

Rapport de stage de fin d'étude

Par

Loan THOMY

Stage réalisé du 10/06/2024 au 30/08/2024

Durée : 3 mois

Enseignant référent : Pierre RODRIGUEZ

Tuteur de stage : Martial SÉRON

Remerciements

Je voudrais remercier toutes les personnes qui ont joué un rôle essentiel dans mon intégration et mon parcours au sein d'Alturing pendant ce stage. Je suis également reconnaissant envers tous les professeurs et les intervenants de l'université de La Rochelle pour la qualité de leurs enseignements cette année.

Un merci tout particulier à l'équipe du Back-Office, avec qui j'ai partagé le bureau pendant trois mois. Les différents échanges enrichissants et leurs anecdotes ont rendu mon environnement de travail plus agréable.

Je souhaite exprimer ma reconnaissance envers M. Thierry Renaudin (responsable du Centre des Services) et M. Martial Séron pour m'avoir offert l'opportunité de réaliser mon projet de fin d'études, en me confiant une autonomie quasi totale. J'ai énormément appris, tant de manière autodidacte que grâce aux conseils de Martial, tant sur le plan technique que humain.

Table des matières

Remerciements	2
1. Introduction et contexte	4
2. État de l'art d'un domaine.....	9
3. Mon projet	16
a. Cahier des charges initial.....	16
b. Outils et méthodologies.....	16
c. Organisation	17
4. Détail de mes réalisations	20
a. Architecture du projet.....	20
b. Design de l'application.....	21
c. La création de l'API	25
d. Intégration web.....	29
5. Ensuite ?	31
6. Conclusion	32
Glossaire.....	33

1. Introduction et contexte

Alturing est une entreprise de services dans le support, l'administration et la supervision de systèmes d'informations spécialisés dans le domaine du transport de colis. C'est une filiale du groupe La Poste.

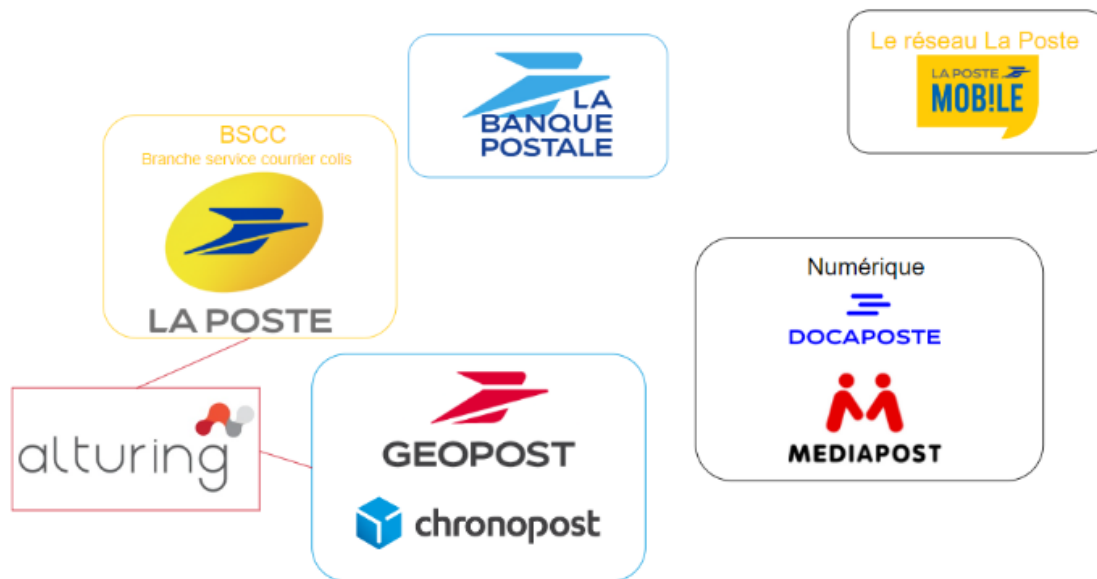


Figure 1: Schéma du groupe La Poste

L'entreprise est séparée en 2 sites, le premier qui est le siège social se situe à Paris où se trouve les ressources humaines et les équipes OPS (systèmes, réseau, base de données...). Et le deuxième site, lieu de mon stage, est le centre de services (CSV) à Verrières-en-Anjou (proche d'Angers).

Le centre de services réalise l'administration, la supervision et le support pour deux différents clients : métier et shipping.

- Clients métier : Les clients affiliés au groupe La Poste comme Chronopost, Colissimo, Geopost ainsi que Pickup Services.
- Clients shipping : Les clients utilisant les services de livraison des clients métiers

Le CSV réalise donc ces différentes activités :

- La supervision : vérifie l'état de santé du système d'information en temps réel ainsi que les contrôles production
- L'habilitation : attribution des droits aux utilisateurs
- Le E-learning : administration et réalisation de modules d'apprentissage en ligne
- Le pilotage incidentel : coordination de partenaires externes et des cellules de crises¹, reporting client
- Le support : assistance utilisateur, support avant-vente et technique
- La sécurité : administration de l'antivirus et lutte contre la Cybercriminalité de La Poste

Et ensuite ces activités sont réparties parmi les différentes équipes présentes au CSV (voir le schéma des équipes du CSV, figure 2) :

- Le centre de contact, composé de techniciens chargés du support direct des clients Shipping et/ou Métier.
- Le H24 est une équipe qui garantit le service 24h/24 7j/7 présente chez Alturing, qui est chargée de superviser les systèmes d'informations des clients du groupe La Poste. Et également de l'administration des habilitations pour les diverses applications.
- L'équipe sécurité, sous la responsabilité du RSSI de Chronopost, s'occupe de la sécurité informatique des différents SI
- Le back office est chargé de rédiger des procédures pour le CSV grâce à leurs proximités avec différentes équipes applicatives auxquelles les autres équipes pourront être confrontées.

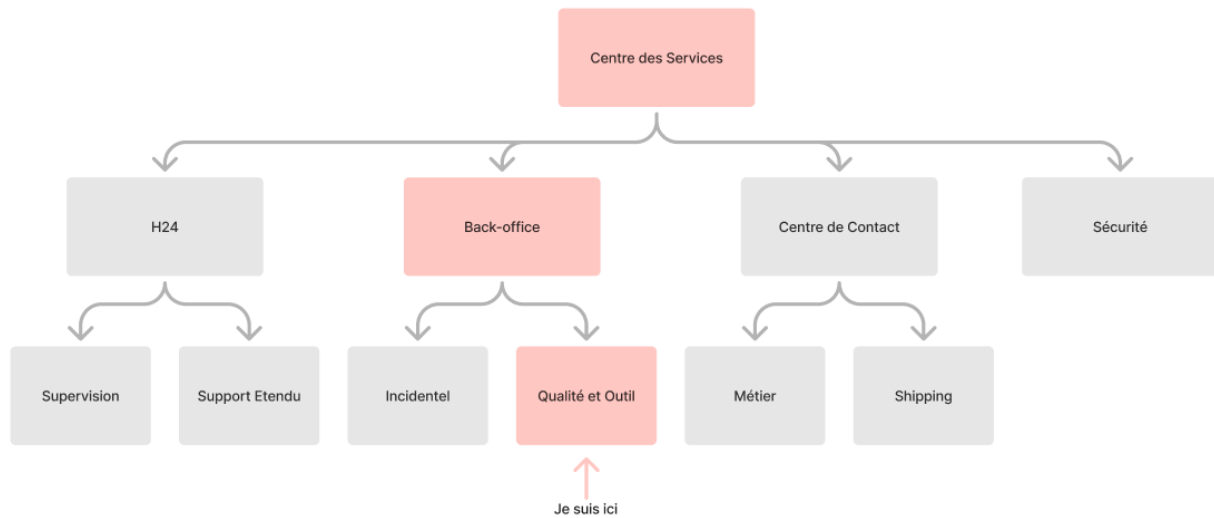


Figure 2 : Schéma des équipes du CSV


Le fonctionnement d'Alturing est basé sur les bonnes pratiques proposées par le référentiel l'ITIL² et est ainsi composé de trois niveaux : CSV, OPS et DSI client permettant de gérer les incidents rencontrés. Le CSV regroupe les collaborateurs qui traitent les alertes directement et accompagnent les utilisateurs. On y retrouve le personnel du CSV. Lorsque le traitement de l'incident ou de la demande n'est pas réalisable de leur côté, ils sollicitent les équipes OPS, en charge de l'exploitation du Système d'Information (SI) et divisées en plusieurs services comme : système, réseau ou encore base de données. Le dernier niveau est la DSI client, comprenant des experts sur des domaines et solutions techniques précis comme l'infrastructure Linux ou des applications.

J'ai rejoint le Centre de Services en tant que stagiaire développeur full-stack au sein de l'équipe Qualité & Outil. L'équipe Outils dont je fais partie est composée de 3 développeurs full-stack, Martial SÉRON le responsable des outils du centre des services, Mickael EXILE et Nicolas PELLETIER qui termine son alternance cet été pour valider un cycle ingénieur.

L'équipe Qualité & Outil possède le statut de référent technique dans la gestion des tickets. On peut donc lui demander des demandes d'accès à des applications, des problèmes sur les outils comme GLPI³ qui sont remontés par les utilisateurs. L'équipe est également référente technique des collègues du Back-Office qui ont une vision plutôt métier avec de bonnes connaissances techniques, mais qui peuvent manquer de connaissances sur certains sujets comme des requêtes en base de données ou le fonctionnement d'une application. L'équipe est donc disposée pour les aider en cas de besoin.

