

ESN



Projet XPérienCe

{Suivi du parcours des collaborateurs}

*Période d'application en entreprise dans le cadre du RNCP
31678 de niveau 6*

Concepteur développeur d'application

ESRP AMIO 2isa Millau

PATRICK NARDI

CDA07 | Stage

Du 19 février

au

11 mai 2024



Remerciements

Lors de cette période d'application en entreprise, j'ai rencontré des magnifiques personnes, conviviales, et professionnelles.

Je remercie pour l'ensemble des personnes que j'ai rencontré, et avec qui j'ai travaillé lors de ce stage. Tout particulièrement :

- Nicolas Turc, pour m'avoir permis d'intégrer l'équipe de Ntico pendant trois mois suite à notre rencontre au Job Stick de Montpellier le 17 octobre 2023
- Aurélien F., le chef de projet, pour son partage de compétence et sa disponibilité
- Guillaume L., mon référent full stack, pour ces conseils et les moments passés à discuter du métier de développeur,
- Vincent P., Pour son temps à m'expliquer le framework Spring Boot et les différentes bibliothèques du projet. Je le remercie aussi pour tous ces bons moments passés ensemble,
- Julien H. Pour ses conseils et son accueil,
- Achille G. Pour le temps passé à m'expliquer les principes de VueJS,
- L'équipe Opération, Anna D, Gilles A., Elodie R, Sébastien G, avec qui j'ai passé un mois formidable avec ce voyage à Lille mémorable
- Et bien entendu toute l'équipe des ressources humaines, qui ont permis mon intégration au sein de Ntico, en portant une forte attention à ma condition de personne en situation de handicap.

A toutes les équipes de Ntico, je voudrais dire un grand merci pour ce moment.

Table des matières

1	Présentation de l'entreprise	6
2	Résumé de la période d'application en entreprise	7
3	De la formation CDA à la période d'application en entreprise	9
4	Liste des compétences du référentiel couverts par le projet	10
5	Expression de la demande pour l'espace d'administration du projet XPérienCe	11
6	Analyse du besoin, et planification	13
6.1	Cas d'utilisation	13
6.2	Planification et méthode de travail	15
6.2.1	Générale	15
6.2.2	Méthode Scrum dans le projet XPérienCe	15
7	Spécifications techniques	17
7.1	Spécification de conception du client web	17
7.2	Maquettage de la solution	18
7.2.1	Vue d'ensemble du projet sous Figma	18
7.2.2	Vue d'accueil du projet XPérienCe	19
7.2.3	Vue de gestion des groupes d'action	19
7.2.4	Vue de gestion d'un groupe d'action	20
7.2.5	Vue de gestion d'une action	22
7.2.6	Popup de prévisualisation	23
7.3	Spécification architecturale	24
7.4	Spécifications techniques du client	25
7.5	Spécifications techniques du serveur	26
8	Développement de la solution	27
8.1	Définition du contrat en client et serveur	27
8.2	Partie administration du Client	29
8.2.1	Remarques sur VueJS et le framework Quasar	29
8.2.2	Développement du Client web	30
8.2.2.1	Modification du fichier de routage, route.js	30
8.2.2.2	Vue d'accueil du projet XPérienCe	31
8.2.2.3	Vue de gestion des groupes d'action,	32
8.2.2.4	Vue de gestion d'un groupe d'action	36
8.3	Partie administration du Serveur	43

8.3.1	Remarques sur l'abstraction de haut niveau avec Spring Boot et les bibliothèques du projet.....	43
8.3.2	Développement de la partie administration du Serveur	48
8.3.2.1	Contrôleurs ajoutés à l'API REST	48
8.3.2.2	Méthodes de Service ajoutés à l'API REST	49
9	Description d'une situation de travail ayant nécessité une recherche	54
10	Conclusion.....	55

1 Présentation de l'entreprise



Le groupe Ntico¹, est une Entreprise de Services Numériques, prestataire chez de grand groupe français des secteurs secondaire et tertiaire. Elle a vu le jour en mars 2010 par suite de la volonté de ses fondateurs de créer une entreprise à dimension humaine, offrant à ses clients une double expertise métier et technique, dans un esprit de collaboration mutuellement bénéfique. Elle est composée d'environ 140 employés, travaillant avec plus de 40 entreprises partenaires, avec un chiffre d'affaires lors du dernier exercice de 12,5 M€. L'ESN Ntico a déployé son expertise sur le territoire français sur trois sites, le première dans les Hauts-de-France à Lille, le second dans le Centre-Val de Loire à Olivet, et le dernier en Occitanie à Montpellier, lieu de ma période d'application en entreprise. Cette dernière unité en croissance est composée à l'heure actuelle de 12 personnes réparties dans trois entités sur quatre du groupe. Ntico est composé de quatre entités :

- Ntico Services service, spécialisé dans le conseil numérique, la Data et le développement logiciel. Elle répond aux besoins d'analyses et de développements de projets digitaux, utilisant un panel de technologies diversifiées : Oracle, PostgreSQL, MySQL, MongoDB, Java Spring Boot, Python, Angular, VueJS, ReactJS, Quasar
- Ntico Opération, spécialisé dans l'ordonnancement avec le logiciel Opcon de SMA.
- Ntico Logistique, spécialisé dans l'implantations de nouveaux sites industriels, dans l'optimisation de la chaîne logistique et des flux afin d'améliorer les sites existants.
- Ntico Solution, est une entité spécialité dans les solutions digitales dédiées aux entreprises et aux collectivités. Les experts Data Ntico ont développé une plateforme unique et innovante, l'application web Locxia², qui valorise plus de 250 données issues d'images satellites et d'informations publiques permettant de mieux connaître son territoire ou ses clients.

Remarque sur l'ordonnanceurs (Schedule) et OpCon :

Les ordonnanceurs sont apparus dans l'histoire avec le développement de l'industrie et la nécessité d'organiser la production de manière efficace et rationnelle. Aujourd'hui, l'ordonnancement est un métier clé dans de nombreux secteurs d'activité, qui requièrent une adaptation constante aux besoins des clients et aux contraintes du marché

OpCon³ de l'entreprise SMA est un logiciel d'ordonnancement dans le domaine de l'IT. Il permet d'automatiser, de planifier et de synchroniser les tâches informatiques, qu'elles soient répétitives, chronophages ou complexes. Il s'agit d'un outil qui optimise les capacités de production, réduit les coûts et les risques, et garantit les délais de livraison. OpCon est une plateforme d'automatisation et d'orchestration low-code⁴ qui élimine les tâches manuelles et gère les charges de travail. OpCon est un logiciel d'ordonnancement conçu entre autres pour les institutions financières.⁵

1 <https://www.ntico.com/>

2 <https://www.locxia.com/>

3 <https://smatechnologies.com/fr/products-opcon-automation>

4 <https://appmaster.io/fr/blog/low-code-devops-combinaison-puissante>

5 <https://www.next-decision.fr/autres-editeurs/automatisation/sma-technologies-opcon>

2 Résumé de la période d'application en entreprise

Ma période de stage en entreprise a débuté par une immersion au sein du département Opérations, où j'ai acquis une compréhension de l'application de planification OpCon de SMA, entreprise avec laquelle Ntico est devenu leur partenaire exclusif pour la zone Europe francophone en avril dernier. Dans le cadre de ce service, j'ai contribué au projet Fluence, visant à identifier les différents cas d'utilisation de cette future application destinée aux clients de l'ordonnanceur OpCon.

Après trois semaines, j'ai rejoint l'équipe Ntico Service pour travailler sur l'administration de leur application Web XPérince en cours de développement, conçue pour gérer le parcours des collaborateurs. Cette application, via des groupes d'action, facilite l'adoption de bonnes pratiques, que ce soit pour la tenue de réunions, la nomination des variables en Java, ou encore le processus d'intégration des nouveaux employés.

Dans le cadre du projet XPérienCe, mes responsabilités incluaient

- L'analyse du besoin, suivant l'expression de la demande par le chef de projet,
- La conception de la maquette, avec l'application web de maquettage Figma,
- Le développement côté client et le coté serveur de l'application web, avec :
 - Côté client une application à page unique en VueJS 3 avec le framework Quasar 2, consommant une API RESTful (interface de programmation, API ou PAI web, qui respecte les contraintes du style d'architecture REST, Representational State Transfert, d'OpenAPI⁶)
 - Côté back, un serveur développé en Java avec le SDK 17 et le framework Spring Boot 3, assurant la logique métier et le requêtage vers la base de données PostgreSQL.

Dans un premier temps, mon travail comprenait l'analyse des exigences, traduites ensuite en scénarios d'utilisation sous forme de cas d'utilisation. Par la suite, j'ai acquis les compétences nécessaires sur les différentes technologies utilisées dans le projet, suivi de la familiarisation avec les composants déjà développés, tant côté client que serveur, ainsi que la structure de base de données.

Dans un deuxième temps, j'ai axé mes efforts sur la définition des patterns DTO⁷ et des URI⁸ pour établir le contrat entre le client et le serveur pour la partie administration. Avec le développement du côté client qui a suivi les meilleures pratiques de l'entreprise et a été aligné autant que possible sur la méthodologie déjà en place. Pour ce front-end, l'utilisation du framework Quasar a permis une découpe modulaire de l'application, où j'ai combiné des composants existants avec mes propres développements.

Dans un troisième temps, j'ai mis en œuvre la partie serveur en utilisant des méthodes déjà présentes, en les adaptant si nécessaire pour intégrer les règles métiers. Le développement avec Spring Boot a impliqué une abstraction de haut

⁶ <https://swagger.io/specification/>

⁷ Concept utilisé pour transférer des données entre différentes parties d'une application Logicielle

⁸ Uniform Resource Identifier (URI), est une courte chaîne de caractères qui permet d'identifier une ressource sur un réseau

niveau. L'utilisation de la librairie MapStruct, a facilité la transformation des objets de DTO à Entité et vice versa. Mon développement devait respecter les tests unitaires existants.

Mon travail était régulièrement suivi lors de réunions quotidiennes avec l'équipe de développement, ainsi qu'au cours des réunions hebdomadaires de projet. Le suivi des tâches était effectué via un tableau Kanban sous Trello, permettant au responsable de projet de suivre mes avancées.

Après des sessions de revue de code avec les chefs de service et les développeurs, j'ai soumis mes pull requests via l'IDE IntelliJ pour le back et VSCode pour le front, vers le référentiel GitHub du projet, en prenant en compte les commentaires des Lead Dev en vue des merges vers la branche finale.

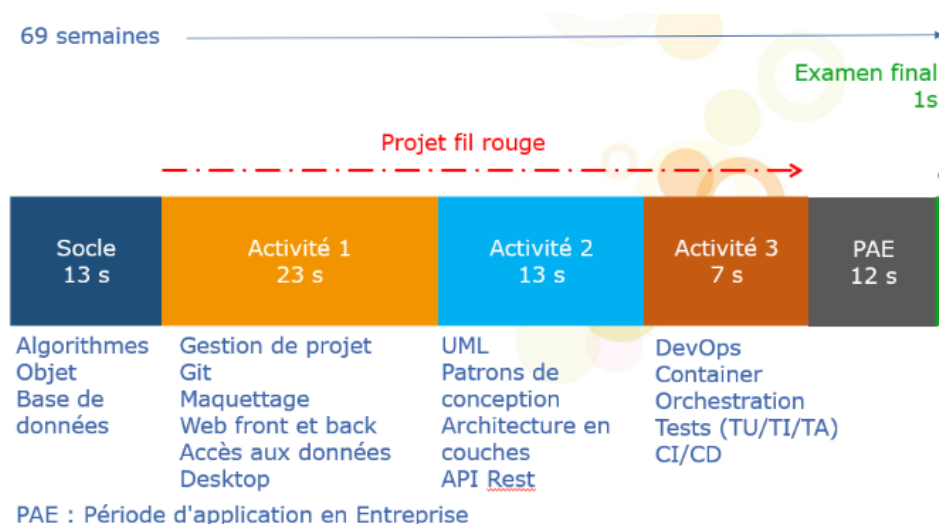
3 De la formation CDA à la période d'application en entreprise

La période de stage en entreprise représente une immersion professionnelle dans le cadre de ma formation en tant que Concepteur Développeur d'Applications (CDA). Au cours des dix-huit derniers mois, entre mon passage à l'ESRP AMIO 2isa et chez l'ESN Ntico, j'ai acquis des compétences professionnelles transférables en entreprise, dans une ESN ou au sein d'un SI.

Cette période d'application en entreprise ayant débuté le 19 février et s'étant achevée le 11 mai 2024, m'a permis de mettre en pratique et d'acquérir de nouvelles compétences variées et pertinentes dans un contexte professionnel :

- Le travail en équipe,
- Le partage de connaissances,
- La conception de solution digitale dans un contexte Agile⁹,
- Le développement d'un client en VueJS avec le framework Quasar¹⁰,
- L'emploi du client HTTP Axios basé sur les promesses JavaScript¹¹,
- Le développement d'un serveur de type API REST en Java avec le framework Spring Boot
- L'emploi de Git et GitHub pour le versionning, et l'intégration en continue

Ces compétences avec ceux acquises tout au long de la formation de Concepteur Développeur d'application réalisée au sein de l'ESRP 2isa à Millau, seront des atouts cruciaux dans ma reconversion professionnelle, de l'industrie au développement applicatif.



⁹ <https://www.atlassian.com/fr/agile>

¹⁰ <https://quasar.dev/>

¹¹ <https://axios-http.com/fr/docs/intro>

4 Liste des compétences du référentiel couverts par le projet

Le concepteur développeur d'applications a pour responsabilité de concevoir et mettre en œuvre des services numériques en conformité avec les normes et standards de l'industrie. Il doit démontrer des compétences dans la conception d'interfaces utilisateur, le développement de bases de données, ainsi que dans l'architecture et le développement logiciel. Cela nécessite une capacité d'adaptation aux évolutions technologiques et une stricte observance des recommandations de sécurité.

La validation de ce titre professionnel requiert la présentation de livrables attestant de mon savoir-faire professionnel, correspondant aux blocs de compétences définis dans le Répertoire national des certifications professionnelles, sous la référence RNCP31678. Mes deux premiers rapports concernant le développement d'une application web et d'un autre desktop couvrent l'ensemble des compétences requises. Concernant la mise en situation professionnelle dans l'entreprise Ntico, les compétences couvertes sont :

- Concevoir et développer une application web :
 - Maquetter une application
 - Développer l'interface utilisateur
 - Développer le traitement serveur
- Compétences transverses :
 - Actualiser et partager ses compétences en conception et en développement informatique
 - Utiliser l'anglais dans son activité professionnelle

5 Expression de la demande pour l'espace d'administration du projet XPérienCe

Objectifs :

- Développer une interface d'administration, client et serveur du site XPérienCe, interne à Ntico.
- Suivre le collaborateur ; cette application web utilisable sur poste de travail a pour objectif le suivi des collaborateurs dans leurs démarches à accomplir tout au long de leurs collaboration avec Ntico en mettant en avant les 3 valeurs de l'entreprise : Partage, Progrès, Plaisir.
- Gérer les action et groupes d'action à réaliser par le collaborateur.
- Permettre d'affecter les groupes d'actions à des collaborateurs
- Décomposer l'application en un client web développée en VueJS 3 avec le framework Quasar 2 et un serveur de type API REST développé Java SDK 17 avec le Framework Spring Boot 3.
- Intégrer les développements déjà réalisées tout en gardant les bonnes pratiques et l'architecture présentes dans l'application XPérienCe est en cours de développement.

Gouvernance :

- Architect, référent fonctionnel et Product Owner : Aurélien Foultier
- Référent technique niveau 1 : Vincent P.
- Référent technique niveau 2 : Guillaume L.
- Validation des PR : Vincent P. et Guillaume L.

