

DOSSIER DE PROJET 2022-2023



Margaux
MARTINEAU



ENI
Rennes



Confidentiel

Remerciements

Je tiens à remercier dans un premier temps, toute l'équipe pédagogique de l'ENI Rennes, pour avoir assurée cette formation et de m'avoir permis de découvrir plus en profondeur le métier de développeuse web.

Je tiens à remercier **OPEN** de m'avoir fait confiance en me proposant ce contrat d'apprentissage après mon stage, ce qui m'a permis d'intégrer la formation Concepteur développeur d'application à leur côté.

Je tiens à remercier chaleureusement toute l'équipe du projet TMA-PNUM (Tierce Maintenance Applicative – Parcours Numérique) au sein du groupe **OPEN**, notamment ceux m'ayant accompagné sur ma partie projet. Merci à Frédéric mon directeur de projet, Camille ma cheffe de projet, Alexandre mon tuteur, Steeven notre lead front et Jean-Yves mon binôme de formation, pour leur professionnalisme, leur patience, leur implication et le partage de leur expertise au quotidien.

Et je tiens également à remercier mon mari, qui m'a donné toutes les cartes en main pour me lancer dans cette aventure, qui m'a accompagné dans mon apprentissage, soutenu quand ça devenait compliqué, et pour le partage de ses connaissances dans le domaine, qui ont permis de développer mes compétences en dehors de mon environnement professionnel.

Table des matières

REMERCIEMENTS	2
1. INTRODUCTION.....	5
2. GLOSSAIRE	6
3. PRESENTATION DE L'ENTREPRISE.....	8
4. PRESENTATION DU POSTE ET DE SON ENVIRONNEMENT TECHNIQUE.....	8
5. COMPÉTENCES DU REFERENTIEL COUVERTES PAR LE PROJET	12
6. RESUME DU PROJET EN ANGLAIS.....	13
7. CAHIER DES CHARGES	14
7.1. Présentation	14
7.2. Expression des besoins.....	14
7.3. Gestion de projet.....	14
7.3.1. Planning et suivi	14
7.3.2. Environnement humain et technique	17
7.4. Spécifications fonctionnelles	18
8. SPECIFICATIONS TECHNIQUES.....	19
8.1. Présentation	19
8.2. Use Case.....	20
8.3. Maquettes.....	21
8.4. Schéma de base de données	24
8.5. Architecture du projet	25
9. REALISATION	26
9.1. Présentation	26
9.2. Réalisation de la base de données	29
9.3. Réalisation des composants d'accès aux données	31
9.4. Réalisation du back-end	33
9.4.1. Présentation.....	34
9.4.2. Entités	35
9.4.3. MAPPER.....	35
9.4.4. REPOSITORY.....	36
9.4.5. SERVICE	37
9.4.6. CONTROLLER	38

9.5. Réalisation du front-end.....	40
9.5.1. Présentation.....	42
9.5.2. Partie TypeScript	44
9.5.3. Partie HTML.....	46
9.5.4. Partie CSS	47
9.5.5. Réalisation de test unitaire front-end	49
10. PRESENTATION DU JEU D'ESSAIE AVEC FONCTION LA PLUS REPRESENATIVE	51
10.1. Présentation	51
10.2. Utilisation.....	51
10.3. Conclusion	54
11. SECURITE.....	55
11.1. Présentation	55
11.2. Recherche effectuée	55
11.3. Mise en œuvre	56
11.4. Conclusion	57
12. RECHERCHE EFFECTUEE	57
12.1. Le besoin d'information.....	57
12.2. La recherche	57
12.3. La mise en œuvre.....	58
13. BILAN ET PERSPECTIVES DU PROJET	60
14. CONCLUSION	61
ANNEXES.....	62

1. INTRODUCTION

Ce rapport d'activité est l'aboutissement de ma formation Concepteur Développeur d'application au cours de laquelle j'ai pu acquérir les compétences en informatique nécessaires à mon intégration sur un projet de développement d'application. Après ma première formation en Développeur Web et Web mobile de 6 mois et un stage chez **OPEN** de 2 mois, j'ai pu poursuivre mon apprentissage au sein de leurs équipes pendant un an, en alternance.

Après plusieurs années d'expériences dans le secteur du commerce et après avoir réalisé mes objectifs, j'ai souhaité entamer une reconversion professionnelle et travailler dans le secteur du numérique. Consciente des réalités du métier par des enquêtes métiers, d'un cercle de connaissances travaillant dans le domaine, et de témoignages d'anciens étudiants de l'ENI, j'ai souhaité intégrer cette école pour ma reconversion dans le développement web.

Acteur local et reconnu dans l'écosystème numérique, **OPEN** est une société centrée sur la qualité de ses relations clients et de ses savoir-faire techniques et digitaux. C'est pour cela que j'ai choisi cette entreprise régionale et de taille humaine.

Les technologies utilisées sur le projet m'intéressaient beaucoup, notamment Spring et Angular que nous n'avions pas étudiés durant la formation et que j'ai pu découvrir pendant mon stage. Cela allait me permettre de continuer ma montée en compétences dans ces langages, en plus de celles acquises.

Le projet concerne le site gouvernemental de l'**ANTAI** (Agence Nationale de Traitement Automatisé des Infractions), établissement public français rattaché au ministère de l'Intérieur. Cette agence contribue à la politique publique de lutte contre l'insécurité routière sur le territoire national. Les objectifs d'**OPEN** pour son client sont d'assurer la continuité de la production, consolider la chaîne de traitement automatisé et de poursuivre la transformation numérique de celle-ci.



Figure 1 : Logo Open



Figure 2 : Logo ANTAI

2. GLOSSAIRE

~A~

ANTAI: Agence Nationale de Traitement Automatisé des Infractions

API: Application programming interface ou « interface de programmation d'application »

~B~

Backend: Partie non visuelle de l'application permettant de traiter la donnée et faire le lien entre le Frontend Web et la base de données

Backlog: Liste des fonctionnalités à développer dans la constitution d'une application.

Bearer Authentication: Schéma d'authentification HTTP qui implique des jetons de sécurité appelés jetons du porteur.

~C~

Commit (Git): Le fait de préparer l'envoi des modifications ou ajouts de code de notre environnement local vers un gestionnaire de versions en ligne (Gitlab dans notre cas).

Composant (Angular): Elément réutilisable de l'application, constitué d'une vue et d'un contrôleur permettant de gérer un ensemble de traitements associés à cette vue.

~D~

DOR (Definition Of Ready): Permet d'aligner les membres de l'équipe sur une vision commune d'une user story type.

DTO: Objet de transfert de données ou data transfer object en anglais est un patron de conception utilisé dans les architectures logicielles objets.

~F~

Framework: Cadre de travail proposant une architecture « prête à l'emploi ». Un Framework peut être surchargé de librairies.

Frontend: Partie visuelle et graphique d'une application.

~H~

HTTP: Protocole de la couche application permettant de faire communiquer un client (Frontend) et un serveur (Backend).

HTTPS: Combinaison du http avec une couche de chiffrement SSL pour une communication plus sécurisée.

~J~

JSON: Format de données textuelles.

~L~

Librairie : Utilisation d'un code déjà fonctionnel permettant de satisfaire un besoin précis pour un développeur, lui permettant ainsi de gagner du temps.

~M~

Mapper : Permet de transférer un Object d'une couche à une autre.

~N~

nfIf : Directive native d'Angular permettant de mettre une condition.

nfFor : Directive native d'Angular permettant de faire une boucle.

~P~

Postman : Logiciel facile à prendre en main, possédant une interface graphique simple permettant d'interroger rapidement des web services en transmettant ou non des données et ainsi analyser leur comportement via leur retour de données.

Product Backlog : Contient une liste d'User Stories priorisées pour un sprint sous la responsabilité du Product Owner.

~R~

RAF (Reste A Faire) : Temps restant avec la finalisation de la tâche.

~S~

Scrum Master : Garant de la bonne utilisation de méthode Scrum, protège son équipe des potentielles perturbations extérieures.

SQL : Langage informatique permettant d'exploiter (ajouter, modifier, supprimer) des bases de données relationnelles.

Spécification fonctionnelle : Également nommée spécification fonctionnelle détaillée (SFD) décrit les fonctions d'un logiciel ou d'une application dans le but de la réaliser.

Swagger : C'est un langage de description d'interface permettant de décrire des API RESTful exprimées à l'aide de JSON.

~T~

TMA-PNUM : Tierce Maintenance Applicative – Parcours Numérique

~U~

User Story : Description précise d'une tâche présente dans le Product Backlog.

3. PRESENTATION DE L'ENTREPRISE

Open est un acteur local et reconnu depuis plus de 20 ans dans l'écosystème numérique de Rennes et de l'Ille-et-Vilaine.

L'activité de l'agence se caractérise par sa forte diversification sectorielle, permettant d'adresser les enjeux de ses clients locaux, régionaux et nationaux : télécommunications et média, secteur public (territorial, hospitalier et Etat), banque, industrie.

Les principaux sites d'OPEN et leurs spécialités :

- ▶ Rennes / Lannion : Centre de production et expertises portails et plateformes web, applications mobiles et géo, solutions digitales Open Swizi et FullMaps,
- ▶ Nantes : Centre de Services dédié aux tests d'applications omni-canales,
- ▶ Tours : Centre de Services dédié au support applicatif et au maintien en conditions opérationnelles, solution digitale Open Primpromo,
- ▶ Toulouse / Bordeaux : Des agences ancrées localement, développant des expertises AS/IS et métiers en lien avec le territoire

Les équipes Open couvrent un panel de compétences techniques large : .Net, JEE, PHP, Technologies mobiles (iOS, Android, ...) ...

4. PRESENTATION DU POSTE ET DE SON ENVIRONNEMENT TECHNIQUE

Open est basé à Cesson-Sévigné mais le client du projet TMA-PNUM est situé à l'ANTAI, allée Ermengarde d'Anjou à Rennes. Nous avons droit à 3 jours de télétravail et 2 jours sur site.

Les applications de la TMA-PNUM :

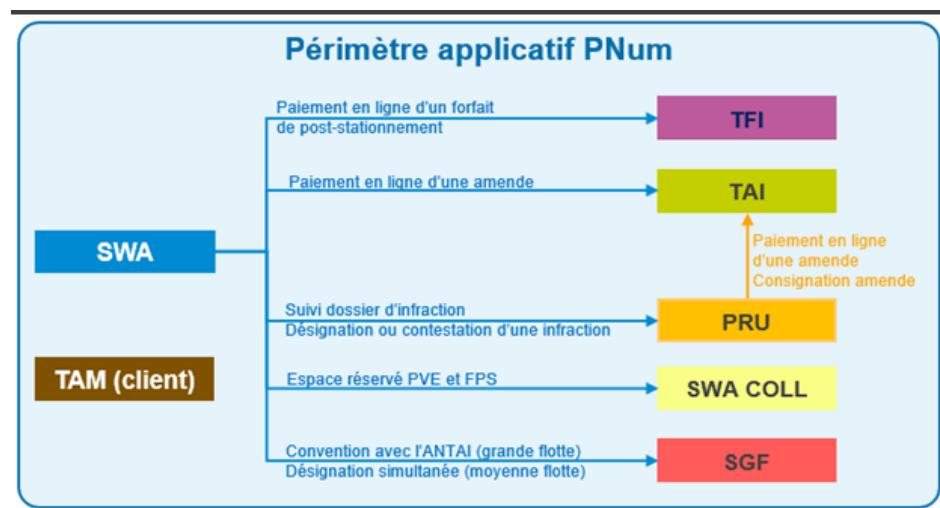


Figure 3 : périmètre applicatif PNUM

Mission du projet :

3.5 Prestations attendues

Les prestations attendues dans le présent marché sont précisées dans le tableau ci-après :

N°	Intitulé	Description synthétique
P0	Reprise	Activités de reprise des prestations des titulaires sortants du CNT4
P1	Pilotage du marché	Activités de pilotage des prestations du marchés
P2	Gestion du contenu éditorial	Gestion et animation du contenu éditorial du site web de l'ANTAI
P3	Tierce Maintenance Applicative (TMA)	Activités de maintenance corrective, préventive, adaptative, évolutive et demandes complémentaires
P4	Support de niveau N3	Activités de support d'expertise de niveau 3
P5	Formation	Activités de préparation et de réalisation de formations
P6	Sécurité des systèmes d'information (SSI)	Activités liées au respect des règles de SSI
P7	Plan de Continuité d'Activité (PCA)	Activités liées au PCA de l'ANTAI et réalisation d'exercices réguliers
P8	Challenge d'innovation	Organisation de challenges d'innovation
P9	Etudes	Réalisation d'études d'architecture, d'études d'impact technico-fonctionnelles et d'études statistiques
P10	Transfert	Activités de préparation et de réalisation du transfert de responsabilité vers un ou plusieurs tiers

Figure 4 : Mission du projet

Environnement technique :



Figure 5 : logo IntelliJ

Pour l'ensemble des développements liés au projet, nous utilisons l'**IDE IntelliJ**. C'est un environnement de développement intégré destiné au développement de logiciels informatiques reposant sur la technologie Java.



Figure 6 : Logo Angular

La partie Frontend est réalisée en Angular, qui est un **Framework** open source côté client créé par Google en 2013 succédant à AngularJS (2009). Ce dernier permet de réaliser des interfaces graphiques. Il est aujourd’hui le framework Frontend le plus complet, fournissant nativement tout le nécessaire (module de routing, formulaire, test, ...). Il permet d’écrire son code en TypeScript qui est ensuite transpilé en JavaScript.



Figure 7 : Logo Swagger

L’outil utilisé pour réaliser la conception technique est Swagger, qui est un **logiciel** open source développé en 2011. Il permet de créer la documentation d’une API REST (Swagger UI) et aussi de générer son code afin d’utiliser cette API REST (Swagger Codegen). Cette génération de code est utilisée dans le projet, elle est de plus essentielle. Swagger fonctionne grâce à un fichier JSON*. C’est dans ce fichier que nous pouvons définir l’ensemble des objets (ou modèle) à utiliser en entrées ou en sorties ainsi que les points d’entrées.



Figure 8 : Logo Spring

La partie Backend est développée en Java EE via le **Framework** open source Spring, qui est actuellement le Framework le plus populaire en Java EE.



Figure 9 : Logo Git

Git est le **logiciel** de gestion de version libre créé par Linus Torvald en 2005, C'est la solution utilisée par le groupe OPEN. Il est aujourd'hui le logiciel de version le plus utilisé dans le monde du développement. L'outil a été choisi pour sa fiabilité, sa popularité (tout le monde sait l'utiliser) et enfin pour respecter la norme imposée par le projet. En effet chaque ligne de code doit être relue par un autre développeur afin de limiter au maximum les erreurs, la fonctionnalité Merge Request couplée à l'interface de GitLab permet de comparer les lignes de façon très visuelle.

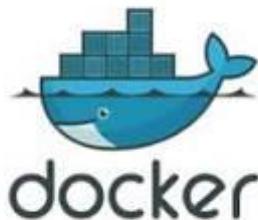


Figure 10 : Logo Docker

Docker est une plateforme permettant de lancer certaines applications dans des conteneurs logiciels. Le fichier docker mis en place sur les environnements de test du projet crée les containers suivants :

- ▶ Un serveur MariaDB (port 6666) initialisé à chaque lancement avec un backup des bases de données de test
- ▶ Gestion_Acces: docker/gestion_acces.sql
- ▶ SGF_Designation: docker/sgf_designation.sql
- ▶ Gestion_Documents: docker/gestion_documents.sql
- ▶ PhpMyAdmin (port 667)
- ▶ Adminer (port 668)
- ▶ Portainer (une interface de gestion pour Docker, port 9000)
- ▶ mailhog (port 25)
- ▶ openldap (port 389) initialisé avec le fichier docker/ldap.ldif.
- ▶ clamav (port tcp 3311): antivirus

5. COMPÉTENCES DU REFERENTIEL COUVERTES PAR LE PROJET

Tout au long de mon alternance, j'ai pu mettre en pratique les compétences du référentiel pour le projet (fiche « Compétences Rapport CDA.docx » en annexe 1) :

Activité 1 : Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité :

- ▶ Maquetter une application avec l'outil Moqups
- ▶ Développer la partie front-end en Angular
- ▶ Développer la partie back-end d'une application web avec Java et SpringBoot
- ▶ Développer les composants d'accès aux données avec SpringBoot

Activité 2 : Concevoir et développer la persistance des données en intégrant les recommandations de sécurité :

- ▶ Mettre en place une base de données

Activité 3 : Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité :

- ▶ Concevoir une application
- ▶ Développer des composants métier
- ▶ Construire une application organisée en couches



Figure 11 : Logo Moqups



Figure 12 : Logo Angular



Figure 13 : Logo Spring boot



Figure 14 : Logo Java

6. RESUME DU PROJET EN ANGLAIS

A company specialized in supporting companies in their industrial and digital transformation challenges, OPEN Group has more than 500 customers in several sectors of activity:

- ▶ Banking and Insurance (Société Générale, Crédit Agricole, BNP Paribas, ...)
- ▶ Services and transportation (SNCF, RATP, La Poste, ...)
- ▶ Energy (EDF, Engie, Suez, ...)
- ▶ Industry (PSA, Renault, Arcelor, ...)
- ▶ Public sector (RSI, Ministry of the Interior, ...)
- ▶ Telecom and media (Orange, SFR, ...)
- ▶ Commerce and distribution (Auchan, Boulanger, Système U, ...)

The PNUM project is a project of the ANTAI (Agence Nationale de Traitement Automatisé des Infractions), a French public institution attached to the Ministry of the Interior. The project started in October 2019 and consists of 7 different applications:

- ▶ SWA: ANTAI's institutional website, all the links to the other applications can be found here.
- ▶ SWAPART: Communities and other ticketing services area for communities to manage violations.
- ▶ PRU (Parcours Relation Usagers): Space for users to contest their tickets.
- ▶ SGF: Professional space for Fleet Managers. Allows the management of average fleet (between 11 and 1000 vehicles) or in mass (from 1001 vehicles and +).
- ▶ TAI : Telepayment / Payment Interface.
- ▶ TFI: Telepayment site for the post-parking fee
- ▶ TAM: Mobile application for fines.gouv

We are a team of 18 people spread over the different applications. The application I worked on is SwaPart with Java/Spring for the backend part and Angular for the frontend part.

The project is an implementation of a messaging system between the communities and the ANTAI administrators for the application Swa-Part : application bringing together organizations that wish to use the ANTAI infrastructures for the management of fines and their printing. Today, exchanges are only done by email, and the interest of messaging is to centralize these exchanges.

7. CAHIER DES CHARGES

7.1. Présentation

Le client ne nous a pas fourni de cahier des charges précis, il a été fait par les fonctionnels du projet par des ateliers de mûrissement.

Double usage :

- ▶ Usage DAF sur facturation : échanges messages.
- ▶ Usage opérationnel.

Côté opérationnel, actuellement, il existe un fichier Excel avec l'historique des échanges avec les collectivités.

Il sert de base de connaissance dans les échanges

Des fichiers Word de templates de réponses existent également pour traiter ces échanges.

7.2. Expression des besoins

Le besoin consiste en la mise à disposition d'un outil de communication de type messagerie conventionnelle permettant de gérer les échanges avec les collectivités de manière efficiente, tout en conservant l'historique de ces échanges et les adresses mails. L'outil de communication doit pouvoir être évolutif.

Côté admin, on accède à tous les messages d'une collectivité et on écrit à toute l'organisation.

Côté utilisateur, on envoie en groupes à l'administrateur avec le nom de l'émetteur. La fenêtre de visualisation est commune pour tous les membres de l'organisation. L'utilisateur a accès à plusieurs thématiques dans un menu déroulant pour envoyer son message groupé à l'admin correspondant : PVE, FPS, SGF (plus tard), SI fourrière avec des propriétés complémentaires pour ensuite mettre des files de traitement côté admin.

Impacts côté SWA-Part et côté appli Administration.

7.3. Gestion de projet

7.3.1. Planning et suivi

L'ANTAI a choisi d'appliquer un modèle SAFe afin de faire travailler les différentes TMAs en parallèle et en suivant le même planning.

Cette méthodologie procède par programmes de développement de trois mois (appelés PI), découpés en 4 sprint de 3 semaines et avec une semaine de PI Planning entre chaque.

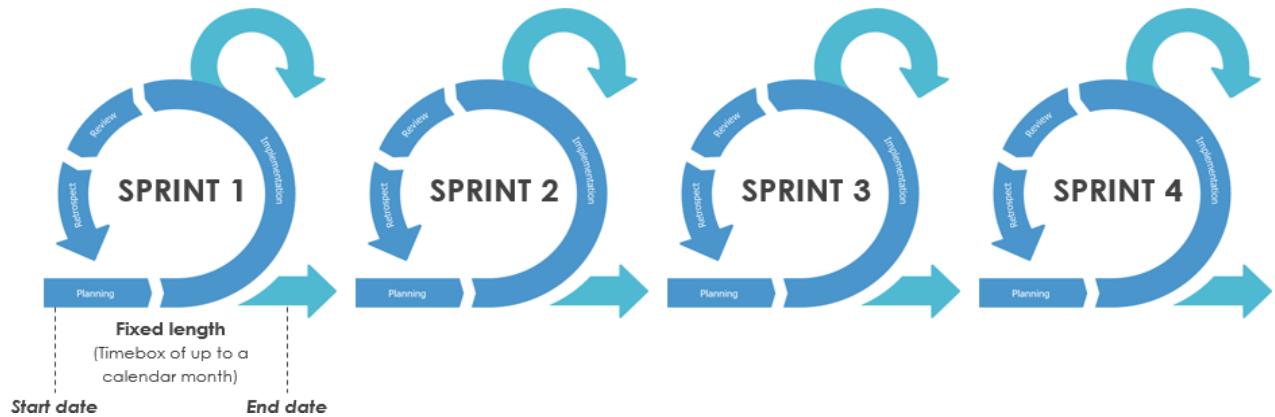


Figure 15 : Présentation PI Planning

L'équipe Open fonctionne suivant la méthodologie Scrum. Ce framework implique la tenue d'un nombre non négligeable de réunions dites « cérémonies agiles ». Celles-ci structurent littéralement la vie de l'équipe. Elles donnent le rythme du projet, favorisent les échanges et la collaboration. Elles doivent se tenir dans un climat de partage et de confiance mutuelle.