Les projets sont notre cœur d'activité qu'ils soient pour les clients à l'exemple du portail du centre de services qui permet à des utilisateurs de générer des demandes, de constater des incidents via une interface intuitive. La création d'un outil de communication pour des incidents qui ont un impact critique sur les livraisons. Les projets peuvent avoir une vocation d'outil interne au CSV comme la QS4. Un outil qui permet d'afficher un écran pour le statut des techniciens (en appel, traitement, pause ...), et également la base de connaissances qui permet de centraliser la connaissance de tout le centre de services.

Personnellement, on m'a confié la tâche de créer un nouveau projet web de zéro permettant à un employé d'administrer différents tableaux de bord appelés « FLASH PROD ». Actuellement, il n'y a pas d'interface web afin d'administrer ces rapports, un mail est écrit manuellement indiquant l'état de services informatiques (voir Figure 3).



colissimo

✓ Tous les systèmes sont opérationnels

Périmètres critiques

> Intégration Collis (T&T)	✓
> Livraison	✓
> Gestion de Plateforme de Tri	✓
> International	✓
> Affranchissement	✓
> Service aux clients	✓
> Communiquer	✓

Faits marquants des dernières 24 heures

Incidents Majeurs

RAS

MEP HNO

RAS

Appels à l'Assistance Managériale & Astreinte N2 et N3

RAS

Figure 3 : Exemple d'un mail envoyé pour Colissimo

2. État de l'art d'un domaine

Le développement web est un domaine en constante évolution, influencé par des avancées technologiques rapides et des changements dans les besoins des utilisateurs. Cet état de l'art vise à fournir une vue d'ensemble des technologies, méthodologies et tendances actuelles dans le domaine du développement web. Il s'agit également de comprendre le web moderne par l'étude de ses évolutions dans le passé et voir les différents dilemmes auxquels les développeurs sont confrontés quotidiennement pour choisir la bonne technologie avant de démarrer un projet.

Le développement web a débuté vers les années 1960, à cette époque, l'origine d'Internet est liée à des projets militaires, notamment avec la création du réseau ARPANET aux États-Unis. Ce réseau financé par le département de la Défense américain visait à relier des ordinateurs pour partager des ressources et des informations. Ce réseau était selon une étude de l'IEEE (Institute of Electrical and Electronics Engineers) le premier réseau à utiliser le protocole de communication TCP/IP, fondement d'internet encore utilisé aujourd'hui, notamment pour sa garantie de livraison des données de bout en bout.

En 1989, Tim Berners-Lee, un chercheur au CERN (Organisation européenne pour la recherche nucléaire) a proposé un système de gestion de l'information révolutionnaire qui permettait aux scientifiques de travailler ensemble malgré la distance. C'est en décembre 1990 que le premier site web a été lancé, il contenait un index de projets et de personnes au CERN, accessible via le tout premier navigateur web nommé WorldWideWeb.

Ces pages web reposent sur des technologies fondamentales : HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), Javascript et HTTP (Hypertext Transfer Protocol). HTML permet la création de pages web structurées tandis que HTTP a facilité leur accès et leur liaison. Le CSS permet de gérer l'apparence (couleurs, polices, mise en page ...) des pages web dans le navigateur. Et le Javascript apporte de l'interactivité aux pages web, permet

de mettre à jour le contenu dynamiquement, d'animer des images ... Ces technologies sont les bases d'Internet, transformant un réseau de communication en une plateforme d'échange mondiale.

L'histoire du web est marquée par une croissance explosive et une évolution constante. En octobre 1994 Tim Berners-Lee fonde le World Wide Web Consortium (W3C), un organisme de standardisation destiné à promouvoir la compatibilité des technologies utilisées sur le web, afin d'obtenir « un seul web partout et pour tous ». Le W3C met à disposition un validateur qui permet de s'assurer que toutes les pages web respectent bien les standards du web.

En 2024, un site web est une nécessité absolue pour toute entreprise souhaitant réussir dans l'économie numérique actuelle. Une présence en ligne forte permet d'augmenter la visibilité et la crédibilité d'une entreprise, d'engager ses clients ou même d'analyser ses performances. Les technologies web permettent l'accès à l'information partout dans le monde, le commerce, le divertissement, les communications instantanées ... C'est pourquoi le web a un rôle crucial dans l'innovation, dans le développement de nouvelles applications et services, dans l'économie et la collaboration mondiale.

Avant 2010, le développement web était largement dominé par le langage PHP. PHP (Hypertext Preprocessor) est un langage de programmation libre créé en 1995 par Rasmus Lerdorf, plus souvent utilisé côté serveur qui a été créé dans le but de permettre la création d'applications dynamiques. PHP conserve encore sa première place en tant que langage de programmation côté serveur sur le web avec près de 79% d'utilisation (voir Figure 4). Cette première place est expliquée notamment par l'utilisation de systèmes de gestion de contenu (CMS) comme Wordpress développé en PHP, qui à lui seul représente 43,5% de tous les sites web.

This report shows the historical trends in the usage of server-side programming languages since January 2010.

	2010 1 Jan	2011 1 Jan	2012 1 Jan	2013 1 Jan	2014 1 Jan	2015 1 Jan	2016 1 Jan	2017 1 Jan	2018 1 Jan	2019 1 Jan	2020 1 Jan	2021 1 Jan	2021 14 Sep
PHP	72.5%	74.8%	76.6%	77.7%	80.3%	80.6%	80.0%	80.0%	80.2%	78.9%	78.9%	79.1%	78.9%
ASP.NET	24.4%	23.2%	21.4%	19.9%	17.8%	16.7%	15.6%	14.8%	13.5%	11.8%	10.6%	9.3%	8.3%
Ruby	0.5%	0.5%	0.6%	0.5%	0.6%	0.9%	1.1%	1.3%	1.6%	2.4%	3.0%	4.3%	5.2%
Java	4.0%	3.8%	3.9%	4.0%	2.6%	2.8%	3.1%	3.3%	3.4%	4.0%	3.7%	3.2%	3.6%
Scala						0.2%	0.2%	0.3%	0.5%	1.2%	1.6%	1.8%	2.0%
JavaScript			<0.1%	<0.1%	0.1%	0.1%	0.2%	0.3%	0.4%	0.7%	0.8%	1.2%	1.5%
static files							1.5%	1.5%	1.6%	2.1%	1.8%	1.6%	1.5%
Python	0.3%	1.0%	1.3%	1.5%	1.7%	1.6%	1.7%	1.6%	1.3%	1.1%	1.3%	1.4%	1.4%
ColdFusion		1.3%	1.2%	1.1%	0.8%	0.7%	0.7%	0.6%	0.6%	0.5%	0.5%	0.3%	0.3%
Perl		1.1%	1.0%	0.8%	0.6%	0.5%	0.5%	0.4%	0.3%	0.3%	0.2%	0.2%	0.1%
Erlang						0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
Miva Script							0.1%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%

Figure 4 : Report d'utilisation des langages de programmation côté serveur depuis 2010

Cependant, depuis 2010, de nombreuses nouvelles technologies ont vu le jour dans le domaine du développement web côté serveur. Des frameworks comme Django écrit en Python, Ruby on Rails écrit en Ruby ou bien Express basé sur Node.js. Ces frameworks que l'on peut traduire par « cadre de travail » sont très utiles pour les développeurs car ils permettent de créer une structure pour le développement qui est à la fois optimisée, performante et durable dans le temps. Des frameworks PHP ont aussi été créés comme Symfony sorti en octobre 2005 et Laravel sorti en juin 2011. Ces deux frameworks sont basés sur l'architecture MVC (Modèle Vue Contrôleur). Cette architecture très populaire pour les applications web comporte 3 types de modules, un modèle qui contient les données à afficher, une vue qui contient l'interface graphique et un contrôleur qui contient la logique des actions de l'utilisateur.

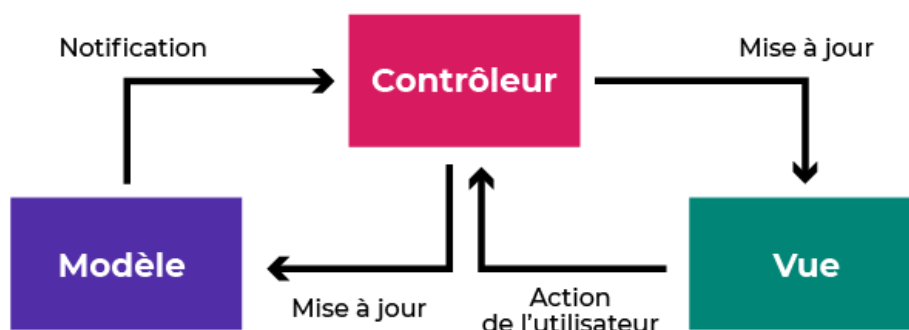


Figure 5 : Schéma de l'architecture MVC

Contrairement à l'architecture MVC, REST (Representational State Transfer) est un type d'architecture d'API apparu dans les années 2000 inventé par Roy Fielding. REST a été conçu pour répondre aux besoins spécifiques du web, il est basé sur le standard URI (Uniform Resource Identifier) et repose également sur le protocole HTTP, les différentes contraintes imposées par cette architecture en font des applications de meilleure qualité. L'API REST est également performante car beaucoup de clients peuvent se connecter sur un serveur à la fois. Les API REST permettent aussi de séparer notre application en deux, la partie frontend et la partie backend. Ces deux parties sont donc maintenant deux applications web indépendantes, le frontend correspond à l'interface utilisateur graphique que l'utilisateur voit. Elle communique avec le backend via des requêtes HTTP, tandis que le backend communique directement avec la base de données afin de traiter la requête et renvoyer une réponse.

