effectuer cette première année de Master 1 d’informatique en alternance 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 ont rendu cette année possible dans les meilleures conditions en se rendant disponibles lorsque j’avais des interrogations et en me conseillant.

Je tiens aussi à remercier toutes les équipes présentes à BSI-6 pour leur accueil chaleureux en m’apportant leur soutien au quotidien durant toute cette première année.

SOMMAIRE

INDEX DES FIGURES PRÉSENTES DANS LE DOCUMENT

INTRODUCTION

Durant mes années d'études j'ai pu réaliser de nombreux projets proposés par mes professeurs, mais je n'étais que du côté développeur. Or il existe deux manières de voir un projet en informatique :

- il y a la vision des développeurs, celle qui est principalement abordée en étude supérieure d'informatique,
- et la vision des pilotes/superviseurs, qui consiste à regarder l'avancement du projet, à vérifier qu'il n'y a pas de problème et le tout dans un temps donné.

C'est cette deuxième vision qui va nous intéresser ici, car l'application PRISME sur laquelle j'ai été affecté durant cette année d'apprentissage est une application de pilotage permettant d'avoir une vision globale de l'avancée d'une autre application appelée PAYSAGE.

PRISME, créée en 2019, est une application de pilotage qui permet d'avoir principalement un tableau de bord synthétique pour toutes les équipes. Le but de PRISME est d'avoir une vision rapide sur l'avancée, les besoins et les problèmes du projet PAYSAGE. PRISME est utilisée par les agents internes de la Direction Générale des Finances Publiques ayant un lien avec PAYSAGE, c'est-à-dire qu'il y a la MOA[1], la MOE[2] et les développeurs de PAYSAGE (ESI38[3]), mais aussi les agents de BSI-6 tels que le pôle pilotage ou encore les analystes.

PRISME utilise les informations présentes sur divers sites de suivi (telles que CORE[4] ou encore Bugzilla[5]), et de celles fournies par les équipes de la MOA indiquant les maintenances qui doivent être réalisées, de la MOE (BSI-6) analysant et chiffrant ces demandes et de l'ESI38 développant ces évolutions sur l'application PAYSAGE.

PRISME est utilisée par de multiples équipes réparties géographiquement sur le territoire. En effet, il y a plusieurs organismes qui travaillent en simultané sur le projet afin de réaliser les demandes et de résoudre les problèmes liés, dans les délais prévus par la MOA.

Dans un premier temps je commencerai par une présentation de l'entreprise et de PRISME de manière détaillée, dans un second temps par le détail de quelques missions que j'ai réalisées durant cette année. Enfin je finirai sur une conclusion avec une réflexion professionnelle et personnelle.

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.

ENTREPRISE

DGFiP

La Direction Générale des Finances Publiques (DGFiP) est une administration qui dépend du ministère de l'Économie, des Finances et de la Souveraineté industrielle et numérique. La DGFiP a été créée en 2008 suite à la fusion de la Direction Générale des Impôts et de la Direction Générale de la Comptabilité Publique. Elle est actuellement dirigée par Amélie VERDIER depuis le 4 mars 2024. Elle comporte une administration centrale, localisée essentiellement à Bercy (75 112) avec les services informatiques situés à Montreuil (93 100) et Noisy-le-Grand (93 160), et une administration décentralisée en province.

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.