Environnement de travail :

- Outils Google pour le partage de documents, les courriels et les visioconférences
- Trello pour la planification du projet
- Git et GitHub pour le versionning et la réalisation des pull-request
- VaultWarden pour la gestion des mots de passe d'application et de base de données, et bien d'autres
- Figma pour le maquettage de l'application
- VSCode comme IDE de développement de l'application Front-end
- VueJS 3 comme langage de développement Front-end avec le Framework Quasar 2
- Eclipse comme IDE de développement de l'application Back-end
- Java SDK 17 comme langage de développement Back-end avec le Framework Spring Boot 3
- Sonarlint comme bibliothèque pour le clean code

Étape de développement :

La première étape fut de proposer une « maquette simplifiée » et grossière des écrans d'administration permettant de se représenter le chemin utilisateur (UX) ainsi que l'organisation de la ou les pages (UI). Puis de me familiariser avec Java Spring Boot, VueJS et Quasar

L'étape suivante fût le développement de la gestion des groupes d'action et des actions. L'administrateur devant pouvoir ajouter / modifier / archiver un groupe d'actions et ses actions. Concernant le thème de du groupe d'action, l'administrateur peut choisir dans la liste des valeurs existantes en base de données. Il ne peut pas en ajouter ni supprimer. Concernant les actions à réalisé par le collaborateur, l'administrateur peut au sein d'un groupe d'action, en ajouter, en modifier, en supprimer, et peut réorganiser leur ordre de réalisation.

La dernière étape, qui n'a pas été implémenté, devait être l'affectation des groupes d'action à des collaborateurs.

6 Analyse du besoin, et planification

6.1 Cas d'utilisation

Pour analyser le besoin et réaliser le développement de la partie administration de l'application j'ai exprimé le besoin par des cas d'usage reflétant les besoins fonctionnels du système d'informations. Ceux-ci m'ont permis la compréhension des attentes de Ntico, et la planification des tâches à réaliser suivant les fonctionnalités attendues.

Les cas d'utilisation ont une vraie valeur métier. Ils permettent de décrire simplement et de manière compréhensible les fonctionnalités devant être réalisées. Par l'utilisation de décomposition de chacun d'eux comme une histoire, ils permettent au client et à l'équipe de développeur de confirmer leurs compréhensions du besoin.

Chaque cas d'utilisation est décomposé en quatre parties. La première représente le rôle de l'utilisateur. La deuxième représente le besoin, l'action devant être réalisée. La troisième représente le bénéfice, la valeur métier de la fonctionnalité. Enfin le comment permet de décrire la manière dont la fonctionnalité doit être réalisée. Cette dernière est optionnelle lors de la discussion avec un client, mais permet de cibler le mode de réalisation souhaité.

Ci-dessous quelques cas d'utilisation ou user story que j'ai identifiés, puis qui furent validés par le chef de projet :

N°	En tant que	Je souhaite	Afin de	Comment
UC101	Administrateur du site XPérienCe	Accéder à la liste des groupes d'action	D'administrer chaque groupe d'action	En cliquant sur un bouton "administration" accessible de la page d'accueil
UC201		Pouvoir ajouter un groupe d'action	Compléter le parcours collaborateur	En cliquant sur un bouton "ajouter" accessible depuis la liste des groupes d'action

N°	En tant que	Je souhaite	Afin de	Comment
UC202		Pouvoir archiver un Action Group	D'archiver ce groupe d'action	<p>En cliquant sur un bouton "supprimer" accessible depuis la liste des groupes d'action sur le groupe d'action concerné</p> <p>En cliquant sur un bouton "supprimer" accessible depuis la liste des groupes d'action sur le groupe d'action concerné avec une demande de confirmation de suppression (Yes/No)</p> <ul style="list-style-type: none"> • Cliquer sur Yes me renvoie sur la page listant les des groupes d'action • Cliquer sur NO me laisse sur page du groupe d'action concerné
UC203		Pouvoir modifier un groupe 'action		En cliquant sur un bouton "éditer" accessible depuis la liste des groupes d'action sur le groupe d'action concerné
UC301		Editer les éléments d'un groupe d'action		<p>En accédant à une page spécifique comprenant l'ensemble des éléments du groupe d'action et permettant de :</p> <ul style="list-style-type: none"> • Editer le titre avec du texte • Editer le sous-titre avec du texte • Choisir un thème suivant une liste de thème existant • Ajouter une image du groupe • Ajouter une image de bannière • Ajouter une action ou des actions • Visualiser le nombre d'action • Editer chaque action, titre et sous-titre • Supprimer une action <p>Remarque : un groupe d'action a au minimum une action et peut comporter n actions</p>

6.2 Planification et méthode de travail

6.2.1 Générale

Ma familiarisation et ma contribution au projet XPérienCe ont débuté la semaine du 13 mars, environ un mois après mon arrivée chez Ntico. Pour organiser les différentes tâches et répondre efficacement aux exigences de Ntico, le responsable du service développement de Ntico Service a choisi d'utiliser la méthodologie Scrum d'Agile avec comme support le kanban Trello, avec comme backlog les cas d'utilisation. Ainsi, mes tâches étaient planifiées en fonction de chaque fonctionnalité à développer.

Chaque matin à 9h15, nous avons une réunion de mêlée quotidienne des méthodes Agiles, appelée Daily Team Dev, d'une durée de 1 minute par personne et réalisée en visioconférence avec les équipes des trois sites de Ntico, soit. Ces réunions ayant pour objectif pour chaque personne de dire ce qu'il a fait la veille, ce qu'il fera aujourd'hui et ce dont il a besoin.

Concernant le projet XPérienCe, nous avons spécifiquement une réunion avec les développeurs du projet, chaque lundi à la suite du Daily, d'une durée maximale de 30 minutes.

En cas de difficultés techniques, il était possible de contacter l'un des Lead Dev.

6.2.2 Méthode Scrum dans le projet XPérienCe



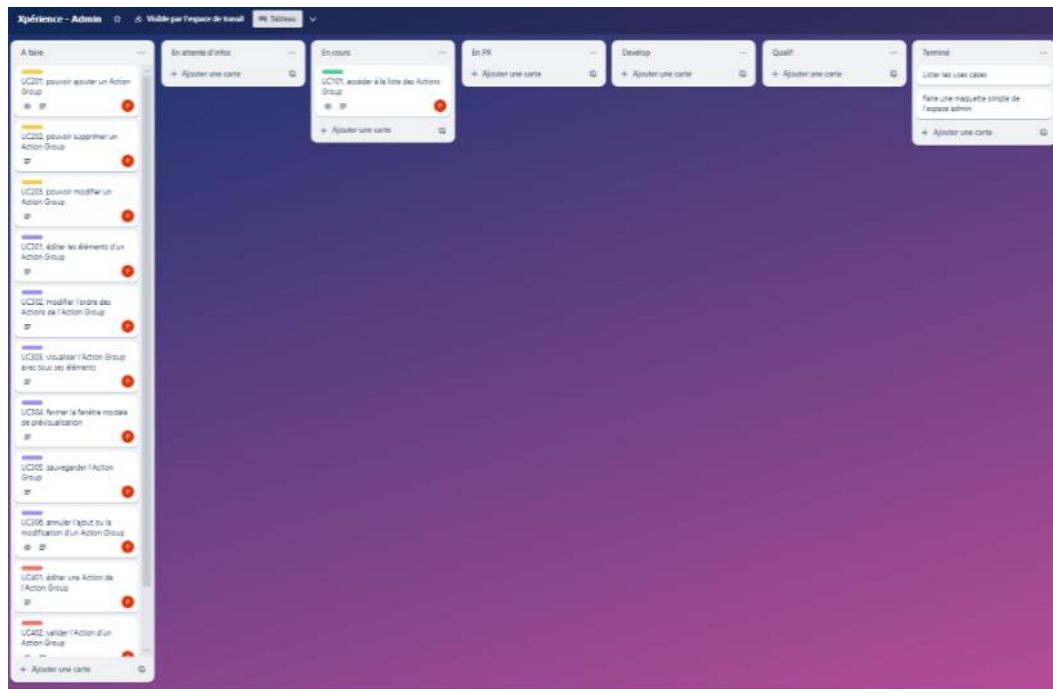
J'ai organisé les tâches à accomplir pour atteindre mes objectifs en suivant une approche similaire au cycle PDCA ou Roue de Deming (Planifier, Faire, Vérifier, Agir) tout au long des phases de conception et de développement du projet. Cela en plus de la méthode Agile employée au sein de l'ESN Ntico.

Au cours de ces étapes, j'ai intégré de nouvelles connaissances acquises par l'échange avec mes collègues, complétées avec mes recherches sur le web et avec différentes intelligences artificielles conversationnelles. Cette méthodologie m'a permis d'améliorer la qualité, respect des bonnes pratiques, la factorisation mon code, et réalisé de l'amélioration en continue. Au sein de l'entité Service de Ntico, la planification suivant Scrum de la méthode Agile est employée. Une fois la définition des cas d'utilisation définis je les ai intégrés au backlog d'un kanban sous Trello.

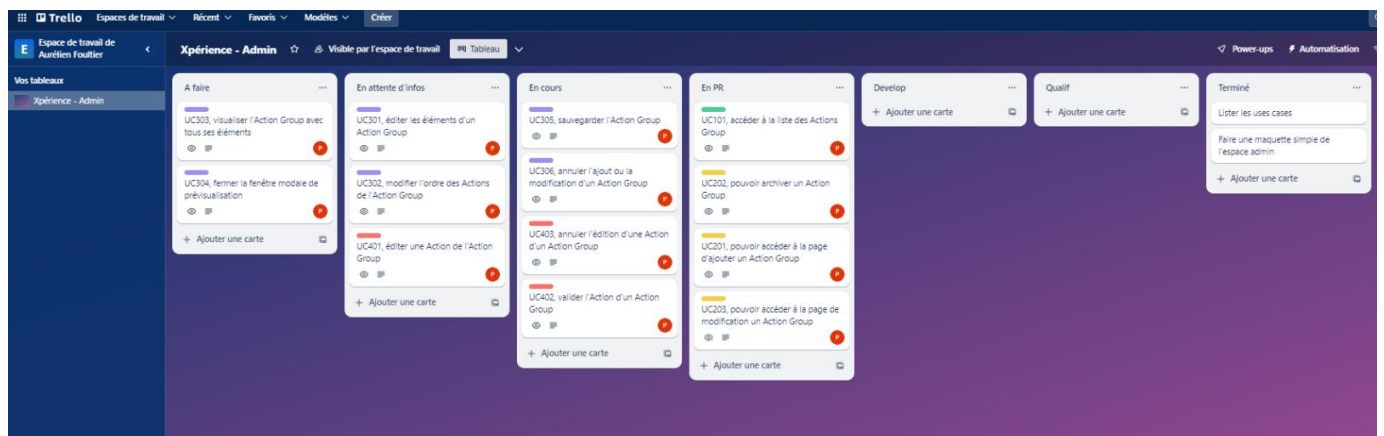
Le Product Owner, Aurélien Foutier, me définissait chaque lundi, lors du point XPérienCe, le sprint que je devais réaliser dans la semaine et m'interrogeant sur mes besoins techniques en liens avec les réalisations à produire. Je devais aussi estimer pour chaque sprint le temps de réalisation en journée. Ces estimations furent pondérées par le Product Owner qui faisait office de Scrum Master, au regard des nouvelles connaissances que je devais acquérir tout au long du projet.

Cette méthodologie nous a permis d'obtenir une interface d'administration du projet XPérienCe, conforme aux attentes de la direction, le client, dans le temps qui m'était imparti.

Kanban Trello en début de projet



Kanban Trello au bout de cinq semaines



7 Spécifications techniques

7.1 Spécification de conception du client web

Avec M. Foulter nous avons commencé par établir les spécifications de l'interface utilisateur de la partie administrative du projet XPérienCe. Cela m'a permis de retenir une spécification permettant de concevoir une interface simple, intuitive, et offrant une XPérienCe utilisateur optimale.

Pour ce faire, je me suis inspiré des interfaces du projet XPérienCe déjà existantes, en les complétant des bonnes pratiques internes à l'entreprise et celles glanées sur le web dans des sites de références comme [ionos.fr¹²](https://www.ionos.fr/digitalguide/web-marketing/vendre-sur-internet/quest-ce-quune-bonne-convivialite-pour-un-site-web/) ou encore [openedition.org¹³](https://journals.openedition.org/edc/5379). Cette démarche m'a permis d'analyser le maquetage en fonction de quatre concepts clés. Pour chacun de ces concepts, j'ai défini des spécifications listées dans le tableau suivant, afin de répondre aux besoins spécifiques de l'entité Ntico Service.

Concept de Maquetage	Spécificité technique de réalisation du maquetage
Architecture de l'information	<ul style="list-style-type: none">• Structure claire et logique pour le contenu en utilisant des catégories,• Hiérarchie visuelle claire.
Stratégie de contenus	<ul style="list-style-type: none">• Informations pertinentes et utiles, sans informations superflues.• Langage simple et compréhensible, sans jargon technique.• Structurer le contenu visuel : titres, sous-titres
Convivialité, interactivité et accessibilité	<ul style="list-style-type: none">• Eléments interactifs (boutons, liens, formulaires, etc.) positionnés de manière cohérente et facilement repérable, en correspondance avec les réalisations du projet déjà existantes.• Indicateurs visuels (changements de couleur, popup, effets d'animation, etc.) pour signaler les interactions et les rétroactions.• Optimiser la navigation en fournissant des liens de retour (pas de page morte).
Interface utilisateur, ergonomie et charte graphique	<ul style="list-style-type: none">• Esthétique et cohérente, charte graphique en correspondance avec les pages déjà existantes (couleurs, typographie, icônes, etc.).• Page épurée et aérée, sans éléments superflus ou encombrants.• Des visuels, pour faciliter la reconnaissance rapide des fonctionnalités.• Design pour ordinateur. Interface non utilisable sur smartphone• Emploi des composants d'interface utilisateur (UI) natifs du Framework Quasar en les adaptant à la charte graphique du site en développement déjà existant.

¹² <https://www.ionos.fr/digitalguide/web-marketing/vendre-sur-internet/quest-ce-quune-bonne-convivialite-pour-un-site-web/>

¹³ <https://journals.openedition.org/edc/5379>

7.2 Maquettage de la solution

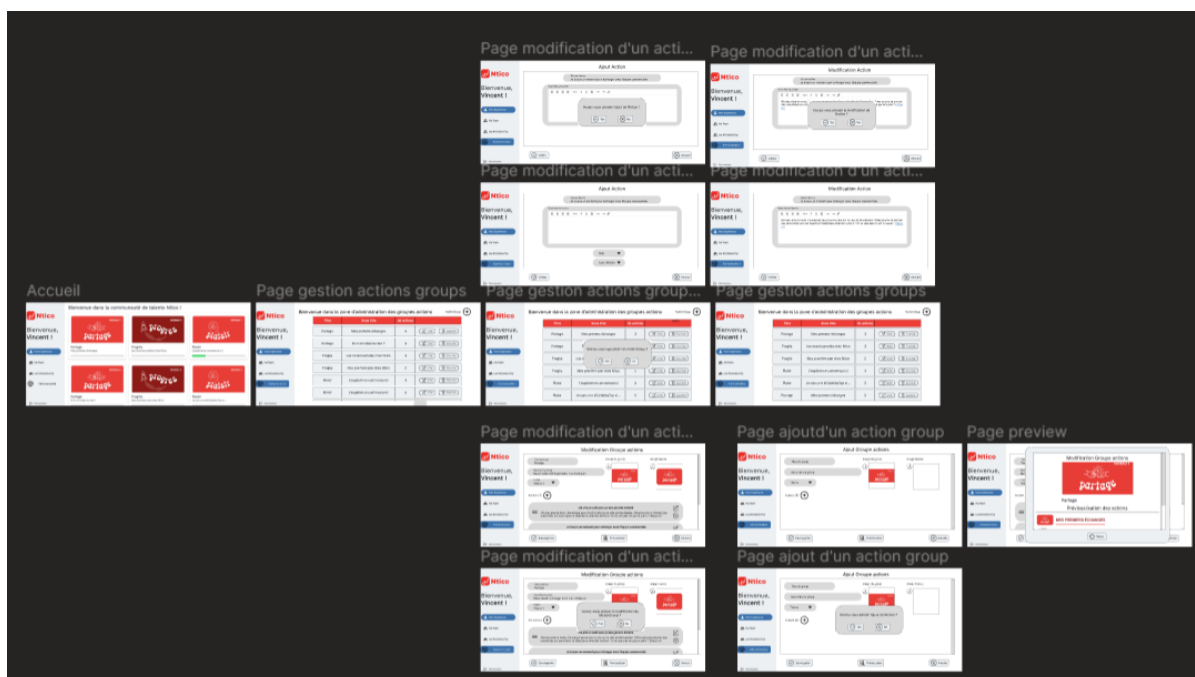
À la demande du référent fonctionnel, j'ai été chargé de créer une maquette simple représentant les fonctionnalités de l'application. J'ai suggéré d'utiliser l'outil de design collaboratif en ligne, Figma, pour réaliser cette maquette. Après une première version et une discussion approfondie, notamment sur les possibilités offertes par le Framework Quasar, j'ai proposé une nouvelle maquette qui a été validée par mon référent.