De nombreuses alternatives pour le développement rapide et flexible d'applications web ont vu le jour depuis 2010 dont notamment Node.js. Node.js n'est ni un framework, ni un langage de programmation, c'est un outil très populaire qui permet d'exécuter du Javascript en dehors du navigateur. Node.js a révolutionné le développement côté serveur avec son approche asynchrone qui permet d'être plus rapide. L'arrivée de la syntaxe ES6 (normes ECMAScript) en 2015 a considérablement boosté sa popularité notamment grâce aux Promises qui ont pu mettre fin au « callback hell » dans le code, un problème qui se produisait lorsque plusieurs appels sont imbriqués dans une fonction, ce qui rendait le code difficile à comprendre et à maintenir. Pour résumer, Node.js est le parfait alliage entre rapidité, performance et stabilité.

De nombreux frameworks Node.js se sont créés avec le temps dont notamment Express en 2010 et plus récemment NestJS lancé en 2017, propulsé par la société Vercel. Ces deux frameworks sont très populaires, étant respectivement à la sixième et quatorzième place des frameworks Node les plus utilisés. Quant à Node, environ quatre pour cent de tous les sites web l'utilisent. Express est un framework qui prône le minimalisme et la flexibilité, notamment apprécié par les développeurs car il n'embarque pas de dépendance superflue. Contrairement à NestJS, qui lui est un framework avec

une forte opiniâtreté, c'est-à-dire que le code est moins flexible, avec plus de contraintes et des règles prédéfinies par le framework. Il utilise également TypeScript par défaut, un sur-ensemble syntaxique strict de Javascript sorti en 2012 qui ajoute principalement le typage et une programmation orientée objet offrant une réduction des risques d'erreurs lors de l'exécution du code. Ces dernières années, TypeScript est de plus en plus adopté par les développeurs à la place de Javascript (voir Figure 6), c'est pourquoi de plus en plus de framework dont Express et NestJS encouragent son utilisation.

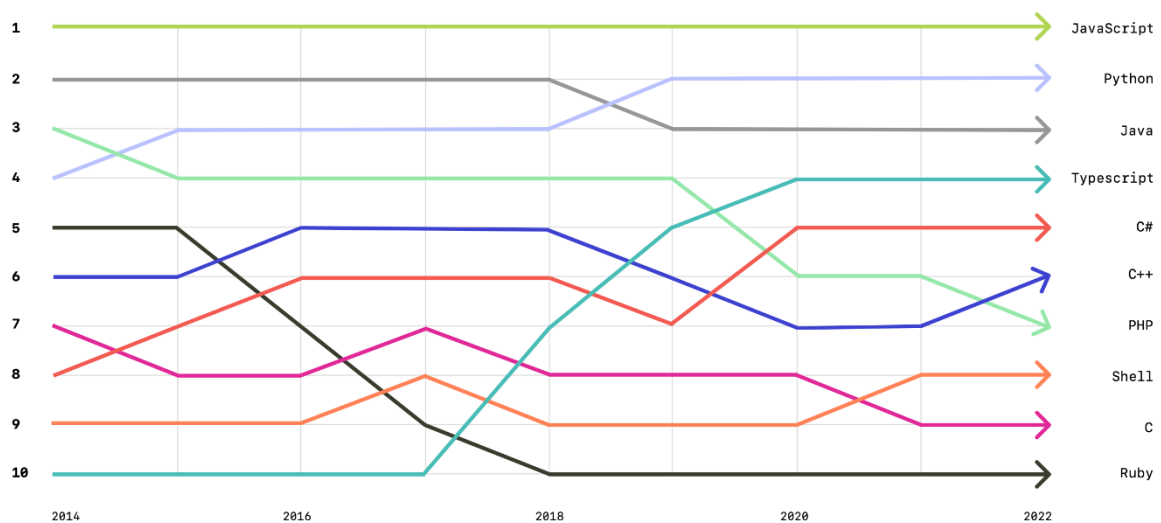


Figure 6 : Graphique de l'utilisation des langages de programmation de 2014 à 2022

C'est pourquoi nous avons choisi d'utiliser NestJS comme framework pour le backend de mon projet. Privilégiant un cadre de travail plus strict qui met en avant les bonnes pratiques actuelles de programmation. Ensuite, le projet sera sûrement amené à l'avenir à être partagé à d'autres développeurs pour continuer de maintenir et d'améliorer le projet, et un cadre de travail strict est préférable lorsque plusieurs développeurs travaillent sur le même projet. Enfin, contrairement à Express, Nest intègre un environnement de tests par défaut, à chaque intercepteur ou contrôleur créé, un fichier de spécification associé est créé.

Enfin, ces dernières années, la complexité et les demandes de fonctionnalités dans le web ont beaucoup évolué. Cela a conduit à la naissance d'un nouveau type d'application web : les SPA (Single Page Application). Une SPA est une

application monopage qui exécute principalement l'interface utilisateur dans le navigateur et qui communique avec le serveur via des services web. Ce qui a permis de fluidifier l'expérience utilisateur en évitant de charger une nouvelle page à chaque action de l'utilisateur, et permet également de s'exécuter localement sur le navigateur.

Les SPA ont rendu l'utilisation de Javascript Vanilla compliqué dû à la complexité de certaines applications. C'est pourquoi le marché a vu naître de nouveaux frameworks modernes pour le frontend maintenant, encourageant un code plus structuré, plus maintenable dans le temps et moins sujet aux erreurs. Trois technologies se distinguent et dominent aujourd'hui le marché principalement, Angular sorti en 2010 et développé par Google, React sorti en 2013 et développé par Facebook et le dernier Vue.js sorti en 2014 par Evan You, un ancien ingénieur de Google. Angular a été réécrit en TypeScript en 2014, il propose une architecture orientée composant basée sur un modèle MVVM (Model View Viewmodel, voir figure 7).

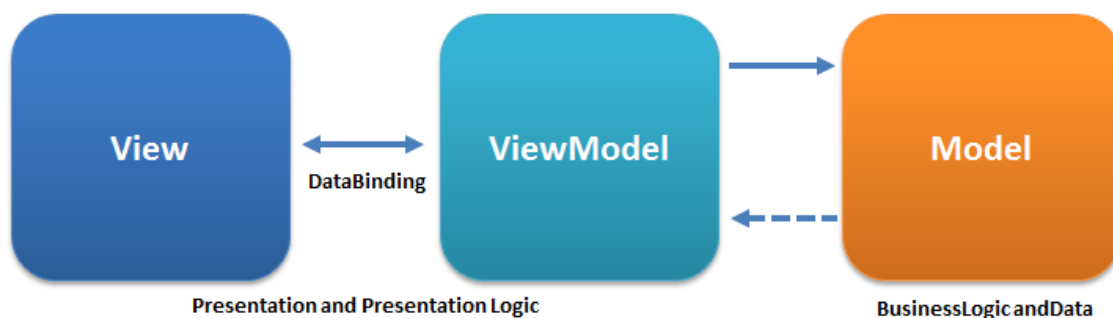


Figure 7 : schéma de l'architecture MVVM

React est comme Angular orienté composant mais il s'apparente plus à une bibliothèque qu'à un framework et n'adresse que la vue du MVVM. Comme React, Vue.js propose une approche modulaire de type bibliothèque orientée composant. Contrairement à React et Angular, Vue.js est indépendant et développé par la communauté.

Nous avons donc choisi React comme technologie pour le backend de notre application, tout d'abord car React est déjà utilisé par l'équipe de développement dans les nouvelles fonctionnalités comme le portail du centre des services. Ce qui veut dire que l'équipe a déjà passé du temps à apprendre

cette technologie, c'était donc le choix idéal. Afin de faire une application plus optimisée, plus performante, avec un système de routage basé sur le système de fichiers, le rendu côté serveur (Server Side Rendering ou SSR), une optimisation SEO ainsi que l'optimisation d'images. Nous avons utilisé le framework Next.js qui contient tous ses avantages par rapport à React. Next.js est un framework React open source sorti en 2016 et développé par Vercel. De plus, la documentation de Next.js est bien réalisée et pratique pour les développeurs qui veulent apprendre cette technologie, car j'ai par exemple pu faire au début de mon stage un tutoriel assez complet disponible sur la documentation officielle du framework.

Pour conclure, il est important en tant que développeur d'étudier chaque technologie pour savoir lesquelles utiliser avant de démarrer le développement d'un nouveau projet. En prenant compte des avantages et inconvénients de chacun, mais également des compétences et des technologies déjà utilisées dans l'entreprise. Il n'existe pas de technologie parfaite, chaque technologie à ses propres caractéristiques.

3. Mon projet

a. Cahier des charges initial

Le cahier des charges initial était donc de démarrer un nouveau projet de zéro dont le but était de réaliser une application web permettant d'administrer différents tableaux de bords appelé « Flash Prod ». Cela permet de faciliter le travail, d'automatiser ce processus afin de faire gagner du temps aux personnes dont le travail est d'indiquer l'état des services informatiques. Car pour rappel, avant mon arrivée, un rapport écrit à la main était envoyé par mail résumant l'état de tous les services, avec des commentaires sur les faits marquants des dernières vingt-quatre heures et un commentaire sur l'appel à l'assistance (voir Figure 3, page 8).

