**Dossier de Projet**

Projet WEATHER

Du 29 août 2022 au 7 juillet 2023



Préparation au titre de Concepteur·trice Développeur·se d’Applications

Tudwal CORLOUER



Code RNCP : RNCP31114

Table des matières

[1 - Remerciements 4](#_Toc137542489)

[2 - Présentation de l’entreprise 5](#_Toc137542490)

[3 - Liste des compétences du référentiel qui sont couvertes par le projet 6](#_Toc137542491)

[4 - Résumé du projet (ANGLAIS) 7](#_Toc137542492)

[5 - Cahier des charges 8](#_Toc137542493)

[Présentation 8](#_Toc137542494)

[Expression des besoins 8](#_Toc137542495)

[Spécifications fonctionnelles 9](#_Toc137542496)

[6 - Gestion de projet 10](#_Toc137542497)

[Introduction 10](#_Toc137542498)

[Outils de gestion de projet 11](#_Toc137542499)

[Workflow 11](#_Toc137542500)

[7 - Spécifications technique 13](#_Toc137542501)

[Présentation 13](#_Toc137542502)

[Sécurité et gestion des utilisateurs 13](#_Toc137542503)

[Maquettes 15](#_Toc137542504)

[Schéma de base de données 16](#_Toc137542505)

[Ressources logicielles 17](#_Toc137542506)

[Architecture du projet 18](#_Toc137542507)

[8 - Réalisations 20](#_Toc137542508)

[Présentation 20](#_Toc137542509)

[Réalisation de la base de données 20](#_Toc137542510)

[Réalisation des composants d’accès aux données 22](#_Toc137542511)

[Réalisation des contrôleurs et services côté back 23](#_Toc137542512)

[Partie Front-end et IHMS 24](#_Toc137542513)

[9 - Présentation de la fonctionnalité la plus représentative 26](#_Toc137542514)

[Présentation 26](#_Toc137542515)

[Utilisation 26](#_Toc137542516)

[Réalisation 27](#_Toc137542517)

[Test 30](#_Toc137542518)

[Conclusion 31](#_Toc137542519)

[10 - Sécurité 32](#_Toc137542520)

[Présentation 32](#_Toc137542521)

[Mise en œuvre 32](#_Toc137542522)

[Conclusion 33](#_Toc137542523)

[11 - Recherche effectuée 34](#_Toc137542524)

[Le besoin d’information 34](#_Toc137542525)

[Le problème 35](#_Toc137542526)

[La recherche 35](#_Toc137542527)

[La solution 37](#_Toc137542528)

[La mise en œuvre 37](#_Toc137542529)

[13 - Conclusion 39](#_Toc137542530)

[Annexes 40](#_Toc137542531)

# 1 - Remerciements

Tout d’abord, je tiens à remercier l’ENI pour m’avoir permis de concrétiser cette reconversion professionnelle.

Ensuite, je remercie l’ensemble des formateurs que j’ai pu avoir, qui tous, à leurs manières ont su me pousser et me conforter dans mon choix de réorientation.

Enfin, je tiens à remercier toutes les équipes de SII Ouest avec qui j’ai eu le plaisir de travailler et plus particulièrement Mme Pascaline HALLARD qui a su m’accueillir au sein de son équipe et me faire évoluer aussi bien humainement que professionnellement. Je tenais également à remercier M. Dominique LABOUR qui m’a suivie durant toute la fin de mon alternance, et pour finir, je remercierais tous les membres de l’équipe WEATHER avec qui j’ai pu travailler.

# 2 - Présentation de l’entreprise

Pour préparer mon titre CDA (Concepteur Développeur d’Application) j’ai eu la chance de faire mon alternance au sein du groupe SII (Société pour l’informatique industrielle), entreprise fondé par Bernard Huvé en 1979. C’est une Entreprise de Services du Numérique (ESN) à dimension internationale composé de 13 000 collaborateurs répartie sur une centaine d’implantations aux quatre coins du monde. En France SII, c’est 9 agences et plus de 4 700 collaborateurs. J’ai rejoint l’agence SII Ouest basée sur Rennes en septembre 2022. SII a pour but d’accompagner les entreprises dans l’intégration des nouvelles technologies et méthodes de management de l’innovation pour faire évoluer les systèmes d’informations et contribuer au développement de leurs futurs produits ou services. Le groupe SII intervient dans différents domaines tel que l’aéronautique, le spatial, la défense, les télécoms ou encore le secteur bancaire et celui des assurances.

J’ai d’abord commencé mon alternance au sein de l’équipe WEATHER composée initialement de Pascaline HALLARD, ma tutrice et également cheffe de projet sur le projet WEATHER et de Julien PERRIER alternant lui aussi. Au cours de mon temps passé dans cette équipe, plusieurs développeurs en inter-contrats nous ont rejoints et ont su chacun m’apporter leurs connaissances. J’ai ensuite rejoint l’équipe Dev&Co qui fait de la Tierce Maintenance Applicative (TMA) sur différents projets clients. Ce passage au sein de deux équipes m’a permis de voir différents projets et d’autres méthodes de travail.

# 3 - Liste des compétences du référentiel qui sont couvertes par le projet

Le projet WEATHER m’a permis de mettre en application des compétences que j’ai pu apprendre au cours de ma première année au sein de l’ENI ainsi que durant mon précédent stage :

* Maquetter une application ;
* Participer à la gestion de projet ;
* Réaliser une interface utilisateur web statique et adaptable ;
* Développer une interface utilisateur web dynamique ;
* Développer les composants d’accès aux données ;
* Développer la partie back-end d’une application web ;
* Développer la partie front-end d’une application web.

# 4 - Résumé du projet (ANGLAIS)

WEATHER is an internal project management application that enables project managers and directors to easily track the progress of their various projects. Before the creation of this application, the different users of the application used to fill out an Excel file to update the progress of a project.

Now, thanks to WEATHER, all projects are accessible to everyone involved. Depending on their role within SII, a user can create or modify a project. If they are a Project Manager or Project Director, they can also create or modify a report. In the report, they can update the progress of a project each week using indicators. A text editor is also available to allow for a more detailed description of the project’s status.

The Delivery Manager can then read and take note of the project’s progress. The creator of the report will be notified that the Delivery Manager has reviewed the report. A template system is also in place to allow the author of the report to save a format for the report’s description. The report is accessible for reading to all persons involved in the project.

A user can have rights to export the latest or all reports in Excel format, and choose to keep the upload images and formatting. WEATHER allows for authentication SII LDAP for added security. The application is currently only used in the SII West agency but aims to be deployed across all SII group agencies.

# 5 - Cahier des charges

## Présentation

Au sein de l’entreprise SII, il y a des directeurs d’opérations, de qualité et des delivery manager (DM) qui ont pour objectif d’évaluer l’état global des projets qu’ils soient internes ou client. Pour cela chaque semaine, les chefs de projet (CP) et les directeurs de projet (DP) saisissaient des synthèses Excel pour renseigner l’état de santé des projets et permettre d’identifier les points à surveiller.

Pour faciliter cette tâche et améliorer l’historisation de ces reportings, l’entreprise à décider de mettre au point une application web qui permettra de saisir et de visualiser des reportings hebdomadaires entre les CP, DP et DM. Cette application est accessible par l’ensemble des utilisateurs ayant un accès au LDAP SII.