Pour cette réalisation, nous avons décidé de créer trois vues distinctes en plus de modifier la page d'accueil pour permettre l'accès à la zone d'administration :

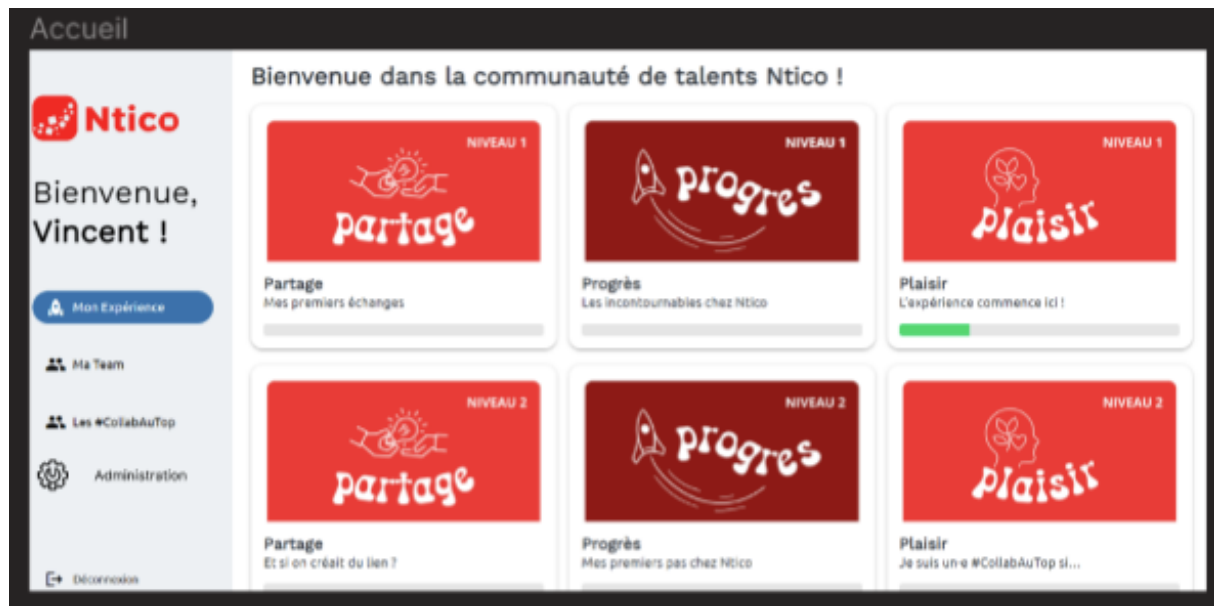
- Une vue listant les actions de groupe pour en faciliter la gestion,
- Une vue dédiée à la gestion des actions de groupe,
- Et une vue spécifique pour la gestion des actions individuelles. Cependant, cette dernière a été éliminée lors du développement, au profit de composants Quasar intégrés dans la vue des actions de groupe, permettant l'édition.

7.2.1 Vue d'ensemble du projet sous Figma

J'ai réalisé, sous Figma, via des composants imbriqués, les différentes vues. Cela avec un enchainement interactif permettant la vision du projet avec son jeu d'interface.

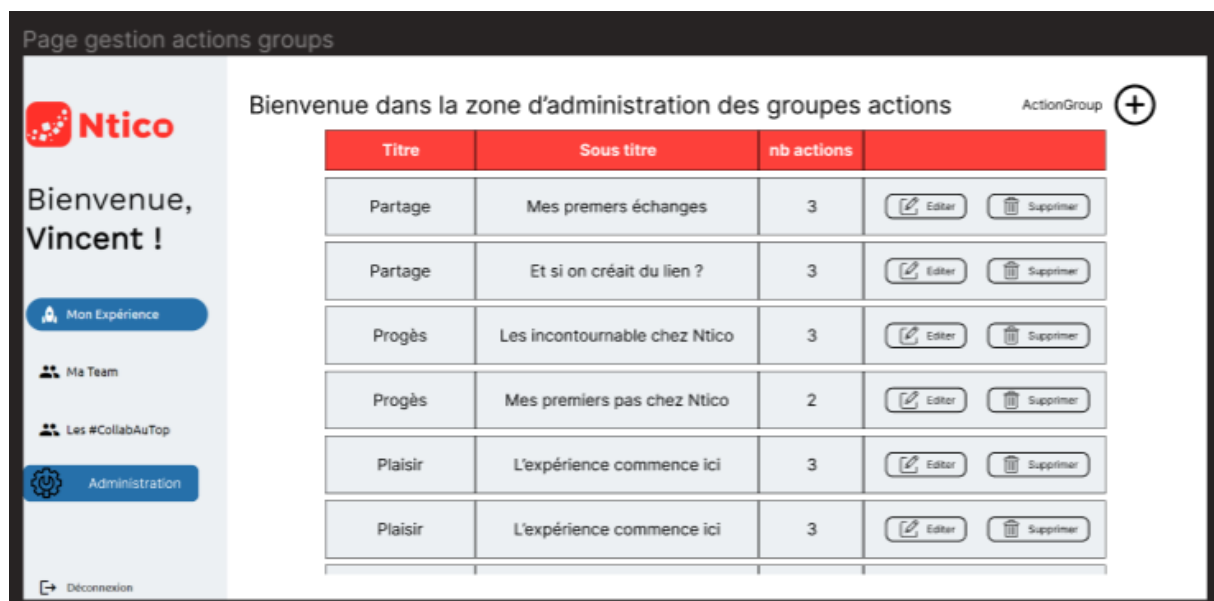


7.2.2 Vue d'accueil du projet XPérienCe

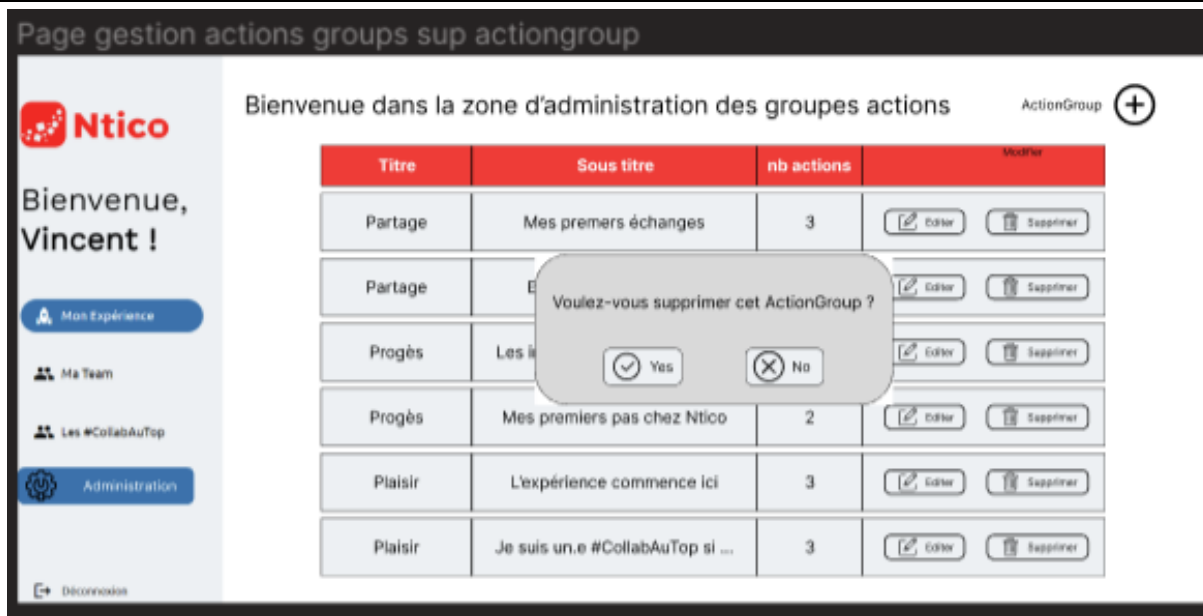


Dans cette interface déjà développée, je devais y intégrer l'accès à la zone d'administration, accessible uniquement aux rôle administrateur. J'ai opté pour un bouton dans la partie gauche de menu. Celui-ci devra être en correspondance avec ceux déjà présents, icône, police d'écriture et activation compris.

7.2.3 Vue de gestion des groupes d'action

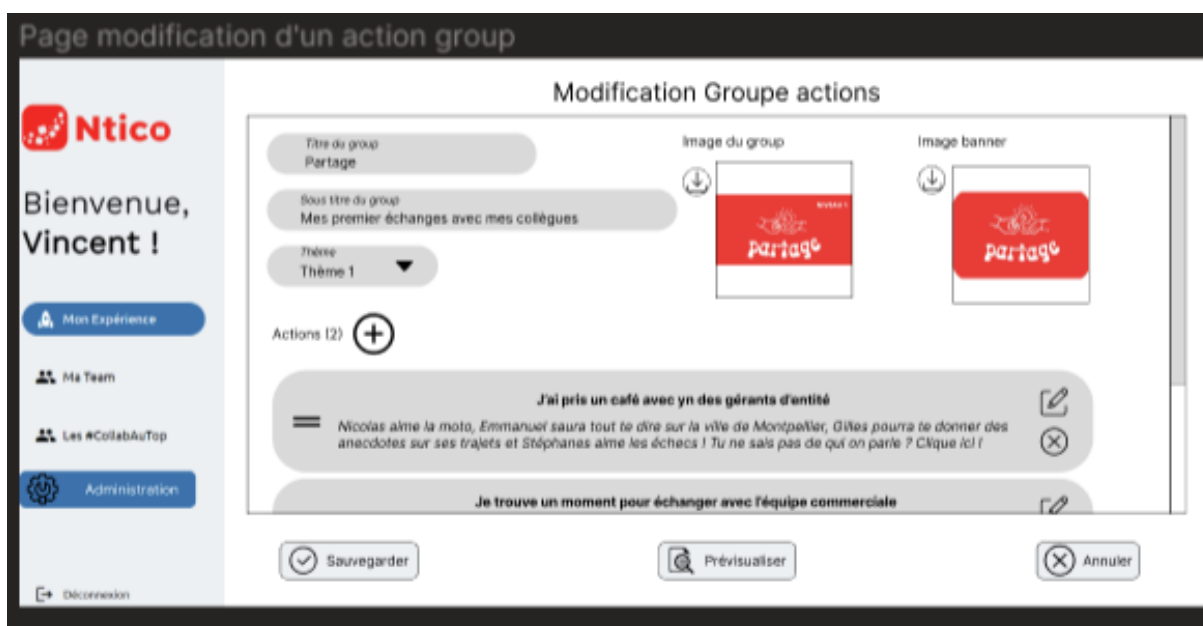


Cette interface devra permettre à l'utilisateur de visualiser l'ensembles des groupes d'action au sein d'un tableau dont l'en-tête sera fixe. Cette page doit permettre l'ajout d'un nouvel AG, la modification de ceux déjà existant ou leur suppression (archivage en base de données).




La suppression d'un groupe d'action devra être confirmé par l'utilisateur avec l'utilisation d'une fenêtre modale.

7.2.4 Vue de gestion d'un groupe d'action



Cette interface devra permettre la gestion de l'ensemble des composants d'un groupe d'action, titre, sous-titre, thème, image de l'action-groupe, image de bannière, thème, et des actions associées (ajout, suppression, modification). Le bouton d'administration de la zone menu devra être activé. De-plus l'ordre des actions devra être modifiable par drag-and-drop. Les informations de l'AG de devront être compris dans une zone scrollable avec un pied de page comprenant les boutons « sauvegarder », « prévisualiser », et « annuler »

Page ajout'un action group



Bienvenue,
Vincent !

Mon Expérience

Ma Team

Les #CollabAuTop

Administration

Déconnexion

Ajout Groupe actions

Titre du group
Sous titre du group
Thème ▼

Image du group
Image banner

Actions (0) +

Sauvegarder
Prévisualiser
Annuler

L'annulation de modification et l'annulation d'ajout d'un groupe d'action devra être confirmé par l'utilisateur avec l'utilisation d'une fenêtre modale.

7.2.5 Vue de gestion d'une action

Page d'ajout d'une action



Ntico

Bienvenue, Vincent !

Mon Expérience

Ma Team

Les #CollabAuTop

Administration

Déconnexion

Ajout Action

Titre de l'action
Je trouve un moment pour échanger avec l'équipe commerciale

Sous-titre de l'action

Rôle

Type d'Action

Valider

Annuler

Page modification d'une action

Clique ici !'. The 'Valider' and 'Annuler' buttons are at the bottom." data-bbox="116 379 879 645"/>

Ntico

Bienvenue, Vincent !

Mon Expérience

Ma Team

Les #CollabAuTop

Administration

Déconnexion

Modification Action

Titre de l'action
Je trouve un moment pour échanger avec l'équipe commerciale

Sous-titre de l'action

Nicolas aime la moto, Emmanuel saura tout te dire sur la ville de Montpellier, Gilles pourra te donner des anecdotes sur ses trajets et Stéphanes aime les échecs ! Tu ne sais pas de qui on parle ? [Clique ici !](#)

Valider

Annuler

Cette interface devra permettre la gestion des composants, titre et sous-titre d'une action (ajout et modification). Le sous-titre devra être modifiable par un wysiwyg comprenant les fonctionnalités, gras, italique, sous-lignage et ajout de liens hypertexte.

Comme pour les vues précédentes, l'annulation devra être confirmée par l'intermédiaire d'une fenêtre modale.

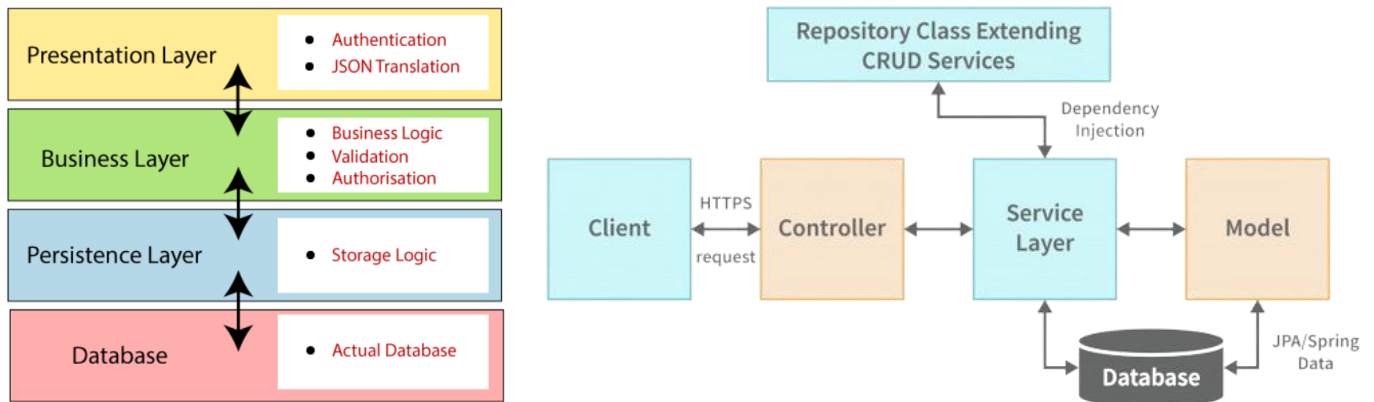
7.2.6 Popup de prévisualisation



La prévisualisation de l'action-groupe et de ses actions devra être accessible sur la page de l'action-group et permettra une prévisualisation du rendu final.