Ce projet impliquait donc des compétences full-stack puisque j'ai commencé par réaliser le design de l'application finale qui comprenait une page d'administration de tous les dashboards, une page d'ajout d'un nouveau dashboard et la page d'un seul dashboard dans laquelle on peut indiquer l'état des services et ajouter un ou des commentaires. J'ai ensuite dû démarrer le développement technique du projet, ce qui implique la création de deux projets : le frontend réalisé à l'aide de Next.js ainsi que le backend réalisé à l'aide de NestJS. J'ai donc créé une API dans le backend pour que le frontend puisse interagir avec le backend et donc avec la base de données. J'ai également effectué des tests unitaires côté backend premièrement et puis des tests unitaires pour tester le fonctionnement des composants du frontend.

b. Outils et méthodologies

Concernant les outils et méthodes de travail utilisés pendant mon stage, j'ai travaillé sur un environnement Windows 11 professionnel. Concernant les logiciels et l'environnement de travail, j'ai effectué les designs sur Figma, mon IDE était Visual Studio Code, je travaillais sur une machine virtuelle Ubuntu utilisant Docker. Docker est un projet open-source distribué à partir de mars 2013, il permet notamment de gérer des conteneurs isolés pour exécuter des applications. En entreprise, Docker est précieux car il facilite la gestion des environnements de développement, assure la

cohérence entre les différentes étapes de la chaîne de production et permet de déployer des applications rapidement et efficacement tout en réduisant le risque de conflits entre les environnements. On m'a conseillé de ne pas utiliser d'intelligence artificielle quelconque, que ce soit ChatGPT ou Copilot pour m'aider dans le développement de l'application afin de mieux comprendre les concepts et logiques de programmation que j'ai appris au cours de ces trois mois de stage. Enfin, au sein d'Alturing, les développeurs utilisent Gitlab comme logiciel de versionning facilitant la collaboration en équipe. Dans mon cas, je n'ai pas pu exploiter tous les bénéfices d'un logiciel de versioning comme Gitlab vu que j'ai essentiellement travaillé seul dans ma branche et sur ce projet au global.

c. Organisation

J'ai réalisé le diagramme suivant (voir Figure 8) afin de pouvoir visualiser dans le temps les différentes tâches que j'ai pu faire durant mon stage. On peut remarquer que la création de l'API avec les tests unitaires associés (en rouge) ainsi que l'intégration de la maquette sont les deux tâches qui m'ont occupé le plus de temps. En effet, le projet repose principalement sur l'API et ce sont notamment les tests unitaires associés aux ressources qui ont pris la majorité de mon temps. Ces ressources étaient chacune composées d'un module, un service, un contrôleur, une entité et un DTO (Data Transfer Object).

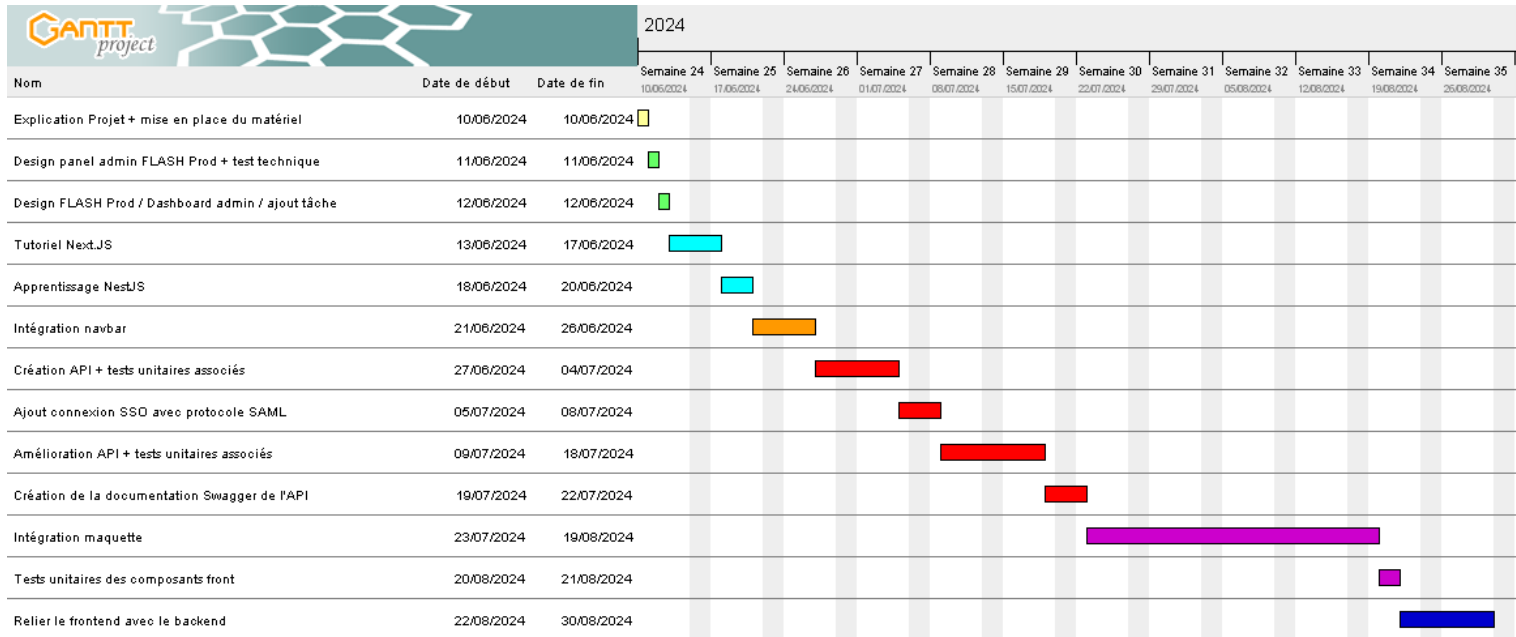


Figure 8 : Diagramme de Gantt répertoriant toutes les tâches effectuées

Ensuite, comme je l'ai dit auparavant, j'étais majoritairement le seul développeur à travailler sur ce projet. Je n'ai donc pas eu à utiliser de méthodologies agiles particulières ou des logiciels de gestion de projet comme Trello par exemple. Pour m'organiser dans mon travail, j'échangeais souvent avec mon tuteur qui me donnait les tâches à faire. Nous avons également communiqué avec Google Chat et fait des visioconférences sur Google Meet lorsqu'il travaillait en télétravail afin d'échanger à propos de mon travail réalisé et de la suite des tâches à faire. J'ai su mener mes tâches et m'organiser dans mon travail avec une certaine autonomie, en prenant des initiatives et en résolvant les défis techniques de manière indépendante. A l'exception d'une tâche assez complexe qui reste à réaliser par l'intervention de mon tuteur, il s'agissait de l'implémentation d'une connexion SSO⁴ en utilisant le protocole SAML⁵ (voir Figure 9).