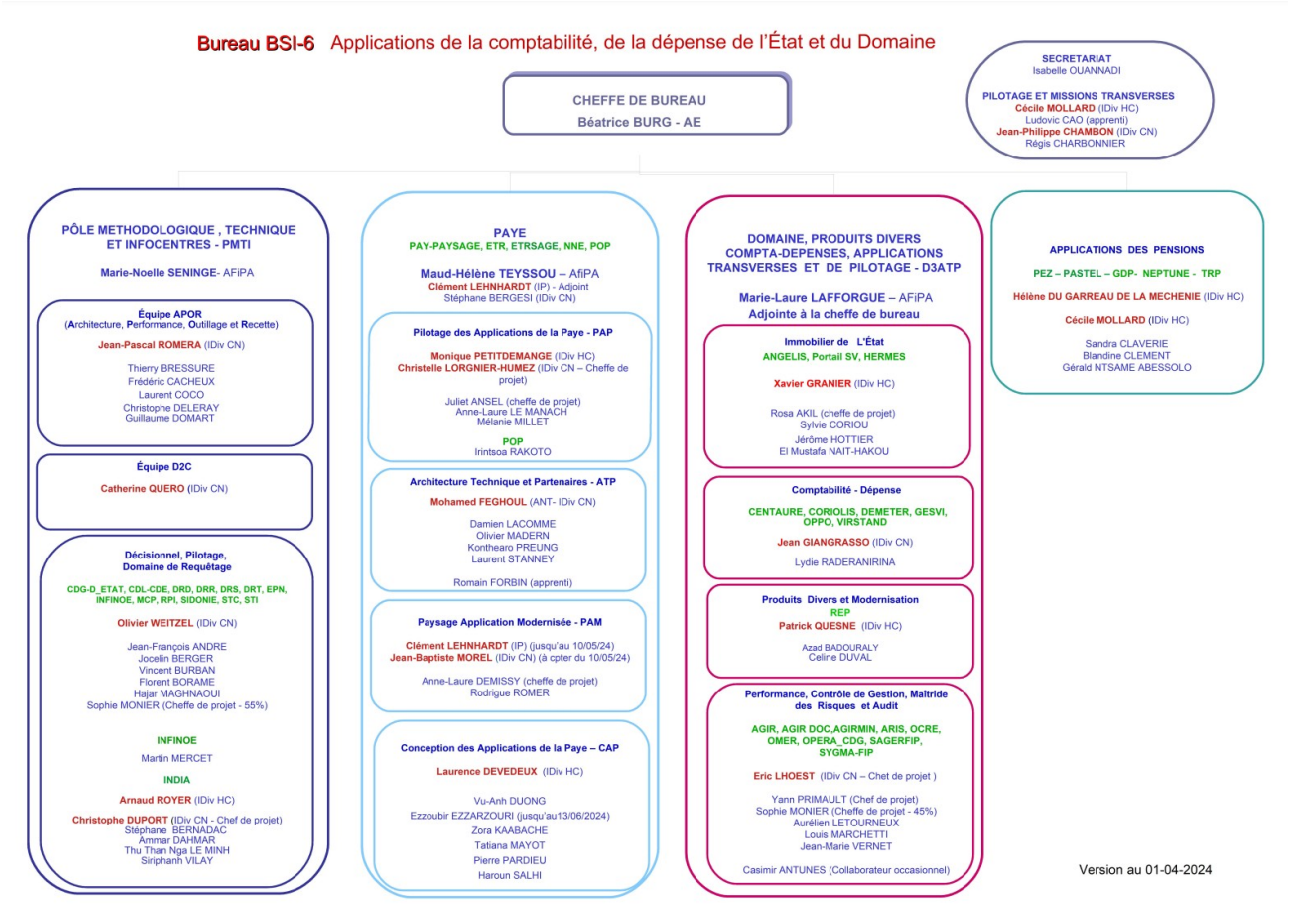
La DGFiP est souvent vue comme l'administration qui gère les impôts, effectivement, mais elle ne possède pas que ce rôle, car elle doit aussi se charger de la rémunération de ses agents, la gestion des Finances Publiques, soit tous les sujets de dépenses de l'État.

La Direction Générale des Finances Publiques reste une administration assez grande. De ce fait, elle se divise en différents services :

- 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 20 bureaux, et celui qui m'accueille est le bureau BSI-6 présidé par Béatrice BURG.

BSI-6



Voici l'organigramme du bureau auquel j'appartiens. Ce bureau est composé de 73 agents et est divisé en 5 divisions :

- 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 ma première année d'apprentissage dans la division PAYE dirigée par Maud-Hélène TEYSSOU. Cette division prend en charge les applications qui gère les payes des fonctionnaires d'État, soit environ 2,5 millions de payes 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, Laurent STANNEY, Konthearo PREUNG et Damien LACOMME.