- ▶ Sprint Planning : Réunion lors du premier jour du sprint avec l'équipe pendant environ 1 heure. Elle permet de définir les objectifs du sprint. Le Scrum Master présente les tâches qui seront définies dans le Product Backlog (Annexe 2) du sprint qui arrive. Ces tâches découpées en User story sont présentées et un niveau d'effort est discuté puis défini par l'équipe. Ce niveau d'effort définit à chaque sprint permet de mesurer plus facilement le périmètre pouvant tenir dans un sprint au fur et à mesure que nous avançons dans le projet (vitesse).
- ▶ Daily meeting (mêlée) : Réunit toute l'équipe quotidiennement pendant 15 minutes. Cette réunion par visio-conférence doit faire parler tous les membres de l'équipe et faire dire à chacun ce qui a été fait la veille, les problèmes rencontrés et ce qui sera fait le jour même. Elle permet le suivi des objectifs du sprint et d'identifier et supprimer des obstacles.
- ▶ Sprint Review : Dernier jour du sprint, réunit l'équipe pendant 1 heure afin de valider les objectifs, passer en revue les sujets terminés et report des sujets non clôturés sur le sprint suivant.
- ▶ La revue de sprint (démonstration) : Dernier jour du sprint également. Réunit toute l'équipe et le client pour présenter les fonctionnalités réalisées et prendre en compte les retours clients.
- ▶ La rétrospective : Ce fait le dernier jour du sprint, cela dure environ 2h. Elle réunit toute l'équipe et permet d'analyser le travail effectué, de recueillir les

retours constructifs et de proposer des axes d'améliorations pour le sprint suivant.

- ▶ Backlog Refinement : Tous les jeudis durant 2h, réunit l'équipe fonctionnelle et le client. Cela peut être les ateliers de mûrissement des sujets, priorisation du backlog, rédaction des User Stories ou faire une macro-estimation des User Stories.
- ▶ Planning Poker : Ce fait en amont du PI Planning avec 2j par itération. Réunit les équipes par projet et permet de découper en tâches des User Stories, de faire le chiffrage des tâches et permet la relecture et la validation des User Stories (DoR respectée).

Afin de permettre le bon suivi du projet, Open utilise le logiciel Jira, nous permettant de créer, suivre et planifier des User Stories ainsi que les tâches associées par rapport à la demande de notre client. Il se présente sous forme d'un tableau Kanban (Annexe 3). Ce Kanban est partagé par toutes les équipes du projet. Chacun devant mettre à jour sa User story (Annexe 4) en la mettant dans la bonne colonne :

- ▶ TODO : la User story n'a pas été prise en compte par les développeurs.
- ▶ IN PROGRESS : la User story est en cours de développement et le développeur met à jour son RAF.
- ▶ TESTING : la User story a été développée et un ou plusieurs tests unitaires ont été effectués.
- ▶ DONE : l'équipe de qualification a validé la User story, la fonctionnalité pourra être embarquée dans la livraison cliente du sprint



Figure 16 : logo Jira

La partie messagerie a été réalisé sur deux PI Planning. Pendant le premier, nous devions mettre en place la messagerie (Messages, conversations) et pendant le deuxième, mettre en place les pièces-jointes.

7.3.2. Environnement humain et technique

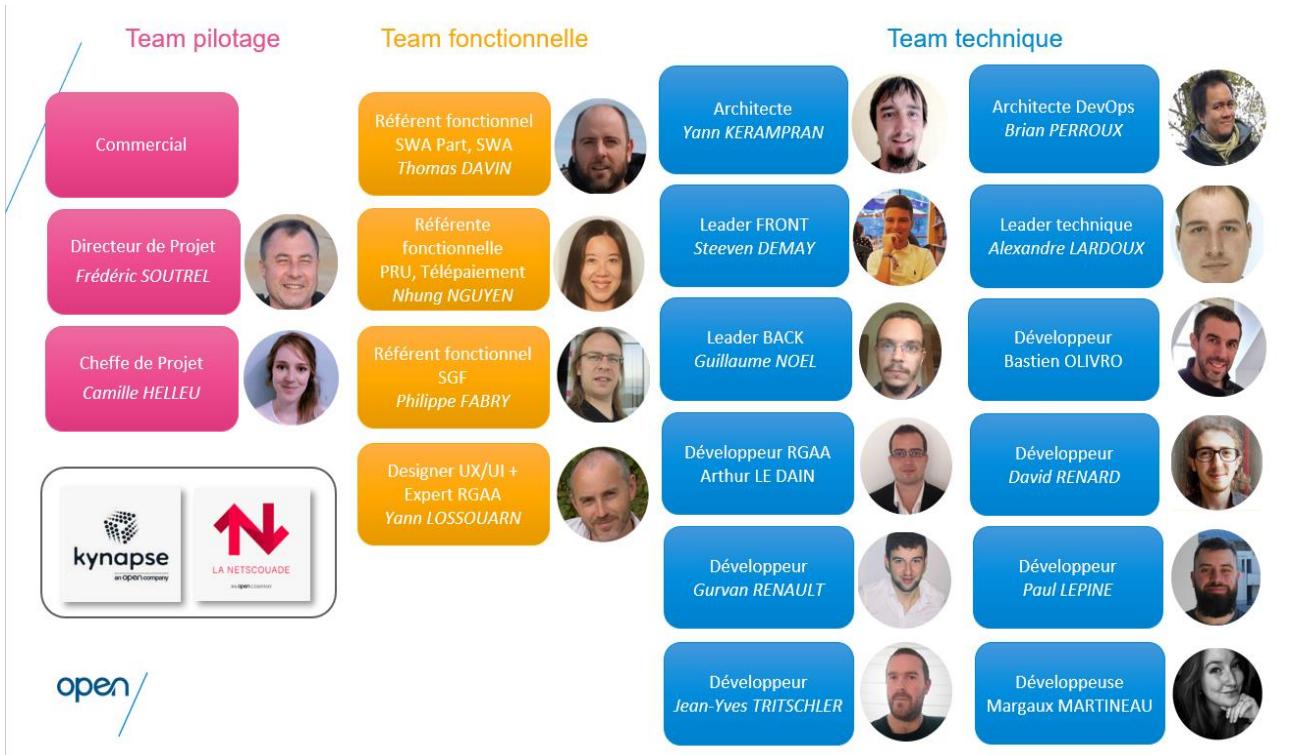


Figure 17 : équipe TMA-PNUM

Nous sommes une équipe de 18 personnes, nous avons :

- ▶ La partie pilotage avec notre directeur de projet et notre cheffe de projet,
- ▶ La partie fonctionnelle avec 3 référents pour les différentes applications du projet et un designer UX/UI + Expert RGAA pour la réalisation des maquettes et des audits RGAA.
- ▶ La partie technique, on y retrouve les architectes, un leader technique qui a été mon tuteur durant mon alternance, deux leader (un back et un front) et 7 développeurs (Back, Front et Full stack)

Pour la réalisation de mon projet, nous étions une équipe de 5 personnes, Steeven en tant que leader Front, Alexandre en leader technique, David et Bastien en développeurs Full stack et moi-même.

Nous avons travaillé sur ce projet avec une approche agile. La méthodologie Agile s'oppose aux méthodes de travail traditionnelles (Cycle en V, Cascade). Elle se veut souple en impliquant au maximum le client et en réagissant rapidement à ses demandes. Cette souplesse permet une meilleure gestion du changement et de pouvoir revenir sur des fonctionnalités déjà spécifiées ou même déjà développées. La méthode agile à un cycle de développement itératif et adaptatif.

Parmi les différentes méthodes agiles existantes, il a été choisi la plus populaire : la méthode SCRUM.

Elle se prête à la taille de l'équipe actuelle et à la modularité que nous pouvons en faire.

La méthode SCRUM se fonde sur trois piliers :

- ▶ La transparence : en interne et vis-à-vis du client
- ▶ Inspection : vérification régulière de la qualité du logiciel.
- ▶ Adaptation : si problème rencontré sur l'inspection, adapter ou corriger le problème au plus vite.

7.4. Spécifications fonctionnelles

Les spécifications fonctionnelles sont stockées sous l'outil client Confluence. Leur mise à jour se fait via le canal Teams "Team fonctionnelle" afin de faciliter le travail en parallèle des fonctionnels, et elles sont sauvegardées une fois par semaine sur Confluence.

Les specs doivent respecter un template et doivent être lu par chaque développeurs avant le commencement du sprint.

Pour les specs messagerie (annexe 5), on y retrouve un récapitulatif de chaque écran souhaité, puis on passe sur chaque écran en présentant :

- ▶ La description générale
- ▶ Les règles d'entrée
- ▶ La maquette
- ▶ La description détaillée de l'écran
- ▶ Les règles d'initialisation
- ▶ Les règles de gestion
- ▶ Les messages d'erreurs

5.8. Ma messagerie

5.8.1. Ecrans

<u>Code</u>	<u>Ecran</u>
<u>CSUL_MESS_ORG</u>	<u>Écran « Ma messagerie »</u>
<u>CREA_MESS_ORG</u>	<u>Écran « Créer un nouveau message »</u>
<u>REP_MESS_ORG</u>	<u>Écran « Répondre à un message »</u>

Figure 18 : Spec messagerie

Ici, nous avons 3 écrans :

- ▶ La page principale pour la consultation de la liste des conversations (CSUL_MESS_ORG)
- ▶ La page pour créer un nouveau message accessible qu'à l'espace Partenaire (CREA_MESS_ORG)
- ▶ La page de conversation avec la liste des messages et l'encart pour répondre aux message (REP_MESS_ORG)

8. SPECIFICATIONS TECHNIQUES

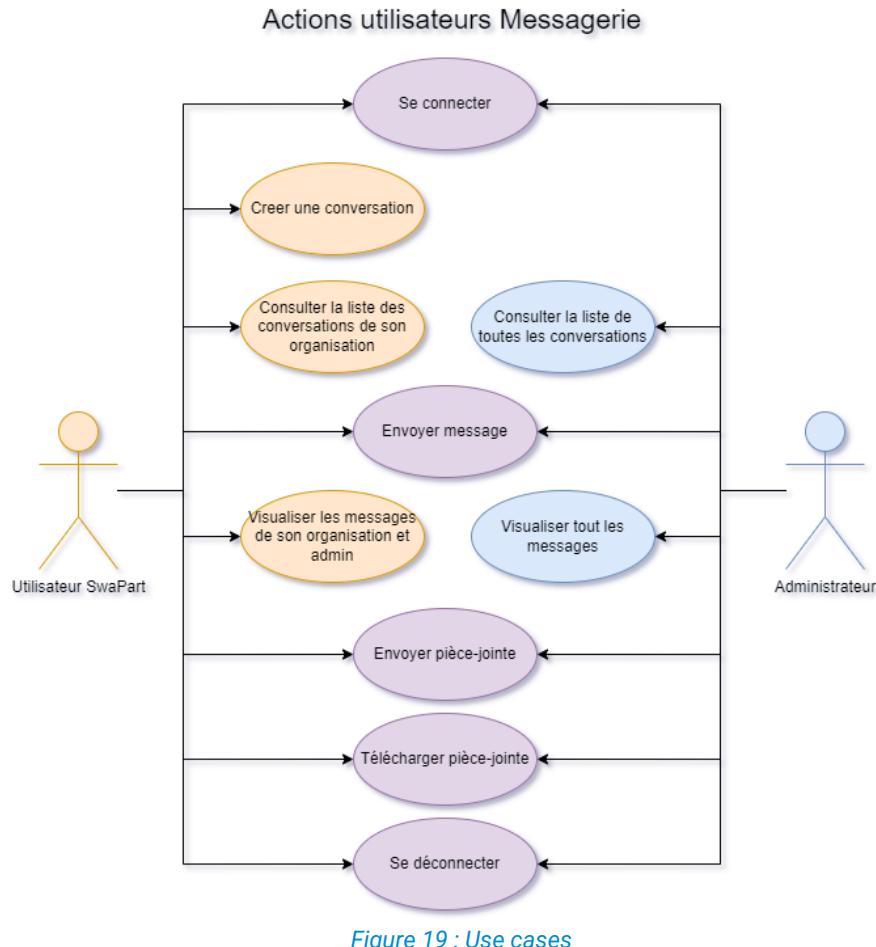
8.1. Présentation

Avant de commencer à développer ma partie du projet, j'ai dû mettre en place :

- ▶ Un Use Case pour me permettre de comprendre les fonctionnalités attendues par le client,
- ▶ Mettre en place des maquettes pour les évolutions des IHM,
- ▶ Créer le schéma de base de données pour comprendre les relations entre les entités à créer,
- ▶ Comprendre l'architecture micro-service qui est utilisée.

8.2. Use Case

Ce diagramme de cas d'utilisation présente une vision globale des fonctions attendues par type de user.



Il y a deux types de Users :

- ▶ Les collectivités (Utilisateurs de l'application SwaPart) : Il n'y a qu'eux qui peuvent créer une conversation. Ils peuvent consulter la liste des conversations de leur propre organisation, et visualiser les messages envoyés de leur organisation et des administrateurs. Ils peuvent également envoyer et télécharger les pièces-jointes
- ▶ Les administrateurs : Après s'être connecté, tout administrateur peut consulter la liste des conversations entre eux et les collectivités, visualiser les messages, y répondre et envoyer et télécharger les pièces-jointes.

8.3. Maquettes

La partie messagerie est une nouveauté pour l'application SWA-PART, il a donc fallu créer les différentes pages en maquettes en respectant l'existant sur les différents onglets, tableaux de l'application.

Après la connexion à SWA PART, dans le haut des pages (header), une icône enveloppe permet d'accéder à la messagerie pour l'utilisateur.

L'enveloppe indiquera si des messages non lus par le groupe sont disponibles, il est demandé que l'enveloppe soit développée en RGAA (Voice over indiquant « accès à la messagerie »).

En cliquant sur l'icône, l'utilisateur accède à ses messages.

Nouvel onglet 'Ma messagerie' en dernière position après 'Mes documents' toujours visible.

Un clic sur le titre 'Ma messagerie' ouvre également la messagerie de l'utilisateur.

The screenshot shows the ANTAI 'espace partenaires' dashboard. At the top, there are logos for the République Française, ANTAI (Agence Nationale de Traitement Automatisé d'Infractions), and a mail icon indicating 1 message. The navigation bar includes 'MON COMPTE', 'MES DOCUMENTS', and 'MA MESSAGERIE' (which is underlined in blue). Below the navigation, the title 'MA MESSAGERIE' is centered. A search/filter section titled 'Filtrer les messages' allows filtering by Service (Tout), Sujet (Tout), and Objet, with buttons for 'FILTRER' and 'X Réinitialiser'. The main area displays a table of messages:

Service	Sujet	Objet	Date	Auteur	Message	Action
FPS	Connexion AGC		02/09/22 10:05	Jules Meunier		
Freeflow	Problème technique		03/09/22 10:55			
SI Fournières	Validation compte		20/08/22 14:26			
PVE	Annulation		10/08/22 16:41			
Freeflow	Traitements des tickets		15/07/22 09:05			

At the bottom of the message list, it says '1 - 5 of 5' and has navigation arrows. A blue button labeled 'NOUVEAU MESSAGE' is located below the table. At the very bottom, there's a footer with links to 'Sites Utiles' (interieur.gouv.fr, ants.gouv.fr, service-public.fr, legifrance.gouv.fr, securite-routiere.gouv.fr) and a link to 'Accueil | A propos de l'ANTAI'.

Figure 20 : Maquette Messagerie

Pour la deuxième maquette, nous nous retrouvons sur la page « Vue Conversation + Ajout d'une PJ ».

Après avoir cliqué sur l'icône œil (action sur tableau de messagerie), la liste des messages de la conversation est affichée, de la plus récente à la plus ancienne.

Faire un affichage type Teams : distinguer message administrateur ou organisation (droite-gauche : à gauche tous les messages administrateurs, à droite tous les messages organisation, Couleur différente admin/orga, indiquer le nom/prénom de l'auteur du message ainsi que la date d'envoi au format JJ/MM/AA HH:MM).

Si des pièces jointes sont attachées au message, en fin de message, les PJ sont affichées précédées d'une icône trombone.

Un clic sur la pièce jointe permet de la télécharger.

Enfin, un espace au-dessus du dernier message permet d'écrire un nouvel item à la discussion.

Il est également possible d'y joindre des pièces jointes de moins de 5 Mo et de type PDF, JPEG/JPG/PNG/TIF/TIFF (Images), DOC/DOCX (Word), XLS/XLSX (Excel) via une icône trombone qui ouvre un pop-up d'ajout de pièces jointes.

Un bouton permet d'envoyer le nouveau message, si du texte a été saisi.

Pas de possibilité de modification de message envoyé.

The screenshot shows the ANTAI espace partenaires messaging interface. At the top, there are logos for the République Française, ANTAI (Agence Nationale de Traitement Automatisé d'Infractions), and the espace partenaires logo. On the right, there is a yellow envelope icon with '0 message' and a 'DECONNEXION' button.

The main navigation bar includes 'MON COMPTE', 'MES DOCUMENTS', and 'MA MESSAGERIE', with 'MA MESSAGERIE' being the active tab. Below the navigation, a link '[← Revenir à la liste des messages](#)' is visible.

MA MESSAGERIE

A message list is displayed:

- Service : FPS**, **Sujet : Connexion AGC**, **Objet : [redacted]**
- Jules Meunier** (ADMIN) - 02/09/2022 10:05: Message body [redacted]
- Aimé Pires** (ORGANISATION) - 02/09/2022 09:32: Attached files: FICHIER1.JPG and FICHIER2.PDF
- Lorraine Tessier** (ADMIN) - 01/09/2022 16:08: Message body [redacted]
- Richard Bourdon** (ORGANISATION) - 01/09/2022 14:55: Message body [redacted]

Sites Utiles section at the bottom:

- [interieur.gouv.fr](#) | [ants.gouv.fr](#) | [service-public.fr](#) | [legifrance.gouv.fr](#) | [securite-routiere.gouv.fr](#)

Footer links:

- [Accueil](#) | [A propos de l'ANTAI](#) | [Accessibilité : non conforme](#) | [Contact](#) | [Données personnelles](#) | [Mentions légales](#)

Figure 21 : Maquette Vue Conversation + Ajout d'une PJ

8.4. Schéma de base de données

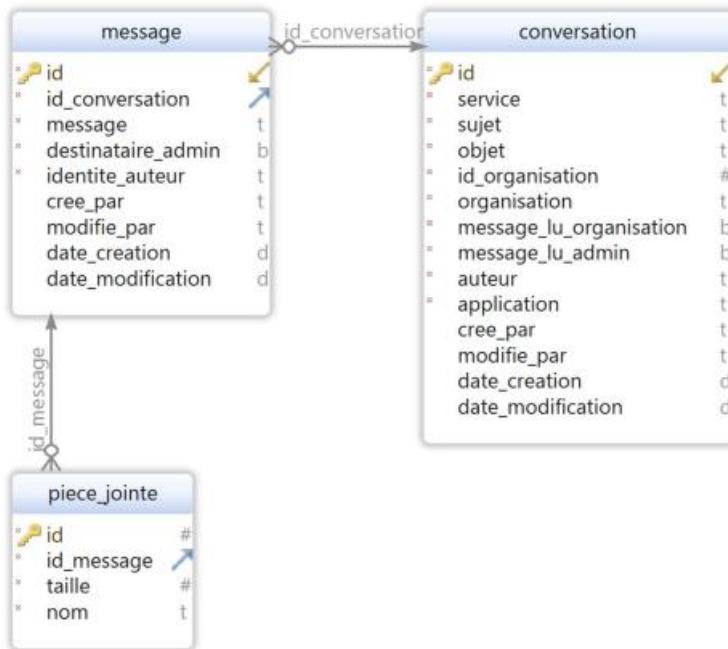


Figure 22 : MCD

Nous avons 3 tables :

- ▶ La table « conversation » : On y retrouve les 3 points obligatoires lors de la création d'une conversation : le service, le sujet et l'objet, l'id de l'organisation qui a créé la conversation, son auteur.
- ▶ La table « message » : La table est reliée à la table conversation par l'id. On y retrouve également le message écrit, si le message est à destination de l'administrateur ou de l'organisation ainsi que l'identité de l'auteur du message.
- ▶ La table « piece_jointe » : On transfère la pièce jointe en Base64 et elle est ensuite enregistrée sur disque dans un répertoire prévu à cet effet. Nous avons juste besoin d'avoir la taille et le nom de la pièce jointe ainsi que l'id du message auquel elle est reliée.

8.5. Architecture du projet

L'architecture utilisée est celle de micro-service :

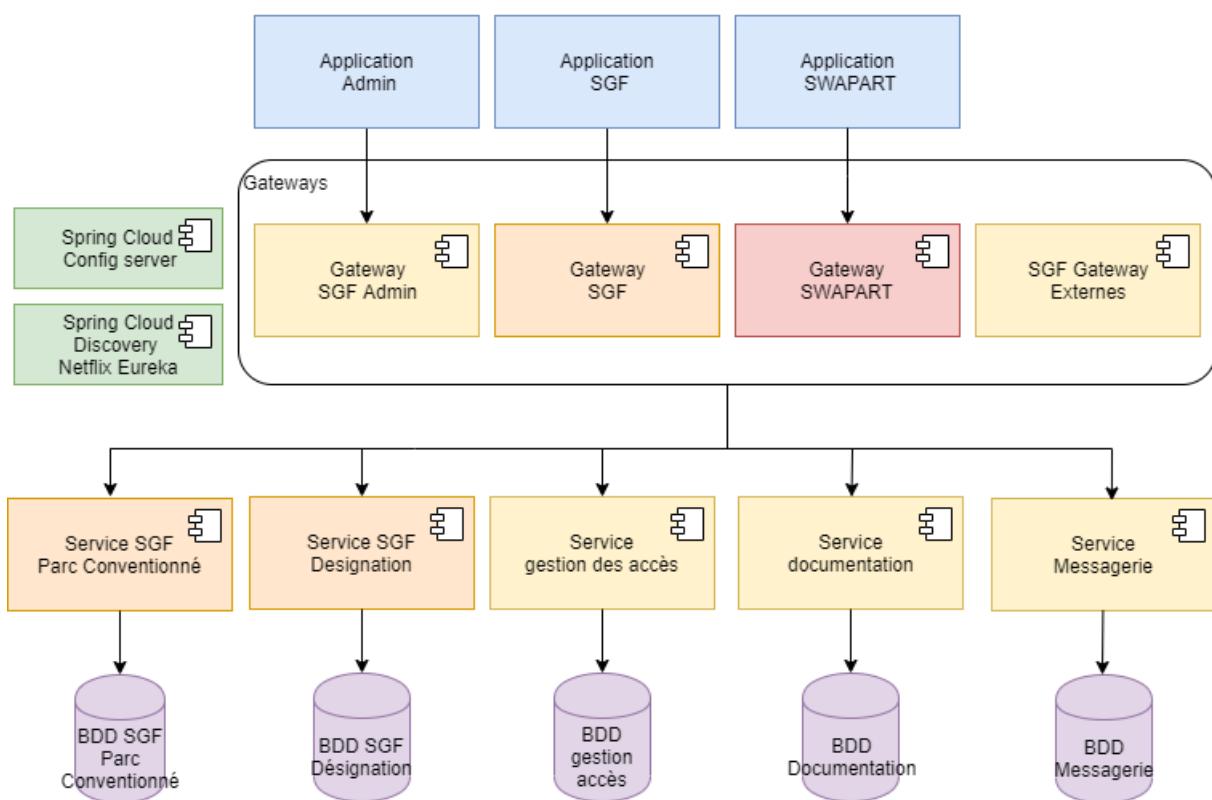
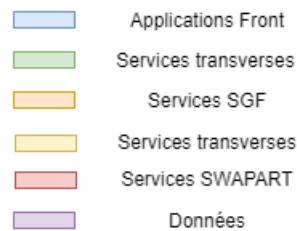


Figure 23 : Architecture Micro-service

Qu'est-ce qu'une architecture micro-service ?