## Expression des besoins

Les besoins étaient donc les suivants :

* L’utilisateur peut se connecter à l’application s’il est présent dans le LDAP SII ;
* Une gestion des rôles est mise en place pour restreindre l’accès à certaines parties de l’application ;
* Permettre la création et la modification de projet et de reportings ;
* Permettre la visualisation de l’ensemble des projets ;
* Faire des exports sous format Excel ;
* Gérer les utilisateurs tout en respectant une hiérarchie, afin de définir des autorisations qui seront utilisées dans différents modules de l’application.

## Spécifications fonctionnelles

L’application web est en production depuis le 15 mars 2021, elle est utilisée par les CP, DP et DM de l’agence de Rennes et a pour but d’être implanté dans les autres agences du groupe.

Lorsqu’un utilisateur valide se connecte à l’application WEATHER, il a accès à une liste des différents projets qu’il a créé et ceux dans lesquels il intervient. Il a également accès à la création d’un nouveau projet.

Un projet est un formulaire découpé en 5 pages. Pour passer d’une page à une autre et enfin valider la création, il doit renseigner l’ensemble des champs requis. Une fois le projet créé, il est possible de saisir un reporting pour décrire l’état d’avancement du projet. Ce reporting pourra ensuite être consulté par le DM.

En fonction de son rôle, l’utilisateur aura accès par exemple à la gestion des différents utilisateurs, à la création des messages à destination des autres utilisateurs ou bien à la gestion des différentes bulles hiérarchique et l’attribution de droits exceptionnels sur certains projets.

Mes missions durant mon alternance étaient le maintien de l’application par la correction d’anomalie et le suivi des règles de sécurité ainsi que par l’ajout de nouvelles fonctionnalités, comme la prise en charge d’un historique des différents statuts par lesquels le projet est passé, je vous décrirais ceci dans une prochaine partie. L’ajout de jeux de tests pour s’assurer la production d’un code propre et le bon fonctionnement de l’application.

WEATHER étant une application interne, j’ai également eu l’occasion de participer activement à la conception de nouvelles fonctionnalités.

# 6 - Gestion de projet

## Introduction

Pendant la durée de mon alternance, j'ai eu l'occasion de faire partie de deux équipes différentes. Tout d'abord, j'ai intégré l'équipe WEATHER chargée du projet du même nom. Ensuite, avec le départ de Mme HALLARD, cette équipe a été absorbée par l'équipe "Dev&Co". Dans cette partie, je présenterai la méthode que nous avons suivie pour la gestion du projet, ainsi que les différents outils utilisés.

Méthode Agile

Durant ce projet, nous avons utilisé la méthode agile Kanban. C’est une méthode qui vise l’amélioration continue, la flexibilité dans la gestion des tâches et l’amélioration du flux de travail. Cette méthodologie prévoit un système basé sur la demande plutôt que sur l’offre, c’est-à-dire que le client émet des besoins qui vont être traités au fur et à mesure.

Au sein de notre équipe, notre cheffe de projet exerce les mêmes missions qu’un PO, ce qu’y lui donne la possibilité d’avoir deux casquettes au même temps.

Les avantages de la méthode agile sont les suivantes :

* **La visualisation et la transparence** : cela permet de mieux comprendre le flux de travail et le statut des tâches grâce à un tableau où chaque colonne représente une étape. Chaque tâche évolue en temps réel jusqu’à ce qu’elle soit finalisé;
* **La flexibilité** : kanban est basé sur la pratique du *’just-in-time’*, ce qui le rend flexible pour s’adapter aux changements du projet. Le PO peut redéfinir l’ordre de priorité des tâches du backlog sans perturber le flux;
* **Allocation des ressources et budget** : vu que le flux de travail est visible sur le tableau Kanban, il est plus facile de reconnaître ses points de blocage et de ralentissements. Ce qui permet à l’équipe, au PO et/ou au CP de cibler les problèmes rencontrés et de les traiter correctement afin de trouver la bonne solution.

Le projet WEATHER a été lancé à la demande des chefs de projet, mais également à la demande de la direction des opérations. Ce qui fait que notre client est interne à l’entreprise et facilite grandement la communication.

La cheffe de projet Mme HALLARD, qui est responsable de rendre compte de l’évolution du projet fait d’abord la construction du backlog en se basant sur le retour des différents CP et DP qui utilisent l’application, les retours du responsable de la qualité et du directeur des opérations qui souhaite que le référentiel des projets chez SII soit dans l’application WEATHER. Mme HALLARD fait régulièrement des ateliers avec les acteurs de la qualité et les delivery managers (DM) pour identifier les besoins prioritaires dans l’outil. Ensuite, la cheffe de projet crée ce qu’on appelle une issue dans git lab, cette issue peut prendre plusieurs forme, comme une US qui correspond à une évolution de l’application, ou bien une correction de bug, appelé FIX au sein de git lab.

Cette US va alors suivre un chemin particulier, tout d’abord une fois sa rédaction finis, elle sera placé dans la colonne ***prêt***. A ce moment-là un développeur peut s’assigner cette tâche et la déplacer dans la colonne ***doing***. Une fois la tâche réalisée, comprenant le développement ou la correction ainsi que l’ajout d’un jeu de test, l’US est déplacé dans la colonne ***à tester***. La cheffe de projet peut alors tester le travail du développeur, si tout est bon, elle sera déplacé dans la colonne ***prêt pour prod***, et fera partie de la prochaine mise en production. Si le test montre des régressions, alors le développement reprend et le ticket est déplacé dans la colonne ***doing***.

## Outils de gestion de projet

En travaillant au sein des deux équipes, j'ai pu utiliser différents outils utiles à la gestion de projet, tout en conservant la méthode agile. Dans l'équipe WEATHER, nous avons utilisé le tableau de bord de Gitlab, qui présente l'avantage d'être très lisible et facilement utilisable.

Dans l'équipe "Dev&Co", nous avons utilisé IPMP, une surcouche créée par SII du logiciel open source appelé ProjetQtor. IPMP ressemble beaucoup à Gitlab en ce qui concerne la partie Kanban. Son point fort réside dans la gestion des imputations sur différents projets, directement au sein de l'application, ce qui facilite le travail des différents chefs de projet.

Au cours de mon travail sur le projet CRF, j'ai également eu l'occasion d'utiliser Wekan, un autre outil utilisé dans la méthode agile. Tous ces outils se ressemblent énormément et permettent de suivre et de gérer les tâches de manière efficace et collaborative.

*Cf. annexe 1*

## Workflow

Je vais ici vous présenter le workflow que nous suivons au sein de l'équipe Dev&Co pour la gestion des tickets et le processus de développement. Ce workflow nous permet de comprendre plus facilement l’avancé du travail de chacun et d'optimiser la qualité du code livré.

Lorsqu'un client a une demande ou un problème à soumettre, il crée un ticket sur l'outil de gestion de projet IPMP. Une fois que le ticket est créé, nous avons une fenêtre de 4 heures pour prendre connaissance de son contenu et le marquer comme ***accepted***. Si des informations supplémentaires sont nécessaires, le ticket passe en statut ***pending*** et est réattribué au client afin de lui demander les informations manquantes. Dans le cas contraire, le ticket est marqué comme ***verified***.