QU'EST-CE PRISME ?

Introduction

PRISME (Pilotage et Restitution d'Informations pour le Suivi des Maintenances Évolutives) est une application créée en 2019 et ramenée sur la forge interministérielle (basée sur GitLab[6]), en interne à la division PAYE. Elle a été développée par mon tuteur Olivier MADERN à partir des demandes exprimées par la responsable de l'équipe Pilotage et en collaboration avec une prestataire. Cette application permet d'obtenir le périmètre d'un palier et de suivre l'avancement des évolutions fonctionnelles de l'application PAY-PAYSAGE.

PAY-PAYSAGE est l'application gérant la paie des agents de l'État. Cette application est composée de PAY, l'application historique écrite en Cobol[7], et de PAYSAGE sa réécriture en cours en Java.

PRISME a été créée afin de faciliter le suivi du développement de PAYSAGE et d'avoir un visuel plus global de l'avancée. L'application est développée en Java à l'aide de frameworks[8] (Spring Boot et BootStrap) et est reliée à une base de données PostgreSQL[9].

Principe de l'application

L'application 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 de sa mise en production mais aussi les évolutions apportées sur l'application avec tous les documents qui leur sont liés. Elle répertorie aussi tous les agents qui participent ou ont participé au développement de l'application PAYSAGE. Elle permet à la MOA de remplir un formulaire décrivant une maintenance en indiquant ses détails comme sa priorité ou encore son palier d'application. Elle permet aussi d'indiquer les tickets qui doivent être pris en charge sur le palier. PRISME est renseignée par les développeurs de la date de prise en charge des maintenances, de leur avancement et de leur date de finalisation.

L'une des pages principales de l'application est la page appelée Météo. Cette page consiste à afficher l'ensemble des maintenances et des tickets qui ont été priorisés sur les différents paliers en indiquant si pour une maintenance il manque des informations comme les documents d'expression initiale du besoin (EIB), d'étude d'impact et de faisabilité (EIF), de spécifications fonctionnelles détaillées (SFD) et de conception technique générale (CTG). Elle indique aussi l'avancement des développeurs sur les maintenances. Pour chaque ticket ouvert la page indique si ce dernier a été livré, corrigé ou pas. La page Météo permet également de visionner les paliers

6 Forge utilisée.
7 Langage de programmation.
8 Modules aidant au développement d'applications.
9 Système de base de données.

précédents et/ou suivants. Sur cette page on peut aussi extraire les informations présentes sous forme de document texte (ODT) et de tableur (ODS) détaillé ou de synthèse.

PRISME met à jour automatiquement les tables ticket et FQR toutes les 30 min en faisant des requêtes via l'API[10] de Bugzilla qui est le site de base où sont créés les tickets du projet PAYSAGE. Cette mise à jour va actualiser la table ticket en ajoutant uniquement ceux qui n'existent pas et en modifiant ceux qui sont déjà dans la base de données de PRISME. Elle peut aussi créer de nouveaux acteurs s'ils ne sont pas répertoriés. PRISME va aussi récupérer et traiter les nouvelles fiches questions/réponses (FQR) qui ont été ajoutées à Bugzilla et modifier celles déjà existantes.

Structure de PRISME

PRISME possède une base de données composée de 34 tables répertoriant, pour les principales, les acteurs (personnes participant au projet), les maintenances, les chantiers des maintenances, les différents documents EIB, EIF, CTG, SFD, les tickets, les tables de jointure (par exemple entre les acteurs et les documents qui ont été rédigés), la table répertoriant les FQR présentes sur les tickets ainsi que celle des priorisations.

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. Cela signifie que la version ne doit pas avoir d'erreurs contraignantes, qu'il n'y ait pas de failles de sécurité connues et que la version soit stable dans le temps donc pas de mise à jour qui puisse casser l'existant.