7.3 Spécification architecturale

- L'application XPérienCe adopte l'architecture de Java Spring Boot¹⁴ côté serveur. Elle fut respectée dans la partie que j'ai développée lors de l'injection d'interface de service.



- Côté client l'architecture sera celle employée dans le développement actuel de l'application XPérienCe, celle du framework Quasar.

Architecture **basée sur les composants** Dans cette structure, chaque dossier joue un rôle spécifique :

- **assets** : Contient les ressources statiques comme les images, les polices, etc.
- **components** : Comprend les composants réutilisables.
- **layouts** : Définit les mises en page globales de l'application.
- **pages** : Contient les composants de page spécifiques à chaque route.
- **router** : Gère les routes de l'application.
- **store** : Stocke l'état global de l'application. Dans les parties déjà présentes, le store ne gérant pas l'état global, il me servira pour la définition des squelettes d'objets.

```
src/
├── assets/
├── components/
│   ├── MyComponent.vue
│   └── ...
├── layouts/
│   ├── MainLayout.vue
│   └── ...
├── pages/
│   ├── HomePage.vue
│   └── ...
├── router/
│   └── index.js
├── store/
│   ├── index.js
│   └── ...
└── ...
```

¹⁴ <https://www.interviewbit.com/blog/spring-boot-architecture/>

7.4 Spécifications techniques du client

La partie administration devra respecter l'architecture du projet XPérienCe. Cette partie de l'application devra respecter aussi :

- Ne pas porter de logique métier. Le front-end se chargera uniquement de réaliser les appels au Endpoint du serveur et l'affichage des composants portant les données brutes reçu de la partie serveur de l'application au travers de pattern DTO spécifiques à la zone d'administration.
- Devra être développé en VueJS 3 avec le Framework Quasar 2. Le front-end devra être codé avec les pseudos langage de ceux framework. Si ceux-ci ne permettent pas la réalisation des fonctionnalités, le langage Javascript ES6 pourra être employé sous l'accord des référents technique.
- Le drag & drop des actions dans la vue d'un action groupe devra être réalisée avec la librairie `Vue.draggable.next`¹⁵.
- Le client http employé sera Axios 1.2¹⁶
- L'interface devra être réalisé avec les composant natifs de Quasar. Si aucun ne correspond à la demande, les référents techniques devront statuer.
- Le principe « parent-enfant » permettant un découpage du code de l'interface en composant devra être réalisé sur chaque élément de la vue. Si une impossibilité technique apparaît, les référents techniques devront statuer.
- Le CSS sera réalisé dans la balise style de la page de code du composant. Aucun style ne devra être codé dans les balises HTML hors natif à Quasar¹⁷.
- Les textes natifs aux vues devront permettre le changement de langue et donc ne pas être codés en dure dans les composants HTML mais être déportés avec la bibliothèque I18next 9¹⁸ au travers de variable si besoin.
- Le compilateur employé pour le projet client est Webpack¹⁹ avec le gestionnaire de packet npm de Node.js

¹⁵ <https://github.com/SortableJS/vue.draggable.next>

¹⁶ <https://axios-http.com/docs/intro>

¹⁷ <https://quasar.dev/components#introduction>

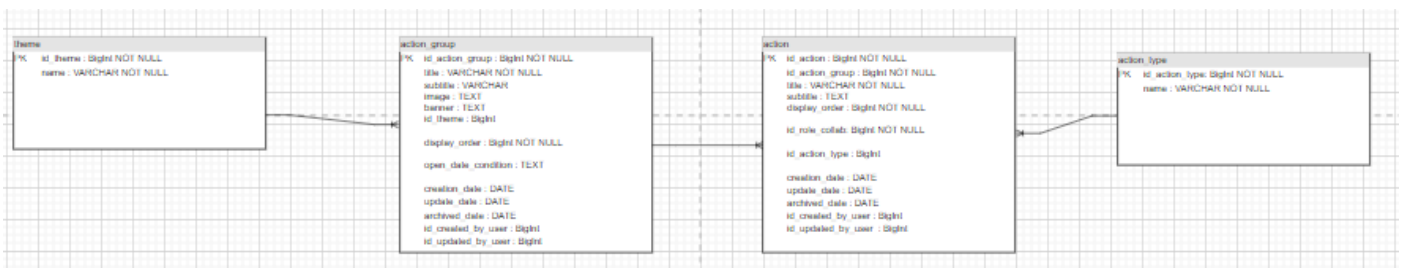
¹⁸ <https://www.i18next.com/>

¹⁹ <https://webpack.js.org/>

7.5 Spécifications techniques du serveur

La partie administration de serveur de l'application web XPérienCe devra s'intégrer aux développements déjà réalisés, et respecter :

- Le principe architectural tiers de l'API REST
- Être développé en Java SDK 17 avec le framework Spring Boot 3
- Employer la bibliothèque MapStruct pour le mappage des entité JPA de Java
- Employer les variables d'environnements au travers au gestionnaire de mot de passe Vaultwarden
- Employer les différentes méthodes déjà développées pour la réalisation des CRUD de la partie d'administration
- Respecter le diagramme de classe ci-dessous :



8 Développement de la solution

Pour initier le développement de la partie administration du projet XPérienCe, j'ai fait le choix de décomposer mon travail en flux suivant les cas d'usage, et pour chacun d'eux, une décomposition en trois parties : Premièrement la définition des Endpoints et des DTO, deuxièmement, le développement de la partie client, et enfin le développement de la partie serveur. J'ai fait le choix de développer le front avant le back, car le client comporté la plus grande difficulté technique au regard de mes connaissances en début de projet.

8.1 Définition du contrat en client et serveur

La méthodologie appliquée concernant le pattern DTO chez l'ESN Ntico est celle d'avoir qu'un seul DTO de requête et de réponse pour la réalisation des requêtes Http Post, Put et Get, combiné au besoin de l'identifiant du groupe d'action aux end-points correspondants. Concernant les requêtes Delete, aucun DTO ne sera utilisé, juste l'id en fin d'Endpoint. Les DTO peuvent porter des Entités pour minimiser le nombre de requête à la base de données et simplifier l'utilisation de la bibliothèque MapStruct utilisée pour le mappage entre les Entités et le DTO.

- Endpoint de l'API REST

GET	https://.../admin/ Récupération de l'ensemble des groupes d'action non archivés avec le nombre d'action non archivée qu'elles portent
GET	https://.../admin/{idActionGroup} Récupération d'un groupe d'action spécifique avec sa liste d'action non archivées
DELETE	https://.../admin/{id} Archivage d'un groupe action spécifique et de ses actions
POST	https://.../admin/ Ajout d'un nouveau groupe d'action et de ses actions
PUT	https://.../admin/{id} Mise-à-jours d'un groupe d'action et de ses actions

- DTO

<p>ActionGroupDTO.</p> <p>Déjà existant, ajout de la propriété pour compter le nombre d'action du groupe d'action</p>	<pre>@Getter @Setter @AllArgsConstructor @NoArgsConstructor @EqualsAndHashCode(exclude = {idActionGroup}) public class ActionGroupDTO { Long idActionGroup; String title; String subtitle; String image; String banner; Theme theme; Long displayOrder; String openDateCondition; LocalDate creationDate; LocalDate updateDate; LocalDate archivedDate; SimpleUserDTO createdBy; SimpleUserDTO updatedBy; Long numberActions; }</pre>
<p>ActionGroupAdminDTO</p> <p>Nouveau DTO, pour la création et ou la modification du groupe d'action et de ses actions.</p>	<pre>@Getter @Setter @AllArgsConstructor @NoArgsConstructor @FieldDefaults(level = AccessLevel.PRIVATE) public class ActionGroupAdminDTO { Long idActionGroup; String title; String subtitle; String image; String banner; Theme theme; Long displayOrder; String openDateCondition; LocalDate creationDate; LocalDate updateDate; LocalDate archivedDate; SimpleUserDTO createdBy; SimpleUserDTO updatedBy; List<ActionAdminDTO> actionList = new ArrayList<>(); }</pre>
<p>ActionAdminDTO</p> <p>Nouveau DTO, pour la création, la modification d'une action</p>	<pre>@Getter @Setter @AllArgsConstructor @NoArgsConstructor @FieldDefaults(level = AccessLevel.PRIVATE) @EqualsAndHashCode public class ActionAdminDTO { Long idAction; String title; String subtitle; Long displayOrder; Role role; ActionType actionType; LocalDate creationDate; LocalDate updateDate; LocalDate archivedDate; SimpleUserDTO createdBy; SimpleUserDTO updatedBy; }</pre>

Voir chapitre 7.3.1 pour un détail sur les annotations de Spring Boot et ses bibliothèques

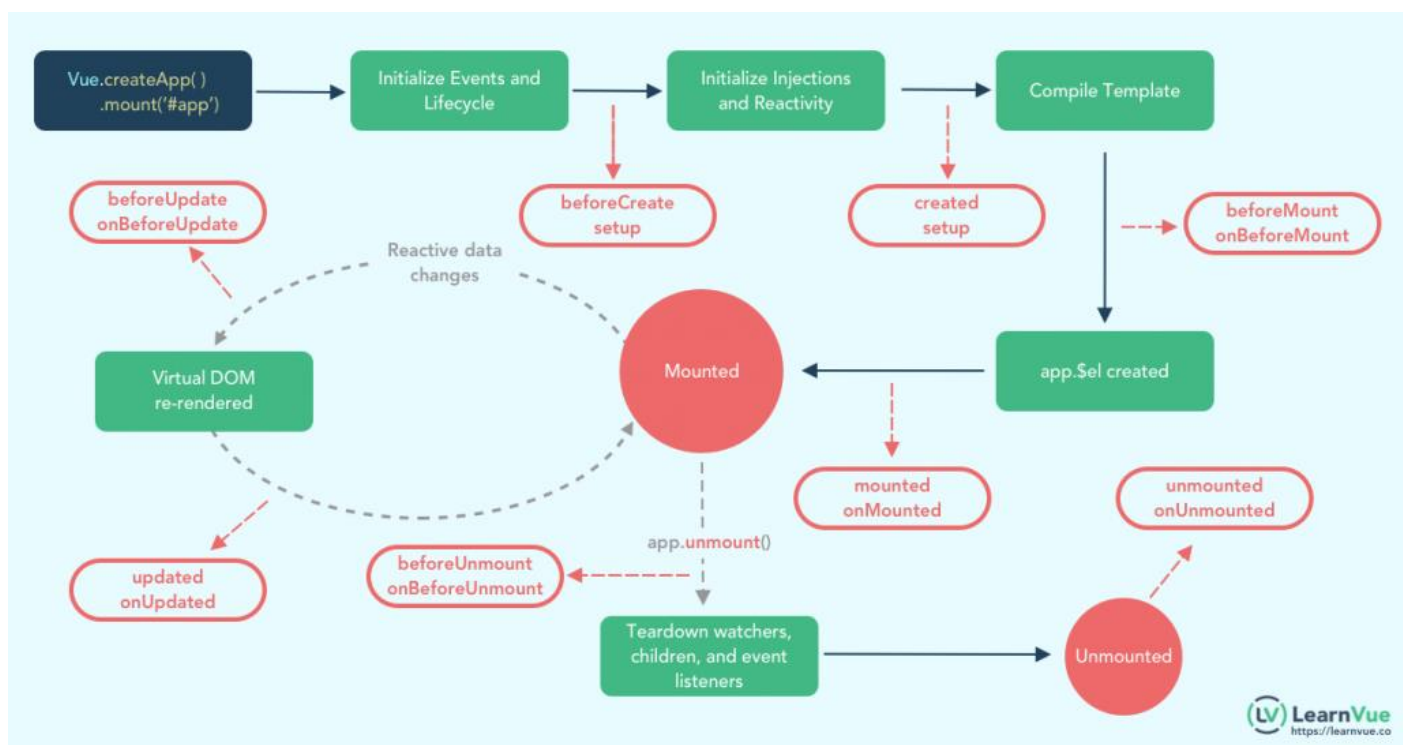
8.2 Partie administration du Client

Pour le développement de la partie Client en VueJS 3 avec le framework Quasar 2, j'ai dû dans un premier temps me former puisque n'ayant aucune connaissance de ces framework en début de projet.

Finalement, j'ai pu développer un espace d'administration répondant aux besoins exprimés par le chef de projet. Cet espace respectant le maquetage, et faisant l'impasse comme demandé par le chef de projet sur le responsive design et le style. Ce dernier devant être minimal. Pour autant, cette interface web étant desktop first d'un point de vue responsive, j'ai mis l'accent sur son adaptabilité avec les différentes tailles d'écrans employés chez Ntico avec un style épuré comme dans les autres parties déjà développées du projet.

8.2.1 Remarques sur VueJS et le framework Quasar

- Quasar est un framework open source multiplateforme construit au-dessus du Framework JavaScript Vue.js qui offre plusieurs avantages pour le développement d'applications web et mobiles. Ils offrent de nombreux avantages, dont sa bibliothèque de plus de 70 composants prêts à être intégrés directement, comme des listes, des Wysiwyg, ou encore des uploader d'image. Le point le plus complexe avec VueJS est le cycle de vie du DOM qui est particulièrement complexe. Lors du développement il implique de comprendre quand une variable est peuplée, et comment la peupler dans des interactions entre les composants parents et enfants. Tous les éléments d'une interface graphique ne sont pas prêts à fonctionner en même temps lors du chargement d'une page VueJS. De plus, les enfants d'une page sont montés avant leurs parents. Ci-dessous le cycle de vie d'un composant dans VueJS²⁰ :



²⁰ <https://fr.vuejs.org/guide/essentials/lifecycle.html>

- Décomposition en composants :

VueJS permet la création d'application web modernes appelée single page application. Dans le cas du projet XPérienCe comme pour la plupart des SPA, le code initial de l'application (HTML, CSS et JavaScript) est téléchargé à partir du serveur, et les ressources suivantes sont chargées dynamiquement sur la page, en réponse aux actions de l'utilisateur. Ces pages sont décomposées en trois parties de code :

- La première en pseudo-code HTMP, compris dans la balise template.
- La deuxième en pseudo-code Javascript de VueJS, dans la balise script.
- La dernière comprenant le CSS ou le SCSS ou encore du SASS suivant le choix du développeur, dans la balise style.

8.2.2 Développement du Client web

8.2.2.1 Modification du fichier de routage, route.js

Ajout des routes pour l'ensemble des pages de la partie administration développée

L'utilisation du router VueJS permet un paramétrage simple à partir du MainLayout, des URI. L'accès à la zone de gestion se fait avec l'URI « /admin » qui importe dans la page listant dans un tableau les groupes d'action, comme définit dans le maquetage. L'accès pour l'ajout d'un groupe d'action, lui est réalisé avec l'URI « admin/action-group/ ». Quant à la modification d'un groupe d'action spécifique, il suffit d'ajout de son identifiant à cette dernière URI.