4 Glossaire: SSO (Single Sign-On)

5 Glossaire: SAML (Security Assertion Markup Language)

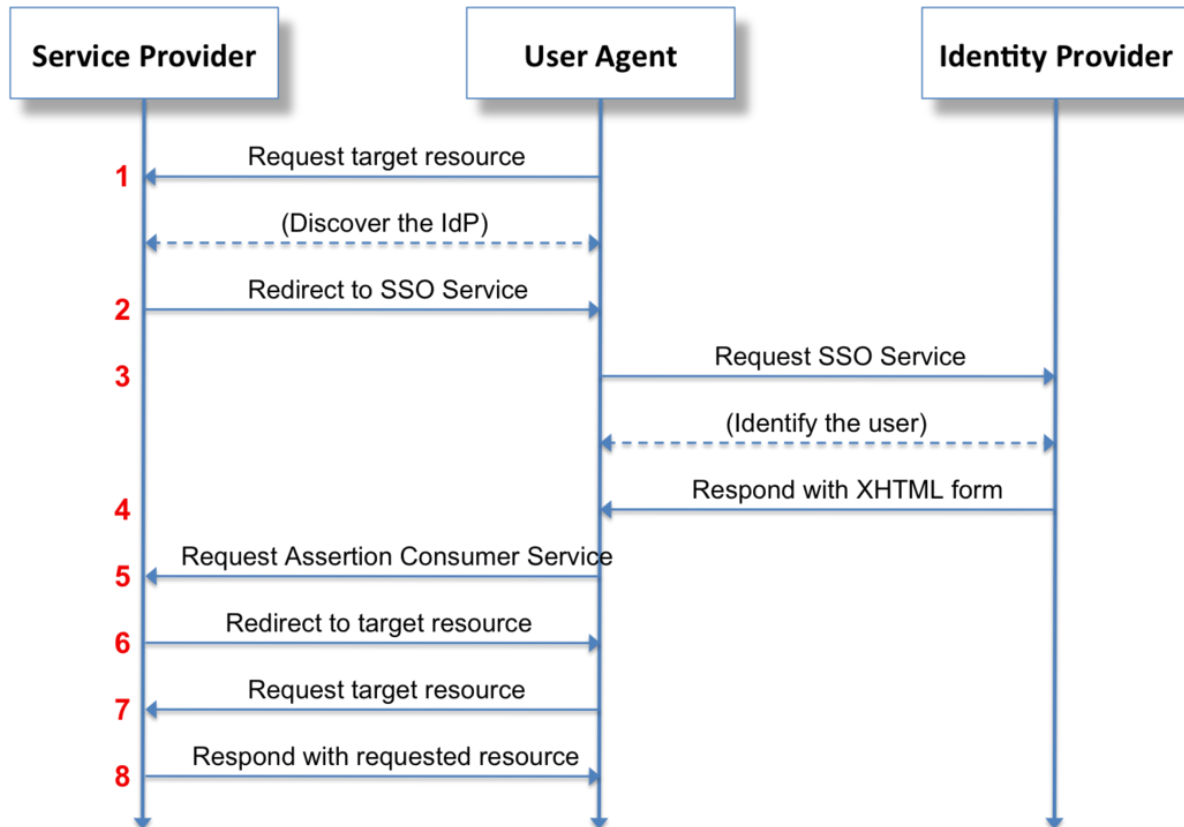


Figure 9 : Explication du protocole SAML avec une connexion SSO

4. Détail de mes réalisations

a. Architecture du projet

Tout d'abord, nous avons fait le choix de choisir une architecture mono-repo⁶ (voir Figure 10) pour structurer notre application. Cette approche mono-repo utilise un dépôt unique pour héberger tout le code des multiples bibliothèques ou services. Cette approche comporte de nombreux avantages comme abaisser les barrières à l'entrée, c'est-à-dire que lorsqu'un nouveau développeur commence à travailler sur un projet, il doit télécharger tout le code et installer les outils nécessaires pour commencer à travailler. Si le projet est dispersé dans de nombreux référentiels ayant chacun son instruction d'installation et ses outils requis, la configuration initiale peut vite devenir complexe. C'est pourquoi le mono-repo est une solution qui simplifie les choses avec un seul emplacement contenant tout le code et la documentation associée. Ensuite, le mono-repo permet une gestion de code centralisée et cela nous donne une visibilité de tout le code à tous les développeurs, ce qui simplifie la gestion du code car un seul outil de suivi des problèmes est nécessaire pour surveiller tous les problèmes tout le long du cycle de vie de l'application. Enfin, cela permet une refactorisation⁷ plus facile du code car l'application comporte toutes les bibliothèques dans la même version.

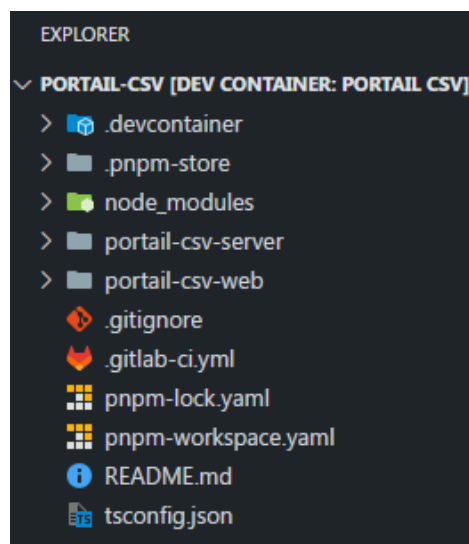


Figure 10 : Structure mono-repo de l'application

6 Glossaire : Repository

7 Glossaire : Refactorisation

Voici un schéma qui résume la différence entre un projet organisé avec une architecture mono-repo et à l'inverse, une architecture poly-repo.

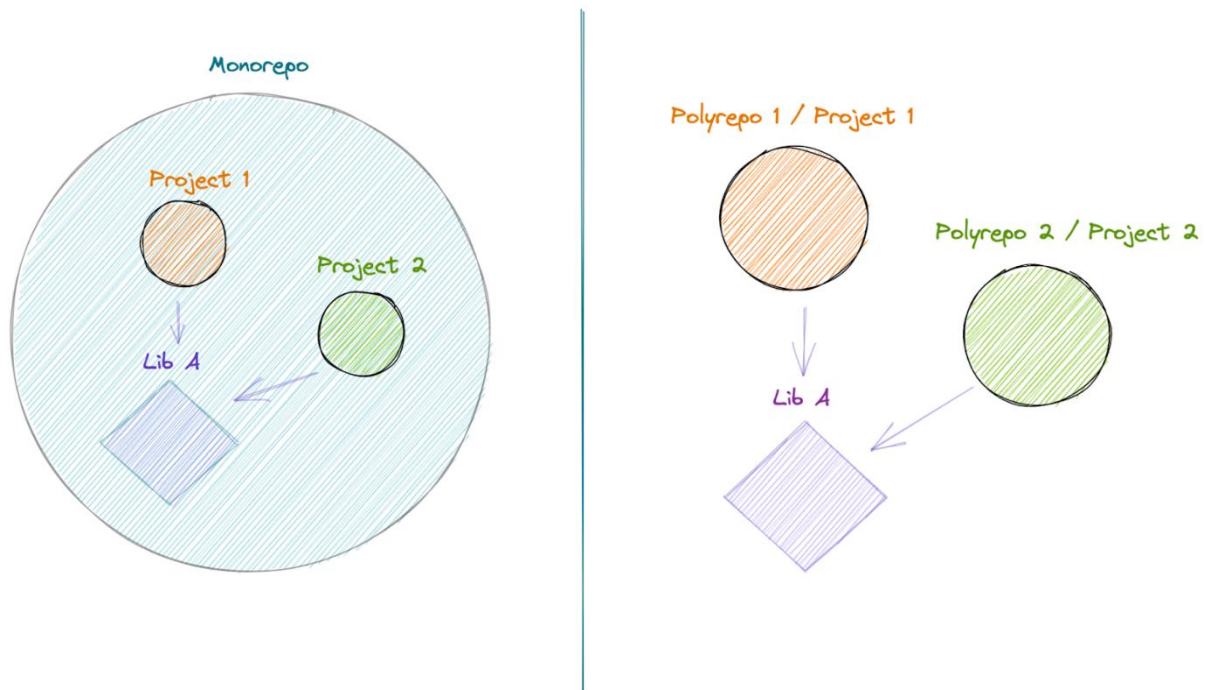


Figure 11 : Schéma explicatif mon-repo / poly-repo

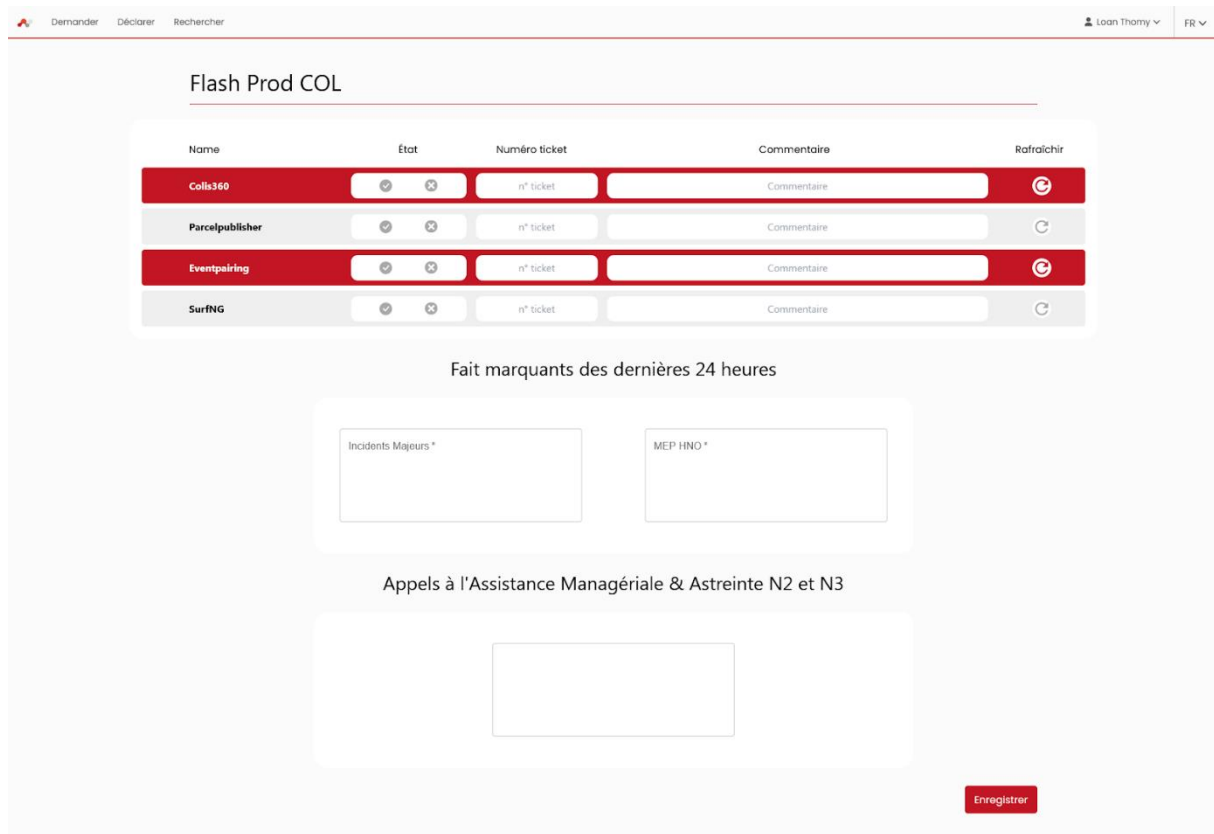
b. Design de l'application

Ensuite, on m'a confié la tâche de réaliser le design de l'application, j'ai commencé par désigner la page principale permettant de renseigner l'état des services d'un dashboard, ici ça sera le dashboard de Colissimo par exemple.

Voici les seules consignes que l'on m'avait données pour cette page était :

- Lister des tests sur une page
- Avoir un accès simple à la liste des tests
- Sélection du statut du test rapide (OK / Perturbation / Interruption)
- Pour chaque test, ajout facultatif d'un commentaire
- Pour chaque test, ajout facultatif d'un numéro de ticket

J'avais commencé à faire une première version de cette page, mais elle a vite été remplacée car le design n'était pas très compact, moins intuitif et nécessitait trop de clics. C'est toujours mieux de réduire au plus le nombre de clics nécessaire pour faire une action afin de simplifier et d'optimiser tous les processus.