De ce fait la version de java utilisée est Java 11, celle de PostgreSQL est PostgreSQL 12. L'application utilise des frameworks afin de faciliter son développement. Pour le côté back-end[11], le framework utilisé est Spring Boot avec sa version de 2.1.7. Pour le front-end[12] le framework utilisé est Bootstrap avec sa version 4.3.1.

Finalité de l'alternance

Le but de mon travail est de collaborer avec mon tuteur Oliver MADERN pour que l'application PRISME ait plus de fonctionnalités afin d'améliorer son ergonomie pour tous les agents qui en ont besoin, car les personnes l'utilisant ne veulent pas forcément naviguer entre différents sites pour avoir une information. 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.

10 Interface logicielle.
11 Partie non visible par les utilisateurs d'une application.
12 Partie qui permet de communiquer entre l'utilisateur et le back-end.

TECHNOLOGIES

Base de Données

PostgreSQL est un système de base de données relationnelle orienté objet[13], open source[14], puissant très utilisé. Il possède de nombreuses fonctionnalités aidant les développeurs à créer leur application. Il permet de gérer des données de manière fiable peu importe la taille de ces dernières.

PostgreSQL a été choisi 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 12 est la dernière version de ce système, approuvé par la DGFiP et mis en place récemment sur tous les postes portant un système PostgreSQL. Cela a permis d'avoir accès à plus de fonctionnalités sur les requêtes SQL et a eu aussi un impact sur la performance de certaines requêtes réalisées.

Au sein de la DGFiP, l'outil PgAdmin est utilisé pour interagir avec une base de données PostgreSQL. PgAdmin est une application pouvant administrer de multiples serveurs et leurs bases de données. Elle permet toutes les interactions avec les bases mais en version graphique (faire des sauvegardes ou encore pouvoir importer des dumps[15] 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.

Même pour des applications internes il y a des normes, comme le versionnage, qui doivent être respectées dans les choix des outils à utiliser. Par exemple quand je suis arrivé, la version de PostgreSQL sur PRISME était PostgreSQL 9. Le changement ne s'est fait qu'en fin d'année 2023.

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

Back-End

PRISME est une application en Java 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 de paramètres à remplir. Il est compatible avec beaucoup modules tels que les bases de données PostgreSQL et le front-end 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 ;
- des requêtes SQL générées automatiquement par l'application ;
- 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 dit précédemment, il s'agit d'un framework qui a besoin de peu de configuration manuelle, facile d'utilisation, bien documenté avec beaucoup d'aide et 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 de front-end et celui utilisé sur PRISME.

Front-End

Pour la partie statique du front-end, 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[16], 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 a des icônes personnalisées fournies par Boostrap ;
- mise en place d'une API utilitaire ;
- de puissants plugins[17] JavaScript.

Les pages WEB de PRISME sont des pages en Java Server Pages (JSP). Ce sont des pages WEB dynamiques qui permettent d'introduire du code Java dans des tags 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[18]. Le JSP permet aussi une utilisation simple du framework Spring Boot directement sur la page en question grâce à des balises de type « <spring> ».

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

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

PLAN D'ACTION

Mon plan d'action pour réaliser les travaux qui me sont demandés se compose de 4 phases :

- Phase d'analyse ;
- Phase de développement ;
- Phase de test ;
- Phase de rétrospective.

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.

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

Phase de développement

La plupart du temps, pendant la phase de développement je suis 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 résolve 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 test où je passe le plus clair de mon temps à essayer diverses fonctionnalités de l'application afin de vérifier que mes changements ont bien été ajoutés sans altérer les autres fonctionnalités et résultats déjà présents dans l'application. Si je constate un résultat imprévu je repasse dans une phase d'analyse afin de comprendre ce qui génère ce problème et de le régler dans une nouvelle phase de développement.