Une fois que le ticket est vérifié, nous entrons dans la phase de chiffrage. Le développeur peut effectuer cette étape individuellement s'il se sent suffisamment sûr de lui. Toutefois, si des incertitudes subsistent, une réunion de poker planning peut être organisée avec les autres membres de l'équipe. Cette réunion permet d'évaluer collectivement l'effort et la complexité de la tâche à réaliser. Une fois le chiffrage terminé, le développement peut commencer.

Avant de commencer le développement, une branche est créée à partir de la branche "develop" et poussée sans aucun code. Cette étape est essentielle pour permettre à l'outil d'analyse statique SONAR d'examiner le code dès le début. Ainsi, à la fin du développement, nous pouvons détecter d'éventuelles erreurs de code et les corriger en conséquence.

Le développement de la fonctionnalité ou de la correction est réalisé, tout en veillant à respecter les bonnes pratiques de codage. Des tests unitaires sont également réalisés pour assurer la qualité et la robustesse du code développé.

Avant de procéder à une merge request, une campagne de test est créée sur l'outil Squash TM. Cette campagne permet d'exécuter les tests nécessaires pour valider la fonctionnalité développée. Si tous les tests sont réussis, le développement est prêt pour la merge request.

Une fois que tous les tests sont validés, une merge request est effectuée pour fusionner la branche de développement avec la branche principale. Une fois la merge request acceptée, le ticket est marqué comme ***done*** et, si cela convient au client, il est clôturé en le passant en statut ***closed***.

Ce workflow, bien établi au sein de l'équipe Dev&Co, nous permet de suivre un processus structuré de gestion des tickets et de développement des fonctionnalités. Il favorise la collaboration, la qualité du code et la satisfaction du client.

*Cf. annexe 2*

En conclusion, la méthode Agile en mode Kanban a été utilisée pour la gestion du projet WEATHER. Cette approche flexible et collaborative nous a permis d'optimiser le flux de travail, de répondre aux besoins changeants du projet et de garantir la transparence et la visibilité des tâches. Les outils tels que Gitlab, IPMP et Wekan ont joué un rôle essentiel dans la gestion et le suivi des tâches tout au long du projet.

# 7 - Spécifications technique

## Présentation

Le projet WEATHER est une application web qui se compose de deux parties principales : le Front-end et le Back-end. Le Front-end est développé à l'aide du Framework Angular, qui nous permet de créer des pages web dynamiques et interactives. Cette technologie nous offre une large gamme de fonctionnalités pour concevoir une interface utilisateur moderne et conviviale.

D'autre part, le Back-end de l'application est développé en utilisant le Framework Spring Boot. Spring Boot est un Framework Java qui facilite la création d'applications web robustes et évolutives. Il fournit un ensemble de fonctionnalités prêtes à l'emploi, telles que la gestion des requêtes HTTP, l'authentification, la sécurité, la gestion des bases de données, etc. Spring Boot favorise le développement rapide en offrant une configuration par défaut et en simplifiant la mise en place de l'architecture MVC (Modèle-Vue-Contrôleur).

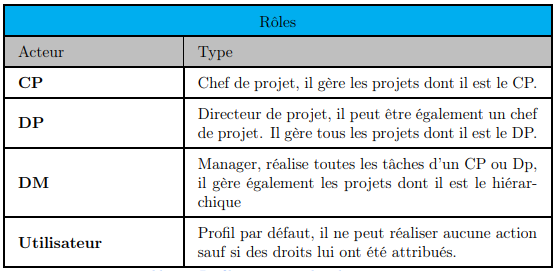
Pour la gestion de la base de données, nous utilisons Maria DB. C’est une base de données relationnelle open source qui est réputée pour sa stabilité, ses performances élevées et sa compatibilité avec MySQL. Elle offre une solution fiable et sécurisée pour stocker les données de l'application.

En combinant Angular pour le Front-end, Spring Boot pour le Back-end et Maria DB pour la gestion de la base de données, nous avons mis en place une architecture technique solide et cohérente pour notre application. Cette combinaison nous permet d'offrir une expérience utilisateur fluide, tout en garantissant la sécurité et la fiabilité des données.

## Sécurité et gestion des utilisateurs

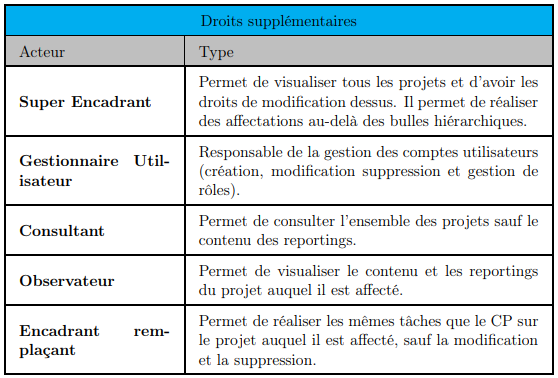
Dans le projet WEATHER, la sécurité joue un rôle primordial, et cela commence par le système d'accès et de connexion à l'application. Le site web de WEATHER est exclusivement accessible depuis le réseau SII, ce qui réduit considérablement les risques d'intrusion externe. De plus, pour se connecter, un utilisateur doit obligatoirement être répertorié dans l'annuaire LDAP de SII, garantissant ainsi l'identification et l'authentification des utilisateurs autorisés.

L'application WEATHER intègre également une gestion des rôles pour restreindre l'accès à certaines parties spécifiques de l'application. Chaque utilisateur se voit attribuer un rôle en fonction de son activité et de sa position au sein de SII, tels que Chef de Projet (CP), Directeur de Projet (DP), Delivery Manager (DM) ou utilisateur standard. Ces rôles définissent les droits et les permissions de chaque utilisateur, leur permettant d'accéder uniquement aux fonctionnalités et aux données qui leur sont pertinentes.



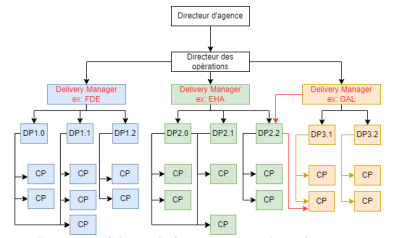
*Schéma du système de rôles*

En plus de la gestion des rôles, l'application WEATHER propose également une gestion de droits spéciaux, pouvant être temporaires ou permanents. Ces droits spéciaux peuvent être accordés à certains utilisateurs pour des besoins particuliers ou pour effectuer des tâches spécifiques au sein de l'application.



*Schéma du système de droits*

La gestion des utilisateurs est organisée selon une structure hiérarchique appelée "bulle". Cette approche permet de cloisonner l'application et de garantir une meilleure sécurité. Chaque utilisateur est rattaché à une bulle spécifique en fonction des projets auxquels il est affecté. Cette granularité dans la gestion des utilisateurs permet de limiter l'accès aux fonctionnalités et aux données uniquement aux utilisateurs autorisés, renforçant ainsi la confidentialité et la sécurité des informations sensibles.



*Schémas des différentes bulles hiérarchiques*

En mettant en place ces mesures de sécurité, telles que l'accès restreint au réseau SII, l'authentification LDAP, la gestion des rôles et des droits spéciaux, ainsi que la structure en bulles hiérarchiques, le projet WEATHER assure un environnement sécurisé pour ses utilisateurs, tout en garantissant la confidentialité et l'intégrité des données manipulées dans l'application.