The image shows a wireframe of a dashboard titled "Flash Prod COL". At the top, there is a navigation bar with links "Demander", "Déclarer", and "Rechercher". On the right, it shows a user profile "Loan Thomy" and a language selector "FR".

The main content area features a table with the following columns: "Name", "État", "Numéro ticket", "Commentaire", and "Rafraîchir". The table contains four rows:

Name	État	Numéro ticket	Commentaire	Rafraîchir
Colis360	<input checked="" type="checkbox"/> <input type="checkbox"/>	n° ticket	Commentaire	<input type="button" value="Refresh"/>
Parcelpublisher	<input checked="" type="checkbox"/> <input type="checkbox"/>	n° ticket	Commentaire	<input type="button" value="Refresh"/>
Eventpairing	<input checked="" type="checkbox"/> <input type="checkbox"/>	n° ticket	Commentaire	<input type="button" value="Refresh"/>
SurfING	<input checked="" type="checkbox"/> <input type="checkbox"/>	n° ticket	Commentaire	<input type="button" value="Refresh"/>

Below the table, there is a section titled "Fait marquants des dernières 24 heures" containing two input fields: "Incidents Majeurs *" and "MEP HNO *".

Underneath, there is a section titled "Appels à l'Assistance Managériale & Astreinte N2 et N3" with a large empty rectangular box.

At the bottom right, there is a red button labeled "Enregistrer".

Figure 12 : Maquette du dashboard Colissimo

Ensuite j'ai réalisé le design de la page de gestion de tous les dashboards qui permet à l'utilisateur de choisir un dashboard à administrer, de modifier ou supprimer un dashboard et également de créer un nouveau dashboard après un clic sur un bouton.

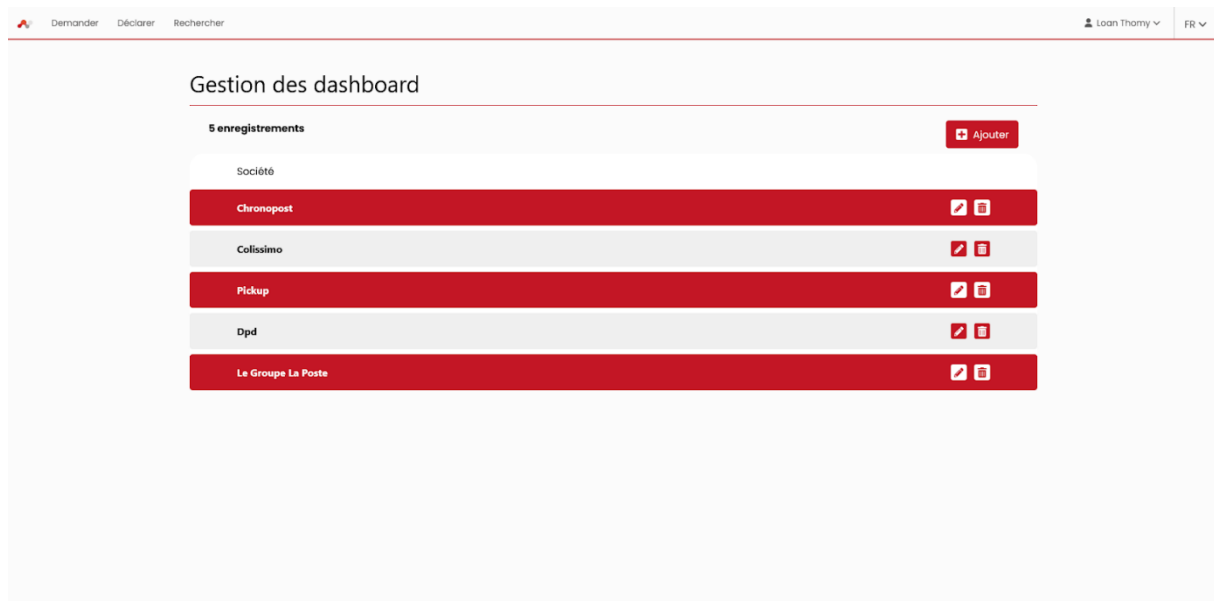


Figure 13 : Maquette de la page de gestion des dashboards

Enfin, j'ai désigné la page d'ajout d'un nouveau dashboard, cette fois-ci avec des couleurs plus sobres, moins de rouge notamment.

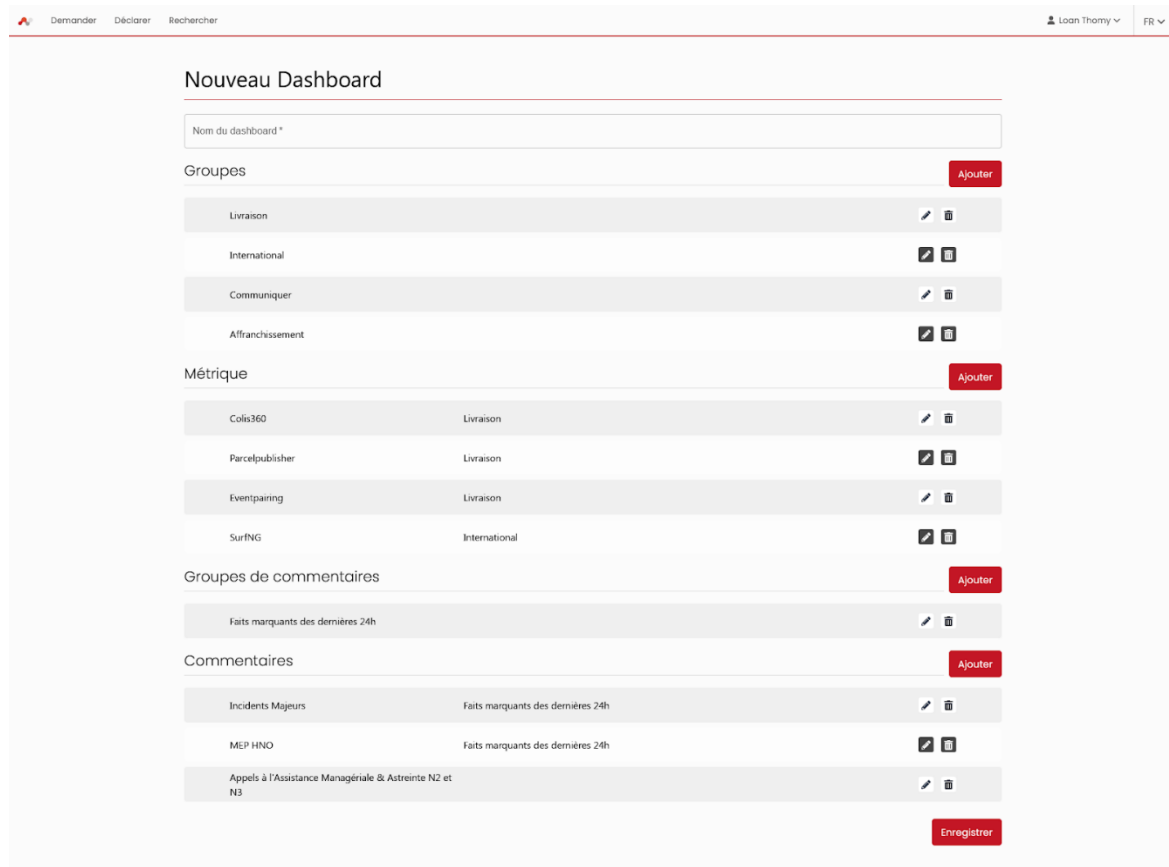


Figure 14 : Maquette de la page d'ajout d'un nouveau dashboard

Afin d'ajouter la possibilité d'ajouter un groupe, une métrique, un groupe de commentaires ou un commentaire, j'ai fait le choix de désigner un pop-up pour chaque action déclenchée au clic du bouton respectif.