L'architecture micro-services est une méthode de développement logiciel qui vise à décomposer une application pour en isoler les fonctions clés, chacune de ces fonctions est appelée « service ». Ces services sont créés pour répondre à un besoin métier précis et unique, par exemple : la gestion des utilisateurs, les paiements, l'envoi d'e-mails ou encore les notifications. De plus, ils sont indépendants et modulables, cela permet à chacun d'être développé et déployé sans affecter les autres. Ce type d'architecture se veut l'opposé des architectures monolithes qui sont construites

comme une seule unité autonome. Chaque module répond à un objectif métier spécifique et communique avec les autres modules. Il est également plus facile d'identifier et de résoudre une panne dans ce type d'écosystème.

9. REALISATION

9.1. Présentation

Après installation de l'environnement et après avoir pris connaissance du besoin client, on m'a confié comme mission :

- ▶ La création d'un nouveau micro-service « Messagerie »
- ▶ La mise en place d'un contrat d'interface (annexe 7)
- ▶ La mise en place d'une base de données
- ▶ Pour la partie back-end j'avais la réalisation de l'action « Envoyer message avec ou sans pièce-jointe » et « Télécharger pièce-jointe »
- ▶ Pour la partie front-end, l'affichage de l'écran « Liste des messages dans une conversation » avec ou sans pièce-jointe.

Pour la création du micro-service métier :

- ▶ J'ai repris le micro-service Gestion-Documents déjà existant en gardant l'architecture et en supprimant les contenus.
- ▶ J'ai ensuite mis à jour la configuration de la Gateway-SWAPART dans le service-config (swapart-gateway.yml).
- ▶ J'ai initialiser le fichier de configuration du micro-service Messagerie à déposer dans le service-config (service-messagerie.yml)
- ▶ Configuration ansible du micro-service

J'ai ensuite réalisé le contrat d'interface en me basant sur les informations ci-dessous :

Création des DTO (Data transfert Object) :

- ▶ MessageDTO
 - Les PJ seront en base64
 - Tous les champs message sauf les fixes (objet, sujet, service) + idOrganisation + bool lu + bool destinataireAdmin (true ->si l'expéditeur est l'organisation) + dateCreation + utilisateurCreation (identité utilisateur mail ou nom/prenom string)
 - Contient une liste de MessagePJ : idPj, taille du fichier, extension

- ▶ ConversationDTO avec objet, sujet, service

Création des routes :

- ▶ Création d'un message avec nouvelle conversation Route POST messagerie/conversation
 - Crée un nouvel Objet CreationConversation qui contient les infos conversation + message + 3 pièces jointes
- ▶ Création d'un message avec conversation existante Route POST messagerie/message/{idConversation}
 - Envoie un MessageDTO
- ▶ Récupération des messages d'une conversation Route GET messagerie/messages/{idConversation}
 - Récupère les messages d'une conversation
- ▶ Route GET /messagerie/conversations renvoie une liste de conversation + dernier message
 - Crée Objet ConversationGenerale qui contient Service / Sujet / Objet / Date / Auteur / Message (tronqué)
 - Prévoir tri et pagination
- ▶ Route GET /messagerie/messagesNonLu qui renvoie un int
 - Même route Admin & non-admin
- ▶ Route GET /messagerie/message/piecejointe
 - Récupération des pièces jointes d'un message pour les télécharger

Le swagger (Langage de description d'interface permettant de décrire les APIRest) permet de faire le lien entre le front et le back et permet d'empêcher les incohérences entre les deux parties. Par exemple, nous avons un attribut nommé EnumOrga dans la partie front et back, nous décidons de le changer dans le swagger en TypeOrgnisationEnum, il sera alors mis à jour des deux côtés.

Une API est une interface de programmation composée de classes, méthodes, fonctions servants de façade pour un logiciel qui offre des services à un autre logiciel. Il existe différents types d'API, Open utilise l'API REST.

Le standard REST a été défini par Roy Fielding en l'an 2000 dans sa thèse de doctorat. Une API REST permet de faire communiquer sans état un serveur et un client via des requêtes HTTP.

La communication HTTP se fait via une URL et une méthode. Les méthodes les plus populaires sont :

- ▶ GET : pour obtenir des informations du serveur.
- ▶ POST : pour envoyer un ensemble d'informations au serveur.
- ▶ PUT : pour mettre à jour des informations.
- ▶ DELETE : pour supprimer des informations.

Le serveur retourne un code (statut HTTP) et une réponse au client le plus souvent sous format JSON.

Les codes les plus populaires :

- ▶ 200 : Tout s'est bien passé.
- ▶ 401 : Authentification échoué.
- ▶ 403 : Pas les droits d'accès à la ressource.
- ▶ 404 : La ressource n'existe pas.
- ▶ 500 : Le serveur rencontre un problème.

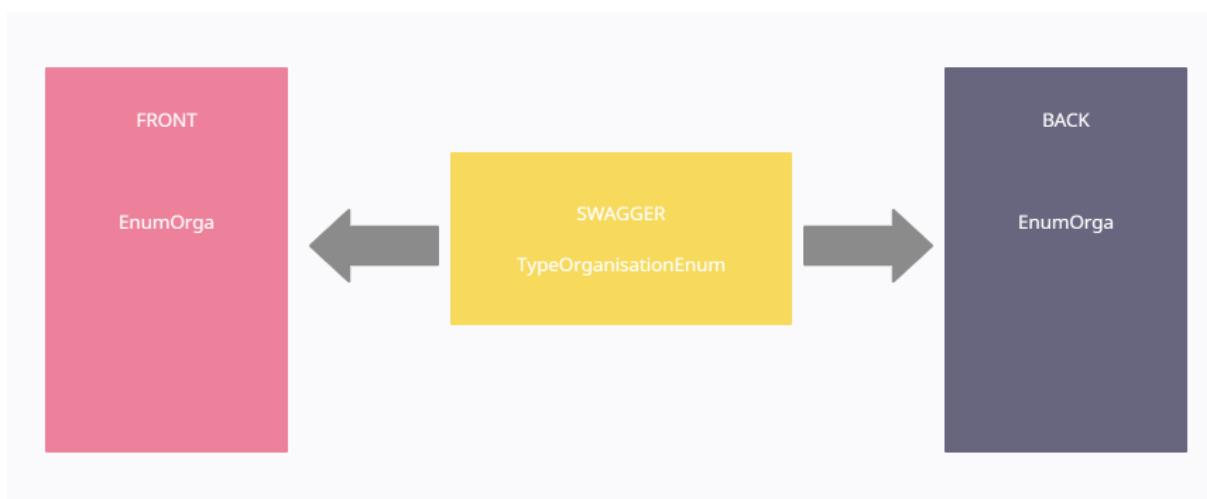


Figure 24 : Swagger

9.2. Réalisation de la base de données

Pour cette partie, j'ai dans un premier temps réaliser le schéma de la base de données en cohérence avec les informations que mon fonctionnel référent m'a fourni et de la création du schéma de la base de données réalisé auparavant.

Ma base de données se nomme « Service_Messagerie » et est en interclassement utf8_general_ci.

```

DROP SCHEMA IF EXISTS `Service_Messagerie`;

CREATE SCHEMA `Service_Messagerie` DEFAULT CHARACTER SET utf8;

USE Service_Messagerie;

create table conversation
(
    id          BIGINT      not null AUTO_INCREMENT primary key,
    service     enum ('FPS','FREEFLOW','SI_FOURRIERE','PVE') not null,
    sujet       varchar(80)  not null,
    objet       varchar(80)  not null,
    id_organisation  BIGINT   not null,
    organisation  varchar(80)  not null,
    message_lu_organisation bit    not null,
    message_lu_admin   bit    not null,
    auteur       varchar(240) not null,
    application   enum ('SWAPART') not null,
    cree_par     varchar(240) null,
    modifie_par   varchar(240) null,
    date_creation  datetime  null,
    date_modification  datetime  null
);

create table message
(
    id          BIGINT      not null AUTO_INCREMENT primary key,
    id_conversation  BIGINT      not null,
    message      varchar(3000) not null,
    destinataire_admin bit    not null,
    identite_auteur  varchar(240) not null,
    cree_par     varchar(240) null,
    modifie_par   varchar(240) null,
    date_creation  datetime  null,
    date_modification  datetime  null,
    constraint message_id_conversation_id_conversation_fk
        foreign key (id_conversation) references conversation (id)
);

create table piece_jointe
(
    id          BIGINT      not null AUTO_INCREMENT primary key,
    id_message  BIGINT      not null,
    taille      BIGINT      not null,
    nom         varchar(240) not null,
    constraint piece_jointe_id_message_id_message_fk
        foreign key (id_message) references message (id)
);

```

Figure 25 : script base de données

J'ai ensuite rajouté un fichier « service_messagerie_data.sql » afin de créer un jeu de données dans la base au lancement de Docker.

```
● ● ●
USE Service_Messagerie;
INSERT INTO Service_Messagerie.conversation (id, service, sujet, objet, id_organisation, organisation,
message_lu_organisation, message_lu_admin, auteur, application, cree_par, modifie_par, date_creation, date_modification)
VALUES (1, 'FPS', 'CONVENTIONNEMENT', 'Conventionnement', 1, 'Organisation 1', 0, 1, 'ANTAI', 'SWAPART', '', '',
'2022-11-10 12:00:00', '2022-11-10 14:00:00'),
(2, 'FREEFLOW', 'QUESTION_SERVICE', 'Question service', 2, 'Organisation 2', 1, 0, 'userfreeflow@test.com', 'SWAPART',
'', '', '2022-11-01 12:00:00', '2022-11-01 12:00:00'),
(3, 'SI_FOURRIERE', 'ANNULATION', 'Annulation', 1, 'Organisation 1', 0, 1, 'swapart@test.com', 'SWAPART', '', '',
'2022-11-12 12:00:00', '2022-11-12 12:00:00'),
(4, 'PVE', 'DEMANDE_ACCEES', 'Demande accès', 2, 'Organisation 2', 0, 1, 'userfreeflow@test.com', 'SWAPART', '', '',
'2022-11-14 12:00:00', '2022-11-14 12:00:00'),
(5, 'FPS', 'PROBLEME_TECHNIQUE', 'Problème technique', 1, 'Organisation 1', 0, 1, 'swapart@test.com', 'SWAPART', '',
'', '2022-11-16 20:00:00', '2022-11-16 20:00:00');

INSERT INTO Service_Messagerie.message (id, id_conversation, message, destinataire_admin, identite_auteur, cree_par,
modifie_par, date_creation, date_modification)
VALUES (1, 1, 'Message 1 conversation 1', 1, 'swapart@test.com', 'User Swapart', 'User Swapart', '2022-11-10 12:00:00', '2022-
11-10 12:00:00'),
(2, 1, 'Message 2 conversation 1', 0, 'ANTAI', 'Admin', 'Admin', '2022-11-10 14:00:00', '2022-11-10 14:00:00'),
(3, 2, 'Message 1 conversation 2', 1, 'userfreeflow@test.com', 'User Freeflow', 'User Freeflow', '2022-11-01 12:00:00',
'2022-11-01 12:00:00'),
(4, 3, 'Message 1 conversation 3', 1, 'swapart@test.com', 'User Swapart', 'User Swapart', '2022-11-12 12:00:00', '2022-
11-12 12:00:00'),
(5, 4, 'Message 1 conversation 4', 1, 'userfreeflow@test.com', 'User Freeflow', 'User Freeflow', '2022-11-14 12:00:00',
'2022-11-14 12:00:00'),
(6, 5, 'Message 1 conversation 5', 1, 'swapart@test.com', 'User Swapart', 'User Swapart', '2022-11-16 20:00:00', '2022-
11-16 20:00:00');
```

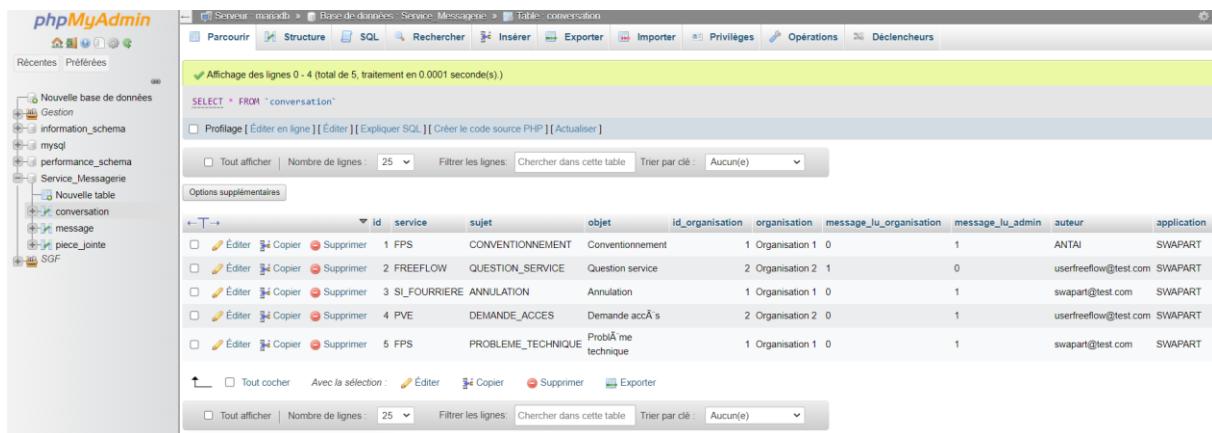
Figure 26 : script base de données data

J'ai créé un script SQL correspondant qui a été configuré pour être joué à chaque lancement de Docker et ainsi mettre à jour la base de données avec les informations.

```
● ● ●
volumes:
- ./maria_custom_conf:/etc/mysql/conf.d
- sgfmariadb_data:/var/lib/mysql
- ./data/sgf_designation.sql:/docker-entrypoint-initdb.d/sgf_designation.sql
- ./data/sgf_designation_data.sql:/docker-entrypoint-initdb.d/sgf_designation_data.sql
- ./data/gestion_acces.sql:/docker-entrypoint-initdb.d/gestion_acces.sql
- ./data/gestion_acces_data.sql:/docker-entrypoint-initdb.d/gestion_acces_data.sql
- ./data/gestion_documents.sql:/docker-entrypoint-initdb.d/gestion_documents.sql
- ./data/gestion_documents_data.sql:/docker-entrypoint-initdb.d/gestion_documents_data.sql
- ./data/service_messagerie.sql:/docker-entrypoint-initdb.d/service_messagerie.sql
- ./data/service_messagerie_data.sql:/docker-entrypoint-initdb.d/service_messagerie_data.sql
- ./data/sgf_parc_conventionne.sql:/docker-entrypoint-initdb.d/sgf_parc_conventionne.sql
- ./data/sgf_parc_conventionne_data.sql:/docker-entrypoint-initdb.d/sgf_parc_conventionne_data.sql
```

Figure 27 : script Docker

Voici le résultat dans ma base de données :



The screenshot shows the phpMyAdmin interface with the 'conversation' table selected. The table contains 5 rows of data:

	id	service	sujet	objet	id_organisation	organisation	message_lu_organisation	message_lu_admin	auteur	application
1	1	FPS	CONVENTIONNEMENT	Conventionnement	1	Organisation 1	0	1	ANTAI	SWAPART
2	2	FREEFLOW	QUESTION_SERVICE	Question service	2	Organisation 2	1	0	userfreeflow@test.com	SWAPART
3	3	SI_FOURRIERE	ANNULATION	Annulation	1	Organisation 1	0	1	swapart@test.com	SWAPART
4	4	PVE	DEMANDE_ACCEES	Demande accès	2	Organisation 2	0	1	userfreeflow@test.com	SWAPART
5	5	FPS	PROBLEME_TECHNIQUE	Problème technique	1	Organisation 1	0	1	swapart@test.com	SWAPART

Figure 28 : PhpMyAdmin résultat

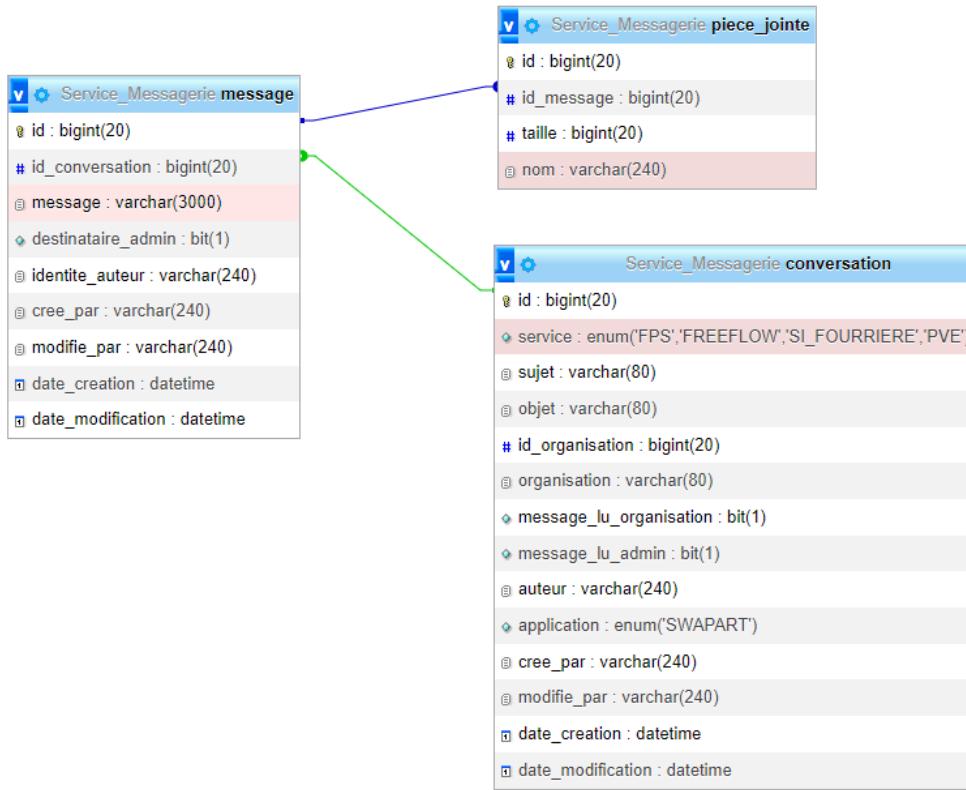


Figure 29 : schéma concepteur phpMyAdmin

9.3. Réalisation des composants d'accès aux données

On retrouve dans le Controller, des objets DTO (data transfer object) générés automatiquement par l'Open API avec l'implémentation Serializable. Il faudra passer par des mapper pour transformer cet objet et ainsi pouvoir le faire circuler entre les différentes couches de l'application. On utilise le Framework Hibernate qui est un ORM

(Object-relational mapping) pour récupérer l'objet Java et le transformer directement dans la base de données.

Le rôle d'un système ORM est de convertir automatiquement, à la demande, la base de données sous forme d'un graphe d'objet. L'ORM s'appuie pour cela sur une configuration associant les classes du modèle fonctionnel et le schéma de la base de données. L'ORM génère des requêtes SQL qui permettent de matérialiser ce graphe ou une partie de ce graphe en fonction des besoins.