## Maquettes

Lors de mon intégration dans l'équipe WEATHER, une partie importante du processus de conception avait déjà été réalisée, et je n'ai pas été directement impliqué dans la création des maquettes initiales. Notre Product Owner (PO), Mme HALLARD, jouait un rôle clé dans la désignation des différentes pages de l'application en se basant sur les retours des utilisateurs. Pour faciliter cette tâche, elle utilisait l'outil Figma pour créer des designs cohérents, en intégrant également les icônes provenant d'Angular Material, que nous utilisions dans l'application.

Lorsqu'une User Story (US) nécessitait des modifications de design, un rendu visuel de la page correspondante était attaché au ticket. Mme HALLARD nous avait également fourni les accès à la maquette Figma, ce qui nous permettait d'avoir une vision globale de l'aspect visuel de l'application.

Au cours de mon expérience en alternance, j'ai eu l'opportunité de participer activement aux réflexions liées au design de l'application. En raison de la taille réduite de l'équipe, Mme HALLARD sollicitait nos avis et suggestions pour l'équilibrage de certaines pages ou simplement pour le placement d'un bouton. Cette collaboration nous offrait une réelle liberté d'action, où chacun d'entre nous pouvait influencer les choix de design et d'ergonomie à sa manière.

J'ai particulièrement apprécié cette approche collaborative, où nous étions tous encouragés à être force de proposition. Cela nous permettait de contribuer activement à l'amélioration du design de l'application, en veillant à ce qu'elle réponde aux attentes des utilisateurs et qu'elle offre une expérience utilisateur optimale.

Bien que je n'aie pas participé directement à la création des maquettes initiales, cette expérience m'a permis d'appréhender l'importance de l'implication de l'équipe dans le processus de conception et de contribuer activement à l'amélioration visuelle et ergonomique de l'application WEATHER.

*Cf. annexe 3*

## Schéma de base de données

Dans le cadre de notre projet WEATHER, la gestion de la base de données est cruciale pour assurer le stockage et la manipulation efficace des données. Nous utilisons Maria DB comme système de gestion de base de données, et pour visualiser et interagir avec celle-ci, nous utilisons l'outil Heidi SQL.

Afin de garantir la cohérence et la traçabilité des modifications apportées à la base de données, nous avons mis en place une approche de gestion des migrations avec l'utilisation de Flyway conjointement avec Spring Boot. Cette approche nous permet de créer des fichiers de migration directement intégrés à notre code back-end.

Les fichiers de migration sont écrits en langage SQL et contiennent les scripts nécessaires pour mettre à jour la structure de la base de données. Lorsque l'application est construite, Flyway examine les fichiers de migration et vérifie s'ils ont déjà été appliqués à la base de données. Si un fichier de migration n'a pas encore été exécuté, il est automatiquement appliqué.

Pour assurer un suivi précis des modifications de la base de données, il est essentiel de respecter des conventions de nommage cohérentes pour les fichiers de migration. Cela permet à Flyway d'identifier les nouveaux fichiers de migration et de les appliquer séquentiellement dans l'ordre approprié.

Cependant, il convient de noter que toute modification apportée à la structure de la base de données nécessite la création d'un nouveau fichier de migration. Cette approche assure la traçabilité des changements et permet de maintenir un historique clair de l'évolution de la base de données.

En utilisant Flyway avec Spring Boot, nous avons pu gérer efficacement les modifications de la base de données, garantissant ainsi une intégrité et une cohérence optimales. Cette approche facilite la collaboration entre les membres de l'équipe de développement et contribue à maintenir une base de données stable et adaptée aux besoins du projet WEATHER.

*Cf. annexe 4*

## Ressources logicielles

La partie Back-end de notre application a été développée avec IntelliJ en Spring Boot, un Framework Java qui nous permet une optimisation de la gestion des dépendances, et l’utilisation de diverses annotations allégeant grandement le travail du développeur.

La partie Front-end a été développée avec Visual code en Angular, un Framework JavaScript qui nous permet un chargement plus rapide des pages et augmente les performances de l’application. Ce Framework étant basé sur l’architecture MVC, il permet une séparation des données, du visuel et des actions, pour une forte maintenabilité. Un des autres avantages est l’utilisation de TypeScript qui fournit un typage, un système de classes et d’interfaces.

Enfin, pour la base de données, nous utilisons Maria DB qui est un système de gestion de bases de données relationnelles, qui remplace MySQL. Maria DB ajoute un certain nombre d’optimisations pour les requêtes liées aux opérations de jointure, au contrôle d’exécution, et même aux instructions d’explication. Enfin, Maria DB offre des moteurs de stockage plus nombreux et de meilleure qualité. Pour la visualisation de la base de données, j’ai utilisé l’application Heidi SQL.

Pour gérer le versionnage du code nous avons utilisé git via l’interface graphique git fork.

*Cf. annexe 5*

## Architecture du projet

L'architecture du projet WEATHER repose sur le modèle MVC (Modèle-Vue-Contrôleur), qui est un pattern de conception largement utilisé dans le développement d'applications web. Le modèle MVC permet de séparer clairement les différentes responsabilités de l'application et favorise une conception modulaire et évolutive.

Modèle (Model):

Dans notre projet WEATHER, le modèle représente la couche de données et de logique métier. Il est responsable de la gestion des projets et de la persistance des données. Les projets sont représentés sous forme d'objets dans le modèle, comprenant des informations telles que leur nom, leur état et les membres de l'équipe associés. Pour le suivi des projets, nous avons également intégré un système de reporting permettant d'expliquer l'avancement du projet. Les données du modèle sont stockées dans une base de données.

Vue (View):

La vue dans notre application WEATHER représente l'interface utilisateur destinée aux différents acteurs concernés. Elle permet la création de nouveaux projets, l'affichage des projets existants, ainsi que la saisie des indicateurs et des commentaires relatifs à l'avancement des projets. Nous utilisons Angular, un Framework JavaScript populaire, pour développer l'interface utilisateur. La vue communique avec le contrôleur pour envoyer les données saisies et les requêtes associées au back-end.

Contrôleur (Controller):

Le contrôleur agit comme une couche intermédiaire entre le modèle et la vue. Dans notre projet, le contrôleur est responsable de la gestion des requêtes et des actions de l'utilisateur. Il reçoit les demandes de la vue, interagit avec le modèle pour effectuer les opérations appropriées sur les projets, et renvoie les résultats à la vue pour affichage. Le contrôleur utilise des API pour communiquer avec le back-end et assurer la persistance des données.

En utilisant l'architecture MVC, nous avons pu concevoir une application qui offre une séparation claire des responsabilités et facilite la collaboration entre les différentes parties du projet. La conception modulaire nous permet d'ajouter de nouvelles fonctionnalités ou de modifier l'interface utilisateur de manière indépendante. L'utilisation d'API pour le lien entre le front-end et le back-end garantit une communication fluide et une persistance des données fiables. L'intégration de Graddle et Lombok facilite le développement et la gestion des dépendances.

En conclusion, l'architecture du projet WEATHER basée sur le modèle MVC offre une structure claire et modulaire pour le développement d'une application de gestion de projets destinée aux Directeurs et aux Chefs de projets. Le modèle gère la logique métier et la persistance des données, la vue représente l'interface utilisateur, et le contrôleur assure la coordination entre la vue et le modèle en utilisant une API pour le lien avec le back-end.