Le développement en VueJS du routage des différentes pages est compréhensible pour les futurs développeurs, et facilement maintenable.

```
{
  path: "/admin",
  component: () => import("layouts/MainLayout.vue"),
  children: [
    { path: "", component: () => import("pages/OnActionsGroupAdmin.vue") },
    {
      path: "action-group",
      component: () => import("pages/OnActionGroupAdminPage.vue"),
    },
    {
      path: "/action-group/:id",
      component: () => import("pages/OnActionGroupAdminPage.vue"),
    }
  ],
  meta: {
    requiresAuth: false,
  },
},
];
export default routes;
```

8.2.2.2 Vue d'accueil du projet XPérienCe



Dans cette interface déjà développée, j'y ai intégré l'accès à la zone d'administration, accessible uniquement aux rôles administrateur, via un bouton utilisant un des icônes la collection Google Fonts déjà employés.

Pour ce faire j'ai modifié la page MainLayout.vue :

- Dans sa balises Template pour ajouter le bouton :

Ajout d'un `<q-btn/>` suivant un rendu conditionnel sur le rôle de l'utilisateur, et un événement sur le click renvoyant vers la page d'administration des groupes d'action définit entre les balises script.

```
<q-btn v-if="roles.includes('Admin')" rounded unelevated no-caps class="q-ma-md
customButton" :class="{
  'bg-primary text-white': $route.path.startsWith('/onActionsGroup-admin'),
}" icon="settings_applications" size="1rem" @click="navigateToListeActionGroup"
:align="'left'">
  <span style="padding-left: 8px">{{ $t("general.administration") }}</span>
</q-btn>
```

- Dans sa balise script pour la redirection vers la page listant les groupes d'action :

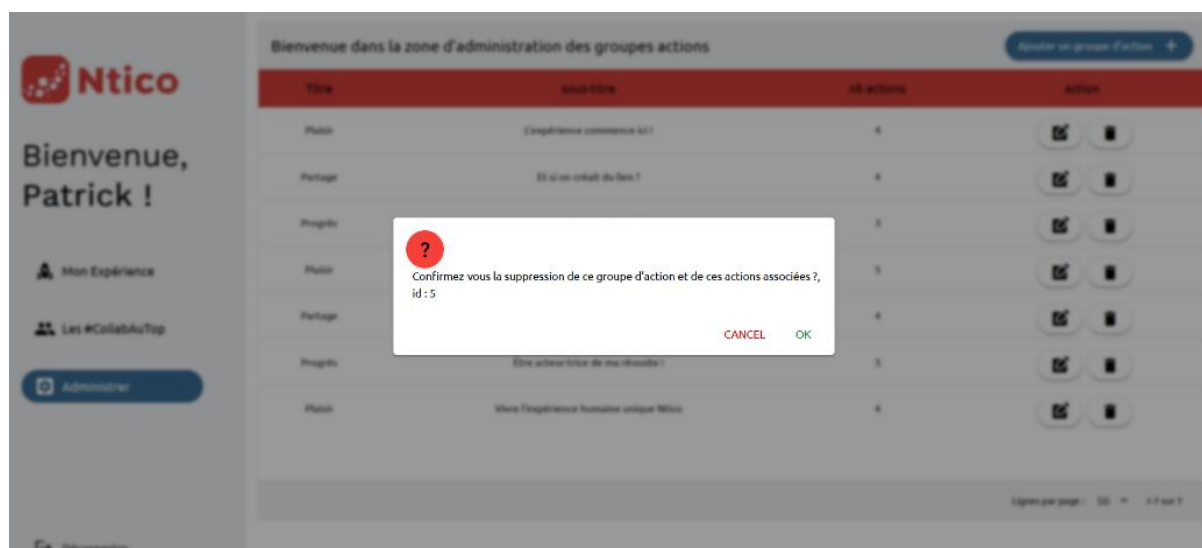
```
import { useRouter } from "vue-router";
const router = useRouter();
const navigateToListeActionGroup = () => {
  router.push({ path: "/OnActionsGroup-admin" });
};
```

Remarque sur l'accessibilité des pages administration par seulement les utilisateurs ayant le rôle admin :

Chacune des pages d'administration comporte en dans sa première balise template, une directive « v-if » de VueJS identique au MainLayout identifiant le rôle de l'utilisateur autorisé. Les rôles de l'utilisateur étant récupérés via un store utilisateur récupérant la liste des rôles de la personne connectée.

8.2.2.3 Vue de gestion des groupes d'action,

- Composant parent : ActionGroupAdmin.vue
- Composant enfant type popup : ConfirmPopup.vue



Cette vue devait comporter un tableau listant les groupes d'action, et pour chacun d'eux, deux boutons, un pour son édition et l'autre pour son archivage. De plus la page devait aussi comprendre un autre bouton pour la rédaction d'un nouveau groupe d'action. A cela, l'ensemble des textes devaient être traductible avec le framework d'internationalisation I18next. Pour cette page j'ai fait le choix de ne pas créer de sous composant afin de faciliter sa maintenance, puisqu'elle est assez simple à développer.

Pour ce faire j'ai codé dans le fichier ActionGroupAdmin.vue :

- Dans la balises Template :

Utilisation d'un <q-table/> pour former le tableau, rempli suivant sa propriété « rows », avec ajout dans le header du tableau du titre et d'un bouton pour accéder à la page d'ajout d'un nouveau groupe d'action.

Chaque ligne du tableau comportant deux boutons pour soit modifier soit archiver le groupe d'action concerné.

Utilisation d'un composant enfant, <ConfirmPopup/> , pour la confirmation de l'archivage d'un groupe d'action.

```
<template>
  <q-page>
    <div class="q-pa-md " v-if="roles.includes('Admin')">
      <q-table :loading="loadingPage" class="my-sticky-header-table " flat bordered
:title=titlePage :rows="myRows"
      :columns="myColumns" :rows-per-page-options="[50, 100]" :wrapCells="true" row-
key="idActionGroup">
        <template v-slot:top-right>
          <q-btn color="primary" icon-right="add" rounded :label="nameBtnAdd" no-caps
            @click="redirectionTaskActionGroup()" />
        </template>
        <template v-slot:body-cell-btnactions="props">
          <q-td :props="props" class="q-gutter-sm">
            <q-btn icon="edit_square" rounded name="edit" :title=nameBtnEdit
              @click="redirectionTaskActionGroupId(props.row.idActionGroup)">
              <q-tooltip>{{ $t("onActionGroup.tooltipEdiActionGroup") }}</q-tooltip>
            </q-btn>

            <q-btn icon="delete" rounded name="delete" :title=nameBtnDelete
              @click="deleteActionGroupConfirmation(props.row.idActionGroup)"></q-btn>
          </q-td>
        </template>
      </q-table>
    </div>
    <q-inner-loading :showing="loadingPage">
      <q-spinner-gears size="50px" color="primary" />
    </q-inner-loading>
    <ConfirmPopup v-model=showConfirmPopupModel @closeOk="ConfirmDelete()"
:texte="txtConfirm"></ConfirmPopup>
  </q-page>
</template>
```

- Dans la balise Script :

Or les imports des composant VueJS comme `onMounted`, `ref` ou `watch`, et de l'imports du composant enfant `ComfirmPopup`, le remplissage du tableau par les groupes d'action (UC101) lors du montage de la vue suivant le cycle de vie VueJS ce réalise lors du montage de la vue avec la fonction VueJS « `onMonted` » qui appelle une fonction employant le client http Axios en asynchrone pour appeler le Endpoint du serveur correspondant au DTO `ActionGroupDTO`. La construction de la liste des groupes d'action utilisable par la vue se faisant dans un `foreach` sur la liste « `actionList` » du DTO afin de réaliser un objet javascript par ligne correspondant aux colonnes du tableau.

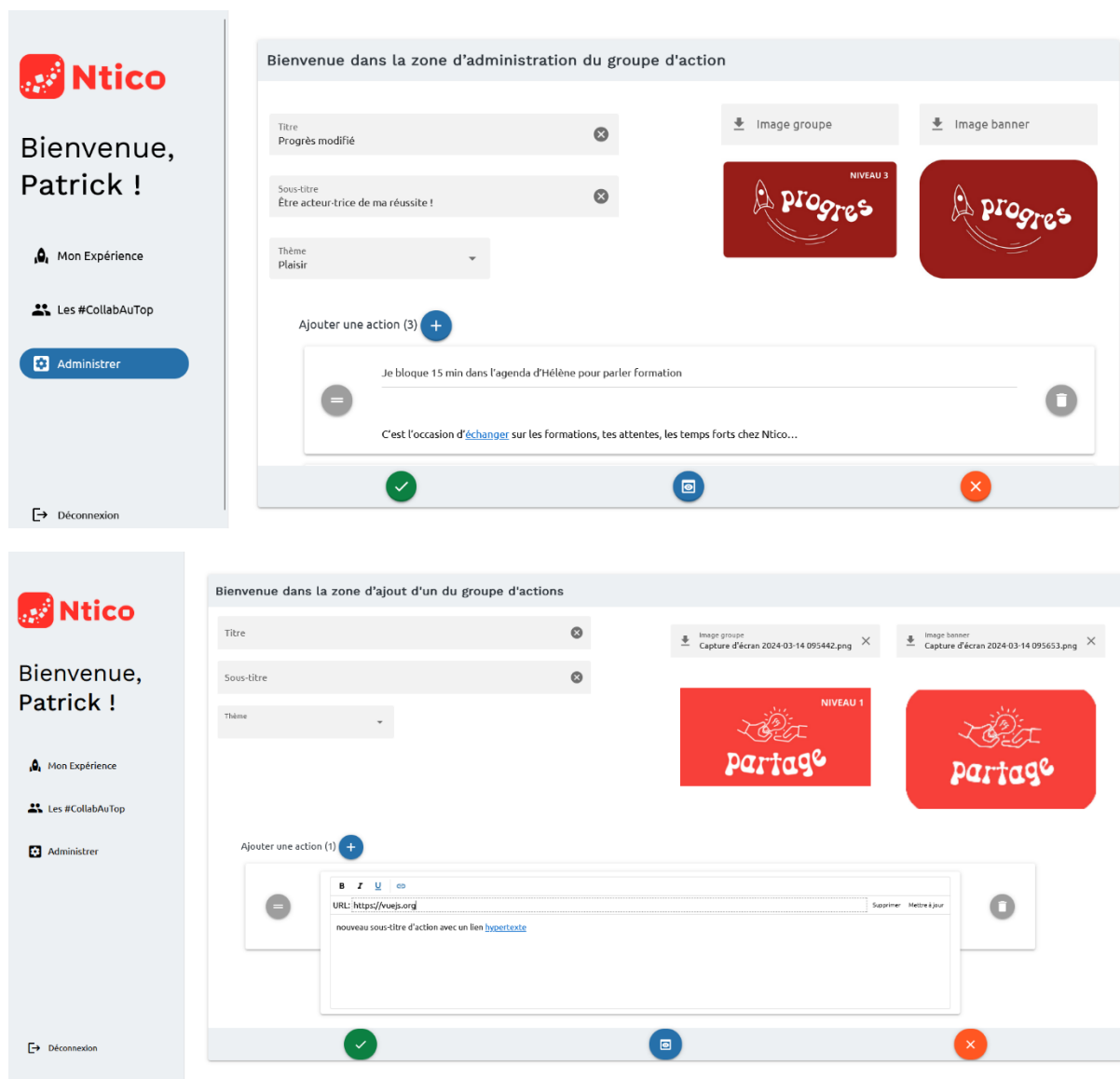
```
onMounted(() => {
  fetchActionsAdmin();
});
function fetchActionsAdmin() {
  loadingPage.value = true;
  api.get(
    `/actiongroup/admin/`
  ).then(response => {
    actionsGroup.value = response.data;
    populateTable();
    loadingPage.value = false;
  }).catch(error => {
    $q.notify({
      color: "negative",
      position: "center",
      message: i18n.t("onActionGroup.qnotify.error.getListActionGroup") + error.message,
      icon: "report_problem",
    });
    loadingPage.value = false;
  });
};
const myRows = ref([]);
function populateTable() {
  actionsGroup.value.forEach((arrayItem) => {
    if (arrayItem.archivedDate !== 'null' || arrayItem.archivedDate !== null) {
      const newRow = {
        idActionGroup: arrayItem.idActionGroup,
        title: arrayItem.title,
        subtitle: arrayItem.subtitle,
        nbactions: arrayItem.numberActions
      };
      myRows.value.push(newRow);
    }
  });
};
```

La réalisation de l'archivage est développée avec la client http Axios et le verbe Delete. Cet archivage est réalisé après confirmation de l'utilisateur au travers de la popup ConfirmPopup qui via le bouton « OK » appelle la méthode ConfirmDelete(). Une fois la promesse du Delete réalisée, la fonction informAndReload() est appelée permettant l'information à l'utilisateur du bon déroulement de l'archivage, et un rafraîchissement de la page au bout de 3s.

```
const deleteActionGroupConfirmation = (id) => {
  idDeleted.value = id;
  txtConfirm.value = i18n.t("onActionGroup.popupConfirm.confirmDeleteActionGroup") + ", id : " + id;
  showConfirmPopupModel.value = true;
};
function informAndReload() {
  $q.notify({
    color: "positive",
    position: "center",
    message: i18n.t("onActionGroup.qnotify.sucess.getListActionGroup"),
    icon: "done",
  });
  setTimeout(() => { location.reload(); }, 3000);
};
const idDeleted = ref();
function ConfirmDelete() {
  api.delete(
    `/actiongroup/admin/${idDeleted.value}`
  ).then(
    informAndReload(),
  ).catch(error => {
    $q.notify({
      color: "negative",
      position: "center",
      message: i18n.t("onActionGroup.qnotify.error.deleteActionGroup") + "id : " + idDeleted.value + ",\n" + error.message,
      icon: "report_problem",
    });
    if (process.env.DEV) {
      console.error(
        "Failed to fetch action group :", + error);
    }
  });
};
```

8.2.2.4 Vue de gestion d'un groupe d'action

- Composant parent : ActionGroupAdminPage.vue
- Composant enfants : ActionQitemSectionCard.vue, ImageActionGroup.vue, InputActionGroup.vue, SelectActionGroupTheme.vue
- Composant enfant type popup : AlertePopup.vue, PreviewPopup.vue



Cette interface est utilisable pour la modification ou la création d'un nouveau groupe d'action.

Elle permet la gestion de l'ensemble des composant d'un groupe d'action, titre, sous-titre, thème, image de l'action-groupe, image de bannière, et des actions associées (ajout, suppression, modification, ordre des actions).

L'ordre des actions est modifiable par drag-and-drop avec la bibliothèque `vue.draggable.next`.

Il est possible d'ajouter de nouvelles actions via le bouton « + » ou d'en supprimer via le bouton « corbeille » sur l'action.

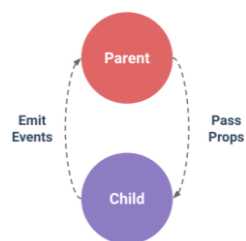
Les sous-titre des actions sont modifiables via un wysiwyg avec la possibilité d'ajout de lien hypertexte, et de mise-en-forme du texte.

Toutes les informations du groupe d'action sont comprises dans une zone scrollable comprenant un formulaire avec ses boutons en pied de page de cette zone : « sauvegarder », « prévisualiser », et « annuler ».

J'ai fait le choix pour cette page de la gérer avec différents composants enfant :

- un composant input, issu du framework Quasar,
- un composant select, issu du framework Quasar,
- un composant image permettant la sélection d'une image sur le poste local et sa visualisation, étant une combinaison de composant Quasar,
- et un composant pour les action comprenant un input et un wysiwyg, issus du framework Quasar, avec un bouton pour la suppression de l'action de la liste des actions du groupe d'action.