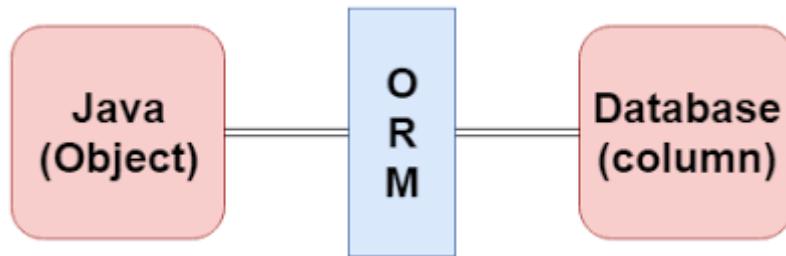


Figure 30 : schéma du fonctionnement de l'ORM

Pour pouvoir accéder aux pièces jointes, j'ai créé une méthode dans une interface dont JpaRepository prend en paramètres l'entity Pièce_Jointe. La méthode correspond à la requête sql que l'on souhaite faire pour accéder à l'information dont nous avons besoin, ici, la méthode va permettre de récupérer une liste de pièces-jointes qui est associés à un message spécifique et va nous permettre de l'afficher dans la conversation ou la récupérer pour la télécharger.

```

● ● ●

package fr.antai.services.pnum.messagerie.infrastructure.adapter.persistence;

import fr.antai.services.pnum.messagerie.infrastructure.model.PieceJointeEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Component;

import java.util.List;
import java.util.Optional;

@Component
public interface IPieceJointeJpaRepository extends JpaRepository<PieceJointeEntity, Long> {

    Optional<PieceJointeEntity> findById(Long piecejointeId);

    /**
     * Récupérer la liste des Pjs associé à un message spécifique
     * @param idMessage l'ID du message lié aux pjs
     * @return la liste des pjs
     */
    List<PieceJointeEntity> findAllByMessage_Id(Long idMessage);

    /**
     * @param idDocument ID de la pièce jointe
     * @param idOrganisation ID de l'organisation liée à la pj
     * @return la liste de pj
     */
    Optional<PieceJointeEntity> findByIdAndMessage_Conversation_IdOrganisation(Long idDocument, Long idOrganisation);
}

```

Figure 31 : IPieceJointeJpaRepository

9.4. Réalisation du back-end

Avant de commencer les développements BACK, il est important de faire une mise à jour sur les bonne pratiques de l'équipe. :

- ▶ Tout d'abord, définir le contrat d'interface (OpenAPI). Le développement FRONT en dépend.
- ▶ Créer une branche nommée PNUM-XXXX

Pour le Controller

- ▶ Restreindre l'accès au(x) rôle(s) requis, exemple :
(@PreAuthorize("hasAnyRole(T(RoleConstantes).ROLE_SWAPART_ACCES)"))
- ▶ Mapper les DTO vers les objets métiers
- ▶ Faire les contrôles sur les paramètres s'ils ne peuvent pas être faits dans l'OpenAPI (SECURITE : le back ne doit JAMAIS faire confiance à ce que le front envoie)
- ▶ Appeler le(s) service(s) existant(s) dans la mesure du possible
- ▶ Après retour des services : Mapper les objets métiers vers les DTO
- ▶ Retourner un objet ResponseEntity<T>

Services

- ▶ Appeler les méthodes repository existantes dans la mesure du possible
- ▶ Logguer les erreurs techniques pour tracer les erreurs en production (log.error(...))
- ▶ Renvoyer les exceptions (techniques / fonctionnelles)
- ▶ Ecrire des tests unitaires

Repositories

- ▶ Mapper les objets métiers vers les Entities
- ▶ Utiliser en priorité les méthodes de JpaRepository : findByXXX(), save(), saveAll()
- ▶ Requérir le minimum de données en BDD, exemple : Boolean existsByNumeroAco(Long numeroAco)
- ▶ Activer les logs SQL pour vérifier que seules les requêtes nécessaires sont exécutées
- ▶ Mapper les Entities vers les objets métiers avant le "return"

Transverses JAVA

- ▶ Utiliser un élément de MessageErreurEnum pour les erreurs remontées
- ▶ Pour des questions de sécurité, il est préférable d'utiliser des listes blanches, exemple :

```
if (statut == "ACTIF") { traitement...}
```

au lieu de :

```
if (statut != "INACTIF") { traitement...}
```

- ▶ Pour tester une String (éviter les NPE) :
("STRING_EXEMPLE").equals(designation.getCivilite().getValeur())
- ▶ Corriger les anomalies Sonar avant les commits

Base de données

- ▶ Typer correctement les champs (char, varchar, datetime, enum...)
- ▶ Si varchar(n), n doit correspondre au besoin, il ne doit pas être trop petit car cela empêchera tout enregistrement avec un nombre de caractères supérieur, mais s'il est trop grand, cela à pour conséquence de réserver plus de mémoire que nécessaire pour chaque entrée dans la table.

9.4.1. Présentation

La fonctionnalité la plus représentative pour le back est l'ajout de pièce-jointe. J'ai d'abord commencé par coder le back avec Spring/Java.

Je vais d'abord vous présenter la partie backend de la méthode :

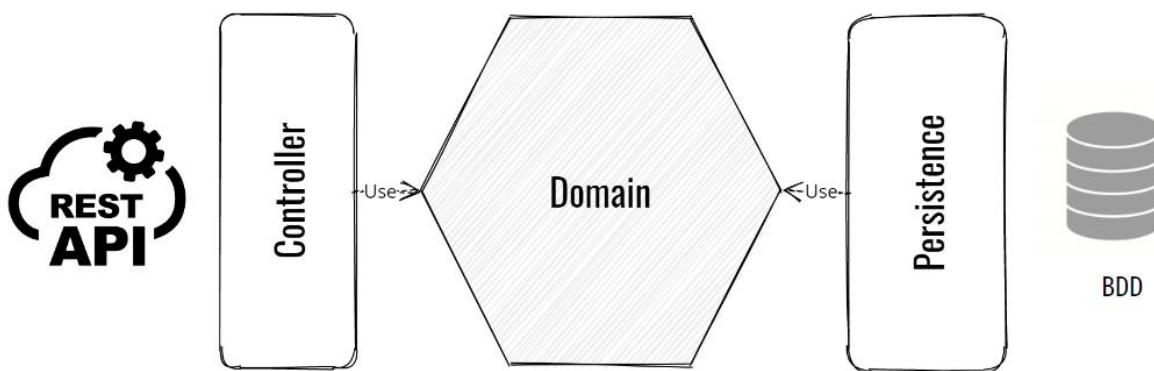


Figure 32 : structure back-end

La structure de notre backend est divisé en 5 couches :

- ▶ OPEN API
- ▶ Application : C'est notre couche controller, on y retrouvera notre CRUD API.
- ▶ Domaine : C'est notre couche métier, aussi appelé service.
- ▶ Infrastructure : C'est notre couche repository qui fera le lien avec la base de données

► Base de données

Pour la partie Back, il a été demandé que les pièces-jointes soient envoyé dans un message au sein d'une conversation :

- Envoie des binaires sur la même route que la création d'un message d'une conversation
- Affichage des PJ dans une bulle :
- Renvoie dans un objet Message, ce dernier comprend les infos des PJ (idPJ, nom/extention du fichier et la taille)

9.4.2. Entités

J'ai, dans un premier temps, créé les entités correspondant à mes tables de la base de données dans le repository. Pour la partie pièce jointe, on y retrouve son id qui est générée automatiquement avec l'annotation de Spring @GeneratedValue(strategy = GenerationType.IDENTITY), le nom de la pièce jointe ainsi que sa taille que l'on mettra ensuite dans la partie front en Méga-Octet (Mo). Nous avons également la relation @ManyToOne qui réfère à la table messageEntity pour associer la pièce-jointe au message. Les annotations @Column de Spring va permettre de directement faire le lien avec la base de données.

```

● ● ●

@Entity
@EqualsAndHashCode(of = {"id", "poidsFichier", "nomFichier"})
@ToString(of = {"id", "poidsFichier", "nomFichier"})
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Getter
@Setter
@Table(name = "piece_jointe")
public class PieceJointeEntity implements Serializable {

    private static final long serialVersionUID = 1727219831635333646L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "nom")
    private String nom;

    @Column(name = "taille")
    private Long taille;

    @ManyToOne
    @JoinColumn(name = "id_message", referencedColumnName = "id")
    @Cascade({org.hibernate.annotations.CascadeType.MERGE})
    private MessageEntity message;
}

```

Figure 33 : PieceJointeEntity

9.4.3. MAPPER

J'ai ensuite créé mon mapper, PieceJointeEntityMapper, qui va me permettre d'utiliser et de transformer mon objet pièce jointe dans les couches supérieures ou inférieur. L'annotation @Mapper marque l'interface en tant qu'interface de mappage et permet au processeur MapStruct d'intervenir lors de la compilation.

```
● ● ●
@Mapper(componentModel = "spring", injectionStrategy = InjectionStrategy.CONSTRUCTOR)
public interface PieceJointeEntityMapper {

    PieceJointeEntity domainToEntity(PieceJointe pieceJointe);

    List<PieceJointeEntity> listDomainToEntity(List<PieceJointe> pieceJointe);

    List<PieceJointe> listEntityToDomain(List<PieceJointeEntity> pieceJointe);

    PieceJointe entityToDomain(PieceJointeEntity entity);
}
```

Figure 34 : PieceJointeEntityMapper

9.4.4. REPOSITORY

Nous passons maintenant à la partie Repository. Dans PieceJointeRepository (Cf annexe 8), j'ai créé la méthode « Créer ». On vérifie dans un premier temps s'il y a un message de présent par son id. J'ai fait une boucle sur la liste de pièce-jointe car il peut y en avoir jusqu'à trois. Pour chaque pièce-jointe, on récupère le nom indiqué ainsi que l'id du message, puis on utilise un buildCheminPj pour sauvegarder le fichier sur disque. Il y a ensuite un Try/Catch qui va permettre de vérifier que les ressources sont bien disponibles par le nom et le chemin du fichier, sinon on envoie un message

```
● ● ●
@Override
public List<Long> creer(List<PieceJointe> pieceJointe, HashMap<String, InputStream> listFichiersPjs, Long
messageId) throws PnumNonAutoriseException, TechnicalException {
    List<PieceJointeEntity> listPieceJointeEntity = pieceJointeEntityMapper.listDomainToEntity(pieceJointe);
    Optional<MessageEntity> messageEntityOptional = messageJpaRepository.findById(messageId);
    if (messageEntityOptional.isPresent()) {
        for (PieceJointeEntity pj : listPieceJointeEntity) {
            InputStream fichier = listFichiersPjs.get(pj.getNom());
            pj.setMessage(messageEntityOptional.get());
            // Sauvegarde du fichier sur disque
            String cheminFichier = buildCheminPJ(pj, messageEntityOptional.get());
            File targetFile = new File(cheminFichier);
            try {
                FileUtils.copyInputStreamToFile(fichier, targetFile);
            } catch (IOException e) {
                String messageErreur = MessageErreureEnum.RESSOURCE_INDISPONIBLE.getMessage();
                PnumLogger.errorDebug(log, MessageErreureEnum.RESSOURCE_INDISPONIBLE.getCodeTechnique()
                    + " : " + messageErreur, e);
                throw new TechnicalException(MessageErreureEnum.RESSOURCE_INDISPONIBLE.getMessage(),
                    MessageErreureEnum.RESSOURCE_INDISPONIBLE.getReferenceMetier());
            }
        }
    }
    List<PieceJointeEntity> piecesJointesEntityList=pieceJointeJpaRepository.saveAll(listPieceJointeEntity);
    return piecesJointesEntityList.stream().map(PieceJointeEntity::getId).collect(Collectors.toList());
}
```

Figure 35 : méthode créer Repository

d'erreur. La variable <>piecesJointesEntityList>> va ensuite enregistrer toutes les pièces jointes dans une liste par le saveAll().

Ci-dessous, la méthode qui va permettre de stocker le fichier sur le disque :

```
● ● ●
/*
 * Permet d'aller stocker la PJ sur le disque
 *
 * @param pJEntity une piece jointe
 * @param message message lié à la piece jointe
 * @return emplacement sur le disque
 */
@NotNull
private String buildCheminPJ(PieceJointeEntity pJEntity, MessageEntity message) {
    StringBuilder cheminConversation = buildCheminConversation(message.getConversation());
    return cheminConversation
        .append(File.separator)
        .append(message.getId())
        .append(File.separator)
        .append(pJEntity.getNom())
        .toString();
}
```

Figure 36 : méthode buildCheminPj repository

9.4.5. SERVICE

Maintenant que la partie persistance a été présenté, je vais passer à la partie Domaine (Service). On y retrouve toute la partie métier et c'est dans cette partie que ce fera les vérifications.

Pour cette partie, je vais vous présenter la méthode CreerMessage qui intègre les pièces-jointes. J'ai créé le constructeur et importée les classes des autres couches avant de démarrer.

Dans un premier temps, je vérifie que l'utilisateur connecté à bien les droits et je récupère ses informations. Je viens setter (mettre à jour) les informations du message, avec les informations de l'utilisateur récupérer ci-dessus. Je peux setter l'identité de l'auteur, récupérer son mail pour le « CreerPar », et la date du jour pour la

```
● ● ●
PnumUserDetails utilisateurConnecte = ((PnumUserDetails) SecurityContextHolder.getContext().getAuthentication().getPrincipal());

message.setDestinataireAdmin(utilisateurConnecte.getOrganisationId() == null);
message.setIdentiteAuteur(utilisateurConnecte.getUsername());
message.setCreerPar(utilisateurConnecte.getEmail());
message.setDateCreation(Timestamp.from(Instant.now()).toLocalDateTime());
message.setDateModification(new Timestamp(System.currentTimeMillis()));
message.setIdConversation(conversationId);
conversationRepository.miseAJourDateModification(conversationId);
conversationRepository.miseAJourMessageLu(conversationId, utilisateurConnecte.getOrganisationId() == null, false);
conversationRepository.miseAJourAuteur(
    conversationId,
    utilisateurConnecte.getOrganisationId() == null ? "ANTAI" : message.getIdentiteAuteur());
Long nouveauMessageId = messageRepository.create(message, conversationId);
```

Figure 37 : méthode CreerMessage service

création. Puis on vient mettre à jour les informations de la conversation qui est lié : Date de modification, Mise à jour Message Lu et Mise à jour Auteur (Si c'est une personne de l'ANTAI (administrateur) ou de l'organisation). La variable nouveauMessageld va permettre de créer l'id du message, en récupérant la variable message et l'id de la conversation qui est relié, en appelant la méthode créer de messageRepository.

La première partie permet de créer un message sans pièce-jointe, la deuxième partie va permettre, s'il y a une ou des pièces-jointes, de vérifier la taille (doit faire moins de 5Mo), de vérifier s'il n'y a pas de virus, et de vérifier si le fichier a bien une des extensions autorisés par des méthodes créer à part (cf : voir la partie recherche).

```

● ● ●

boolean isOK = false;
if (listFichiersPjs != null) {
    //Vérification de la taille de la PJ (doit être inférieur à 5Mo soit 5242880 octets)
    long tailleMax = 5242880;
    for (PieceJointe pieceJointe : message.getPiecesJointes()) {
        if (pieceJointe.getTaille() > tailleMax) {
            throw new SwaPartException(String.format(MessageErreurEnum.UPLOAD_TAILLE_FICHIER.getMessage(), tailleMax),
                MessageErreurEnum.UPLOAD_TAILLE_FICHIER.getReferenceMetier());
        }
    }
    for (Map.Entry<String, InputStream> pj : listFichiersPjs.entrySet()) {
        String nomFichier = pj.getKey();
        isOK = antivirusPort.scanFichier(pj.getValue(), nomFichier);
        //Méthodes externalisées pour une meilleure compréhension
        verifierTypeEtExtension(pj, nomFichier);
        //Vérification Antivirus
        if (!isOK) {
            throw new SwaPartException(String.format(MessageErreurEnum.UPLOAD_FICHIER_VIRUS.getMessage(), nomFichier),
                MessageErreurEnum.UPLOAD_FICHIER_VIRUS.getReferenceMetier());
        }
    }
    //Si antivirus ok : création pieceJointe
    if (isOK) {
        pieceJointeRepository.create(message.getPiecesJointes(), listFichiersPjs, nouveauMessageId);
    }
}

```

Figure 38 : suite méthode CreerMessage service

9.4.6. CONTROLLER

Le controller contient 2 annotations sur la classe :

► L'annotation @RestController :

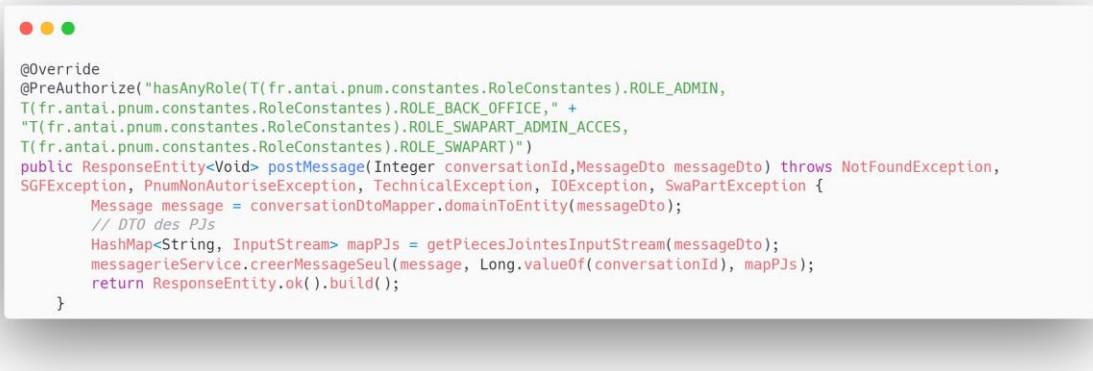
C'est une annotation Spring qui permet d'indiquer qu'il s'agit d'un contrôleur spécialisé pour le développement d'API Web. @RestController est simplement une annotation qui regroupe @Controller et @ResponseBody. Il s'agit donc d'un contrôleur dont les méthodes retournent par défaut les données à renvoyer au client plutôt qu'un identifiant de vue.

► L'annotation @RequestMapping :

Annotation Spring pour le mappage des requêtes Web sur les méthodes des classes de gestion, avec des signatures de méthodes flexibles.

Dans ServiceMessagerieController, on vient implémenter ConversationsApi, ConversationApi, MessagesApi, MessageApi, MessagesnonluApi, PieceJointeApi pour récupérer les méthodes CRUD (Create, Read, Update et Delete).

Pour le controller, nous allons nous intéresser à la méthode postMessage : Je viens récupérer l'objet message puis, j'utilise la méthode que je vais présenter ci-dessus pour récupérer les pièces-jointes. J'appelle ensuite la méthode CreerMessage de la partie service pour venir construire mon message.



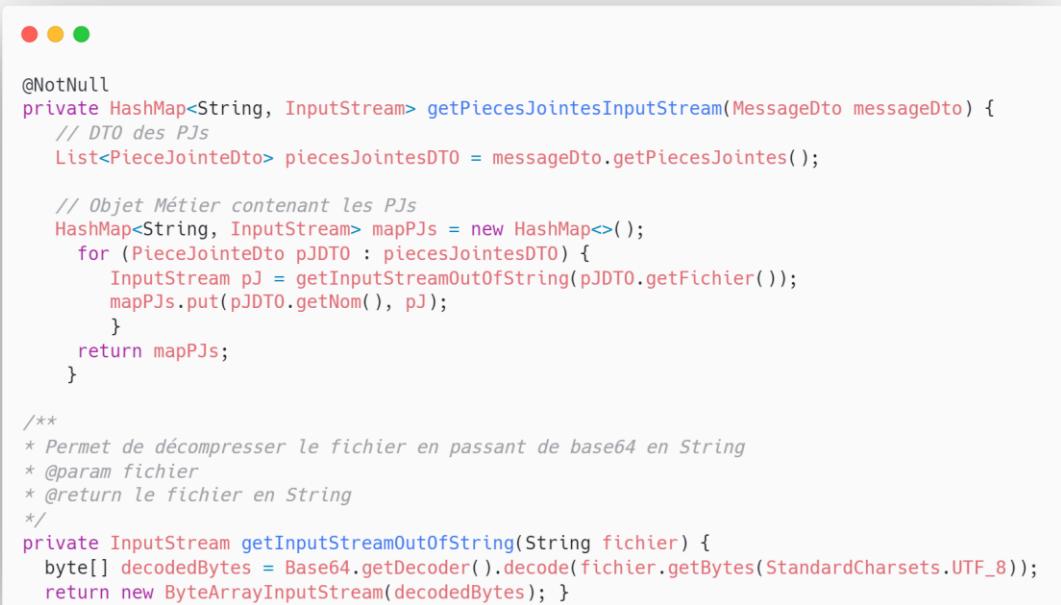
```

@Override
@PreAuthorize("hasAnyRole(T(fr.antai.pnum.constants.RoleConstantes).ROLE_ADMIN,
T(fr.antai.pnum.constants.RoleConstantes).ROLE_BACK_OFFICE, " +
"T(fr.antai.pnum.constants.RoleConstantes).ROLE_SWAPART_ADMIN_ACES,
T(fr.antai.pnum.constants.RoleConstantes).ROLE_SWAPART")")
public ResponseEntity<Void> postMessage(Integer conversationId, MessageDto messageDto) throws NotFoundException,
SGFException, PnumNonAutoriseException, TechnicalException, IOException, SwaPartException {
    Message message = conversationDtoMapper.domainToEntity(messageDto);
    // DTO des PJ
    HashMap<String, InputStream> mapPJs = getPiecesJointesInputStream(messageDto);
    messagerieService.creerMessageSeul(message, Long.valueOf(conversationId), mapPJs);
    return ResponseEntity.ok().build();
}

```

Figure 39 : postMessage controller

Lorsque je crée un message et qu'il y a une ou des pièce(s)-jointe(s) associé, je dois également créer ma pièce-jointe qui est en base64. Pour cela, je viens récupérer ma liste d'objet pièces-jointes relié au message et par une boucle for, décompresser un par un le fichier en passant de base64 en String.



```

@NotNull
private HashMap<String, InputStream> getPiecesJointesInputStream(MessageDto messageDto) {
    // DTO des PJ
    List<PieceJointeDto> piecesJointesDTO = messageDto.get件Jointes();

    // Objet Métier contenant les PJ
    HashMap<String, InputStream> mapPJs = new HashMap<>();
    for (PieceJointeDto pJDTO : piecesJointesDTO) {
        InputStream pJ = getInputStreamFromString(pJDTO.getFichier());
        mapPJs.put(pJDTO.getNom(), pJ);
    }
    return mapPJs;
}

/**
 * Permet de décompresser le fichier en passant de base64 en String
 * @param fichier
 * @return le fichier en String
 */
private InputStream getInputStreamFromString(String fichier) {
    byte[] decodedBytes = Base64.getDecoder().decode(fichier.getBytes(StandardCharsets.UTF_8));
    return new ByteArrayInputStream(decodedBytes);
}

```

Figure 40 : get件JointesInputStream

Nous utilisation avant chaque commit SonarLint, qui est un plugin permettant de réaliser les analyse de code SonarQube au fil des développements, directement dans l'IDE et alerte si une partie du code n'est pas correct. Il faut alors faire un review et faire des modifications si nécessaire avant de pousser notre commit.



Figure 41 : Logo Sonarlint

Une fois le développement fait, avec Maven, nous utilisons la configuration clean install pour mettre à jour notre environnements. Puis avec Sping boot, nous démarrons les services nécessaires pour démarrer le projet.