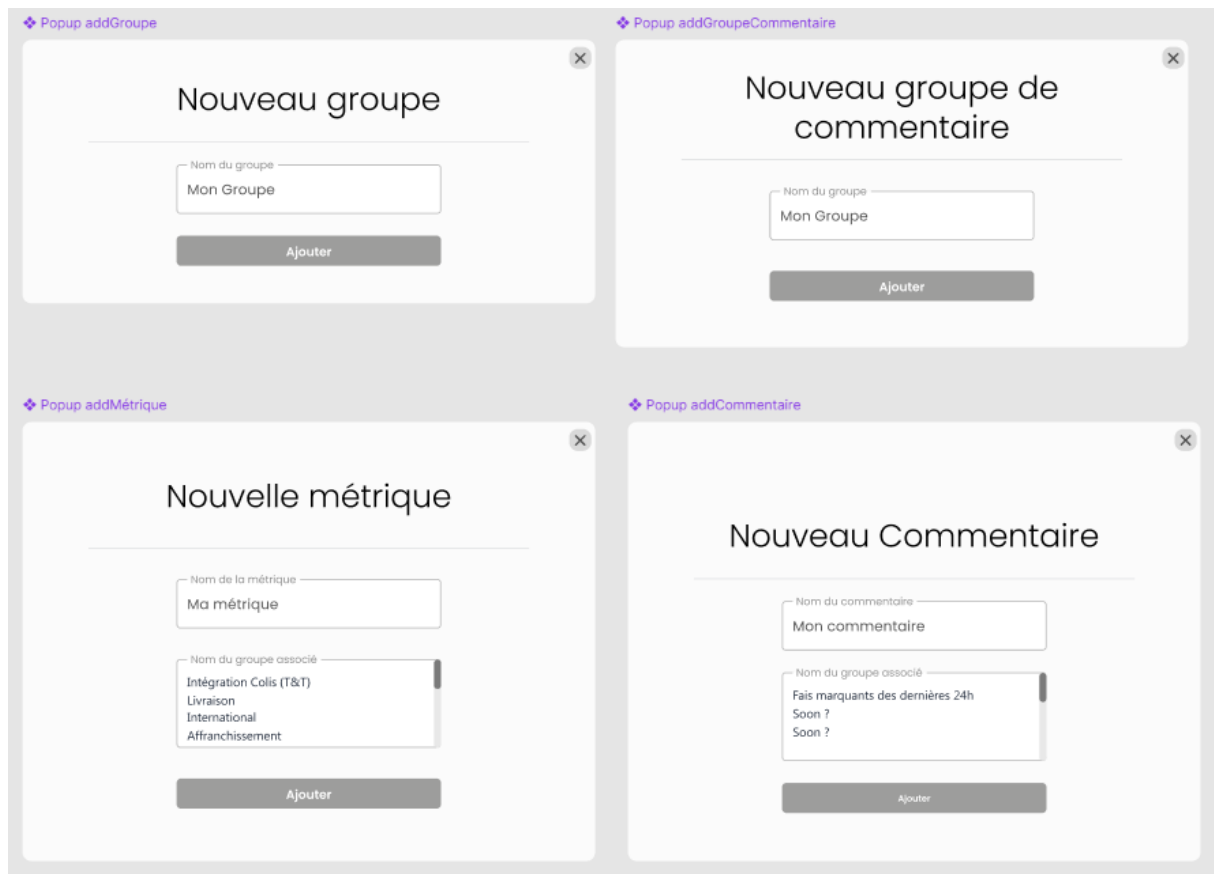


Figure 15 : Maquette des différents pop-ups de la page d'ajout d'un dashboard

On peut remarquer que j'ai essayé de respecter une certaine cohérence dans le design de ces trois pages, que ce soit dans les couleurs (le rouge rappelle le logo d'Alturing) les boutons ou les icônes qui sont identiques. Les polices de caractères présentes dans le design sont les mêmes que d'anciens projets d'Alturing, ce sont, Poppins disponible sur Google Font ainsi que Segoe UI utilisé notamment par Microsoft. Enfin, j'ai utilisé la librairie d'icônes gratuite et open-source Font Awesome pour le design de certains boutons comme le pinceau de modification, la poubelle de suppression et le plus du bouton ajouter.

c. La création de l'API

Après avoir réalisé et validé le design par Martial, nous avons d'abord commencé à deux par créer le mono-repo contenant le front-end en Next.JS et le back-end en NestJS. Ensuite, Martial a commencé la création de l'API incluant la création de ressources composées d'un module, un service, un contrôleur, une entité et un DTO.

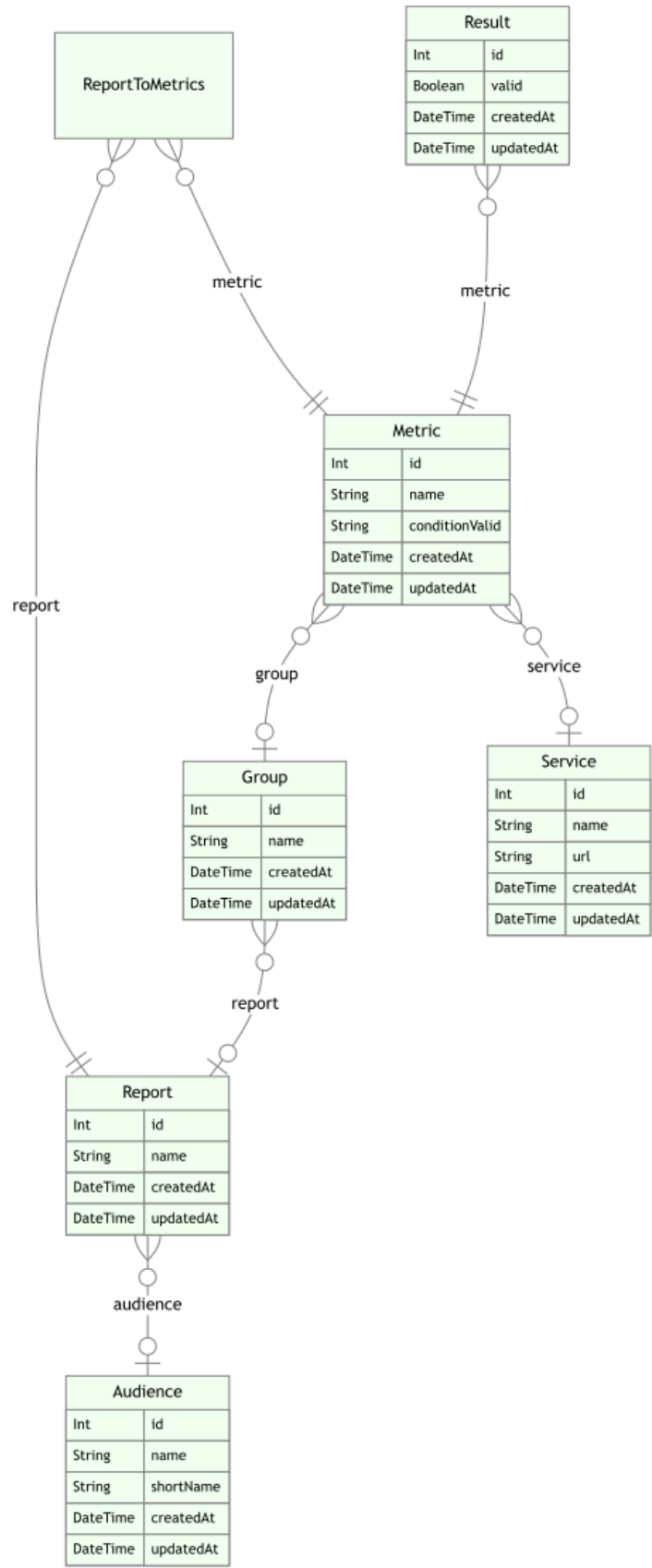
Afin d'interagir avec la base de données, nous avons utilisé Prisma, c'est un ORM (Mapping Objet-Relationnel) qui permet de faire correspondre la base de données avec des classes Javascript. Un ORM permet aux développeurs de penser en objets sans avoir à générer des requêtes SQL. C'est également un plus en matière de sécurité puisque Prisma aide à prévenir les injections SQL car les requêtes sont générées automatiquement et les données sont souvent échappées automatiquement.



Figure 16 : Schéma du fonctionnement d'un ORM

Voici un schéma qui représente la base de données générée par Prisma, la base de données comporte 7 tables : User, Result, Metric, Group, Service et Audience. Un dashboard est représenté par la table Report, la table Audience permet de savoir qui à accès à ce Dashboard. La table Metric représente les tests, une métrique fait partie d'un groupe. Ensuite, la table Service se charge d'aller tester une métrique et son résultat est stocké dans la table Result.

User	
Int	id
String	email
String	username
String	password
Boolean	active
String	comment
DateTime	createdAt
DateTime	updatedAt
DateTime	tokensGeneratedAt
String	apiKeyPrefix
String	apiKeyHash



Ensuite, Martial m'a également montré la démarche à suivre pour faire un test unitaire associé à la ressource précédemment créée. Pour réaliser nos tests dans ce projet, nous allons utiliser Jest, un framework de test Javascript qui met l'accent sur la simplicité. Il ne nécessite pas de configuration, les tests sont parallélisés en les exécutant dans leurs propres processus afin de maximiser les performances.

Pour faire les tests, j'ai appris le principe de "mock", un mock est un objet simulé qui reproduit le comportement d'objets réels de manière contrôlée. Par exemple, dans les tests unitaires des services d'une ressource, nous devons mocker Prisma car l'objectif de notre test n'est pas de vérifier si Prisma fonctionne correctement. On teste ici si notre fonction nous donne le bon résultat attendu. Voici un exemple d'un test unitaire du service de la ressource User.

```
Run | Debug
describe('findAll', () => {
  Run | Debug
  it('Should return an array of users', async () => {
    // Data passed to service
    const paginatedQueryDto: PaginatedQueryDto = {};

    // Returned by prisma.user.findMany()
    const findAllUsers: PrismaUser[] = usersInDB;

    jest
      .spyOn(prismaService.user, 'findMany')
      .mockResolvedValueOnce(findAllUsers);

    // Expected result from userService.findAll()
    const expected: User[] = [user01, user02];

    const allUsers = await userService.findAll(paginatedQueryDto);

    expect(allUsers).toEqual(expected);

    expect(prismaService.user.findMany).toHaveBeenCalledWith({
      skip: 0,
      take: 50,
      orderBy: { createdAt: 'asc' },
    });
  });
});
```

Figure 17 : Exemple d'un test unitaire du service de la ressource User

La fonction `“jest.spyOn().mockResolvedValueOnce”` permet d’espionner le résultat de la fonction `findMany()` afin de lui mocker le résultat par la variable `findAllUsers`. Ensuite, deux tests sont effectués avec la fonction `expect`, le premier test si le résultat attendu est le bon et le deuxième test si la fonction `findMany` qui est présente dans notre fonction `findAll` a bien été appelée avec les bons paramètres. Ceci est un exemple d’un seul test, j’ai donc effectué cette méthode de test pour tester toutes les fonctions des services et des contrôleurs de toutes mes ressources.