*Cf. annexe 6*

# 8 - Réalisations

## Présentation

Je vais maintenant vous présenter les différentes réalisations que j'ai effectuées dans le cadre du développement de l'application WEATHER. Dans un premier temps, je vous présenterai mes contributions à la création et à la modification des tables de la base de données. Ensuite, je vous montrerai les composants que j'ai développés pour accéder aux données. Avant de passer à la présentation des interfaces utilisateur, je vous présenterai également les contrôleurs et les services que j'ai réalisés. Ces différentes réalisations ont été essentielles pour le bon fonctionnement et la fluidité de l'application WEATHER.

## Réalisation de la base de données

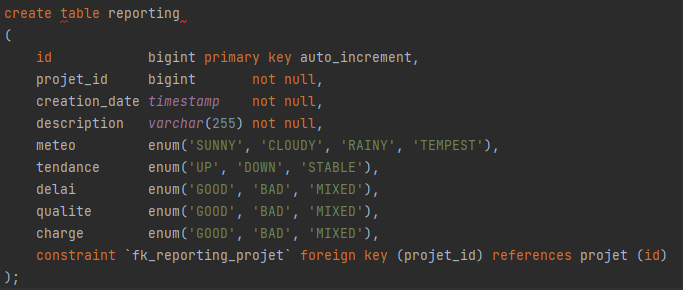
Lorsque j'ai intégré le projet WEATHER, la base de données était déjà créée. Cependant, je vais vous présenter la manière dont cela a été réalisé. Dans le projet, nous utilisons Flyway, un gestionnaire de schémas de base de données qui fonctionne avec un système de versionnement des fichiers de migration SQL.

Dans l'arborescence du projet, se trouve un dossier nommé "ressources" qui contient l'ensemble des fichiers de migration. Ces fichiers permettent la création ou la modification des différentes tables de la base de données. Ils doivent respecter des conventions de nommage spécifiques (par exemple, "V1.0\_\_description\_du\_fichier"), où le numéro correspond à l'ordre dans lequel les fichiers seront lus lors du processus de construction de l'application.

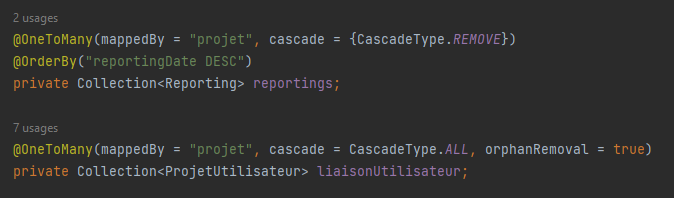
Chaque fichier de migration contient une ou plusieurs requêtes SQL. Il est possible de créer de nouvelles tables, de les modifier ou d'ajouter des données. Il est important de noter qu'il est déconseillé de modifier un fichier de migration déjà lu lors d'un build précédent. En effet, notre base de données comprend une table appelée "flyway\_schema\_history" qui enregistre l'historique de tous les fichiers de migration. Chaque fichier est associé à un checksum qui correspond aux données présentes dans le fichier. Ainsi, si nous modifions un fichier de migration, le checksum sera différent et le processus de construction de l'application détectera une incohérence avec Flyway.

En ce qui concerne les relations entre les tables, nous indiquons les clés étrangères dans les fichiers de migration pour associer les tables souhaitées. Ensuite, lors de la création de nos entités, j'utilise les annotations de Spring pour définir les relations entre ces tables.

Cette approche nous permet de gérer efficacement l'évolution de la structure de la base de données tout en préservant l'intégrité et la cohérence des données.



*Script de création de la table reporting originel*



*Exemple de relation qu’il peut exister entre la table projet et la table reporting.*

## Réalisation des composants d’accès aux données

Une fois notre base de données créée, il est maintenant temps de permettre l'accès à celle-ci. Pour cela, nous utilisons des interfaces appelées Repository, en utilisant Spring Data JPA et plus précisément l'interface JPARepository. L'utilisation de JPARepository, une extension de JPA, nous offre la possibilité d'implémenter automatiquement toutes les méthodes du CRUD (Create, Read, Update, Delete), ainsi que l'accès aux fonctionnalités de pagination et de tri fournies par cette interface.

L'avantage majeur de JPARepository est qu'il permet de simplifier considérablement le code en évitant d'avoir à écrire manuellement les requêtes SQL. En utilisant les conventions de nommage, il génère automatiquement les requêtes SQL appropriées pour récupérer les entités correspondant aux critères spécifiés.

Par exemple, en déclarant une méthode dans notre interface Repository comme "Projet findByCodeProjet(String codeProjet)", JPARepository traduit automatiquement cette méthode en une requête SQL qui récupère le projet correspondant au codeProjet fourni en argument. Il n'est pas nécessaire d'écrire du code SQL à l'intérieur de la méthode, seul le nom de la méthode est important.

De plus, JPARepository offre la possibilité de personnaliser et de surcharger certaines méthodes du CRUD pour répondre à des besoins spécifiques.

En résumé, c'est l'utilisation de JPARepository dans le contexte de Spring Data JPA qui nous permet de bénéficier de méthodes pratiques telles que "Projet findByCodeProjet(String codeProjet)". Cela simplifie considérablement le développement en évitant d'écrire manuellement des requêtes SQL et en fournissant une interface expressive et intuitive pour interagir avec la base de données.



## Réalisation des contrôleurs et services côté back

Tout d'abord, pour établir la communication entre le front-end et le back-end de l'application, nous avons besoin de contrôleurs côté back-end. Un contrôleur est une classe Java qui joue le rôle de point d'entrée et de liaison entre les différentes parties de l'application. Pour indiquer à Spring le rôle du fichier, nous utilisons l'annotation @RestController. Cette annotation permet de définir que le fichier représente un contrôleur.

Ensuite, l'annotation @RequestMapping("/adresse") est utilisée pour établir le lien entre le front-end et le back-end. Lorsque le front-end appelle le back-end, il utilise une adresse qui correspond à un contrôleur spécifique. Si plusieurs méthodes dans le contrôleur utilisent le même type de requêtes (GET, PUT, POST, DELETE), il est nécessaire de préciser un complément d'adresse pour chaque méthode en utilisant des annotations telles que @GetMapping("/complément"). Il est également possible d'envoyer des données, telles qu'un ID, dans ces adresses.

Dans le cas de WEATHER, l'accès à certains contrôleurs est restreint en fonction du rôle de l'utilisateur effectuant l'appel. Pour cela, nous utilisons l'annotation @PreAuthorize("hasAuthority('GLOBAL\_MANAGE\_USERS')"). Cette annotation permet de définir des autorisations spécifiques basées sur les rôles des utilisateurs. Ainsi, seuls les utilisateurs ayant l'autorité appropriée pourront accéder aux méthodes du contrôleur.

La plupart de nos contrôleurs servent de points d'accès à l'application, ce qui signifie que chaque méthode dans le contrôleur effectue un appel au service correspondant pour exécuter la tâche demandée. Cette approche respecte le principe de séparation des responsabilités. D'autres annotations présentes dans les arguments envoyés aux méthodes du contrôleur permettent un contrôle supplémentaire et une sécurisation de l'application.