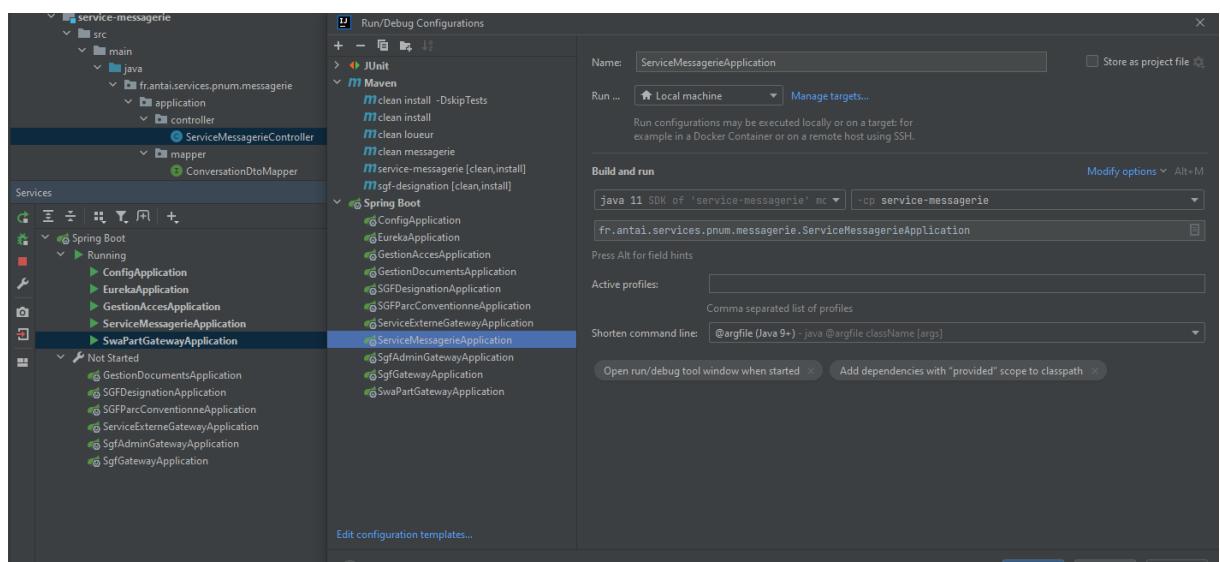


Figure 42 : configuration Maven

9.5. Réalisation du front-end

Pour une bonne réalisation, comme pour le back-end, nous avons des bonnes pratiques avant de commencer nos développements :

Pour le HTML +CSS :

- ▶ Ne jamais utiliser l'attribut : "style=", car en html, cela rend la maintenabilité et l'évolution très compliqué et ne respecte donc pas les règles W3C.
- ▶ Ne jamais utiliser l'attribut «! important» car cela rend difficile la maintenabilité à cause d'éventuel !important surchargeant du style alors que ce n'est pas voulu.
- ▶ Variabiliser les couleurs dans un fichier dédié afin de pouvoir avoir uniquement un seul endroit à modifier si le code couleur du site vient à évoluer.

- ▶ Pour que l'application reste adaptative, il faut utiliser (% , em , vh , vw) pas de pixel en dur (sauf quelques exceptions), afin de pouvoir plus facilement s'adapter aux différent format d'écran utiliser par les usagers.
- ▶ N'abusez pas des class (Bootstrap, Mixin) (1 à 2 directives par attribut "class").
- ▶ En css comme dans tout langage on évite de créer 2 règles qui font quasiment la même chose (on préférera la hiérarchisation).
- ▶ Utiliser le SASS (Sass est un langage de script préprocesseur qui est compilé ou interprété en CSS. SassScript est le langage de script en lui-même.).
- ▶ Eviter de mettre de la complexité trop importante dans le html, on préférera la mettre dans le TS avec une variable mise à jour et inclus dans le html.

TypeScript :

- ▶ Typer toutes vos variables, éviter de concaténer des types maVar : type1 && type2 ;
- ▶ Lorsque vous créez une méthode ou que vous en modifiez une il faut faire la JSdoc :

```

    /**
     * NomDeLaMéthode, description de ce qu'elle fait
     * @param nomDuParametre1 description du paramètre (Optionnel)
     * @return description de ce qui est retourné (Optionnel)
     */

    NomDeLaMéthode(nomDuParametre1 :typeParam) : TypeDeRetour {
        a+c = c;
        return c;
    }

```

- ▶ Exception si vous avez une méthode qui est très simple et qui a du sens rien qu'avec son nom.
- ▶ Pour le nom des méthodes, premier mot en anglais (verbe) reste en français.

Exemple : <verbeTechniqueEnAnglais><metierEnFrançais> (get,search,update)

- ▶ On évite de faire des fonctions avec un trop gros degré de complexité, il vaut mieux scinder en plusieurs méthodes regroupées dans une méthode parente.
- ▶ Pour les composants génériques, attention à ne pas faire de composant trop générique qui ne sont pas maintenable facilement (ex : account-or-convention-state-form).
- ▶ Utiliser les ternaires ils sont puissants (ne pas faire de ternaire de ternaire).

- ▶ N'hésitez pas à créer des fonctions dans des services partagés (ex : fonction de téléchargement de fichier).
- ▶ Utiliser RXJS dès que nécessaire (manipulation d'appel REST) (pipe, switchmap, tap).
- ▶ Typage des tableaux : Privilégier la notation Type [] au lieu de Array<Type>. Par exemple : number [], string [], ...
- ▶ Condition/égalité : Privilégier la triple égalité.
- ▶ Ordonnancement des variables @Input/@Output/autre décorateurs/variables.

9.5.1. Présentation

Pour la partie FRONT du projet, j'ai travaillé en Angular. J'ai pris connaissance de la User Story concernant ma tâche :

- ▶ Mettre le champs texte message en haut de l'écran
- ▶ Message le plus récent en haut
- ▶ Les bulles de messages doivent être positionnées à droite & gauche selon l'auteur
- ▶ Utiliser le bool destinataireAdmin pour positionner ces bulles
- ▶ Bulle de couleur
- ▶ Faire un composant réutilisable pour la partie non-admin

Pour récupérer les méthodes réalisées côté back, j'ai dû générer un contrat d'interface grâce à NodeJS afin de récupérer le swagger Messagerie de l'open Api. Pour cela, j'ai indiqué dans mon terminal la commande :

```
npx ng-openapi-gen --input ../open-api/service_messagerie_swagger.yaml -  
output projects/common/src/lib/api/api-service-messagerie
```

Il m'a fallu ensuite créer mon component avec cette commande :

```
ng generate component PartnerSpaceMessagerie
```

Une fois l'environnement en place, il ne restait plus qu'à coder.

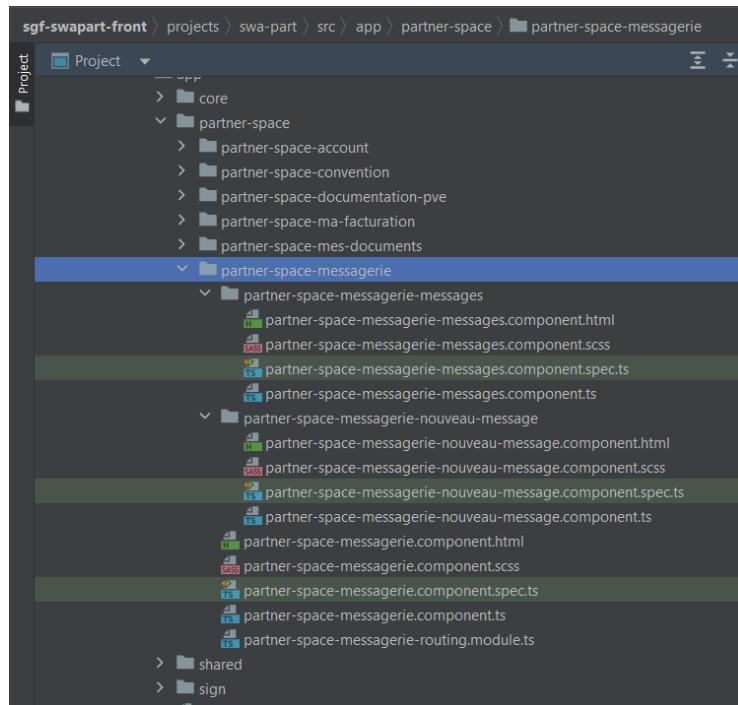


Figure 43 : Structure environnement Front

Pour la partie front, j'ai créé un composant commun qui est utilisé dans le composant administrateur mais également dans le composant espace-partenaire. J'ai créé le composant « List-messages » dans le fichier common (commun) du projet.

Dans Angular, nous avons 3 fichiers, un fichier TS (pour le TypeScript) qui permet de créer les méthodes liées au front et est reliée au fichier HTML, qui lui permet de faire la mise en page. Il y a également le fichier SCSS qui aide à la mise en page.

Le composant est ensuite appelé dans la partie administrateur et partner-space-messagerie, ce qui évite une répétition de code trop important.

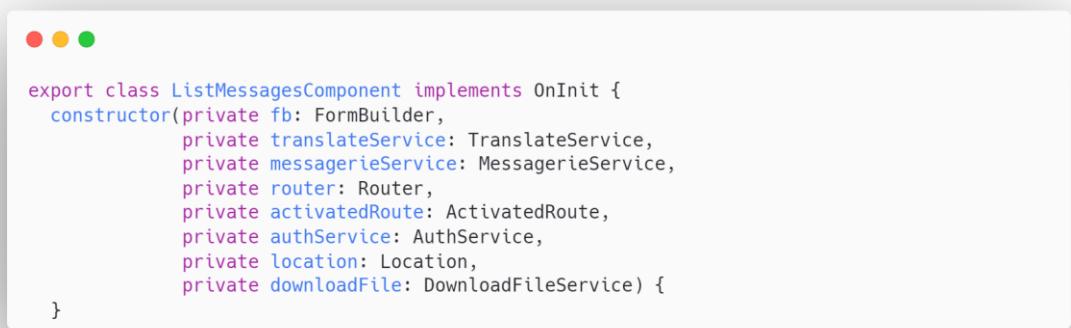


Figure 44 : appel composant

9.5.2. Partie TypeScript

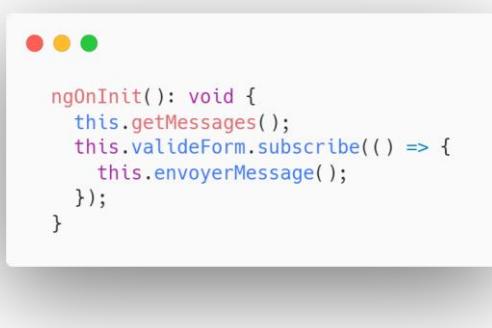
C'est le point d'entrée de l'application. On y retrouve la partie « métier » car c'est ici que l'on va créer les méthodes qui seront utilisés dans le fichier HTML.

Lors de la création du composant, j'ai initialisé le constructeur que j'ai ensuite au fur et à mesure rempli des dépendances nécessaires aux méthodes. On y retrouve un FormBuilder pour la partie envoyer message, MessagerieService (en lien avec notre swagger), DownloadFileService pour la partie téléchargement de notre pièce-jointe.



```
export class ListMessagesComponent implements OnInit {
  constructor(private fb: FormBuilder,
              private translateService: TranslateService,
              private messagerieService: MessagerieService,
              private router: Router,
              private activatedRoute: ActivatedRoute,
              private authService: AuthService,
              private location: Location,
              private downloadFile: DownloadFileService) {
```

Figure 45 : TypeScript composant ListMessage partie 1



```
ngOnInit(): void {
  this.getMessages();
  this.validateForm.subscribe(() => {
    this.envoyerMessage();
  });
}
```

On retrouve ensuite le NgOnInit, qui est spécifique à Angular, qui va permettre d'initialiser la page. Ici, j'ai appelé les méthodes getMessages pour l'affichage des messages et envoyerMessage qui sera appellé lorsque l'événement validateForm sera soumis pour le formulaire côté HTML.

Figure 46 : TypeScript ngOnInit

Pour la méthode envoyerMessage, je crée une constante de tableau vide pour les pièces-jointes, si il n'y a pas de pièces jointes, on passe directement à la partie message. On fait ensuite une boucle qui passe une par une les pièces-jointes. Cela va permettre de récupérer l'objet, son contenu transformé en base64 avec this.downloadFile.getBase64, son nom et sa taille et le rajouter au tableau. Ensuite, avec le controlMessageInput.value, on récupère le contenu du message pour l'associer à l'objet MessageDto, ainsi que les pièces jointes et on appelle la méthode postMessage de notre swagger messagerieService. Une fois l'appel à la méthode postMessage, on vient faire un « subscribe », qui permet de souscrire à un Observable et être notifié des nouvelles valeurs, des erreurs ou de la fin du "stream". Ici on vient récupérer l'id de la conversation et faire une mise à zéro des inputs message et pièce-

jointe, et vient afficher le nouveau message qui vient d'être envoyé dans la conversation.

```
● ● ●

/*
 * Methode pour appeler l'API afin de créer un nouveau message dans la conversation
 */
async envoyerMessage() {
    const nouvellePieceJointe: PieceJointeDto[] = [];
    // Les pieces jointes ne sont pas obligatoire, donc verification si il y a bien une pj
    for (const pj of this.piecesJointes.files) {
        const pjDto = {
            fichier: await this.downloadFile.getBase64(pj) as string,
            nom: pj.name,
            taille: pj.size,
        };
        nouvellePieceJointe.push(pjDto);
    }
    const body: MessageDto = {
        message: this.controlMessageInput.value,
        piecesJointes: nouvellePieceJointe
    };
    this.messagerieService.postMessage({idConversation: this.idConversation,
body}).subscribe(() => {
        this.controlMessageInput.reset();
        this.piecesJointes = new SgfUploadFile();
        this.getMessages();
    });
}
```

Figure 47 : TypeScript composant ListMessage méthode EnvoyerMessage

On y retrouve également la méthode telechargerPieceJointe, elle va permettre de convertir le fichier qui est en base64 en PDF. Lorsque l'on cliquera sur l'onglet de la pièce-jointe, le getPieceJointe est alors appelé et va souscrire un observable qui va convertir puis télécharger grâce au nom du fichier le bon pdf.

```
● ● ●

telechargerPieceJointe(idPieceJointe: number, nomFichier: string) {
    this.messagerieService.getPieceJointe({idPieceJointe: idPieceJointe}).subscribe((pj) => {
        const a = document.createElement('a');
        a.href = window.URL.createObjectURL(pj);
        a.download = nomFichier;
        a.click();
    });
}
```

Figure 48 : TypeScript composant ListMessage méthode telechargerPieceJointe

9.5.3. Partie HTML

Dans le composant listMessage.HTML, je présente d'abord la partie où l'on va écrire un nouvel item message et l'insertion de pièce-jointe. J'utilise un composant Angular <mat-form-field> pour déterminer l'affichage du formulaire. Il y a ensuite un <textarea> qui va permettre à l'utilisateur de venir insérer le texte, et dans ce label, je lui ai mis un [FormControl] qui est instancié dans la partie TypeScript. Ce FormControl va servir à vérifier qu'il y a bien du texte car c'est une valeur obligatoire (required). Le message ne doit pas dépasser un certain nombre de caractères (Pas plus de 3000), j'ai donc rajouté un maxlength qui empêchera l'utilisateur d'écrire plus que ce qui est possible. A la suite, j'ai mis un mat-error qui va venir alerter si le message est vide. On passe ensuite au bouton de validation. Dans un composant <antai-footer-with-btn>, je centre le bouton avec centerBtn, je lui associe la couleur bleu, je lui rajoute un label et un clic évènement, qui permet de valider le formulaire, et il y a une particularité disabled, qui vient griser et empêcher l'utilisateur de cliquer sur le bouton si le formulaire du message est invalide (pas de caractères dans le textarea). Il y a ensuite le bloc pour l'insertion des pièces-jointes qui a été réalisé par un de mes collègues. Il a créé un composant commun qui va permettre à d'autre composant de l'utiliser avec les mêmes caractéristiques.

```

● ● ●

<!-- Bloc pour l'insertion du message-->
<p class="field-hint">* {{ 'field.mandatory' | translate }}</p>
<div class="with-input_form flex">
  <p class="fg-1">
    <mat-form-field appearance="fill">
      <mat-label>{{'field.message' | translate}}</mat-label>
      <textarea matInput
        #message
        [formControl]="controlMessageInput"
        [maxlength]="'MESSAGE_MAX_LENGTH'"
        required>
      </textarea>
      <mat-hint align="end">{{message.value?.length }}/{{MESSAGE_MAX_LENGTH}}</mat-hint>
      <mat-error *ngIf="controlMessageInput.hasError('required')">
        {{ 'field.error.required' | translate }}
      </mat-error>
    </mat-form-field>
  </p>
<!-- Bloc pour le bouton de validation-->
<div class="footer-btn">
  <antai-footer-with-btn class="flex"
    [value]="{{
      centerBtn: {
        color:'blue',
        label: 'messagerie.send',
        clickEvent: {event: valideForm},
        disabled: controlMessageInput.invalid
      }
    }}"
  >
    </antai-footer-with-btn>
  </div>
</div>
<!-- Bloc pour les pièces-jointes-->
<div class="bg-blue-light p-5">
  <antai-upload-pj-messagerie [idTechnique]="'messagerie-piecesJointes-listMessages'">
    [value]="'piecesJointes'"</antai-upload-pj-messagerie>
  </div>

```

Figure 49 : HTML composant ListMessage partie 1

La deuxième partie de cette page HTML est l'affichage des messages et des pièces jointes. On a une div class messages qui va permettre de faire un bloc, on va boucler dessus pour chaque message, et vérifier si le message vient de l'expéditeur. Si c'est true, on vient afficher le message à gauche, sinon si c'est false, on vient le mettre à gauche. Nous avons deux <>span> va venir instancier l'identité de l'auteur du message et la date de création, qui seront affiché dans l'en-tête du bloc message. Puis je récupère le contenu avec {{message.message}} pour venir l'afficher. On passe ensuite à la partie pièce-jointe. Il n'est pas obligatoire d'avoir des pièces-jointes à un message. On va venir faire une boucle for pour les afficher que s'il y en a. On est sur un mat-button-toggle, ça permet d'afficher le contenu sous forme de boutton. On y associe un évènement à un clic, avec la méthode vu dans le partie TypeScript : telechargerPieceJointe avec en paramètre l'id et le nom du fichier. Pour chaque pièce-jointe, il y a un icône trombone comme demandé par le client, puis le nom et la taille du fichier convertie de octets en Mégaoctets, avec deux chiffres après le virgule grâce autoFixed(2).



```

<div class="messages">
  <div *ngFor="let message of conversation.messages" class="message" tabindex="0"
    [class]="message.isExpediteur ? 'message-droite' : 'message-gauche'">
    <div class="header-message">
      <span class="auteur">
        {{message.identiteAuteur}}
      </span>
      <span class="date">
        {{message.dateCreation | date:'dd/MM/yyyy HH:mm'}}
      </span>
    </div>
    <p>
      {{message.message}}
    </p>

    <div class="pieceJointe" *ngFor="let pj of message.piecesJointes">
      <mat-button-toggle class="boutonPj" (click)="telechargerPieceJointe(pj.id, pj.nom)">
        <mat-icon>attach_file</mat-icon>
        <span class="pieceJointeNom">{{pj.nom}}</span>
        <span class="pieceJointeTaille">{{(pj.taille/CONVERSION_EN_M0).toFixed(2)}} Mo</span>
      </mat-button-toggle>
    </div>
  </div>
</div>
</div>
</antai-card>

```

Figure 50 : HTML composant ListMessage partie 2

9.5.4. Partie CSS

Pour chaque génération de composant, un fichier CSS est également créée. Ça permet de modifier la mise en page en ajoutant des contours, des marges ou des couleurs par exemple. Pour le composant ListMessage, j'ai modifié la couleur des blocs des messages en fonction de l'utilisateur : pour le message à droite, j'ai ajouté un fond bleu clair, pour le message à gauche, j'ai mis un fond blanc. Pour le bloc Message où l'on retrouve les

valeurs, j'ai joué sur les marges (margin-bottom, padding) et ajouté un box shadow qui permet de légèrement noircir le tour du bloc. Les messages sont affichés en colonne dans la classe messages. Et j'ai également modifier le header (en-tête) du message pour que le texte auteur soit en gras et la date de couleur plus clair. J'ai utilisé un justify-content : space-between pour afficher les deux paramètres de chaque côtés du bloc.

```

● ● ●

@import "../../../../../style_variables";

.message-droite {
    background: map-get($color-blue, 'light');
    align-self: flex-end;
}

.message-gauche {
    background: white;
}

.message {
    width: 50em;
    margin-bottom: 3.2em;
    box-shadow: 0.2em 0.25em 0.37em 0.06em rgb(0 0 0 / 20%);
    padding: 0.62em;
}

.messages {
    display: flex;
    flex-direction: column;
    height: 44em;
    overflow: auto;
    padding: 2em;
}

.header-message {
    display: flex;
    justify-content: space-between;

    .auteur {
        font-weight: bold;
    }

    .date {
        color: map-get($color-black, 'default');
    }
}

```

Figure 51 : CSS composant ListMessage

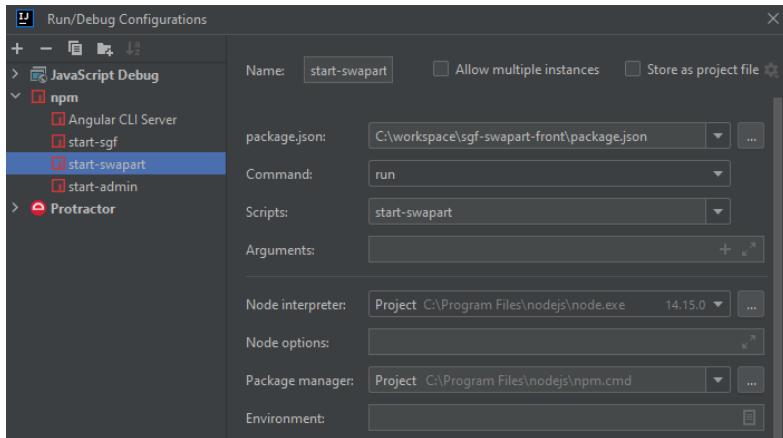


Figure 52 : configuration npm Front

Une fois les développements réalisés, nous démarrons avec npm, notre configuration start-swapart qui va permettre de démarrer notre IHM, celui-ci fonctionne que si les services côté back sont démarrés.