Enfin, après avoir fini l’API ainsi que ses tests unitaires associés, j’ai créé une documentation de l’API avec Swagger. Swagger nous fait gagner beaucoup de temps car il permet directement de créer une documentation claire et interactive de l’API à partir du code de l’application. En entreprise, c’est important de documenter son API car d’autres développeurs seront peut-être amenés à travailler sur ce projet, cette documentation permettra ainsi de comprendre comment l’API fonctionne, ce qui réduit considérablement le temps d’intégration. De plus, Swagger suit la spécification OpenAPI, un standard largement adopté. Cela assure que la documentation est conforme aux meilleures pratiques de l’industrie, ce qui est particulièrement important pour les entreprises qui collaborent avec des partenaires externes ou qui publient leurs API publiquement. Voici à quoi ressemble notre documentation Swagger. Pour chaque ressource, le fonctionnement de chaque requête est expliqué avec des exemples, on peut même tester des requêtes pour voir la réponse de l’API et mieux comprendre son fonctionnement.

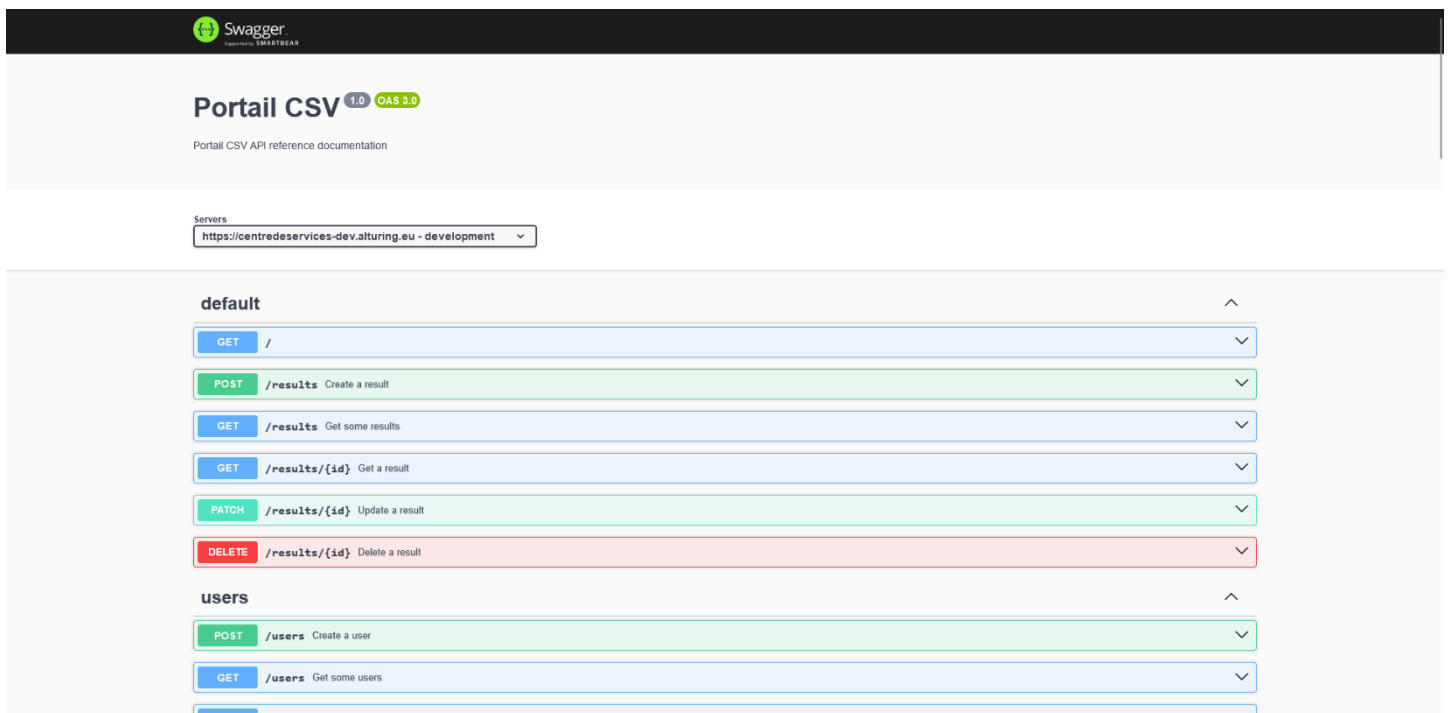


Figure 18 : Documentation de l'API

d. Intégration web

Après avoir terminé la création de l'API, ses tests unitaires associés et la documentation Swagger. J'ai intégré les pages de la maquette Figma avec Next.js, d'abord avec des données en brut. J'ai suivi ces étapes pour chaque page :

- Faire la page en bloc
- Définir tous les états possibles de chaque élément, par exemple, un bouton peut contenir trois états : l'état normal, l'état au survol de la souris et l'état après un clic.
- Découper la page en composant si le code risque d'être réutilisé dans d'autres pages pour éviter la répétition de code.
- Faire les tests unitaires pour chaque page

Les tests unitaires côté frontend ont également été faits avec le framework Jest. Pour tester des composants React, il est nécessaire d'utiliser une bibliothèque qui va générer un DOM virtuel du composant en question

et permettre de faire des assertions avec Jest sur ce qui est présent ou non dans ce DOM. Les deux bibliothèques les plus utilisées sont Enzyme et Testing-library, sauf que Enzyme n'est plus vraiment maintenue et n'est plus utilisable dans la dernière version de React. C'est pourquoi j'ai choisi d'utiliser Testing-library, cette bibliothèque plus récente a été développée directement par les développeurs de React et est indiquée comme la bibliothèque officielle à utiliser.

Testing-library est composée principalement de deux fonctions très utiles :

- Render : c'est une fonction qui permet de générer le DOM virtuel à partir du composant React
- Screen : c'est une fonction permettant de scanner le contenu du DOM virtuel généré pour chercher un élément. Elle possède plusieurs méthodes pour trouver des éléments du DOM comme `screen.getByText()`, `screen.getByRole` ou `screen.getByTestId()` qui sont les plus utilisées.

5. Ensuite ?

Enfin, nous allons voir ce que le projet va devenir après mon départ chez Alturing le 30 août et ce qu'il reste à faire pour le terminer. En effet, le projet ne sera pas complètement fini, c'est Martial qui le terminera. Premièrement, peut-être que l'API ne sera pas complète et il restera quelques améliorations à faire. Ensuite, la plus grande partie du travail à terminer est la liaison entre le frontend et le backend, que j'aurais peut-être commencée mais pas finie. Il restera également quelques petites modifications pour apporter une meilleure cohérence à l'expérience utilisateur. Par exemple, dans mon design du dashboard Colissimo, les lignes du tableau n'ont pas la même couleur (rouge et gris) alors qu'elles ont le même niveau d'importance. Peut-être qu'un utilisateur pourrait croire qu'une ligne grise signifie que la métrique est désactivée. Martial m'a également fait la remarque que les textes sont parfois légèrement trop petits et que chaque bouton qui a la même fonction est censé être de la même couleur, contrairement à mes boutons modifier et supprimer dans la page d'ajout d'un nouveau dashboard et dans la page de gestion des dashboard qui alternent de couleur selon la ligne, ce qui n'est pas cohérent. Quelques petites optimisations de code notamment dans la partie frontend seront nécessaires, comme par exemple, le remplacement de States par des objets afin de ne pas re-render le composant. Pour simplifier la première version de l'application, Martial m'avait dit de ne mettre que deux états (en marche et interrompu) pour les métriques mais plus tard, il faudra rajouter un troisième état « perturbation », ce qui implique une mise à jour de l'API et du frontend.

Une fois le projet terminé et mis en production, il sera utilisé quotidiennement par des employés de l'équipe H24 pour automatiser l'envoi du mail récapitulatif de l'état des services des FLASH Prod.

6. Conclusion

Pour conclure, je suis satisfait d'avoir pu faire mon stage chez Alturing car j'ai enrichi mes connaissances sur toutes les tâches qu'un développeur web full-stack, cela passe par du design, du développement backend et frontend dans des technologies récentes que je n'avais encore jamais utilisées (Next.JS et NestJS). J'ai eu l'opportunité de démarrer un projet de zéro, ce qui m'a permis de voir toutes les étapes de création d'un projet en entreprise. J'ai également appris à réaliser des tests unitaires avec Jest, j'ai appris à développer en TypeScript, à réaliser une documentation d'API avec Swagger.

Je suis fier de savoir que mon travail va être utile à Alturing, répondant à un réel besoin. Je remercie Martial pour m'avoir transmis son savoir sur le plan technique et également de m'avoir laissé autonome dans mon travail. Mon contrat se termine le 30 août et malheureusement Alturing n'a pas pour projet d'engager un nouveau développeur. L'année prochaine, je compte donc rechercher un poste de développeur web soit en tant que développeur full-stack ou spécialisé frontend ou backend selon les propositions que l'on me fera à l'avenir.

Glossaire

- Cellule de crise : Lors d'un incident critique, une équipe se réunit dans une conférence en ligne contenant les responsables des SI et des différentes équipes
- GLPI (Gestionnaire Libre de Parc Informatique) : Il s'agit d'un outil libre permettant d'assurer une gestion des services informatiques (inventaire des applications, serveurs, comptes, etc...) et une gestion des services d'assistance (création et attribution de tickets pour des demandes de services ou de changement sur le parc informatique).
- SSO (Single Sign-On) : C'est un service d'authentification de session et d'utilisateur qui permet à un utilisateur d'utiliser un ensemble d'informations d'identification (par exemple, nom et mot de passe) pour accéder à plusieurs applications.
- Le protocole SAML (Security Assertion Markup Language) : Il s'agit d'une norme XML qui permet aux domaines Web sécurisés d'échanger des données d'authentification et d'autorisation d'utilisateurs.