La logique métier de notre application se trouve dans les services, qui sont également des classes Java. Pour indiquer à Spring que le fichier est un service, nous utilisons l'annotation @Service. Une méthode dans le service sera appelée par le contrôleur, qui lui enverra des données en entrée et attendra un certain type de données en retour. Dans le service, la méthode traitera ces données de différentes manières. Par exemple, pour accéder à la base de données, elle appellera le repository correspondant avec la méthode souhaitée. Pour effectuer un traitement de données, la méthode du service effectuera les calculs nécessaires avant de renvoyer les données au contrôleur. Dans de tels cas, l'utilisation de mappers est courante. Les mappers sont responsables de la conversion des entités en DTO (Data Transfer Object) et vice versa. Cela permet d'envoyer uniquement les données requises sans surcharger les appels.

En résumé, les contrôleurs et les services côté back-end sont essentiels pour la communication et la logique métier de l'application. Les contrôleurs agissent en tant que point d'entrée entre le front-end et le back-end, tandis que les services contiennent la logique métier et effectuent les opérations nécessaires, telles que l'accès à la base de données via les Repository, et le traitement des données. L'utilisation de différentes annotations facilite la gestion des autorisations, le contrôle de l'accès et la sécurisation de l'application.

## Partie Front-end et IHMS

Dans l'application WEATHER, la partie Front-end est développée avec le Framework Angular et accompagnée de la bibliothèque Angular Material. Angular est un Framework populaire pour la création d'applications Web robustes et évolutives, tandis qu'Angular Material fournit des composants et des directives prêts à l'emploi pour une interface utilisateur cohérente et esthétique.

Au cours de mon travail sur WEATHER, j'ai participé à la mise à jour du Framework Angular de la version 11 à la version 15. Cette migration de version avait pour objectif d'améliorer la maintenabilité de l'application. J'ai pu contribuer à une partie de la gestion de projet liée à cette migration.

Dans la partie Front-end de WEATHER, les différents composants s'appellent mutuellement pour former l'application. Chaque composant est composé de plusieurs fichiers.

Le fichier HTML contient le code des différentes pages de l'application. Il définit la structure de la page et inclut des appels vers d'autres composants de l'application. Certains composants peuvent être créés en externe de la page, ce qui permet de réutiliser des éléments tels qu'un formulaire sur différentes pages. Cette approche réduit considérablement la duplication de code.

Le fichier SCSS (Syntactically Awesome Style Sheets) contient les styles de l'application. On y définit l'emplacement des différents éléments, les polices utilisées et les couleurs des éléments de texte. Dans WEATHER, il existe un fichier SCSS global situé à la racine du projet, qui contient la majorité des styles de l'application. Chaque composant possède également son propre fichier SCSS, permettant une gestion plus spécifique des styles du composant en question.

Enfin, deux fichiers TypeScript sont utilisés pour le composant. Le fichier ".component.ts" contient la logique métier de la page et le fichier ".component.spec.ts" contient les jeux de tests associés. Le fichier métier fait référence aux fichiers HTML et SCSS avec lesquels il interagit. Il est composé de différentes méthodes qui interagissent avec le code HTML pour afficher des données ou communiquer avec les services responsables des appels vers la partie back-end de l'application.

En dehors des composants, la partie Front-end comprend également un ensemble de services qui facilitent la communication entre la partie Front-end et la partie Back-end. Ces services sont également des fichiers TypeScript qui contiennent les méthodes nécessaires pour effectuer des appels via une API.

En résumé, dans la partie Front-end et IHM de l'application WEATHER, Angular et Angular Material sont utilisés pour créer une interface utilisateur réactive et esthétiquement agréable. Les composants, les fichiers HTML, SCSS et TypeScript, ainsi que les services, jouent un rôle clé dans la construction et le fonctionnement de l'application.

# 9 - Présentation de la fonctionnalité la plus représentative

## Présentation

Au cours de mon alternance, j'ai développé une fonctionnalité clé qui permet l'historisation des différents statuts des projets dans l'application. Dans cette section, je vais vous présenter cette fonctionnalité en détail. Les utilisateurs de l'application peuvent désormais bénéficier de cette fonctionnalité en accédant à l'historique des statuts d'un projet donné. Cela leur permet de suivre l'évolution du projet au fil du temps et d'analyser les changements de statut. Je vais vous décrire le processus complet, de l'utilisation de la fonctionnalité par l'utilisateur à sa réalisation, en passant par la transmission des données du front vers le back, jusqu'à leur affichage dans la page de détail d'un projet.

## Utilisation

Je vais maintenant vous présenter comment les utilisateurs peuvent interagir avec la fonctionnalité que j'ai développée. Lorsqu'un utilisateur crée ou modifie un projet, il accède à un formulaire composé de cinq pages distinctes. La fonctionnalité en question se trouve sur la quatrième page de ce formulaire. Sur cette page, l'utilisateur doit renseigner la date de début du projet, qui est le seul champ obligatoire, ainsi que la date de fin et le statut. Par défaut, le statut est défini comme "A venir". Ces trois champs sont les principaux éléments qui nous intéressent.

*Cf. annexe 7.1*

La sélection du statut dépend de la date de début et de la date de fin du projet. Par exemple, si la date de début est antérieure à la date du jour, l'utilisateur n'est pas autorisé à choisir le statut "A venir". Un message d'erreur explicatif lui sera affiché pour lui indiquer la raison de cette restriction.

*Cf. annexe 7.2*

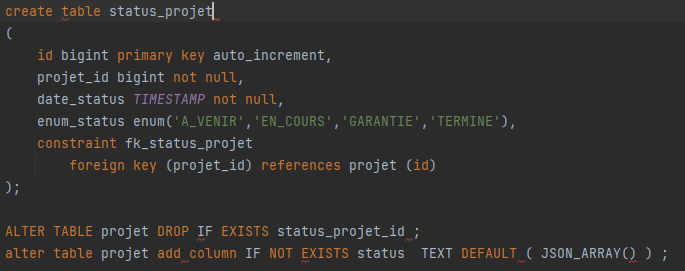
Une fois le formulaire validé, l'utilisateur peut consulter les détails de son projet en cliquant dessus. Dans l'onglet "jalons", une frise chronologique affiche l'avancement du projet avec les différentes dates des statuts par lesquels le projet est passé. Cette représentation visuelle permet à l'utilisateur de suivre l'évolution du projet au fil du temps.

*Cf. annexe 7.3*

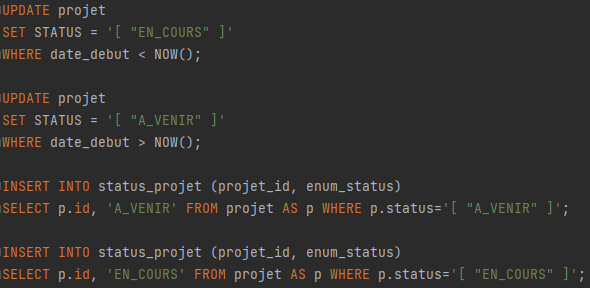
## Réalisation

Après avoir étudié l'User Story, j'ai analysé le code existant afin de comprendre comment ajouter la nouvelle fonctionnalité d'historisation des statuts des projets. L'onglet permettant à l'utilisateur de renseigner le statut d'un projet était déjà présent, mais des modifications étaient nécessaires pour gérer ces données. Cela impliquait l'ajout d'une nouvelle table dans la base de données, des modifications dans la partie Back-end pour traiter efficacement ces données, ainsi que des ajouts dans la partie Front-end pour récupérer et afficher ces informations de manière claire dans les détails d'un projet.