9.5.5. Réalisation de test unitaire front-end

Après avoir réalisé mes méthodes et mon visuel pour la partie front, j'ai créé des test unitaires pour tester mes fonctionnalités. J'ai dans un premier temps fait un Mock (bouchon) qui va permettre de créer des objets sans passer par la base de données. Ici je crée une conversation, avec des messages et des pièces-jointes. J'importe également les modules dont je vais avoir besoin pour la réalisation.

```

describe('ListMessagesComponent', () => {
  let component: ListMessagesComponent;
  let fixture: ComponentFixture<ListMessagesComponent>;
  let mockMessagerieService: MessagerieService;

  const CONVERSION_EN_MO = 1048576;
  const mockConversation: ConversationDto = {
    messages: [
      {
        messageLu: true,
        dateCreation: '01/01/1970 01:16',
        identiteAuteur: 'Jean',
        isExpediteur: true,
        message: 'Ceci est un test',
        piecesJointes: [
          {
            nom : 'fichierTest.pdf',
            taille : 144813,
          }
        ],
        {
          messageLu: true,
          dateCreation: '01/01/1970 01:16',
          identiteAuteur: 'Toto',
          isExpediteur: false,
          message: 'Reponse test 2',
          piecesJointes: [
            {
              nom : 'fichierTest2.pdf',
              taille : 125025
            }
          ]
        },
        objet: 'string',
        service: TypeService.Fps,
        sujet: Sujet.Annulation
      };
    ];
  };
}
);

```

Figure 53 : Test unitaires partie 1

Dans ces tests unitaires, j'ai testé les méthodes getMessages, boutonEnvoyer, envoyerMessage et téléchargerPieceJointe. Pour le getMessages, on vérifie que l'élément affiché correspond bien à l'élément que l'on a créé dans le mock. Pour le bouton envoyé, on vérifie que c'est bien le même message affiché dans le bouton. Dans la méthode envoyerMessage, on vérifie que la méthode PostMessage est bien appelé. Et dans la méthode telechargerPieceJointe, on vérifie que la méthode getPieceJointe avec l'id et le nom de la pièce jointe du mock est bien appelé.

```

● ● ●

it('should create', () => {
  expect(component).toBeTruthy();
});

it('should pass accessibility test', async () => {
  const test = await axe(fixture.nativeElement);
  expect(test).toHaveNoViolations();
});

it('getMessages method', () => {
  const listMessages = fixture.debugElement.query(By.css('div.messages')).nativeElement;
  expect(listMessages).toBeDefined();
  const listMessagesMessage = fixture.debugElement.query(By.css('div.messages > div:nth-child(1) > p')).nativeElement.innerText;
  expect(listMessagesMessage).toEqual(mockConversation.messages[0].message);
  const listMessagesAuteur = fixture.debugElement.query(By.css('div.messages > div:nth-child(1) > div.header-message > span.auteur')).nativeElement.innerText;
  expect(listMessagesAuteur).toEqual(mockConversation.messages[0].identiteAuteur);
  const listMessagesDate = fixture.debugElement.query(By.css('div.messages > div:nth-child(1) > div.header-message > span.date')).nativeElement.innerText;
  expect(listMessagesDate).toEqual(mockConversation.messages[0].dateCreation);
  const listMessagesPieceJointeNom = fixture.debugElement.query(By.css('div.messages > div:nth-child(1) > div.pieceJointe span.pieceJointeNom')).nativeElement.innerText;
  expect(listMessagesPieceJointeNom).toEqual(mockConversation.messages[0].piecesJointes[0].nom);
  const listMessagesPieceJointeTaille = fixture.debugElement.query(By.css('div.messages > div:nth-child(1) > div.pieceJointe span.pieceJointeTaille')).nativeElement.innerText;
  expect(listMessagesPieceJointeTaille).toEqual(((mockConversation.messages[0].piecesJointes[0].taille) /
CONVERSION_EN_M0).toFixed(2));
});

it('boutonEnvoyer method', () => {
  const boutonLabel = fixture.debugElement.query(By.css('button.antai-submit-btn')).nativeElement.innerText;
  expect(boutonLabel).toEqual('messagerie.send');
});

it('envoyerMessage method', () => {
  const spy = spyOn(mockMessagerieService, 'postMessage').and.returnValue(of());
  component.envoyerMessage();
  expect(spy).toHaveBeenCalled();
});

it('telechargerPieceJointe method', () => {
  const spy = spyOn(mockMessagerieService, 'getPieceJointe').and.returnValue(of());
  component.telechargerPieceJointe(mockConversation.messages[0].piecesJointes[0].id,
  mockConversation.messages[0].piecesJointes[0].nom);
  expect(spy).toHaveBeenCalled();
});
}
);

```

Figure 54 : Test unitaires partie 2

10. PRESENTATION DU JEU D'ESSAIE AVEC FONCTION LA PLUS REPRESENTATIVE

10.1. Présentation

La fonction la plus représentative est la partie envoyer message (avec pièce-jointe) dont j'ai présenté la partie back-end et la partie front-end ci-dessus. Cette fonctionnalité est utilisée sur les deux applications, Administrateur et SwaPart. Pour pouvoir envoyer une ou des pièces-jointes, il doit impérativement y avoir un message de créé avec. Une fois l'utilisateur connecté, il a accès à sa messagerie par l'onglet « ma messagerie » ou l'enveloppe dans le header.

Les données en entrée : L'utilisateur peut ajouter du texte dans le bloc jusqu'à 3000 caractères, et ajouter jusqu'à 3 pièces-jointes aux formats possibles et dont la taille ne dépasse pas 5Mo par fichier.

The screenshot shows the 'MA MESSAGERIE' (My Messaging) section of the Swa-Part application. At the top, there are navigation links: MON COMPTE, FOURRIÈRE, MES DOCUMENTS, MA MESSAGERIE (which is underlined), and MA FACTURATION. Below these, a link 'Retour à la liste des messages' (Back to message list) is visible. The main area is titled 'MA MESSAGERIE'. It displays a new message form with fields for 'Service : FPS', 'Sujet : Problème technique', and 'Objet : Démonstration CDA'. There is a note '* Champs requis'. A text input field 'Message *' contains placeholder text 'Message *'. To the right is a button 'Envoyer message' and a character counter '0/3000'. Below this is a 'CHARGER UN FICHIER' (Upload file) section with instructions about supported formats (PDF, JPEG/JPG/PNG/TIF/TIFF (Images), DOC/DOCX (Word), CSV/XLS/XLSX (Excel)) and size limits (5 Mo). It includes a file upload area with a '+ Glissez et déposez vos fichiers ici' placeholder and an 'AJOUTER' (Add) button. An incoming message from 'swapart@test.com' dated '18/05/2023 16:14' is shown, containing a file attachment 'TEST.docx (0.01 Mo)'.

Figure 55 : IHM Swa-Part messagerie

10.2. Utilisation

Les données attendues et obtenues : si l'utilisateur ne met pas de message, le bouton reste gris et on ne peut pas cliquer dessus, même si des pièces-jointes sont ajouter. Si une ou plusieurs pièces-jointes ne sont pas aux bons formats, la pièce-jointe n'est pas ajouté. Des alertes d'erreurs apparaissent en rouge. (**Ce champs est requis / Votre document doit être au format PDF, JPEG/JPG/PNG/TIF/TIFF (Images), DOC/DOCX (Word), CSV/XLS/XLSX (Excel)**)

MON COMPTE FOURRIÈRE MES DOCUMENTS MA MESSAGERIE MA FACTURATION

[← Retour à la liste des messages](#)

MA MESSAGERIE

Service : FPS

* Champs requis

Message *

Ce champ est requis

CHARGER UN FICHIER

Formats possibles: PDF, JPEG/JPG/PNG/TIF/TIFF (Images), DOC/DOCX (Word), CSV/XLS/XLSX (Excel). Taille maximale 5 Mo par fichier. Vous pouvez ajouter 3 pièces jointes maximum.

Votre document doit être au format PDF, JPEG/JPG/PNG/TIF/TIFF (Images), DOC/DOCX (Word), CSV/XLS/XLSX (Excel).

Sujet : Problème technique

Objet : Démonstration CDA

Figure 56 : IHM Swa-Part messagerie avec erreur

Mais si l'utilisateur ajoute un message, le bouton devient bleu et cliquable et les messages d'erreurs disparaissent.

MON COMPTE FOURRIÈRE MES DOCUMENTS **MA MESSAGERIE** MA FACTURATION

[← Retour à la liste des messages](#)

MA MESSAGERIE

Service : FPS

* Champs requis

Message *

Bonjour, ci-joint un document complémentaire

Sujet : Problème technique

Objet : Démonstration CDA

44/3000

CHARGER UN FICHIER

Formats possibles: PDF, JPEG/JPG/PNG/TIF/TIFF (Images), DOC/DOCX (Word), CSV/XLS/XLSX (Excel). Taille maximale 5 Mo par fichier. Vous pouvez ajouter 3 pièces jointes maximum.

Fiche suivie des compétences mises en oeuvre en entreprise Titre CDA 12_2022 (3).pdf (0.16 Mo)

swapart@test.com 18/05/2023 16:14

Bonjour, veuillez trouver ci-joint le document, cordialement

Figure 57 : IHM Swa-Part messagerie avec message et pièce jointe

Une fois le message envoyé, il apparaît en premier dans la liste des messages, et à gauche. Il contient également la pièce-jointe associé. Le formulaire et la pièce-jointe sont réinitialisé lors de l'envoie sans avoir besoin de rafraîchir la page.

The screenshot shows the "MA MESSAGERIE" section of the application. At the top, there are fields for "Service : FPS", "Sujet : Problème technique", and "Objet : Démonstration CDA". Below these, a message input field is labeled "Message *". To the right of the message field is a button "Envoyer message". A note "0/3000" indicates the character limit. Below the message input is a "CHARGER UN FICHIER" section with a note about file formats and size limits. Two messages are listed in the conversation:

- swapart@test.com (18/05/2023 16:39): Bonjour, ci-joint un document complémentaire
Fiche suivie des compétences mises en oeuvre en entreprise Titre CDA 12_2022 (3).pdf (0.16 Mo)
- swapart@test.com (18/05/2023 16:14): Bonjour, veuillez trouver ci-joint le document, cordialement
TEST.docx (0.01 Mo)

Figure 58 : IHM Swa-Part messagerie

Côté administrateur, les messages venant de Swa-Part s'affiche en bleu à droite. Il va pouvoir ensuite cliquer sur le bouton de la pièce-jointe qui va venir directement se télécharger. Il peut également répondre dans la conversation tout comme les autres utilisateurs.

The screenshot shows the "MESSAGERIE" section of the application. At the top, there are fields for "Organisation : LA VILLE DE PARIS", "Service : FPS", "Sujet : Problème technique", and "Objet : Démonstration CDA". Below these, a message input field is labeled "Message *". To the right of the message field is a button "Envoyer message". A note "0/3000" indicates the character limit. Below the message input is a "CHARGER UN FICHIER" section with a note about file formats and size limits. Two messages are listed in the conversation:

- ANTAI (18/05/2023 16:51): Bonjour, merci, je prend bonne réception.
- swapart@test.com (18/05/2023 16:39): Bonjour, ci-joint un document complémentaire
Fiche suivie des compétences mises en oeuvre en entreprise Titre CDA 12_2022 (3).pdf (0.16 Mo)
- swapart@test.com (18/05/2023 16:14): Bonjour, veuillez trouver ci-joint le document, cordialement

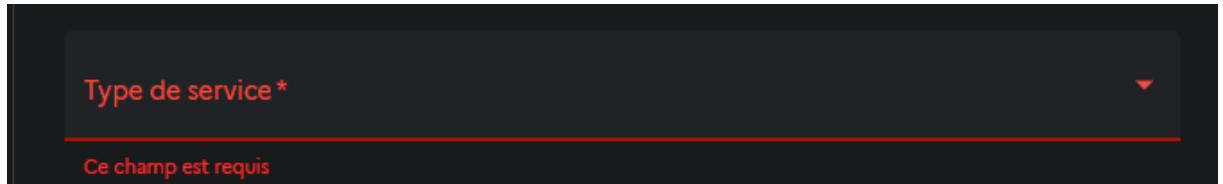
At the bottom right of the message list, there are buttons "Tout afficher" and "X".

Figure 59 : IHM ADMIN messagerie

Il est important de tester facilement si les contrôles des champs des formulaires sont réalisés côté front et côté back.

► Front

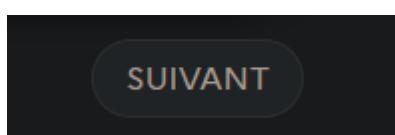
Si un champ obligatoire n'est pas renseigné, il doit y avoir une erreur sur le champ :



Type de service *

Ce champ est requis

Le Bouton de validation du formulaire doit être désactivé :



► Back

Faire un clic gauche sur le bouton de soumission du formulaire, puis "**Inspecter**", l'élément HTML correspondant au bouton s'affiche

```
▼ <button _ngcontent-lny-c163 mat-flat-button class="mat-focus-indicator mat-flat-button mat-button-base antai-submit-btn btn-blue mat-button-disabled" ng-reflect-ng-class="btn-blue" ng-reflect-disabled="true" disabled="true">
  ▼<span class="mat-button-wrapper">
    <!--bindings={}-->
    <span _ngcontent-lny-c163>SUIVANT</span> == $0
    </span>
    <span matripple class="mat-ripple mat-button-ripple" ng-reflect-disabled="true" ng-reflect-centered="false" ng-reflect-trigger="[object HTMLElement]"></span>
    <span class="mat-button-focus-overlay"></span>
  </button>
```

En supprimant l'attribut 'disabled="true"'

Le bouton est désormais cliquable

Si le formulaire n'est pas correctement renseigné et qu'il n'y a pas d'erreur en cliquant sur le bouton, alors il y a un problème de sécurité que l'on doit corriger.

10.3. Conclusion

Cette méthode m'a permis de réaliser le côté backend et le côté frontend de A à Z, et ainsi de comprendre le fonctionnement du Swagger. C'est la fonctionnalité qui a été la plus complexe pour moi car j'ai dû transformer mes pièces-jointes en base64, ce qui n'avait jamais été réalisé sur l'une des applications auparavant et donc m'a valu beaucoup de recherche et de test.

Cette fonctionnalité est opérationnelle, elle a été présentée pendant une démonstration aux clients, validée puis mise en production.

11. SECURITE

11.1. Présentation

Une faille de type injection se produit quand une donnée non fiable est envoyée à un interpréteur en tant qu'élément d'une commande ou d'une requête. Elle peut avoir un impact technique sévère, comme un vol de données, une perte d'intégrité des données ou bien un déni de service. L'ORM mis en place sur le projet permet d'empêcher les types d'injections SQL : si on lui a préciser que l'on veut un type String, un utilisateur ne pourra mettre un type Int.

Il existe les rôles suivants pour les utilisateurs des applications :

Pour SGF :

- ▶ GF_ADMIN : les gestionnaires de flottes administrateurs de la société
- ▶ GF : pour les gestionnaires de flottes "standards"

Pour SWAPART :

- ▶ SWAPART : les utilisateurs de l'application SWAPART ("ESPACE PARTENAIRES")

Pour admin :

- ▶ ADMIN : administrateurs de l'ANTAI
- ▶ BACK_OFFICE : utilisateurs de SGF admin ne pouvant pas créer d'utilisateur

11.2. Recherche effectuée

Chez Open, la sécurité est un point important. Tous les ans, chaque collaborateur doit repasser une formation SMSI (Système de Management de la Sécurité de l'Information) et une formation de l'OWASP afin de rappeler les mesures de sécurité, leur mise en place, gérer les risques et améliorer la sécurité en continu. Pendant mon alternance, je n'ai pas eu besoin de faire de recherche sur la sécurité car j'ai pu réaliser ces deux formations sécurité qui m'ont données toutes les réponses pour le besoin de sécurité. Sur le projet TMA-PNUM, c'est le LDAP qui gère la partie sécurité et authentification. Le LDAP est un protocole permettant l'interrogation et la modification des services d'annuaires. La TME (Tierce Maintenance Exploitation) gère l'OpenLDAP qui est une logiciel implémentant le LDAP. Lors d'une inscription, l'utilisateur renseigne un identifiant et son mot de passe, qui sera envoyé au LDAP (annuaire géré par la TME). Il sera ensuite mis dans un groupe en fonction de l'utilisateur (Si c'est un utilisateur classic, une collectivité, un loueur, un administrateur, ...).

Quand celui-ci se connecte, il y a un appel à la LDAP pour vérifier son id et son mot de passe ainsi qu'une vérification que son groupe lui permet d'accéder au site (Exemple : Error 403, Forbidden ou Error 401 Unauthorized). Si dans la base de données son rôle est ok : Création d'un token bearer (Chiffre clé en base64) qui est enregistrer en base de données pour une durée de 15 minutes. Lors d'une requête, il y a une vérification du token et du rôle.

11.3. Mise en œuvre

Un mapping des rôles SWAPART vers les groupes LDAP est présent dans les fichiers de configuration des services sgf-gateway, sgf-gateway-admin et service-gestion-acces. Seuls les rôles ayant accès à chaque application sont mappés dans la configuration des gateways, ce qui permet de renvoyer une erreur lors de l'authentification d'un utilisateur présent dans le ldap mais n'ayant pas accès à une application. Le LDAP récupère l'utilisateur ainsi que son mot de passe (et le hashage).

Les rôles LDAP permettent l'accès à chaque application. En complément, chaque utilisateur possède un rôle positionné en base de données (table utilisateur) permettant une gestion des permissions par fonctionnalité.

Pour la mise en place de la messagerie, seuls les rôles ROLE_ADMIN, ROLE_BACK_OFFICE, ROLE_SWAPART_ADMIN_ACCES (qui sont les rôles pour la partie administrateur), et le rôle ROLE_SWAPART, sont autorisés.

Ce sont les rôles que nous retrouvons dans les fichiers des « Rôles Constantes » et que j'ai ensuite mis en place dans le controller.

```
● ● ●

/**
 * @param idPieceJointe Récupération de la pièce jointe liée à un message (required)
 * @return la pièce jointe qui est sélectionnée par son ID
 * Contrôle des droits d'accès par le rôle de l'utilisateur
 * @throws NotFoundException
 * @throws IOException
 */
@Override
public ResponseEntity<org.springframework.core.io.Resource> getPieceJointe(
    Integer idPieceJointe) throws NotFoundException, IOException {
    final Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    boolean isAdmin = authentication.getAuthorities().contains(new
SimpleGrantedAuthority(RoleEnum.ROLE_SGF_ADMIN_ACCES.getRole()));

    final DocumentFichier documentFichier;
    if (isAdmin) {
        documentFichier = documentService.recupererDocument(Long.valueOf(idPieceJointe));
    } else {
        final PnumUserDetails utilisateurConnecte = (PnumUserDetails) authentication.getPrincipal();
        documentFichier = documentService.recupererDocument(utilisateurConnecte.getOrganisationId(),
Long.valueOf(idPieceJointe));
    }

    return ControllerUtil.getResourceResponseEntity(documentFichier, log);
}
```

Figure 60 : méthode getPieceJointe avec vérification du rôle autorisé

Pour la méthode ci-dessus, on récupère l'authentification de l'utilisateur qui va nous permettre de savoir s'il a un rôle autorisé. On définit ensuite dans un boolean si c'est un administrateur ou un rôle swapart, qui va permettre d'autoriser la récupération

des pièces-jointes, si c'est un administrateur, il pourra récupérer toutes les pièces-jointes, et si c'est un utilisateur Swapart, il pourra récupérer (comme pour les conversations et messages) ceux de son organisation avec utilisateurConnecte.getOrganisationId. C'est une autre sécurité pour empêcher les autres entreprises d'avoir accès à des informations qui ne leur appartiennent pas.

11.4. Conclusion

Cette mise en place de sécurité permettra aux utilisateurs ayant le rôle SWAPART, d'avoir accès aux conversations de leur organisation et bloquera l'accès aux autres utilisateurs.

Il est primordial d'avoir une sécurité infaillible. Cela permet d'éviter les failles de sécurité telles que les injections SQL mais également les violations de gestion d'authentification et de session, les failles XSS (Cross-site-scripting), éviter les références directes non sécurisées à un objet, une exposition des données sensibles et bien d'autres encore...

12.RECHERCHE EFFECTUEE

12.1. Le besoin d'information

Lors de cet apprentissage, j'ai eu besoin de m'auto-former sur Spring pour le backend et Angular pour le frontend. Ces recherches ont permis de comprendre plus facilement le projet malgré quelques points de blocages nécessitant la recherche d'informations. Ces recherches se faisaient en autonomie via une recherche sur internet, ou par du support d'un membre de l'équipe, après quelques heures de recherche infructueuse.

12.2. La recherche

Lors d'un envoi de pièce-jointe, il a été demander que les extensions des fichiers soient vérifier car toutes les extensions ne sont pas autorisées. Il est également possible de modifier une extension de fichier quand on l'enregistre. J'ai donc fait une recherche sur mon navigateur web en anglais pour avoir plus de résultat : java verify file type. Je suis donc tombé sur cette page qui explique les différentes façons de vérifier le type ainsi que son avis sur leur pertinence. En dernier paragraphe, il présente Apache Tika, que c'est une bonne librairie et détecte bien les types. Il permet de scanner le fichier sans prendre en compte l'extension dans le nom du fichier. Après en avoir discuté avec mon leader technique, vérifier la librairie, que celle-ci soit toujours en date et maintenu, il a été possible de mettre en place Apache Tika dans le projet.

DZone REFCARDS TREND REPORTS EVENTS Over 2 million developers have joined DZone. Log In / Join  

Culture and Methodologies | Data Engineering | Software Design and Architecture | Coding | Testing, Deployment, and Maintenance | Partner Zones

If we call this method for the file that we changed the extension of to PPTX, we get the following result as file type:

Output: content/unknown

4. Apache Tika

Previous three approaches are provided by the JDK. However, there are others like Apache Tika. Apache Tika is a very successful library and is good at detecting file type via analyzing file content independently of its extension.

Our method gets InputStream as parameter and uses detect method of Apache Tika:

```
Java
1 public static String getFileTypeByTika(InputStream istream) {
2
3     final Tika tika = new Tika();
4     String fileType = "";
5     try {
6         fileType = tika.detect(istream);
7     } catch (IOException e) {
8         System.out.println("**** getFileTypeByTika - Error while detecting mime type from InputStream ***");
9         System.out.println("**** getFileTypeByTika - Error message: " + e.getMessage());
10        e.printStackTrace();
11    }
12    return fileType;
13}
```

If we convert the file that was originally in JPEG format to FileInputStream, but we change the extension of it to PPTX and give it as a parameter to `getFileTypeByTika`, we get the following result:

Output: image/jpeg

Tika detected the type of file correctly.