La difficulté posée par cet imbriquage de composant, soit directement issus du framework Quasar ou bien développé par mes soins, fut la communication bilatérales entre le parent et ses enfants lors du montage du DOM et de la modification des données par l'utilisateur. Pour ce faire j'ai employé trois principes de VueJS :



- Les « props », permettant au parent d'envoyer des données à l'enfant
- Les « emit », permettant à l'enfant d'envoyer des données à son parent
- Les watch sur les props, permettant à un enfant de mettre à jours ses variables faisant référence aux props utilisées dans sa partie Template et ayant une valeur null à l'initialisation pour le cas d'ajout de groupe d'action.

8.2.2.4.1 Exemple de code sur les actions draggables

```

<div class="row">
  <draggable tag="ul" :list="actionGroupCurrent.actionList"
    item-key="element.idDragAndDrop" @end="onEnd" class="list-group">

    <template #item="{ element, index }">

      <li class="list-group-item listDraggable">
        <ActionQitemSectionCard :titleaction="element.title"
          :subtitleaction="element.subtitle"
          :idDragAndDrop="element.idDragAndDrop" :position="index"
          @removeAction="handleRemovalAction"
          @valuesActionChanged="hadleValuesAction">
        </ActionQitemSectionCard>
      </li>
    </template>
    <template #header>
      <p>Ajouter une action ({{ nbAction }})</p>
      <q-btn push round color="primary" icon="add" @click="addAction" />
    </template>
  </draggable>
</div>

```

Au sein du Template du parent (ActionGroupAdminPage.vue) la zone des actions permet de lister un tableau d'actions et de modifier leur ordre via un drag and drop de celles-ci.

J'y ai fait le choix d'ajouter aux actions dans le frontend un identifiant, « idDragAndDrop » pour les différencier les unes des autres via le « item-key », et non d'utiliser les identifiant de l'entité. Cela fut un choix de ma part car les actions pouvaient être soit mises-à-jour, soit ajoutées, soit supprimées, ou soit ordonnancés différemment.

La librairie permettant cette fonctionnalité fût simple à mettre en œuvre grâce à sa documentation et ses exemples.

Pour permettre au backend de facilement identifier les nouvelles actions à ajouter, et ainsi éviter les méthodes trop complexes dans le back, j'ai fait le choix d'avoir un id négatif pour les actions ajoutées dans la liste des actions du groupe d'action. De ce fait cet identifiant ne pouvait pas être employé comme clé pour le drag and drop. Cette logique se retrouve dans la partie script de la vue parent dans la fonction addAction(). Les actions sont créées en réalisant une copie profonde du squelette définie dans le store (ci-contre) (actionGroupAdmin-store.js)

```
actionCurrent: {  
  idAction: null,  
  idActionGroup: null,  
  title: "new title action",  
  subtitle: "new subtitle action",  
  displayOrder: -1,  
  role: { idRoleCollab: defaultIdRole },  
  creationDate: null,  
  updateDate: null,  
  archivedDate: null,  
  idDragAndDrop: null,  
},
```

```
function addAction() {  
  let newAction = JSON.parse(JSON.stringify(actionGroupAdminStore.actionCurrent));  
  if (nbAction.value !== 0) {  
    let newDisplayOrder =  
actionGroupCurrent.value.actionList[actionGroupCurrent.value.actionList.length -  
1].displayOrder + 100;  
    let newIdDragAndDrop = 0;  
    actionGroupCurrent.value.actionList.forEach((arrayItem) => {  
      if (arrayItem.idDragAndDrop >= newIdDragAndDrop) {  
        newIdDragAndDrop = arrayItem.idDragAndDrop + 1;  
      }  
    });  
    newAction.displayOrder = newDisplayOrder;  
    newAction.idDragAndDrop = newIdDragAndDrop;  
    actionGroupCurrent.value.actionList.push(newAction);  
  } else {  
    newAction.displayOrder = 1100;  
    newAction.idDragAndDrop = 100;  
    actionGroupCurrent.value.actionList.push(newAction);  
  }  
  nbAction.value = nbAction.value + 1;  
};
```

L'ordre des actions, suivant le choix de l'utilisateur, avec la fonctionnalité drag and drop, est géré par la fonction `onEnd()`, déclenchée lors du relâchement du clic gauche de la souris via l'évènement « end » de la bibliothèque `vue.draggable.next`.

Cette fonction ayant pour but de redéfinir la valeur de l'ordre d'affichage qui sera envoyé au backend lors du submit avec le verbe Http Post

```
const onEnd = () => {
  actionGroupCurrent.value.actionList.forEach((arrayItem, index) => {
    arrayItem.displayOrder = ((index + 1) * 100) + 1000;
  });
};
```

8.2.2.4.2 Exemple de composant développé par combinaison de composant Quasar

Composant enfant pour les actions, `ActionQitemSectionCard.vue`

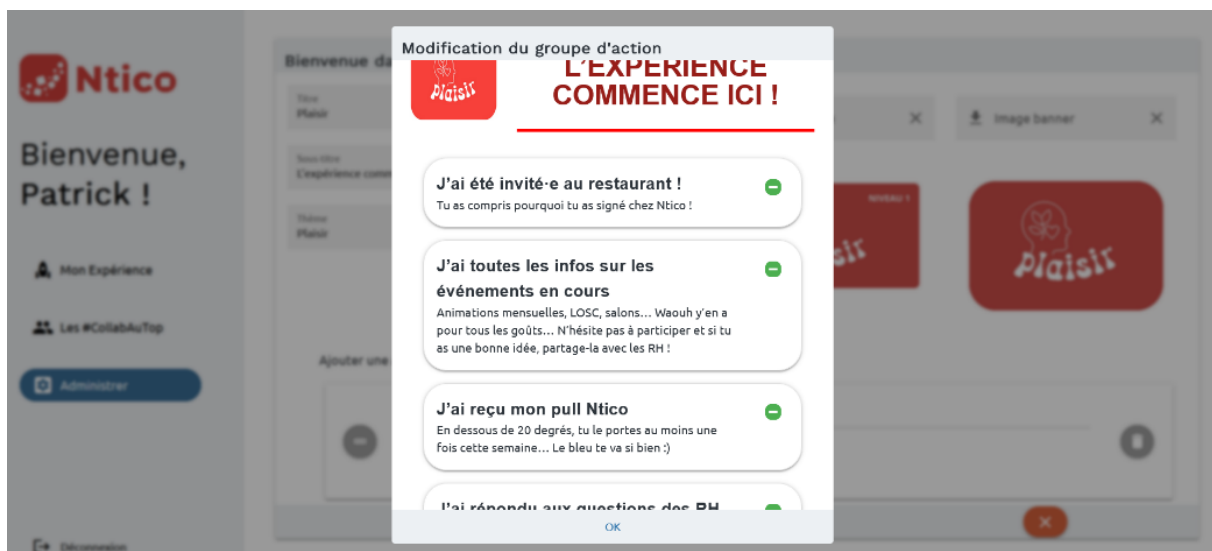
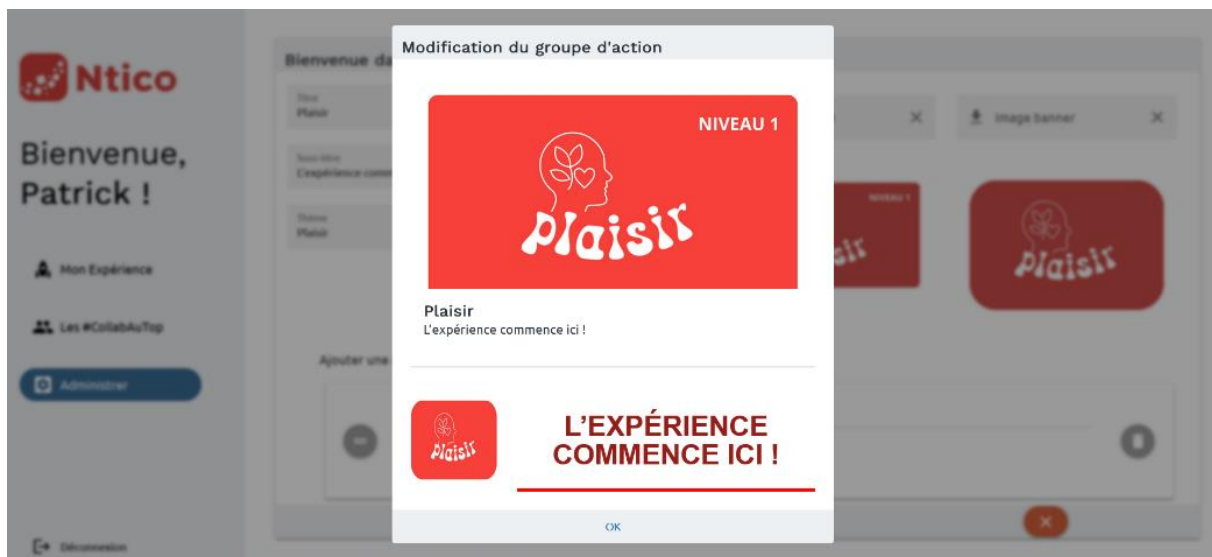
```
<template>
  <q-item-section class="drabbing">
    <div style="margin: 0.5em;">
      <q-card class="myCard text-black">
        <div class="row no-wrap">
          <div class="col-1 centerElmt">
            <q-btn class="dragicon" round color="grey" icon="drag_handle" />
          </div>
          <div class="col-10">
            <q-card-section>
              <q-input autofocus dense v-model="action.title_action" @blur="emitValuesAction"
                :rules="[val => val.length >= 5 || rule]">
            </q-input>
            </q-card-section>
            <q-card-section>
              <div class="text-subtitle2">
                <div class="subtitleaction" v-html="cleanHtml(action.subtitle_action)"></div>
                <q-popup-edit v-model="action.subtitle_action" :validate="val => val.length > 5"
                  auto-save @before-hide="emitValuesAction">
                  <q-editor autofocus dense v-model="action.subtitle_action"
                    :model-value="action.subtitle_action"
                    :toolbar="[['bold', 'italic', 'underline'], ['link']]">
                  </q-editor>
                </q-popup-edit>
              </div>
            </q-card-section>
          </div>
          <div class="col-1 centerElmt">
            <q-btn round color="grey">
          </div>
        </div>
      </q-card>
    </div>
  </q-item-section>
</template>
```

```
function cleanHtml(dirty) {
  const clean = DOMPurify.sanitize(dirty);
  return clean;
}
```

Ce composant comprenant un wysiwyg pour la modification du sous-titre de l'action, accessible par simple clic sur le texte. Celui-ci étant inséré via l'attribut « v-html ». Pour s'abstenir de d'attaque XSS (Cross-site Scripting, injection de code malveillant) j'ai employé la librairie `DOMPurify` via la méthode `cleanHtml()`. Cette librairie transforme en texte toutes les balises html sauf celles sans risque comme les `` et même les `<a/>` pour les liens hypertextes.

8.2.2.4.3 Popup de prévisualisation

Pages de code créés : AlertePopup.vue



Cette interface a pour but la prévisualisation d'un groupe d'action et de ses actions, avec une mise-en-forme identique à celle vue par l'utilisateur final.

```
<template>
  <q-dialog backdrop-filter="blur(2px) saturate(70%)" :auto-close=true>
    <q-layout view="lHh lpr lFf" container>
      <q-header class="headerFooterActionGroup">
        <q-toolbar>
          <q-toolbar-title class="q-mt-md">
            <h4 v-if="idRoute > 0" class="q-mb-lg">{{
              $t("actionGroup.preview.modification") }}</h4>
            <h4 v-else class="q-mb-lg">{{ $t("actionGroup.preview.new") }}</h4>
          </q-toolbar-title>
        </q-toolbar>
      </q-header>
      <q-footer class="headerFooterActionGroup row justify-around">
        <q-btn flat label="OK" color="primary" v-close-popup />
      </q-footer>
    </q-layout>
  </q-dialog>
```



```

</q-footer>
<q-card>
  <q-page-container>
    <q-page padding>
      <q-card class="noShadow">
        <q-card-section class="myImage">
          <q-img v-if="props.actionGroupCurrent.image != null" class="imageCard"
            :src="props.actionGroupCurrent.image" alt="image" :ratio="16 / 9"
fit="fill" />
          <q-img v-else id="imageCard" :src="props.actionGroupOrigin.image"
alt="image" :ratio="16 / 9"
            fit="fill" />
        </q-card-section>
        <q-card-section class="myInfo">
          <h4>{{ props.actionGroupCurrent.title }}</h4>
          <p>{{ props.actionGroupCurrent.subtitle }}</p>
        </q-card-section>
      </q-card>
      <q-separator class="separator" />
      <div class="customContainer">
        <q-img v-if="props.actionGroupCurrent.banner != null"
:src="props.actionGroupCurrent.banner" alt="image"
          class="customImage" :ratio="1 / 1" fit="fill" />
        <q-img v-else :src="props.actionGroupOrigin.banner" alt="image"
class="customImage" :ratio="1 / 1"
          fit="fill" />
        <h2 class="customTitle">{{ props.actionGroupCurrent.subtitle }}</h2>
      </div>
      <q-card-section class="listActions q-pt-none">
        <p v-for="action in props.actionGroupCurrent.actionList" :key="action">
          <q-card class="myCard " bordered>
            <q-card-section horizontal>
              <q-card-section>
                <h6 class="info-title with-left-padding">
                  {{ action.title }}
                </h6>
                <div class="info-description with-left-padding" v-
html="action.subtitle"></div>
              </q-card-section>
              <q-space></q-space>
              <q-card-section class="status">
                <q-checkbox color="grey-4" />
              </q-card-section>
            </q-card-section>
          </q-card>
        </p>
      </q-card-section>
    </q-page>
  </q-page-container>
</q-card>
</q-layout>

```

```
</q-dialog>
</template>
<script setup>
import { ref } from 'vue';
import { useRoute } from "vue-router";
const props = defineProps({
  actionGroupCurrent: {
    type: Object,
  },
  actionGroupOrigin: {
    type: Object,
  },
});
const route = useRoute();
const idRoute = ref(route.params.id ?? null);
</script>
```

8.3 Partie administration du Serveur

Pour le développement de la partie Serveur su projet en Java avec le SDK 17 avec le framework Spring Boot 3, comme pour le frontend, je ne connaissais ni le langage ni le framework employé. J'ai dû dans un premier temps me former pour pouvoir développer les composants backend répondant aux besoins exprimés par le chef de projet.

8.3.1 Remarques sur l'abstraction de haut niveau avec Spring Boot et les bibliothèques du projet

Mon projet de comporté aucune modification de la base de données relationnelle actuelle, ou encore des Entités du projet. Pour autant il m'a fallu comprendre les principes de persistance des données et le mappage qui sont tous deux des abstractions de haut niveau en Java Spring Boot avec des bibliothèques employées dans le projet global :

- **Lombok**, est une bibliothèque Java qui vise à réduire le code dit « boiler plate »²¹, récurrent, dans les projets. Au sein d'un le contexte de Spring Boot, Lombok est souvent utilisé pour simplifier la création de classes de modèle, en éliminant la nécessité de créer manuellement les getters, les setters, les constructeurs et d'autres méthodes de base. En intégrant Lombok dans un projet Spring Boot, il est possible d'annoter les classes avec des annotations telles que `@Data`, `@Getter`, `@Setter`, `@NoArgsConstructor`, `@AllArgsConstructor`, `@EqualsAndHashCode` etc. Ces annotations génèrent automatiquement le code nécessaire lors de la compilation, ce qui le rend plus concis et plus lisible.