Tout d’abord, j’ai créé un fichier de migration pour la base de données, j’ai ajouté une nouvelle table comprenant le nom du statut (en lien avec un fichier JSON qui permet la création du champ statut), l’id du projet auquel ce statut est rattaché ainsi que la date à laquelle le statut a été modifié.



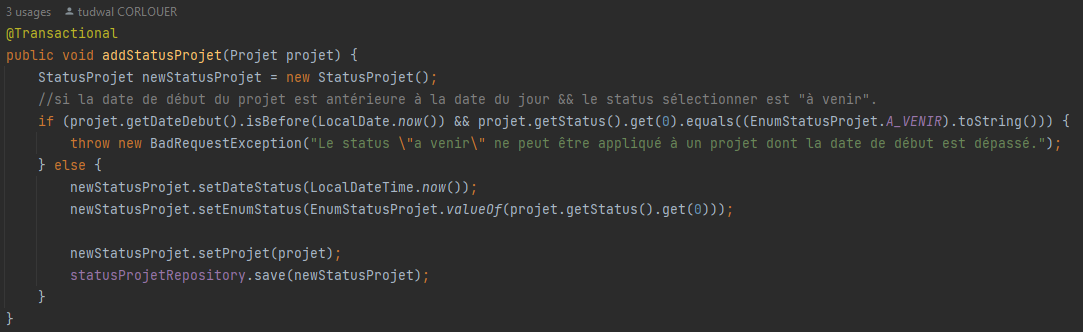
Afin d'éviter tout problème lors de la mise en production de cette fonctionnalité, j'ai également créé un fichier de migration permettant d'initialiser les statuts pour les anciens projets.



Une fois, cette nouvelle table créée, j’ai pu m’occuper de la partie Back-end. Tout d’abord, les données saisies par l’utilisateur sont envoyées via un Controller qui va appeler le service du projet. Ici, dans le fichier « ProjetService », toutes les informations permettant la création d’un projet sont récupérées. J’ai donc eu simplement à me positionner dans ce fichier et à récupérer les informations nécessaires à l’historisation de ces statuts. Pour cela, j’ai donc créé un nouveau service contenant toute la logique métier en lien avec les statuts. J’appelle ce fichier et les méthodes internes depuis la création d’un projet.

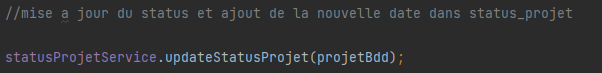


Dans le cas de la création d’un projet, la logique est assez simple, des contrôles ont déjà été effectués côté front, permettant d’avoir des données propres. Pour plus de sécurité, j’effectue un nouveau contrôle sur la date du statut pour être sûr de persister des données correctes.



Si la date de début de projet est passée et que le statut renseigné par l’utilisateur est « A\_VENIR » alors on renvoie une erreur expliquant que ce statut ne peut être associé à un projet dont la date de début est déjà dépassée. Sinon on continue le traitement des données et on persiste dans la nouvelle table la date de ce nouveau statut renseigné.

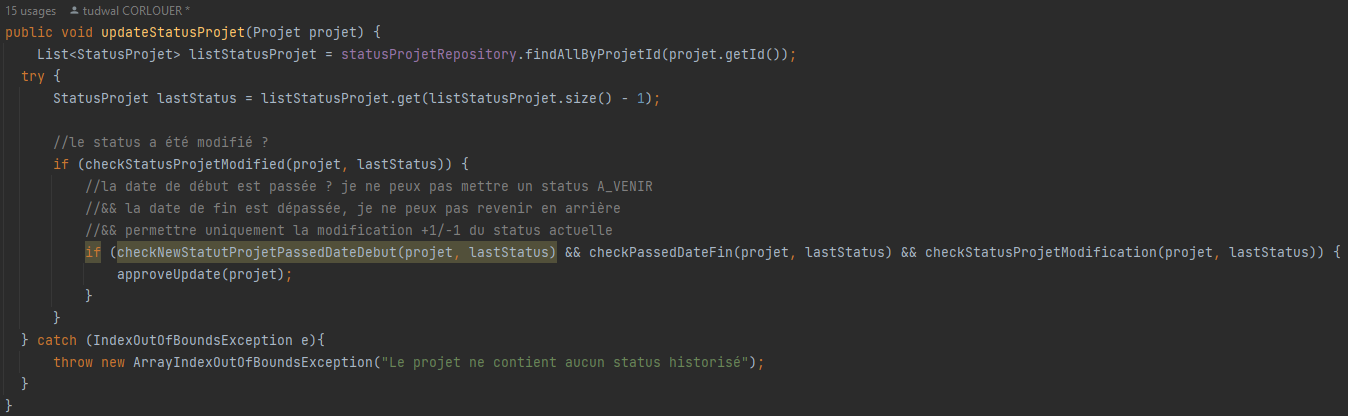
Dans le cas d’une modification de projet, la logique est différente. Dans la méthode liée à l’édition de projet, j’appelle mon service lié au statut via la méthode updateStatusProjet en lui passant comme argument le projet à modifier.



C’est à l’intérieur de cette méthode que les derniers contrôles sont effectués. En effet, l’US précisait tous les cas d’acceptation liés au statut. Ces différents cas sont les suivants :

* Un projet dont la date de début est dépassée ne peut pas avoir un statut à « A\_VENIR » ;
* Le statut ne peut évoluer que un par un, c’est-à-dire un projet qui a un statut à « A\_VENIR » ne peut pas passer à un statut « GARANTIE » ;
* Un projet dont le statut est « TERMINE » ne peut pas revenir en arrière sur ses statuts.

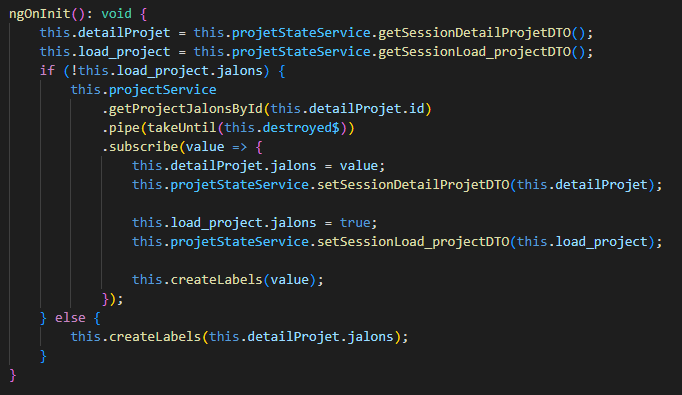
Tout d’abord, je récupère en base de données, la liste de tous les statuts associés à ce projet. Ensuite, je vais vérifier si le statut a été modifié par l’utilisateur, pour cela, je le compare avec le dernier statut présent en base de données. Si le statut a été modifié, je vais alors appeler différentes méthodes qui auront pour but de vérifier chacun des cas d’acceptations listé ci-dessus.



Si à un moment du processus, l'un de ces cas n'est pas respecté, une erreur est renvoyée directement au Front-end, permettant à l'utilisateur de modifier sa saisie s'il souhaite sauvegarder la modification de son projet.