We can use the `detect` method of Apache Tika with parameter has type of File, instead of InputStream. We will use following method to use the detect method of Tika with File parameter:

Figure 61 : page web DZone pour recherche Tika

12.3. La mise en œuvre

Après vérification sur le site tika.apache.org que les versions sont bien à jour et maintenu régulièrement :

Apache Tika - a content analysis toolkit

The Apache Tika™ toolkit detects and extracts metadata and text from over a thousand different file types (such as PPT, XLS, and PDF). All of these file types can be parsed through a single interface, making Tika useful for search engine indexing, content analysis, translation, and much more. You can find the latest release on the [download page](#). Please see the [Getting Started](#) page for more information on how to start using Tika.

The [Parser](#) and [Detector](#) pages describe the main interfaces of Tika and how they work.

For more in-depth documentation, see our [wiki](#), especially for [tika-server](#).

If you're interested in contributing to Tika, please see the [Contributing](#) page or send an email to the [Tika development list](#).

Tika is a project of the [Apache Software Foundation](#), and was formerly a subproject of [Apache Lucene](#).

Latest News

15 May 2023: Apache Tika Release
 Apache Tika 2.8.0 has been released! This release includes optional handling of incremental updates in PDFs, a bug fix that had prevented the running of exiftool and ffmpeg by default, and the move of the GeoTopic parser back to its 1.x namespace: o.a.t.parser.geo.topic. There are several other improvements, bug fixes and dependency upgrades. Note that we are no longer shading the tika-parsers-standard module. Please see the [CHANGES.txt](#) for the full list of changes in the release and have a look at the download page for more information on how to obtain Apache Tika 2.8.0.

03 February 2023: Apache Tika Release
 Apache Tika 2.7.0 has been released! This release includes releasing tika-parser-nlp-package as its own artifact, improvements to handling attachments in RFC822 .eml files, several other improvements and

Apache Tika

- Introduction
- Download
- Contribute
- Mailing Lists
- Tika Wiki
- Tika Server Wiki
- Issue Tracker
- Security
- Tika Support

Documentation

- Apache Tika 2.8.0
 - Getting Started
 - Supported Formats
 - Parser API
 - Parser 5min Quick Start Guide
 - Content and Language Detection
 - Configuring Tika
 - Usage Examples
 - API Documentation
- Apache Tika 2.7.0
 - Apache Tika 2.6.0
 - Apache Tika 2.5.0
 - Apache Tika 2.4.1
 - Apache Tika 1.28.5
 - Apache Tika 2.4.0
 - Apache Tika 1.28.4
 - Apache Tika 1.28.3
 - Apache Tika 1.28.2
 - Apache Tika 2.3.0
 - Apache Tika 1.28.1
 - Apache Tika 2.2.1
 - Apache Tika 1.28
 - Apache Tika 2.2.0
 - Apache Tika 2.1.0

Figure 62 : Site Apache.Tika.org

J'ai modifié le fichier du projet pom.XML pour rajouter la librairie avec la version 2.7.0 et également rajouté dans les dépendances :

```
<dependency>
    <groupId>org.apache.tika</groupId>
    <artifactId>tika-core</artifactId>
    <version>${tika-core.version}</version>
</dependency>
```

Après un clean install de notre projet avec maven, la librairie s'est bien installée. J'ai donc créé la méthode « getFileTypeByTika » dans le service qui m'a permis de vérifier le type du fichier téléchargé.

```
● ● ●

public String getFileTypeByTika(InputStream istream) throws TechnicalException {
    final Tika tika = new Tika();
    String fileType = "";
    try {
        fileType = tika.detect(istream);
    } catch (IOException e) {
        log.error(e.getMessage());
        throw new TechnicalException(String.format(MessageErreurEnum.TYPE_FILE_TIKA_INCORRECT.getMessage(), fileType),
                MessageErreurEnum.TYPE_FILE_TIKA_INCORRECT.getReferenceMetier());
    }
    return fileType;
}
```

Figure 63 : méthode getFileTypeByTika

Après quelques test avec mon IHM, la méthode a été concluante et fonctionne correctement.

13.BILAN ET PERSPECTIVES DU PROJET

Le projet Messagerie est passé par les différentes étapes de validation. Démonstration aux clients après avoir mis nos développements en qualif, une fois validée, nous avons mis le projet en recette puis en pré-production pour que le client puisse tester que cela lui convenait. Une fois validée, le projet a été mis en production en avril 2023. Le projet a été très intéressant et fonctionne bien. Niveau planning, le premier PI Planning a été réalisé très rapidement car nous étions plusieurs développeurs back/front/fullstack (Voir annexe 888) mais pour la deuxième PI Planning, nous avons pris un peu plus de temps même si nous avons été dans les temps par rapport à ce qui était estimé, dû à deux alternants seuls sur la partie pièces-jointes (Voir annexe 889).

Pour l'évolution de ce projet, il a été vu avec le client pour que la messagerie soit aussi accessible pour les utilisateurs de l'application SGF avec les administrateurs. De légères modifications seront à prévoir visuellement.

Pour toute mise en production, il faut :

- ▶ Mettre le correctif en Recette ;
- ▶ Tester la recette ;
- ▶ Envoyer les documents suivants pour pouvoir passer au CAB :
 - Créer un change dans EasyVista en renseignant la date de livraison TMA et la Date de MEP souhaitée (à confirmer par la suite par rapport aux capacités de la TME)
 - La page confluence à partir du modèle "Livraison Change TMA" est renseignée et envoyé par workflow confluence vers la TME
 - Une demande de MEP ;
 - Les livrables applicatifs et documentaires Cahier de tests, résultats tests 2 à 2, PV de VABF, ...
 - Un chronogramme de MEP ;
- ▶ Une fois le CAB validé, le correctif peut être installé en PréProd par la TME ;
 - Action TME : installer en PréProd ;
 - Action TMA : Tester en PréProd ;
 - Action TME : lancer le Rollback en PréProd ;
 - Action TMA : Tester en PréProd après le rollback ;
 - Action TME : Tester la ré-installation;
 - Action TMA : Tester en PréProd après la ré-installation ;
- ▶ Le correctif ou l'évolutif peut être installé en Prod par la TME ;
- ▶ Tester en Prod ; Sanity check si possible.

14.CONCLUSION

Cette alternance a répondu à toutes mes attentes et a permis de me conforter dans ma reconversion. Après mon stage, j'avais peur lors de ma formation, de ne pas réussir en entreprise, d'être perdu et de ne pas savoir m'adapter à ce nouveau domaine. Mais l'équipe Open a su me guider, m'écouter et me conseiller, ce qui m'a donné confiance et permis de réussir mon apprentissage.

J'ai pu découvrir la méthode de travail Scrum dont j'avais entendu parler dans mon entourage et en formation, que j'ai trouvé intéressant en termes de management, suivi du projet et d'esprit d'équipe. J'ai pu approfondir mes connaissances en Java mais surtout appris Spring et Angular alors que je partais de zéro sur ses langages / Framework.

Open m'a également permis de voir plus clair sur mon avenir en me proposant un contrat en CDI à la suite de mon alternance. Après un an dans l'équipe TMA-PNUM, on m'a proposé de me diriger vers le poste de développeuse back/Scrum pour allier mon savoir et mon savoir être et j'espère pouvoir, d'ici quelques années, devenir cheffe de projet.

Je ne retiens que du positif de cet apprentissage qui continue pour moi.

ANNEXES

Annexe 1 : Fiche de suivi de projet en entreprise

Nom et prénom du candidat : MARTINEAU Margaux

Document complété d'un commun accord entre le stagiaire et le responsable du stage en entreprise à joindre au rapport d'activité.

Compétences Voir le détail dans le référentiel d'emploi, d'activités et de compétences	Cocher les compétences mises en œuvre lors du projet en entreprise (en totalité ou partiellement)
C1 - Maquetter une application	X
C2 - Développer une interface utilisateur de type desktop	<input type="checkbox"/>
C3 - Développer des composants d'accès aux données	X
C4 - Développer la partie front-end d'une interface utilisateur web	X
C5 - Développer la partie back-end d'une interface utilisateur web	X
C6 - Concevoir une base de données	X
C7 - Mettre en place une base de données	X
C8 - Développer des composants dans le langage d'une base de données	<input type="checkbox"/>
C9 - Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement	X
C10 - Concevoir une application	<input type="checkbox"/>
C11 - Développer des composants métier	X
C12 - Construire une application organisée en couches	X
C13 - Développer une application mobile	<input type="checkbox"/>
C14 - Préparer et exécuter les plans de tests d'une application	<input type="checkbox"/>
C15 - Préparer et exécuter le déploiement d'une application	<input type="checkbox"/>

Observations éventuelles :

Du stagiaire : Entreprise et équipe bienveillantes. Mes responsables ont su me mettre sur un sujet intéressant et conforme aux compétences que je devais développer.

De l'entreprise : Margaux a beaucoup progressé depuis le début de son alternance. Elle prend en charge des développements en autonomie. Son niveau technique est plus élevé qu'un alternant lambda. Côté humain, Margaux a toutes les qualités nécessaires à une bonne intégration dans une équipe. Son appétence en communication et sa confiance pourrait lui permettre de faire du pilotage, si l'envie lui prend.

Nom et Prénom du stagiaire : Martineau Margaux
Signature :


Entreprise : OPEN
Nom du responsable de stage : Frédéric SOUTREL
Signature :


Annexe 2 : BackLog

ANTAI Tableaux de bord Projets Tickets Tableaux Plans Crée Recherche Tableau Tableau Kanban FILTRES RAPIDES: PI 16 PI 14 PI 15 Mes tickets uniquement Récemment mis à jour

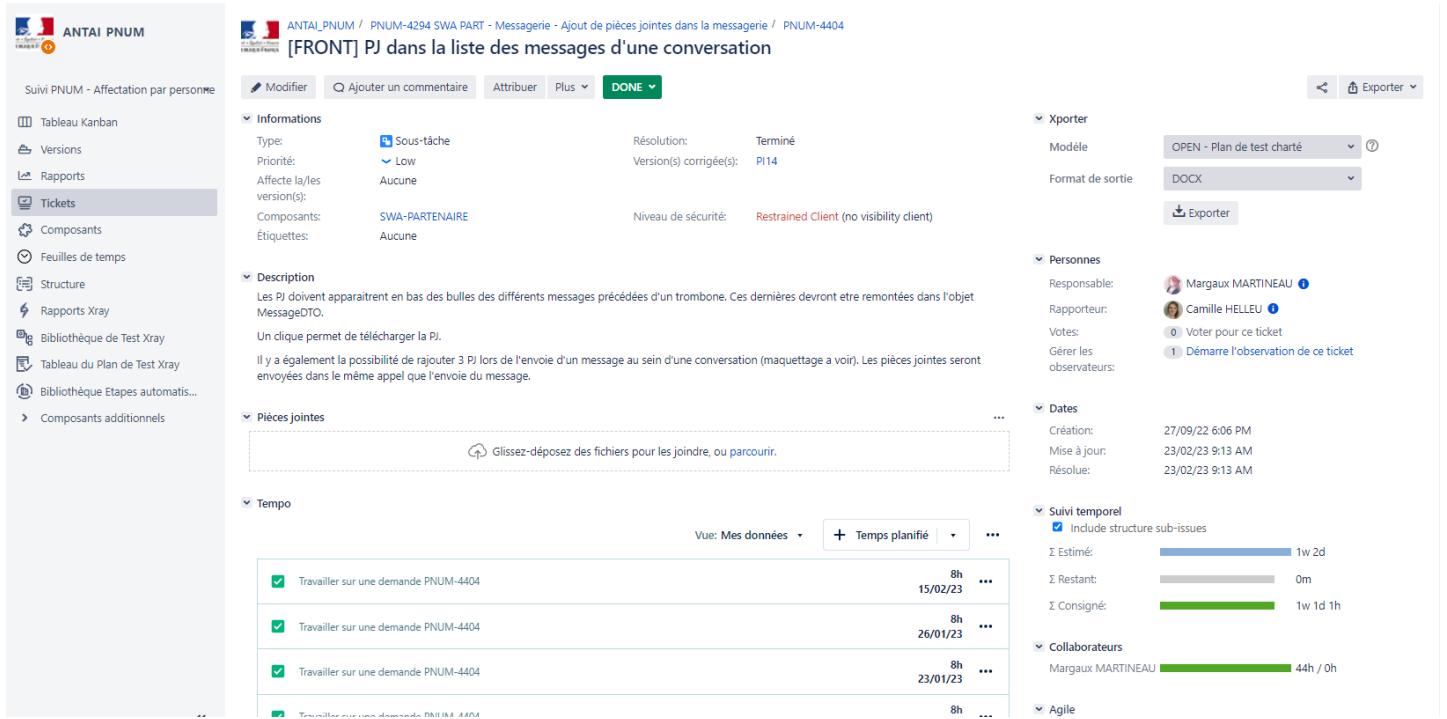
SUSPENDU 3	BACKLOG 32	EN ANALYSE 9	ESTIMÉ 6	EMBARQUÉ 11	EN COURS 12	EN DÉPLOIEMENT 0	EN TEST 0
TAF-76 TAM - Homologation - Corrections failles de sécurité TAM SSI, TAM Aucune Aucune 🕒	TAF-386 TFI - Libellé à modifier pour paiement fps TFI Aucune Aucune 🕒	PRU-786 PRU - Parcours contestation mobile - Permettre la prise de proximité dans les PRU Aucune Aucune 🕒	TAF-415 TAI / TAM - Ajouter le paiement de proximité dans les TAI, TAM 12 Aucune 🕒	PTR-306 Transverse - Diminution de la dette technique Supervision Aucune 80 🕒	PTR-293 Transverse - Cartographie, PIC PIC, Supervision Aucune 25 🕒	PTR-250 Transverse - Etude, Pilotage, Cérémonies Agile Supervision Aucune 270 🕒	
TAF-17 [TAM] Audit d'accessibilité RGAA TAM Aucune Aucune 🕒	TAF-51 FEATURE - TEMPLATE Aucune Aucune Aucune 🕒	PRU-56 PRU - Systran - évolution du CI du traducteur cas 3 PRU Aucune Aucune 🕒	PTR-18 Transverse - Ajout de la version applicative des IHM Supervision 15 Aucune 🕒	PTR-8 Transverse - Etude feature flipping Supervision Aucune 20 🕒	PRU-1287 PRU - Tests 2à2 - Traces + Rendre impossible une PRU Aucune 0 🕒	PRU-1304 PRU - Retours suite démo PRU PRU Aucune 0 🕒	
TAF-10 TAM - Compatibilité Android 12+ - Changement TAM 10 Aucune 🕒	TAF-28 TAI/TFI - Homologation/Corrections failles de SSI, TAI, TFI 50 Aucune 🕒	PRU-53 PRU - Nouveaux cas de contestation des dossiers LOM et du PRU 50 Aucune 🕒	PRU-722 PRU - Contrôle des âges autorisés pour la contestation PRU 25 Aucune 🕒	PRU-1182 PRU - API PRU Aucune 🕒	PRU-891 PRU - Prise en compte des PRU Aucune 🕒		
	TAF-21 TFI - Optimisation/Compression 🕒	PRU-24 ASSP-1326 🕒					

Annexe 3 : Kanban

Jira open Tableaux de bord Projets Tickets Tempo Tableaux Structure Plans Plus Recherche Tableau Tableau Kanban FILTRES RAPIDES: Mes tickets uniquement Récemment mis à jour

À FAIRE 260	EN COURS 38	EN ATTENTE 11	A RELIRE 15	RELUE (ANOS) 0
➤ Alexandre 19 tickets ➤ Arthur 6 tickets ➤ Bastien 10 tickets ➤ Brian 1 ticket ➤ Cédric 1 ticket ➤ David 14 tickets ➤ Guillaume 15 tickets ➤ Gurvan 11 tickets ➤ Jean-Yves 7 tickets ➤ Margaux 10 tickets ➤ Nhung 11 tickets ➤ Paul 16 tickets ➤ Philippe 13 tickets ➤ Samuel 1 ticket ➤ Steeven 14 tickets ➤ Sylvain 8 tickets ➤ Thomas 11 tickets ➤ Tout le reste 254 tickets				

Annexe 4 : UserStory



[FRONT] PJ dans la liste des messages d'une conversation

Informations

- Type: Sous-tâche
- Résolution: Terminé
- Priorité: Low
- Affecte la/les version(s): Aucune
- Version(s) corrigée(s): PI14
- Composants: SWA-PARTENAIRE
- Niveau de sécurité: Restrained Client (no visibility client)
- Étiquettes: Aucune

Description

Les PJ doivent apparaître en bas des bulles des différents messages précédées d'un trombone. Ces dernières devront être remontées dans l'objet MessageDTO.

Un clique permet de télécharger la PJ.

Il y a également la possibilité de rajouter 3 PJ lors de l'envoie d'un message au sein d'une conversation (maquettage à voir). Les pièces jointes seront envoyées dans le même appel que l'envoie du message.

Pièces jointes

Glissez-déposez des fichiers pour les joindre, ou parcourir.

Tempo

Vue: Mes données	+ Temps planifié	...
Travailler sur une demande PNUM-4404	8h 15/02/23 ...	
Travailler sur une demande PNUM-4404	8h 26/01/23 ...	
Travailler sur une demande PNUM-4404	8h 23/01/23 ...	
Travailler sur une demande PNUM-4404	8h ...	

Xporter

Modèle: OPEN - Plan de test charté

Format de sortie: DOCX

Personnes

Responsable: Margaux MARTINEAU

Rapporteur: Camille HELLEU

Votes: Voter pour ce ticket

Gérer les observateurs: Démarrer l'observation de ce ticket

Dates

Création: 27/09/22 6:06 PM

Mise à jour: 23/02/23 9:13 AM

Résolue: 23/02/23 9:13 AM

Suivi temporel

Include structure sous-issues

Σ Estimé: 1w 2d

Σ Restant: 0m

Σ Consigné: 1w 1d 1h

Collaborateurs

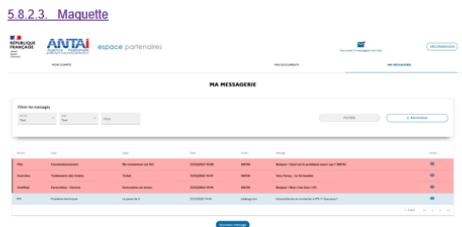
Margaux MARTINEAU 44h / 0h

Agile

Annexe 5 : Spécifications fonctionnelles

[Logo client éventuel]

5.8.2.3. Maquette



5.8.2.4. Description de l'écran

Libellé	Description	Type	Modifiable	Obligatoire
Généralités				
Titre	Ma messagerie	(Texte)	=	=
Filtres				
Service	Choix du Service	(Liste déroulante) Valeurs FPS- Cloudow-SI Fournitures-PvP	=	=
Sujet	Choix du Sujet	(Liste déroulante) Valeurs Problème / Problème technique / Facturation / Facture / Facturation - Reversement / Facturation - Autre / Informations RGPD / Demande de certificat / Informations code service / Annulation / Question liée au service et/ou Validation compte / Connexion AGC / Traitements des tickets / Demande d'accès à l'espace personnel	=	=
Objet	Objet du message	(Espace libre)	=	=

Open | ANTAI | Statut Open - Validé | Classification : Protégé | Version V1.0.20
Ce document est la propriété d'Open - Son utilisation, sa reproduction et sa diffusion en dehors des personnes autorisées sont interdites.

85/93

[Logo client éventuel]

[Logo client éventuel]

Libellé	Description	Type	Modifiable	Obligatoire
Filtrer	Lance la recherche avec les filtres saisis	(Bouton)	-	-
X Réinitialiser	Réinitialise les filtres à vide	(Bouton)	-	-
Contenu du tableau des messages				
Service	Service à prendre	(Texte)	-	-
Sujet	Sujet du message	(Texte)	-	-
Objet	Objet du message	(Texte)	-	-
Date	Date du dernier échange dans le message	(Texte)	-	-
Auteur	Auteur du dernier échange dans le message	(Texte)	-	-
Message	Message	(Texte)	-	-
Action	Visualisation des échanges du message	(Icone-Bouton)	-	-
Nouveau message	Accès à la page de création de message	(Bouton)	-	-

5.8.2.5. Règles d'initialisation

Code	Titre	Règles d'initialisation
RI_CSUL_MESS_ORG_01	Liste des messages	La liste est proposée, triée et filtrable sur chacune des colonnes sauf 'Message'.
RI_CSUL_MESS_ORG_02	Filtres	Le filtre 'Service' est le seul pour lequel un critère de filtre n'est saisi.
RI_CSUL_MESS_ORG_03	Tri	Par défaut, la liste est triée comme suit : D'abord les messages non lu, triés par date de dernier échange décroissant, puis les messages lu, triés par date de dernier échange décroissant.

5.8.2.6. Règles de gestion

Code	Titre	Règles de gestion
RG_CSUL_MESS_ORG_01	Message	Chaque ligne du tableau affiche le dernier échange du message.
RG_CSUL_MESS_ORG_02	Message à l'état 'Non lu'	Les messages à l'état 'Non lu' doivent apparaître de façon avancée dans le résultat de liste, en gras et en couleur rouge avec barre sur le côté.
RG_CSUL_MESS_ORG_03	Accessibilité des messages	L'utilisateur accède à la liste de tous les messages de son organisation (et seulement ceux de son organisation). L'utilisateur pourra visualiser la totalité des messages échangés avec l'administration pour son compte. Il visualise ainsi la liste des messages écrits par les autres utilisateurs du compte SWA PART.
RG_CSUL_MESS_ORG_04	Date	La colonne date indique la date du dernier échange au format JJ/MM/AA HH:MM
RG_CSUL_MESS_ORG_05	Message	L'affichage est limité sur la colonne 'Message', les 80 premiers caractères sont affichés puis '...'.
RG_CSUL_MESS_ORG_06	Auteur	La colonne 'Auteur' indique le dernier auteur ayant envoyé un message. Si c'est un administrateur, on affiche 'ANTAI'. Si c'est de l'organisation, le mail de l'auteur est affiché.
RG_CSUL_MESS_ORG_07	Action	Dans la colonne action, une icône est affichée tout le temps et permet d'afficher l'ensemble des échanges (avec possibilité d'y répondre) dans l'écran REP_MESS_ORG.