Phase de rétrospective

Enfin vient la phase de rétrospective. Une fois le ticket déposé sur la forge, j'attends le retour de mon tuteur Oliver MADERN sur le rendu que j'ai fait afin d'avoir un retour sur mon travail. Il arrive que je doive reprendre le ticket à cause d'une erreur que je n'aurais pas vu afin de la corriger. Dans le cas où tout est bon et qu'il n'y a pas de problème, alors mon travail est ajouté à la branche principale du projet.

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

MISSIONS

L'ensemble de mes missions est en rapport avec le développement de PRISME, c'est-à-dire que lorsqu'il y a un problème présent sur l'application ou que l'application a besoin d'avoir des modifications une demande est faite sur la forge interministérielle. Ces demandes sont décrient par des tickets et peuvent être plus ou moins longues selon la tâche qui est demandée. Les modifications peuvent aller d'un simple ajout de lien en bas de page, tel que j'ai pu le faire à mon arrivée sur le projet, à une refonte majeure du modèle de données ayant des impacts multiples sur le back-end et le front-end.

Missions de refonte du modèle de données

Dès mon arrivée j'ai pris mes marques avec l'application plutôt rapidement. J'ai pu commencer les tickets de refonte de base de données après seulement 3 semaines. Par exemple, un de mes tickets était de séparer des informations communes présentes dans les tables Maintenance et Ticket pour les mettre dans une nouvelle table Evolution afin de rendre le modèle applicatif plus propre. Pour effectuer cette tâche, j'ai dû rédiger un script SQL afin de récupérer les informations qui m'intéressaient sur les tables pour les ajouter à la nouvelle table, et faire les jointures nécessaires.

Une fois cela fait, j'ai vérifié que les données renseignées sur les tables modifiées n'étaient pas erronées. Ensuite j'ai pu commencer la modification de l'application en elle-même en commençant par le back-end qui avait besoin d'une adaptation pour être en accord avec le nouveau modèle. Dans cette mission, j'ai dû ajouter une nouvelle classe entité avec une classe contrôleur, une classe service et une classe dépôt qui lui est spécialement dédiée afin de pouvoir interagir avec, dans toute l'application. Les tables Ticket et Maintenance font partie des tables les plus consultées de l'application, donc il était important de savoir où elles sont appelées pour ne pas avoir d'erreur et faire les modifications nécessaires. Par exemple, changer les requêtes SQL présentes dans les classes dépôts où sont appelées les tables maintenance et ticket mais aussi de changer les méthodes GET et POST dans les contrôleurs afin que la récupération et l'enregistrement se fassent correctement.

Puis s'en est suivi des modifications sur le front-end pour utiliser les nouvelles fonctionnalités ajoutées par les changements. Par exemple changer les références aux champs des classes qui ont été retirées pour ne pas provoquer d'erreurs d'affichage.

Une autre mission de refonte du modèle de données que j'ai réalisée pendant mon année d'alternance est la simplification des livrables. Les livrables sont les documents de type EIB, EIF, CTG et SFD. Ces livrables peuvent apparaître plusieurs fois dans la base de données mais avec des numéros de version différents. Cela permet d'avoir un historique des modifications apportées sur le projet PAYSAGE. Le but de ma mission ici était de rendre ces tables plus petites en gardant uniquement la version la plus élevée du livrable. Or, le but de cette mission n'était pas juste de

supprimer tout ce qu'on ne voulait plus mais de garder des informations sur les modifications apportées sur un seul livrable.

Pour cela, il m'était indiqué les modifications à apporter sur les différentes tables pour reporter les informations de précédentes versions de livrable sur un élément unique. Par exemple en reportant la date de réception de la première version du livrable sur sa dernière pour ne garder que celle-ci. Dans un premier temps, j'ai réalisé un script SQL pour répondre à la demande souhaitée. Puis j'ai mis à jour le code java en modifiant toutes les requêtes ressortant une liste des livrables portant le nom de celui recherché pour qu'elles ne puissent retourner qu'un seul élément, impliquant des changements de type de variables pour correspondre à celui des requêtes.