Exemple de code autogénéré pour l'entité `ActionGroup` avec l'annotation `@AllArgsConstructor` :

```
public ActionGroup(final Long idActionGroup, @Nullable final String title, @Nullable
final String subtitle, @Nullable final String image, final String banner, final Theme
theme, final Long displayOrder, @Nullable final String openDateCondition, @Nullable
final ZonedDateTime creationDate, @Nullable final ZonedDateTime updateDate, @Nullable
final ZonedDateTime archivedDate, @Nullable final User createdBy, @Nullable final User
updatedBy) {
    this.idActionGroup = idActionGroup;
    this.title = title;
    this.subtitle = subtitle;
    this.image = image;
    this.banner = banner;
    this.theme = theme;
    this.displayOrder = displayOrder;
    this.openDateCondition = openDateCondition;
    this.creationDate = creationDate;
    this.updateDate = updateDate;
    this.archivedDate = archivedDate;
    this.createdBy = createdBy;
    this.updatedBy = updatedBy;
}
```

²¹ Code standard, qui est quasiment le même dans tous les programmes.

- **Hibernate** est une spécification Java qui fournit une interface de programmation standard pour la gestion des données dans les applications Java. Elle simplifie le développement d'applications en permettant de travailler avec des objets Java plutôt qu'avec des requêtes SQL directes lorsqu'ils interagissent avec une base de données relationnelle. Ce concept fonctionne avec des annotations comme pour Lombok :
 - **@Entity**: Cette annotation est utilisée pour marquer une classe Java comme une entité persistante. Elle est généralement placée au-dessus de la classe qui représente une table dans la base de données.
 - **@Table**: Cette annotation est utilisée pour spécifier le nom de la table dans la base de données correspondant à l'entité. Elle peut également être utilisée pour spécifier d'autres propriétés de la table, comme le schéma.
 - **@Id**: Cette annotation est utilisée pour marquer une propriété comme identifiant de l'entité. Cela indique à Hibernate quelle propriété utiliser comme clé primaire dans la table de la base de données.
 - **@GeneratedValue**: Cette annotation est utilisée pour spécifier la stratégie de génération de valeurs pour les clés primaires. Elle peut être utilisée en conjonction avec **@Id** pour indiquer à Hibernate comment générer les valeurs des clés primaires.
 - **@Column**: Cette annotation est utilisée pour mapper une propriété à une colonne de la table dans la base de données. Elle peut être utilisée pour spécifier le nom de la colonne, le type de données, la longueur, etc.
 - **@OneToMany**, **@ManyToOne**, **@ManyToMany**, **@OneToOne**: Ces annotations sont utilisées pour définir les relations entre les entités. Elles permettent de spécifier les associations entre les tables dans la base de données.
 - **@JoinColumn**: Cette annotation est utilisée pour spécifier la colonne de jointure pour une relation entre entités.
- **Jakarta Persistence API (JPA) Query Methods**²², ce module permet par la signature d'une méthode au sein d'un repository de réaliser une requête au SGBD. Ci-dessous un exemple réalisé dans le repository des Actions :

```
List<Action> findByActionGroupOrderByDisplayOrder(ActionGroup actionGroup);
```

Par cette simple signature de méthode, une liste ordonnée d'action suivant leur propriété `DisplayOrder`, appartenant à un groupe d'action, est récupérée.

Spring Boot propose un lexique pour la réalisation de ces méthodes de requête sur son site²³

²² <https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html>

²³ <https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html>

Keyword	Sample	JPQL snippet
Distinct	findDistinctByLastNameAndFirstname	<code>select distinct ... where x.lastname = ?1 and x.firstname = ?2</code>
And	findByLastNameAndFirstname	<code>... where x.lastname = ?1 and x.firstname = ?2</code>
Or	findByLastNameOrFirstname	<code>... where x.lastname = ?1 or x.firstname = ?2</code>
Is, Equals	findByFirstname, findByFirstnameIs, findByFirstnameEquals	<code>... where x.firstname = ?1</code>
Between	findByStartDateBetween	<code>... where x.startDate between ?1 and ?2</code>
LessThan	findByAgeLessThan	<code>... where x.age < ?1</code>
LessThanEqual	findByAgeLessThanEqual	<code>... where x.age <= ?1</code>
GreaterThan	findByAgeGreaterThan	<code>... where x.age > ?1</code>
GreaterThanEqual	findByAgeGreaterThanEqual	<code>... where x.age >= ?1</code>
After	findByStartDateAfter	<code>... where x.startDate > ?1</code>
Before	findByStartDateBefore	<code>... where x.startDate < ?1</code>
IsNull, Null	findByAge(Is)Null	<code>... where x.age is null</code>
IsNotNull, NotNull	findByAge(Is)NotNull	<code>... where x.age not null</code>
Like	findByFirstnameLike	<code>... where x.firstname like ?1</code>
NotLike	findByFirstnameNotLike	<code>... where x.firstname not like ?1</code>

- **CrudRepository**²⁴ sont des interfaces fournies par Spring Data JPA qui simplifient l'accès aux données en abstrayant les opérations CRUD (Create, Read, Update, Delete)²⁵ sur les entités persistantes. Ces interfaces héritent de l'interface Repository de Spring Data JPA et fournissent des méthodes prédéfinies pour effectuer des opérations courantes sur les entités annotées @Entity, avec sa propriété @Id annotée aussi, sans avoir besoin d'écrire de requêtes SQL explicites. Les interfaces CrudRepository fournissent des méthodes prédéfinies telles que save, findById, findAll, delete, etc., qui permettent d'effectuer les opérations CRUD de base sur les entités associées.

Table 1. Query subject keywords

Keyword	Description
find...By, read...By, get...By, query...By, search...By, stream...By	General query method returning typically the repository type, a Collection or Streamable subtype or a result wrapper such as Page, GeoResults or any other store-specific result wrapper. Can be used as findBy..., findMyDomainTypeBy... or in combination with additional keywords.
exists...By	Exists projection, returning typically a boolean result.
count...By	Count projection returning a numeric result.
delete...By, remove...By	Delete query method returning either no result (void) or the delete count.
...First<number>..., ...Top<number>...	Limit the query results to the first <number> of results. This keyword can occur in any place of the subject between find (and the other keywords) and by .
...Distinct...	Use a distinct query to return only unique results. Consult the store-specific documentation whether that feature is supported. This keyword can occur in any place of the subject between find (and the other keywords) and by .

- MapStruct²⁶ est une bibliothèque Java open-source qui simplifie la tâche du mappage d'objets en générant automatiquement le code de mappage à partir d'interfaces Java annotées. Au lieu d'écrire manuellement des méthodes de mappage, il suffit de définir simplement une interface source et une interface cible, et MapStruct se charge du reste. Il génère un code Java efficace qui effectue le mappage entre les deux objets.

Exemple avec le mapper des groupes d'actions, le fichier ActionGroupMapper.java :

```
@Mapper(uses = {DateMapper.class, UserMapper.class, ActionMapper.class})
public interface ActionGroupMapper {

    ActionGroupDTO fromActionGroup(ActionGroup actionGroup);
    ActionGroupAdminDTO fromActionGroup(ActionGroup actionGroup);
    ActionGroup fromActionGroupDTO(ActionGroupAdminDTO actionGroupAdmin);
}
```

²⁴ <https://howtodoinjava.com/spring-data/spring-crudrepository-listcrudrepository/>

²⁵ <https://docs.spring.io/spring-data/jpa/reference/repositories/query-keywords-reference.html>

²⁶ <https://mapstruct.org/documentation/1.5/reference/html/>

La méthode `fromActionGroupDTO`, à la suite de la compilation est réalisé par la bibliothèque MapStruct comme ce-ci :

```
@Override
public ActionGroup fromActionGroupDTO(ActionGroupAdminDTO actionGroupAdmin) {
    if ( actionGroupAdmin == null ) {
        return null;
    }

    ActionGroup actionGroup = new ActionGroup();

    actionGroup.setIdActionGroup( actionGroupAdmin.getIdActionGroup() );
    actionGroup.setTitle( actionGroupAdmin.getTitle() );
    actionGroup.setSubtitle( actionGroupAdmin.getSubtitle() );
    actionGroup.setImage( actionGroupAdmin.getImage() );
    actionGroup.setBanner( actionGroupAdmin.getBanner() );
    actionGroup.setTheme( actionGroupAdmin.getTheme() );
    actionGroup.setDisplayOrder( actionGroupAdmin.getDisplayOrder() );
    actionGroup.setOpenDateCondition( actionGroupAdmin.getOpenDateCondition() );
    actionGroup.setCreationDate( dateMapper.toZonedDateTime( actionGroupAdmin.getCreationDate() ) );
    actionGroup.setUpdateDate( dateMapper.toZonedDateTime( actionGroupAdmin.getUpdateDate() ) );
    actionGroup.setArchivedDate( dateMapper.toZonedDateTime( actionGroupAdmin.getArchivedDate() ) );
    actionGroup.setCreatedBy( userMapper.fromDTO( actionGroupAdmin.getCreatedBy() ) );
    actionGroup.setUpdatedBy( userMapper.fromDTO( actionGroupAdmin.getUpdatedBy() ) );

    return actionGroup;
}
```

8.3.2 Développement de la partie administration du Serveur

Le développement de la partie Serveur de l'application c'est déroulé en flux suivant les cas d'utilisation. Cela en miroir du développement du frontend.

8.3.2.1 Contrôleurs ajoutés à l'API REST

Les contrôleurs de cette API devaient seulement appeler le service et la méthode correspondants au besoin de la fonctionnalité développée. La gestion des exceptions étant déjà développée je devais respecter la pratique employée à l'ESN Ntico, sachant que dans l'environnement de développement la BaseApiException renvoyée un message explicite avec le type de réponse http correspondant à l'erreur levée. Dans l'environnement de production, elle ne renverrait qu'une erreur standard, sans message explicite.

GET	<pre>@GetMapping("/admin/") public List<ActionGroupDTO> getAllActionGroupsWithNumberActions() throws BaseApiException { return actionGroupService.getAllActionGroupsWithNumberActions(); }</pre>
GET	<pre>@GetMapping("/admin/{idActionGroup}") public ActionGroupAdminDTO getActionGroupByIdWithActions(@PathVariable Long idActionGroup) throws BaseApiException { return actionGroupService.getActionGroupByIdWithActions(idActionGroup); }</pre>
DELETE	<pre>@DeleteMapping("/admin/{id}") public void archiveActionGroupAdmin(@PathVariable Long id) throws BaseApiException { actionGroupService.archiveAllActionsAndActionGroup(id); }</pre>
POST	<pre>@PostMapping("/admin/") @ResponseStatus(code = HttpStatus.OK) public ActionGroupAdminDTO create(@RequestBody ActionGroupAdminDTO actionGroupAdmin) throws BaseApiException { return actionGroupService.createActionGroupAndActions(actionGroupAdmin); }</pre>
PUT	<pre>@PutMapping("/admin/{id}") @ResponseStatus(code = HttpStatus.OK) public ActionGroupAdminDTO update(@PathVariable Long id, @RequestBody ActionGroupAdminDTO actionGroupAdminDTO) throws BaseApiException { return actionGroupService.updateActionGroupAndActions(actionGroupAdminDTO); }</pre>

8.3.2.2 Méthodes de Service ajoutés à l'API REST

Pour développement des méthodes du Service ActionGroupService, je devais au maximum employer les méthodes déjà existantes. Et si ces méthodes ne convenaient pas ou entrées en conflit avec celles que je développée, comme une injection circulaire entre deux services, je devais résoudre ces problématiques tout en faisant part aux Référents avant modifications.

Lors du développement j'ai rencontré une injection de dépendance circulaire entre les services ActionService, et ActionGroupService. Celle-ci était induite par le fait qu'avant le début de mon développement une action n'était pas dépendante d'un groupe d'action, mais appelait ActionGroupService au besoin pour ajouter l'id du groupe d'action à l'entité Action. De ce fait j'ai dû modifier les méthodes la création et la modification des actions pour qu'elles n'appellent plus le service des groupes d'action, pour y récupérer l'id du groupe, mais que ce soit ActionGroupService qui réalise l'attribution de l'id du groupe à une action.

8.3.2.2.1 Signature des méthodes réalisées :

- ActionGroupController :

```
public List<ActionGroupDTO> getAllActionGroupsWithNumberActions() throws
BaseApiException{...}

public ActionGroupAdminDTO getActionGroupByIdWithActions(@PathVariable Long
idActionGroup) throws BaseApiException{...}
public void archiveActionGroupAdmin(@PathVariable Long id) throws BaseApiException {...}

public ActionGroupAdminDTO create(@RequestBody ActionGroupAdminDTO actionGroupAdmin)
throws BaseApiException {...}

public ActionGroupAdminDTO update(@PathVariable Long id,@RequestBody
ActionGroupAdminDTO actionGroupAdminDTO) throws BaseApiException {...}
```

ActionGroupService :

```
public List<ActionGroup> getAllActionGroupsSortedByUpdateDate() {...}

public ActionGroupAdminDTO getActionGroupByIdWithActions(Long idActionGroup) throws
BaseApiException{...}

public List<ActionGroupDTO> getAllActionGroupsWithCountActions()throws
BaseApiException{...}

public void archiveActionGroupAndItsActions(Long idActionGroup)throws
BaseApiException{...}

private void archiveActionGroup(ActionGroup actionGroup) throws BaseApiException{...}

public ActionGroupAdminDTO createActionGroupAndActions(ActionGroupAdminDTO
actionGroupAdminDTO) throws BaseApiException {...}
```

```
private List<Action> findAllActiveActionGroup(ActionGroup actionGroup){...}

public ActionGroupAdminDTO updateActionGroupAndActions(ActionGroupAdminDTO
actionGroupAdminDTO) throws BaseApiException {...}
```

- ActionService :

```
public Action updateActionAdmin(Action action, ActionAdminDTO actionAdminDTO) throws
BaseApiException {...}

public List<Action> addActions(List<Action> listAction, ActionGroup actionGroup) throws
BaseApiException {...}

public List<Action> archiveActions(List<Action> listAction, ActionGroup actionGroup)
throws BaseApiException {
```

- ActionRepository :

```
List<Action> findByActionGroupOrderByDisplayOrder(ActionGroup actionGroup);

List<Action> findByActionGroupAndArchivedDateNull(ActionGroup myActionGroup);
```

ActionGroupRepository :

```
List<Action> findAllActionGroupOrderByUpdateDateDesc();
```

ActionGroupMapper :

```
ActionGroupAdminDTO toActionGroupAdminDTO(ActionGroup actionGroup);

ActionGroup fromActionGroupDTO(ActionGroupAdminDTO actionGroupAdmin);
```

ActionMapper :

```
ActionAdminDTO toActionAdminDTO(Action action);

Action fromActionAdminDTO(ActionAdminDTO actionAdminDTO);
```

Remarque :

Une des particularités de Java Spring Boot est les transactions. Pour réaliser des transactions dans des méthodes modifiant plusieurs Entités, il suffit d'indiquer par l'annotation `@Transactional` en en-tête de méthode. Cette abstraction comme celle de l'injection automatique des dépendances avec `@Autowired` sur les propriétés d'une classe de contrôleur ou de service, évitent l'écriture de code récurrent. Ces annotations sont surprenantes et déroutantes, mais elles facilitent la lecture du code et sa réalisation.