Open | ANTAI | Statut Open - Validé | Classification : Protégé | Version V1.0.20
Ce document est la propriété d'Open - Son utilisation, sa reproduction et sa diffusion en dehors des personnes autorisées sont interdites.

85/93

[Logo client éventuel]

Code	Message d'erreur
MES.REP.MESS.ADM.03	Le fichier %s contient des virus. Merci de télécharger un fichier non infesté.

5.8.4. Répondre à un message - REP.MESS.ORG

5.8.4.1. Description générale

Cet écran permet de visualiser les échanges d'un message envoyé aux administrateurs, et d'y répondre au besoin.

5.8.4.2. Règles d'entrée

Code	Titre	Description
RE.REP.MESS.ORG.01	Point d'accès à la fonction	L'accès à la fonction se fait à partir : - D'un clic sur l'icône œil dans la colonne action de la liste des messages de l'organisation 'CSUL.MESS.ORG'.

5.8.4.3. Maquette



Open | ANTAI | Statut Open : Validé | Classification : Protégé | Version V1.0.20

91/94

[Logo client éventuel]

5.8.4.4. Description de l'écran

Libellé	Description	Type	Modifiable	Cherchable
Généralités				
Lien	Revenir à la liste des messages	[Hyperlien]	:	:
Titre	Service : XXX Sujet : YYY Objet : ZZZ, où XXX est le service sélectionné pour le message, YYY est le sujet sélectionné pour le message et ZZZ est l'objet saisi pour le message	[Texte]	:	:
Historique des messages				
Message	Zone de saisie d'un nouveau message	[Saisie libre]	Oui	Oui
Pièce Jointe	Zone et bouton permettant l'ajout de pièces jointes (jusqu'à 3)	[Fichiers JPEG, JPG, PNG, TIFF, DOC, DOCX, XLS, XLSX, CSV] 5 Mo max par fichier, 3 fichiers max	Oui	Non
Envoyer message	Envoyer un message	[Bouton]	:	:
Messages	Liste des messages et pièces jointes	[Zones de saisie/fichiers]	:	:

5.8.4.5. Règles d'initialisation

Code	Titre	Règles d'initialisation
RI.REP.MESS.ORG.01	Tri des échanges du message	Par défaut, les échanges du message sont triés du plus récent
RI.REP.MESS.ORG.02	Envoyer message	Le bouton est prisé tant que rien n'a été saisi dans la zone message.

Open | ANTAI | Statut Open : Validé | Classification : Protégé | Version V1.0.20

92/94

[Logo client éventuel]

RI.REP.MESS.ORG.03	Disposition des échanges du message	A droite, tous les messages administrateurs sont affichés, et à gauche, tous les messages des utilisateurs de l'organisation sont affichés. Les échanges des administrateurs sont affichés sous fond blanc. Les échanges des utilisateurs de l'organisation sont affichés sous fond bleu.
RI.REP.MESS.ORG.04	Informations de chaque échange	Pour chaque échange, il est indiqué le mail de l'auteur du message ou la mention 'ANTAI', ainsi que la date d'envoi au format JJ/MM/AA HH:MM.

5.8.4.6. Règles de gestion

Code	Titre	Règles de gestion
RG.REP.MESS.ORG.01	Envoyer message	Le bouton 'Envoyer message' est saisissable à partir du moment où du texte a été saisi dans la zone 'Saisissez votre message'.
RG.REP.MESS.ORG.02	Envoi du message	Après avoir cliqué sur le bouton 'Envoyer message', le message est créé avec les valeurs suivantes : Le message saisi, les valeurs du message (Service-Sujet-Objet), Appli SWAPART, Utilisateur ayant rédigé le message, Organisation de l'utilisateur rédacteur, date et heure d'enregistrement, flag non lu. La liste des échanges est mise à jour avec le nouveau message.
RG.REP.MESS.ORG.03	Revenir à la liste des messages	Un clic sur l'hyperlien 'Revenir à la liste des messages' affiche la liste des messages de l'organisation 'CSUL.MESS.ORG'.
RG.REP.MESS.ORG.04	Ajout de pièce jointe	Une pièce jointe ne peut être utilisée seule en tant que réponse, il est nécessaire d'ajouter un message pour pouvoir envoyer une pièce jointe. On peut ajouter de 0 à 3 fichiers en tant que pièces jointes. L'ajout se fait via le bouton « Ajouter » (navigation dans les répertoires du PC) ou glissant/Déposant fichier dans le cadre à droite du bouton « Ajouter ». Chaque fichier ne peut faire qu'un maximum de 5 Mo. Après l'ajout d'un fichier une ligne est affichée au-dessus du champ d'ajout de fichier et du bouton « ajouter »: ANTAI_convention_PV1.pdf La ligne présente le nom du fichier avec son extension. Un icône d'œil permet de visualiser le fichier et une icône de poubelle permet de le supprimer. Lorsque le nombre maximum de pièces jointes est atteint (3), le champ et le bouton permettant d'ajouter des fichiers disparaissent. Si on supprime un fichier et que l'on repasse à moins de 3 pièces jointes, le champ et le bouton d'ajout de fichier réapparaissent. Si l'utilisateur ajoute un fichier de plus de 5 Mo le message d'erreur MES.REP.MESS.ORG.01 est affiché. Si un virus est détecté par l'antivirus dans la pièce jointe, le message MES.REP.MESS.ORG.02 est affiché et le message et les pièces-jointes ne sont pas envoyées.
RG.REP.MESS.ORG.05	Véification des pièces jointes	Au clic sur une pièce-jointe, celle-ci est téléchargée.

Messagerie-PJ
a mis en forme : Couleur de police : Couleur personnalisée(RVB(0;87;142))

Messagerie-PJ
a mis en forme le tableau

Messagerie-PJ
a mis en forme : Couleur de police : Couleur personnalisée(RVB(0;87;142))

Messagerie-PJ
a mis en forme : Couleur de police : Couleur personnalisée(RVB(0;87;142))

Open | ANTAI | Statut Open : Validé | Classification : Protégé | Version V1.0.20
Ce document est la propriété d'Open - Son utilisation, sa reproduction et sa diffusion en dehors des personnes autorisées sont interdites.

93/95

Annexe 6 : Maquettes



ANTAI
Agence Nationale
Traitement Automatisé d'Infractions

espace partenaires

DECONNEXION

0 message

MON COMPTE
MES DOCUMENTS
MA MESSAGERIE

[← Revenir à la liste des messages](#)

MA MESSAGERIE

Service : FPS
Sujet : Connexion AGC
Objet

Lore ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui.

ENVOYER

U
FICHIER2.JPG
FICHIER3.XLS
AJOUTER

Jules Meunier
ADMIN
02/09/2022 10:05

Aimé Pires ORGA 02/09/2022 09:32

U
FICHIER1.JPG
U
FICHIER2.PDF

Lorraine Tessier
ADMIN
01/09/2022 16:08

Richard Bourdon ORGA 01/09/2022 14:55

Sites Utiles

[interieur.gouv.fr](#)
[ants.gouv.fr](#)
[service-public.fr](#)
[legifrance.gouv.fr](#)
[securite-routiere.gouv.fr](#)

[Accueil](#) | [A propos de l'ANTAI](#) | [Accessibilité : non conforme](#) | [Contact](#) | [Données personnelles](#) | [Mentions légales](#)



DECONNEXION

1 message

MON COMPTE

MES DOCUMENTS

MA MESSAGERIE

MA MESSAGERIE

Filtrer les messages

Service	Sujet	Objet	Date	Auteur	Message	Action
Tout	Tout			Jules Meunier		

FILTRER

X Réinitialiser

Service	Sujet	Objet	Date	Auteur	Message	Action
FPS	Connexion AGC		02/09/22 10:05	Jules Meunier		
Freeflow	Problème technique		03/09/22 10:55			
SI Fourrières	Validation compte		20/08/22 14:26			
PVE	Annulation		10/08/22 16:41			
Freeflow	Traitements des tickets		15/07/22 09:05			

1 - 5 of 5 |< < > >|

NOUVEAU MESSAGE

Sites Utiles

[interieur.gouv.fr](#)  [ants.gouv.fr](#)  [service-public.fr](#)  [legifrance.gouv.fr](#)  [securite-routiere.gouv.fr](#) 

Accueil | A propos de l'ANTAI  | Accessibilité : non conforme | Contact  | Données personnelles | Mentions légales 

Annexe 7 : Contrat d'interface

```

openapi: 3.0.1
info:
  title: Api service messagerie
  description: Service de messagerie pour les applications SWAPART et SWAPART admin
  version: "1.0"
servers:
  - url: '/api/messagerie'
tags:
  - name: Messagerie
    description: Service de messagerie Controller
  - name: Messagerie Admin
    description: Service de messagerie Controller pour les Admins
paths:
  '/message/{idConversation}':
    post:
      tags:
        - Messagerie
      summary: Créer un nouveau message dans une conversation existante
      operationId: postMessage
      parameters:
        - name: idConversation
          in: path
          required: true
          schema:
            type: integer
            description: Id de la conversation existante
            example: 123456789
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/MessageDto'
      responses:
        '200':
          description: OK
        400:
          $ref: '#/components/responses/BadRequest'
        401:
          $ref: '#/components/responses/Unauthorized'
        403:
          $ref: '#/components/responses/Forbidden'
        404:
          $ref: '#/components/responses/NOTFOUND'
        500:
          $ref: '#/components/responses/InternalError'
      security:
        - bearer-key: []
  '/pieceJointe/{idPieceJointe}':
    get:
      tags:
        - Messagerie
      summary: 'Récupération d''une pièce jointe par son id'
      operationId: getPieceJointe
      parameters:
        - name: idPieceJointe
          description: 'Récupération de la pièce jointe liée à un message'
          in: path
          required: true
          schema:
            type: integer
      responses:
        '200':
          description: OK
          content:
            multipart/form-data:
              schema:
                type: string
                format: binary
        '400':
          description: Données en entrée invalides
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ApiError'
        401:
          $ref: '#/components/responses/Unauthorized'
        403:
          $ref: '#/components/responses/Forbidden'
        404:
          $ref: '#/components/responses/NOTFOUND'
        500:
          $ref: '#/components/responses/InternalError'
      security:
        - bearer-key: []

```

```

schemas:
  ApiError:
    type: object
    properties:
      code:
        type: string
      label:
        type: string
  MessageDto:
    type: object
    required:
      - message
    properties:
      messageLu:
        type: boolean
      dateCreation:
        type: string
        format: date-time
      description: Date de création du message
      example: 2017-07-21T17:32:28Z
  identiteAuteur:
    type: string
    description: Auteur du message
    example: Harry Covert
  lsExpediteur:
    type: boolean
    description: Est à vrai si l'utilisateur fait partie de l'organisation qui a écrit ce message (message à afficher à gauche)
  message:
    type: string
    description: Contenu du message
    example: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris lacus metus, pharetra eget congue et, tempor sit amet lorem. Orci.
    maxLength: 3000
  piecesJointes:
    type: array
    items:
      $ref: '#/components/schemas/PieceJointeDto'
  PieceJointeDto:
    type: object
    description: pièces jointes envoyées au format base64
    properties:
      id:
        type: integer
        format: int64
      nom:
        type: string
        description: Nom du fichier
      example: Justificatif.pdf
      taille:
        type: integer
        format: int64
        description: taille de la pièce jointe (Max 5)
      example: 2
      fichier:
        type: string
        description: Contenu de la pièce jointe
      example: Image
  securitySchemes:
    bearer-key:
      type: http
      scheme: bearer
      bearerFormat: JWT

```

Annexe 8 : Repository

```

● ● ●

@Slf4j
@Service
public class PieceJointeRepository implements IPieceJointeRepository {

    @Autowired
    private PieceJointeEntityMapper pieceJointeEntityMapper;

    @Autowired
    private ConversationEntityMapper conversationEntityMapper;

    @Autowired
    private IMessageJpaRepository messageJpaRepository;

    @Autowired
    private IPieceJointeJpaRepository pieceJointeJpaRepository;

    @Override
    public List<Long> creer(List<PieceJointe> pieceJointe, HashMap<String, InputStream> listFichiersPjs,
                           Long messageId) throws PnumNonAutoriseException, TechnicalException {
        List<PieceJointeEntity> listPieceJointeEntity = pieceJointeEntityMapper.listDomainToEntity(pieceJointe);
        Optional<MessageEntity> messageEntityOptional = messageJpaRepository.findById(messageId);
        if (messageEntityOptional.isPresent()) {
            for (PieceJointeEntity pj : listPieceJointeEntity) {
                InputStream fichier = listFichiersPjs.get(pj.getNom());
                pj.setMessage(messageEntityOptional.get());
                // Sauvegarde du fichier sur disque
                String cheminFichier = buildCheminPJ(pj, messageEntityOptional.get());
                File targetFile = new File(cheminFichier);
                try {
                    FileUtils.copyInputStreamToFile(fichier, targetFile);
                } catch (IOException e) {
                    String messageErreur = MessageErreurEnum.RESSOURCE_INDISPONIBLE.getMessage();
                    PnumLogger.errorDebug(log, MessageErreurEnum.RESSOURCE_INDISPONIBLE.getCodeTechnique()
                        + " : " + messageErreur, e);
                    throw new TechnicalException(MessageErreurEnum.RESSOURCE_INDISPONIBLE.getMessage(),
                        MessageErreurEnum.RESSOURCE_INDISPONIBLE.getReferenceMetier());
                }
            }
        }
        List<PieceJointeEntity> piecesJointesEntityList = pieceJointeJpaRepository.saveAll(listPieceJointeEntity);
        return piecesJointesEntityList.stream().map(PieceJointeEntity::getId).collect(Collectors.toList());
    }

    @Override
    public PieceJointe getPieceJointeById(Long id) throws NotFoundException {
        Optional<PieceJointeEntity> pieceJointe = pieceJointeJpaRepository.findById(id);
        if (pieceJointe.isEmpty()) {
            throw new NotFoundException(MessageErreurEnum.ERREUR_PIECE_JOINTE_INTROUVABLE.getMessage(),
                MessageErreurEnum.ERREUR_PIECE_JOINTE_INTROUVABLE.getReferenceMetier());
        }
        return pieceJointeEntityMapper.entityToDomain(pieceJointe.get());
    }

    @Override
    public List<PieceJointe> getPieceJointeByMessageId(Long idMessage) {
        return pieceJointeEntityMapper.listEntityToDomain(pieceJointeJpaRepository.findAllByMessage_Id(idMessage));
    }

    /**
     * Permet d'aller stocker la PJ sur le disque
     *
     * @param pJEntity une piece jointe
     * @param message message lié à la piece jointe
     * @return emplacement sur le disque
     */
    @NotNull
    private String buildCheminPJ(PieceJointeEntity pJEntity, MessageEntity message) {
        StringBuilder cheminConversation = buildCheminConversation(message.getConversation());
        return cheminConversation
            .append(File.separator)
            .append(message.getId())
            .append(File.separator)
            .append(pJEntity.getNom())
            .toString();
    }
}

```

Annexe 9 : ZProd PI 1 et PI 2

Clé	Résumé	TP	P:	Z:	État	Respon:	V	Estimé	Consommé	RAF	BJP	BIP	zProd
	▶ PRU				14			149,50 0,00	95,81 0,00	0,00	95,81 0,00	53,69 0,00	
	▶ SGF				87			87,50 0,00	48,50 0,00	0,00	48,50 0,00	39,00 0,00	
	▶ SWA				3			3,00 0,00	0,00	0,00	0,00	3,00 0,00	
	▼ SWA-PARTENAIRE				32			328,50 0,00	200,00 0,00	0,00	200,00 0,00	128,50 0,00	
✓ PNUM-4284	▶ SWA PART - Mise en place d'un système de messagerie interne entre swa part et admin • Prérequis Aucun				16	DONE		181,00 0,00	87,06 0,00	0,00	87,06 0,00	93,94 0,00	
✓ PNUM-4285	▶ SWA PART/ADMIN - Messagerie - Prérequis Technique Messagerie • A faire dans un premier temps				30	FERMÉE		38,00 0,00	6,75 0,00	0,00	6,75 0,00	31,25 0,00	
✓ PNUM-4286	▶ SWA PART - Messagerie - Accès messagerie • Après la connexion à SWA PART, dans le haut des pages				7	FERMÉE		7,50 0,00	5,63 0,00	0,00	5,63 0,00	1,88 0,00	
✓ PNUM-4287	▶ SWA PART - Messagerie - Créer un nouveau message • Dans la page de visualisation des messages				14	FERMÉE		14,50 0,00	6,13 0,00	0,00	6,13 0,00	8,38 0,00	
✓ PNUM-4288	▶ SWA PART - Messagerie - Affichage des messages côté organisation • En cliquant sur l'icône enveloppe				17	FERMÉE		17,00 0,00	8,13 0,00	0,00	8,13 0,00	8,88 0,00	
✓ PNUM-4289	▶ ADMIN - Messagerie - Accès messagerie et affichage des messages • Après la connexion à l'Admin				7	FERMÉE		7,00 0,00	10,88 0,00	0,00	10,88 0,00	-3,88 0,00	
✓ PNUM-4290	▶ ADMIN - Messagerie - Indication nombre nouveau message et mise à jour • Un indicateur textuel indiquant				12	FERMÉE		13,50 0,00	7,19 0,00	0,00	7,19 0,00	6,31 0,00	
✓ PNUM-4291	▶ SWA PART - Messagerie - Indication nombre nouveau message et mise à jour • Un indicateur textuel indiquant				5	FERMÉE		5,00 0,00	5,50 0,00	0,00	5,50 0,00	-0,50 0,00	
✓ PNUM-4292	▶ ADMIN - Messagerie - Visualiser les messages d'une organisation et y répondre • Après avoir cliqué sur l'icône				31	FERMÉE		31,00 0,00	18,63 0,00	0,00	18,63 0,00	12,38 0,00	
✓ PNUM-4293	▶ SWA PART - Messagerie - Visualiser les messages d'une organisation et y répondre • Après avoir cliqué sur l'icône				14	FERMÉE		14,00 0,00	7,25 0,00	0,00	7,25 0,00	6,75 0,00	
✓ PNUM-4296	▶ SWA PART/ADMIN - Messagerie - Prépa test				12	FERMÉE		12,00 0,00	5,25 0,00	0,00	5,25 0,00	6,75 0,00	
✓ PNUM-4297	▶ SWA PART/ADMIN - Messagerie - Documentation				11	FERMÉE		11,00 0,00	3,00 0,00	0,00	3,00 0,00	8,00 0,00	
✓ PNUM-4298	▶ SWA PART/ADMIN - Messagerie - Non Régression				2	FERMÉE		2,00 0,00	2,00 0,00	0,00	2,00 0,00	0,00	

Clé	Résumé	TP	P:	Z:	État	Respon:	V	Estimé	Consommé	RAF	BJP	BIP	zProd
✓ PNUM-4542	▶ SWA PART - Messagerie V2 • Ajout des Pièces Jointes				6	DONE		63,00 0,00	60,00 0,00	0,00	60,00 0,00	3,00 0,00	
PNUM-4294	▶ SWA PART - Messagerie - Ajout de pièces jointes dans la messagerie • A partir de la page de création				30	IS DONE		36,50 0,00	47,00 0,00	0,00	47,00 0,00	-10,50 0,00	
PNUM-4295	▶ ADMIN - Messagerie - Ajout de pièces jointes dans la messagerie • A partir de la page de visualisation				7	IS DONE		7,50 0,00	5,25 0,00	0,00	5,25 0,00	2,25 0,00	
PNUM-4853	▶ Messagerie - SFD et non-régression				8	IS DONE		8,00 0,00	3,13 0,00	0,00	3,13 0,00	4,88 0,00	
PNUM-4854	▶ Messagerie - Prépa tests				1	IS DONE		1,00 0,00	0,88 0,00	0,00	0,88 0,00	0,13 0,00	
PNUM-4855	▶ Messagerie - Ajout WebSocket • Installation de Netty afin d'utiliser les Websockets en passant par le				10	IS DONE		0,00	3,00 0,00	0,00	3,00 0,00	-3,00 0,00	
PNUM-4856	▶ Messagerie - VABF					IS DONE		10,00 0,00	0,75 0,00	0,00	0,75 0,00	9,25 0,00	

Annexe 10 : Visuel final

Vous avez 4 message(s) non lu(s)

DÉCONNEXION

MON COMPTE
FOURRIÈRE
MES DOCUMENTS
MA FACTURATION
MA MESSAGERIE

MA MESSAGERIE

Filtrer les messages

Service
Sujet
Objet

Tout
Tout

FILTRE
X Réinitialiser

Service	Sujet	Objet	Date	Auteur	Message	Action
FPS	Problème technique	Démonstration CDA	18/05/2023 16:51	ANTAI	Bonjour, merci, je prend bonne réception.	
FPS	Problème technique	Problème technique	16/11/2022 20:00	swapart@test.com	Message 1 conversation 5	
Fourrière	Annulation	Annulation	12/11/2022 12:00	swapart@test.com	Message 1 conversation 3	
FPS	Conventionnement	Conventionnement	10/11/2022 14:00	ANTAI	Message 2 conversation 1	

1 à 4 de 4 | < < > >|

Nouveau message

Sites
[interieur.gouv.fr](#)
[ants.gouv.fr](#)
[service-public.fr](#)
[legfrance.gouv.fr](#)
[securite routiere.gouv.fr](#)

Vous avez 3 message(s) non lu(s)

DÉCONNEXION

MON COMPTE
FOURRIÈRE
MES DOCUMENTS
MA FACTURATION
MA MESSAGERIE

[← Retour à la liste des messages](#)

MA MESSAGERIE

Service : FPS
Sujet : Problème technique
Objet : Démonstration CDA

* Champs requis
Envoyer message

Message *

0/3000

CHARGER UN FICHIER
Glissez et déposez vos fichiers ici
AJOUTER

Formats possibles: PDF, JPEG/JPG/PNG/TIF/TIFF (Images), DOC/DOCX (Word), CSV/XLS/XLSX (Excel).
Taille maximale 5 Mo par fichier.
Vous pouvez ajouter 3 pièces jointes maximum.

ANTAI 18/05/2023 16:51
Bonjour, merci, je prend bonne réception.

swapart@test.com 18/05/2023 16:39
Bonjour, ci-joint un document complémentaire
[Fiche suivie des compétences mises en oeuvre en entreprise Titre CDA 12_2022 \(3\).pdf \(0.16 Mo\)](#)