Ces missions sont les missions les plus longues car elles sont en rapport avec les trois parties de l'application que sont la base de données, le back-end et le front-end. Elles demandent en plus une phase de test beaucoup plus longue, car il faut aussi vérifier qu'aucune autre page, hormis celle voulue, n'ait été modifiée ni même cassée. Malgré cette phase de test réalisée de mon côté, certaines erreurs peuvent échaper ma vigilance mais mon tuteur réalise aussi de son côté une phase de test pour vérifier. Dans le cas où des erreurs subsistent, je reprends le ticket en charge pour les corriger.

Missions de correction

Les missions de correction consistent à résoudre un problème qui a été trouvé pendant son utilisation. Généralement ce sont des missions qui se résolvent plutôt rapidement car cela peut correspondre à un oubli de changement ou un oubli de contrôle d'une variable, ou encore une valeur qui n'a pas été modifiée correctement suite à un changement.

Par exemple, lors de l'ajout d'un document comme une EIB avec des informations erronées à l'intérieur, la page générait une erreur. Lors de l'affichage de cette dernière pour indiquer les erreurs, une information était perdue entre les appels de fonctions. Pour résoudre ce problème, j'ai juste dû faire une nouvelle requête pour pouvoir de nouveau renseigner le champ auquel on n'avait plus accès.

Un autre exemple de mission de ce type est un affichage manquant sur une page de suppression. Lors d'un autre ticket, une ligne s'est retrouvée supprimée par erreur et n'a pas été remarquée tout de suite. De ce fait, j'ai dû remettre cette ligne manquante pour retrouver un affichage correct sur la page concernée.

Ce sont des missions très courtes et qui ne demandent pas beaucoup de temps dans la résolution, mais elles peuvent vite avoir une influence sur d'autres parties du code de code de l'application pouvant allonger son temps de résolution mais que de très peu.

Missions d’amélioration

Ces missions servent, comme leur nom l’indique, à améliorer l’application en ajoutant des fonctionnalités ou en faisant des changements sur l’application. Elles ne provoquent pas de modifications ergonomiques majeures pour l’utilisateur de l’application ni de changement complexe du modèle de données.

- J’ai dû faire en sorte que l’ajout d’une nouvelle maintenance puisse s’insérer entre d’autres maintenances plutôt que faire des changements manuels pour l’ajouter sur une priorité qui été déjà occupée. À cette fin, j’ai dû changer le contrôle qui était fait pour savoir si une maintenance existait déjà sur cette priorité en un décalage de priorité directement via le code.
- J’ai dû ajouter un nouveau champ sur la table acteur permettant de savoir si un acteur est actif sur le projet et dans le cas où il ne l’est pas ne plus le proposer lors d’un ajout de document.

Ces missions peuvent être courtes ou longues en fonction de la demande. S’il s’agit d’un simple ajout de bouton oui/non la mission est courte et est finie rapidement, mais il peut aussi s’agir d’un changement d’interaction avec l’utilisateur, comme le décrit la première mission de ce type. Dans ce cas la mission peut devenir plus longue car il y a plus de code à modifier.

Missions de fonctionnalité

Cette dernière catégorie de missions consiste à ajouter des fonctionnalités qui sont directement utilisées par les utilisateurs de l’application. J’ai fait peu de missions d’ajout de fonctionnalités, car elles demandent plus de spécifications de la part des utilisateurs pour que le rendu convienne mais aussi parce que les missions de refonte du modèle de données peuvent permettre de simplifier ces tickets dans le futur.

- J’ai réalisé la création automatique d’un compte-rendu concis de l’avancement du palier. Pour faire ce ticket j’ai pu réutiliser la plupart des fonctions déjà existantes pour produire le rendu donc je n’ai eu que très peu d’ajout personnel dessus.

La durée de ces missions sont comme celles de la section précédente c’est-à-dire qu’elles dépendent de la demande faite.

La principale difficulté rencontrée durant cette année d’apprentissage fut la prise en main d’un code déjà existant avec certaines technologies que je ne maîtrisais pas forcément comme Spring Boot ou même Bootstrap.