8.3.2.2.2 Méthode de service Update d'un groupe d'action

Dans mon développement de la partie administration du projet XPérienCe, la méthode de mise-à-jour d'un groupe d'action demandait d'ajouter, de modifier ou d'archiver des actions d'un groupe d'action en plus de sa mise-à-jours. En front j'ai fait le choix d'envoyer les nouvelles actions avec un Id négatif, permettant au back de les identifier avec ce discriminant. Pour celles devant être mise-à-jour ou à archiver, une comparaison de la liste des actions actuelles et celles envoyée par le front, était pour moi le chemin le plus rapide. Afin d'identifier celles à archiver j'ai défini une nouvelle liste copiant la liste des actions actuellement en base de données à laquelle je rétractais les actions toujours présente dans la liste envoyée par le front. De ce fait les actions restantes dans la liste à archiver sont celles devant être archivées.

```
@Transactional
public ActionGroupAdminDTO updateActionGroupAndActions(ActionGroupAdminDTO actionGroupAdminDTO) throws BaseApiException {
    if(actionGroupAdminDTO==null) {
        throw new IncorrectDataException("le groupe d'action ne peut être null ");
    }
    if(actionGroupAdminDTO.getIdActionGroup()==null) {
        throw new IncorrectDataException("l'identifiant du groupe d'action ne peut être null ");
    }
    ActionGroup myActionGroup = actionGroupMapper.fromActionGroupDTO(actionGroupAdminDTO);

    if((actionGroupAdminDTO.getImage() != null) && actionGroupAdminDTO.getImage().contains("base64")) {
        myActionGroup.setImage(actionGroupAdminDTO.getImage().split(",")[1]);
    }
    if((actionGroupAdminDTO.getBanner() != null) && actionGroupAdminDTO.getBanner().contains("base64")) {
        myActionGroup.setBanner(actionGroupAdminDTO.getBanner().split(",")[1]);
    }
    updateActionGroup(myActionGroup);
    List<Action> myListActionsCurrent = actionRepository.findByActionGroup(myActionGroup);
    List<Action> myListActionSent = new ArrayList<>();
    for (ActionAdminDTO actionAdminDTO : actionGroupAdminDTO.getActionList()){
        Action action = actionMapper.fromDTO(actionAdminDTO);
        myListActionSent.add(action);
    }
    List<Action> myListActionsForArchive = new ArrayList<>(myListActionsCurrent);
    for (Action actionSent : myListActionSent){
        if(actionSent.getIdAction() < 0) {
            addNewActionFromActionGroup(actionSent, myActionGroup);
        }else {
            for (Action actionCurrent : myListActionsCurrent) {
                if(actionSent.getIdAction() == actionCurrent.getIdAction() && actionCurrent.getArchivedDate()== null) {
                    updateActionFromActionGroup(actionSent, myActionGroup);
                    myListActionsForArchive.remove(actionCurrent);
                }
            }
        }
    }
    if(myListActionsForArchive.size() > 0) {
        for (Action actionCurrent : myListActionsForArchive) {
            if(actionCurrent.getArchivedDate() == null) {
                archiveActionFromActionGroup(actionCurrent, myActionGroup);
            }
        }
    }
    return this.getActionGroupByIdWithActions(myActionGroup.getIdActionGroup());
}

private void addNewActionFromActionGroup(Action action, ActionGroup actionGroup){
    action.setActionGroup(actionGroup);
    actionService.add(action);
}

private void updateActionFromActionGroup(Action action, ActionGroup actionGroup){
    action.setActionGroup(actionGroup);
    actionService.update(action);
}

private void archiveActionFromActionGroup(Action action, ActionGroup actionGroup){
    action.setActionGroup(actionGroup);
    action.setArchivedDate(ZonedDateTime.now());
    actionService.update(action);
}
```

8.3.2.2.3 Restructuration de la méthode de mise à jour de groupe d'action

Par suite de l'emploi de l'analyser de code SonarLint²⁷ avec l'IDE IntelliJ, sur la complexité cyclomatique²⁸ de cette méthode, j'ai cherché à l'améliorer tout en employant les particularismes de Java Spring Boot. Pour ce faire :

- J'ai déclaré des propriétés privées à la classe pour les listes des actions à ajouter, mettre à jour et à archiver
- J'ai dans un premier temps modifier le client web pour qu'il me fournisse des actions à ajouter avec un identifiant à nul, pour permettre une lecture plus facile par d'autre développeur.
- J'ai par la suite décomposé cette méthode pour séparer les fonctionnalités, comme le traitement de l'image.
- Je l'ai aussi reformaté avec l'API Stream²⁹ de Java 8, permettant de traiter des collections d'objet de manière déclarative, plutôt qu'avec des boucles standards, tout en pouvant chaîner les opérations comme filtrage.
- J'ai aussi utilisé la méthode removeAll, applicable sur des listes en tableau (ArrayList), permettant la suppression d'éléments dans une liste. Pour cette astuce, il me fallait porter une attention particulière sur la provenance des objets de la liste : est-ce que cet objet est un objet JPA ou non. Sachant que toute appel à la base de données produit des objets dit JPA à contrario des objets envoyés par le client, même si ces deux sont des instanciations de la même Entité, il ne pointe pas vers le même objet.
- J'ai de plus employé un dictionnaire, avec la classe Map³⁰ pour mettre en regard les DTO des action reçus avec l'entité action en base de données.
- J'ai enfin employé la classe Optional³¹ pour vérifier la présence d'éléments facultatif. Et ainsi éviter l'exception NullPointerException.

²⁷ <https://www.sonarsource.com/products/sonarLint/>

²⁸ https://fr.wikipedia.org/wiki/Nombre_cyclomatique

²⁹ <https://www.jmdoudoux.fr/java/dej/chap-streams.htm>

³⁰ <https://www.codejava.net/java-core/collections/java-map-collection-tutorial-and-examples>

³¹ <https://www.developpez.com/actu/134958/Java-apprendre-a-utiliser-la-classe-Optional-lt-T-gt-pour-eviter-d-utiliser-explicitement-null-par-Gugelhupf/>

```
private List<Action> AdminActionsToUpdate = new ArrayList<>();
private List<Action> adminActionsToAdd = new ArrayList<>();
private List<Action> adminActionsToArchive;
private Map<Action, ActionAdminDTO> adminActionToActionAdminDTOMap = new HashMap<>();
```

```
@Transactional
public ActionGroupAdminDTO updateActionGroupAndActions(ActionGroupAdminDTO actionGroupAdminDTO)
    throws BaseApiException {
    testActionGroupAdminDTO(actionGroupAdminDTO);
    ActionGroup myActionGroup = actionGroupMapper.fromActionGroupDTO(actionGroupAdminDTO);
    updateActionGroup(FormatImageUtils.formatAndSetImageAndBanner(actionGroupAdminDTO, myActionGroup));
    defineActionTypeFromActionGroupDTO(actionGroupAdminDTO);
    List<Action> actionsActive = findAllActiveActionGroup(myActionGroup);
    actionsActive.removeAll(AdminActionsToUpdate);
    adminActionsToArchive = new ArrayList<>(actionsActive);
    addActionFromActionGroupDTO(adminActionsToAdd, myActionGroup);
    updateActionFromActionGroupDTO(AdminActionsToUpdate, actionToActionAdminDTOMap);
    archiveActionFromActionGroupDTO(adminActionsToArchive, myActionGroup);
    return getActionGroupByIdWithActions(myActionGroup.getIdActionGroup());
}

private bool testActionGroupAdminDTO(ActionGroupAdminDTO actionGroupAdminDTO) {
    if (actionGroupAdminDTO == null) {
        throw new IncorrectDataException("le groupe d'action ne peut être null ");
    }
    if (actionGroupAdminDTO.getIdActionGroup() == null) {
        throw new IncorrectDataException("l'identifiant du groupe d'action ne peut être null ");
    }
    if (actionGroupAdminDTO.getIdActionGroup() < 0) {
        throw new IncorrectDataException("l'identifiant du groupe d'action ne peut être null ");
    }
    return true;
}

private void defineActionTypeFromActionGroupDTO(ActionGroupAdminDTO actionGroupAdminDTO) {
    for (ActionAdminDTO actionAdminDTO : actionGroupAdminDTO.getActionList()) {
        if (actionAdminDTO.getIdAction() == null) {
            adminActionsToAdd.add(actionMapper.fromActionAdminDTO(actionAdminDTO));
        } else {
            Optional<Action> action = actionRepository.findById(actionAdminDTO.getIdAction());
            if (action.isPresent()) {
                adminActionToActionAdminDTOMap.put(action.get(), actionAdminDTO);
                adminActionsToUpdate.add(action.get());
            } else {
                adminActionsToAdd.add(actionMapper.fromActionAdminDTO(actionAdminDTO));
            }
        }
    }
}

private bool addActionFromActionGroupDTO(List<Action> actionsToAdd, ActionGroup myActionGroup) {
    if (!actionsToAdd.isEmpty()) {
        actionService.addActions(actionsToAdd, myActionGroup);
        return true;
    } else {
        return false;
    }
}

private bool updateActionFromActionGroupDTO(List<Action> actionsToUpdate,
    Map<Action, ActionAdminDTO> actionToActionAdminDTOMap) {
    if (!actionsToUpdate.isEmpty()) {
        for (Map.Entry<Action, ActionAdminDTO> entry : actionToActionAdminDTOMap.entrySet()) {
            Action action = entry.getKey();
            ActionAdminDTO actionAdminDTO = entry.getValue();
            actionService.updateActionAdmin(action, actionAdminDTO);
        }
        return true;
    } else {
        return false;
    }
}

private bool archiveActionFromActionGroupDTO(List<Action> actionsToArchive, ActionGroup myActionGroup) {
    if (!actionsToArchive.isEmpty()) {
        actionService.archiveActions(actionsToArchive, myActionGroup);
        return true;
    } else {
        return false;
    }
}
}
```

9 Description d'une situation de travail ayant nécessité une recherche

L'ensemble de ma conception en back et en front m'ont demandé de nombreuse recherche, puisque je ne connaissais pas l'ensemble des framework utilisés dans le projet avant la semaine du 13 mars. J'ai dû donc effectuer de nombreuse recherche tout au long du projet.

Une des plus pertinentes fût celle concernant le client http Axios pour la partie front.

Le choix de ce client vient du fait qu'à contrario de Fetch, intégré dans tous les navigateurs modernes, le client Axios permet le traitement des code erreur http. Ainsi il rejettera la promesse de demande si le serveur renvoie un code de réponse supérieur ou égal à 400.

Pour me documenter sur cette librairie, j'ai consulté la documentation officielle sur de l'API Axios :

- https://axios-http.com/docs/api_intro,

Et sur les notions de promesse Javascript sur :

- https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Using_promises
- <https://fr.javascript.info/promise-basics>.

Dans un premier temps je me suis familiarisé avec la notion de promesse et les différentes formes d'écriture de celle-ci, dont le chainage. Cette dernière forme fût celle que j'ai employée pour mes différentes requêtes avec l'API Axios. De plus les promesses Javascript sont nativement asynchrones.

Exemple avec la requête de mise-à-jour d'un groupe d'action au sien du formulaire de création ou de modification :

```
function myPut(actionGroup) {
  api.put(
    `/actiongroup/admin/${actionGroup.idActionGroup}`,
    actionGroup
  )
  .then(response => {
    resultPut.value = response.data;
  })
  .catch(error => {
    $q.notify({
      color: "negative",
      position: "center",
      message: "".concat(i18n.t("onActionGroup.qnotify.error.putActionGroup"), " ",
error.message),
      icon: "report_problem",
    });
    if (process.env.DEV) {
      console.error(
        "".concat("Failed to put an action group :", " ", error));
    }
  });
}
```

10 Conclusion

Après avoir suivi une formation de reconversion professionnelle de 15 mois suite à un accident de travail, visant l'obtention du titre professionnel de Conception Développement d'Applications à l'ESRP 2isa de Millau, j'ai intégré l'ESN Ntico pour ma période d'application en entreprise en fin de formation. Cette phase de trois mois avait pour objectif de m'exposer au métier de concepteur et développeur au sein d'une équipe expérimentée de développeurs full stack.

Pour faciliter mon intégration dans cette ESN, j'ai suivi pendant un mois une formation sur l'ordonnanceur OpCon de l'entreprise SMA, dont Ntico est le revendeur officiel dans les régions francophones de l'Europe. Cette période m'a permis de me familiariser avec ce logiciel, qui représente une part importante des activités chez Ntico tout en m'intégrant dans l'entreprise.

Par la suite, j'ai rejoint une équipe de huit développeurs répartis sur trois sites en France pour continuer le développement de l'application interne XPérienCe, destinée au suivi du parcours professionnel et de formation des employés chez Ntico. Mon rôle était de contribuer à la partie administration de cette application web. Pour ce faire, j'ai étudié les besoins exprimés par le chef de projet ainsi que les technologies utilisées, tant en front-end qu'en back-end. Étant donné que je n'avais aucune eXPérienCe préalable avec les langages utilisés dans le projet, j'ai consacré une dizaine de jours à me former à VueJS, son framework Quasar, Java et son framework Spring Boot.

Ensuite, j'ai commencé à répondre aux exigences du projet en travaillant selon le découpage des cas d'utilisation défini lors de l'analyse des besoins. En suivant la méthode PDCA et en bénéficiant des conseils des référents, j'ai progressivement développé chaque fonctionnalité en respectant les bonnes pratiques de codage de Ntico. Malgré mon engagement dans le développement de la partie administration du projet XPérienCe, j'ai été confronté à plusieurs défis techniques auxquels j'ai dû rapidement et professionnellement répondre, compte tenu du temps imparti. Par exemple, j'ai dû surmonter le niveau d'abstraction élevé du framework Spring Boot et me familiariser avec le pseudo-langage et le cycle de vie des composants de VueJS.

Aucun des langages ou frameworks utilisés dans le projet n'avait été abordé au cours de ma formation de 15 mois, ce qui a représenté un défi que j'ai relevé

Concernant le frontend avec VueJS et son framework Quasar, j'ai pris beaucoup de plaisir à développer. Après une phase doute tant les pseudos langages sont éloignées du pure Javascript HTML, j'ai pris confiance dans capacité à fournir une interface agréable à utiliser et codée suivant les standards de ces framework. Pour ce faire je me suis beaucoup documenté au travers de leurs documentations et de différents sites de développeurs qui mettaient en avant leurs particularismes. Comme la méthode « watch » observant une variable et pouvant fournir son état avant et après modification. Dans une phase deux de ce projet, si le temps me l'aurait permis, j'aurais employé cette fonctionnalité pour réaliser un historique sur les modifications utilisateur d'un action groupe e ainsi lui permettre des retours en arrière sur chaque donnée spécifiquement. Je l'aurais combiné avec un store Pinia pour connaître l'état de l'application à chaque instant.

Pour ce qui est du backend avec Java Spring Boot, mes interventions sur l'API REST déjà développée furent limitées au CRUD de la partie administration. De ce fait elle n'intégrait pas toute la définition de l'architecture métier ni les paramétrages des différentes bibliothèques employées. Je devais fondre mon développement dans celui déjà présent en employant au maximum les méthodes de CRUD sur les actions et les groupes d'action déjà développés. Mon développement concerné principalement la logique métier dans le respect des consignes du chef de projet. Dans une phase deux j'aurais mis en place une autre gestion des images afin de limiter leurs téléchargements depuis le SGBD. Vu que leur nombre est limité, une dizaine, je les aurais intégrées de manière optimisée au front, du fait que chaque utilisateur télécharge l'ensemble dans une navigation normale. De plus j'aurais mis en place des DTO spécifique pour les requêtes et les réponses en utilisant que des identifiants et non des entités. Cela aurait demandé plus de calcul pour employer la bibliothèque MapStruct ou de JPA lors des attaques à la base de données, mais éviterait un alourdissement inutile de transfert de données avec le client.

Au bout de sept semaines et avec à l'assistance de Vincent le développeur junior, Guillaume un des développeurs seniors et Aurélien, le chef de projet, j'ai réussi à développer une partie de l'administration de l'application web XPérience répondant aux exigences formulées. Cette expérience de travail dans un environnement professionnel m'a conforté dans ma reconversion de l'industrie lourde vers la conception et le développement logiciel, tout en m'offrant l'opportunité d'acquérir de nouvelles compétences très utiles sur le marché du travail. Ces compétences seront particulièrement pertinentes pour ma poursuite d'études en alternance pour devenir Architecte logiciel ou pour trouver un emploi de développeur full-stack junior dans les semaines à venir.

Je tiens à exprimer ma profonde gratitude envers mes formateurs, qui m'ont accompagné tout au long de ma formation, ainsi qu'à toute l'équipe de Ntcio pour m'avoir permis d'acquérir de nouvelles compétences au sein d'une équipe de développeurs professionnels et agréable à vivre.