Mission actuelle

Ma mission actuelle est d'améliorer l'application PRISME pour rendre l'application plus complète que cela soit pour les utilisateurs ou pour les développeurs en faisant du code propre. Pour cela il me reste des tickets qui n'ont pas encore été traités sur la forge interministérielle, certains doivent être spécifiés avec la personne qui a initié la demande.

L'un des objectifs pour l'avenir est de pouvoir migrer l'application sur NUBO, un cloud interministériel opéré par la DGFiP, en utilisant les technologies DevOps[19].

Compatibilité avec le M1

Je pense que je n'aurais pas pu trouver une meilleure alternance pour cette année de Master 1 Informatique, car les notions de développement application en langage Java que j'ai vues en cours ont pu directement être appliquées pendant ma période en entreprise. Cela m'a permis d'avoir une prise en main plus rapide du projet de Java donné en cours grâce aux missions que j'ai réalisées sur l'application PRISME. Cela m'a aussi amené à développer une vision différente des méthodes de travail avec différentes approches selon la demande. Cette période en entreprise m'a aussi permis d'améliorer ma vitesse de développement en langage Java. J'ai également amélioré les notions en base de données en écrivant plusieurs scripts SQL qui ont pu être exécutés. Pendant cette année j'ai pu aussi améliorer mes compétences en matière de communication grâce aux échanges que je pouvais faire entre développeurs en posant des questions directement à mes tuteurs mais aussi avec les échanges entre développeurs et utilisateurs qui étaient encore jusque-là une notion vaste que je ne maîtrisais peu.

Dès mon arrivée en entreprise, je me suis renseigné au travers de différentes documentations afin de comprendre l'application et pour avoir une prise en main plus facile sur les missions qui m'ont été confiées. De plus, en cas difficulté, j'ai pu directement demander du soutien à mes tuteurs pour qu'ils m'aident à résoudre le problème.

Cette année, en alternance au sein de la DGFiP, a confirmé la raison pour laquelle je voulais devenir un ingénieur informatique. Le développement est la raison pour laquelle j'ai choisi l'informatique, car c'est un univers vaste où il existe des points de vue différents, de nombreux de langages avec un panel de technologies différentes.

19 Regroupement des développements et des exploitations d'une application.

CONCLUSION

Le rythme de 2 jours d'entreprise pour 3 jours à l'université est assez difficile à associer. En effet la transition entre les deux était assez compliquée car passer d'un rythme d'entreprise avec des collègues à un rythme scolaire avec des camarades et des professeurs est totalement différent. L'autre difficulté est la reprise le lundi en entreprise après 5 jours hors du projet.

Cette année en alternance m'a permis de mettre en pratique les 4 années d'études supérieures en informatique, de prendre conscience des différences entre le monde du travail et le monde universitaire.

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, ou encore une communication fréquente avec les utilisateurs pour savoir ce qui leur convient ou doit être modifié.

J'ai aussi pu voir la difficulté de prendre un projet en cours. Dans un projet déjà commencé, on débute forcément par une lecture 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 d'améliorer diverses compétences comme l'autonomie, la communication et la 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

RÉSUMÉ

Français

Les suivis de projet sont souvent complexes car il faut réunir des informations qui proviennent de différents endroits comme des sites ou l'attente de réponses des différents acteurs. C'est pour cela que PRISME a été développé car cette application réunit toutes les informations nécessaires pour suivre l'avancement du projet PAYSAGE. Il permet aussi d'avoir un historique clair et rapide sur ce qui a été fait ou de ce qu'il reste à faire. Il permet aussi de générer des rendus 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.

English

Project monitoring are 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 join all necessary information to follow the progress of the project PAYSAGE. It makes it possible to have a clear history and fast to access on what has been done or what it remains to do. It also allows to generate renderings that could be shown during meetings, or highlights actual problems on the application indicating whether they are solved, delivered or blocking.