Du côté du Front-end, les informations du projet créé ou modifié sont envoyées via un service qui communique avec le Back-end. L'élément qui nous intéresse ici est l'affichage de l'historique des différents statuts. Pour cela, nous utilisons un composant Angular qui crée une frise chronologique répertoriant les différents statuts par lesquels le projet est passé, ainsi que leurs dates.

Lors de l'initialisation du composant, la méthode "getProjetJalonById" est appelée dans notre service dédié aux projets. Cet appel me permet de récupérer tous les statuts du projet ainsi que leurs dates associées.



Une fois ces informations récupérées, je mets à jour les différents jalons et les allume si le jalon est passé. Pour cela, j'appelle la méthode "createLabels()" qui initialise chaque libellé et modifie leurs valeurs si nécessaire. Enfin, une dernière méthode permet de vérifier si la date de fin du projet a été atteinte, et ainsi d'allumer l'ensemble de la frise. Comme chaque jalon de la frise est associé à une date, j'ai créé une méthode pour personnaliser l'affichage de cette dernière en utilisant un "pipe" personnalisé qui me permet de définir le format souhaité pour la date.

Pour finir, je vérifie si la date de fin est dépassée pour signifier dans le détail du projet que le projet est fini.

## Test

Afin de finaliser le développement de cette fonctionnalité, j'ai réalisé un ensemble de tests pour la partie Back-end de l'application. J'ai principalement axé mes tests sur cette partie, car c'est là que se trouve la majeure partie de la logique métier liée à ce développement.

Pour rédiger les tests, j'ai utilisé Mockito, qui nous permet de créer des objets fictifs (mocks) pour simuler les interactions avec d'autres parties du système. La logique des tests est assez simple. J'ai créé un test pour chaque cas possible en fonction de la date de début du projet, de sa date de fin, ainsi que du statut sélectionné par l'utilisateur.

Le temps consacré à l'écriture de ces tests m'a permis de garantir l'efficacité de mon développement. Ces tests ont permis de valider que la logique des statuts était correctement implémentée et qu'elle répondait aux différentes exigences spécifiées dans l'User Story.

En plus des tests unitaires, j'ai également participé à une campagne de tests avec l'utilisation de SquashTM, une plateforme de gestion des campagnes de test. Cela nous a permis d'effectuer des tests d'intégration en simulant des scénarios réels avec différentes combinaisons de dates de début, de dates de fin et de statuts.

L'ensemble de ces tests a contribué à assurer la qualité et la fiabilité de la fonctionnalité développée, en nous permettant de détecter et de corriger les éventuels problèmes ou erreurs avant la mise en production.

## Conclusion

En conclusion, travailler sur cette User Story a été une expérience enrichissante qui m'a permis de relever de nombreux défis et de toucher à tous les aspects de l'application. La phase de conception a été particulièrement gratifiante, car elle m'a demandé d'analyser l'ensemble de l'application afin de développer cette nouvelle fonctionnalité de la manière la plus optimale possible. Dans un souci de maintenabilité, j'ai opté pour la création d'une nouvelle table dans la base de données.

La partie Back-end a été relativement simple, car une grande partie des données nécessaires était déjà présente. J'ai cependant dû ajouter des contrôles supplémentaires pour assurer la validité des informations. La complexité résidait dans la recherche d'une implémentation propre qui respecte les règles de sécurité.

La partie Front-end a représenté le plus grand défi pour moi, car j'ai développé cette fonctionnalité de manière autonome peu de temps après mon arrivée chez SII. Cela m'a permis d'approfondir ma compréhension du fonctionnement de l'application et de mettre en pratique mes compétences dans le développement Front-end.

Dans l'ensemble, travailler sur cette fonctionnalité m'a permis d'étendre mes connaissances et de développer mes compétences techniques. J'ai pu mettre en pratique des principes de conception solides et acquérir une meilleure compréhension de l'architecture de l'application. Je suis fier du travail accompli et confiant dans la qualité de la fonctionnalité que j'ai développée.

# 10 - Sécurité

## Présentation

Cette alternance m’aura permis de me challenger sur l’aspect technique du développement. En effet, j’ai souvent été amené lors de la réalisation de tâches, aussi bien évolutions qu’anomalies, à effectuer des recherches sur internet ou bien solliciter l’aide de différents membres de SII. Je dois bien avouer que l’accès à un grand nombre de profils différents permet de se sortir de la plupart des situations problématiques. Chacun a su m’aiguiller dans la bonne direction lorsque je faisais face à un trop gros challenge. J’ai été amené à effectuer différentes recherches aussi bien sur le fonctionnement de certaines méthodes, que sur la partie sécurité de l’application. Ce qui m’a permis de mieux appréhender les différents systèmes de sécurité mit en place.

La gestion de la sécurité est quelque chose de très important lorsque que l’on crée une application, c’est pourquoi différents outils ont été mis en place sur le projet WEATHER. Dans un premier temps, je pourrais vous parler de la sécurité liée à la bonne utilisation de l’application, puis enfin, je vous parlerais de la sécurité en termes d’intrusion et de modification du fonctionnement de l’application.

## Mise en œuvre

Au sein de l’application WEATHER il y à différents niveaux de sécurités. Tout d’abord, le contexte d’accès à l’application est une première sécurité en soi. En effet, il faut être connecté sur un réseau SII pour pouvoir accéder au site. Soit en étant sur site ou bien via l’utilisation d’un vpn lorsque je suis en télétravail. Ensuite, la connexion se fait par le biais d’un identifiant lié au LDAP de SII. Cela nous permet de ne donner l’accès à l’application qu’aux seules personnes travaillant chez SII et ainsi limité au maximum les intrusions extérieures. Le protocole LDAP ou Lightweight Directory Access Protocol est un annuaire centralisé qui permet de stocker des données et d’authentifier les utilisateurs qui veulent y accéder. Au sein de SII, cela permet de créer un écosystème entre toutes les applications utilisées. Enfin, il y a un niveau de sécurité plus directe avec l’accès à la base de données, l’identification ou encore les diverses autorisations.

Sur WEATHER, les utilisateurs n’ont pas tous les mêmes droits. Seules quelques personnes ont la possibilité de modifier les droits accordés aux différents utilisateurs. Cette gestion des droits nous permet de contrôler l’accès à l’application. En effet du côté back de l’application, au niveau des points d’accès via l’API, nous utilisons une annotation de Spring (@PreAuthorize) qui nous permet de bloquer l’accès à certains de nos services en fonction du rôle de l’utilisateur.

Autre petit point de sécurité qui peut sembler futile mais pourtant assez important. Le retour des messages d’erreurs lors de la connexion d’un utilisateur. Il est recommandé de ne pas prévenir l’utilisateur du champ qu’il a mal renseigné. En effet, si on lui indique uniquement (mot de passe ou identifiant incorrect), cela rend le forçage de mot de passe plus long.

Un autre grand point de sécurité est la protection contre l’injection de données. Ce type d’attaque a pour but d’injecter en base, des données non désirées ce qui peut avoir pour effet de donner le contrôle de l’application à un intrus. Pour remédier à cela, il y a plusieurs solutions. L’utilisation de requête paramétrée pour contrôler ce qui est envoyé en base de données et ainsi limiter les failles ou bien utilisé grâce à spring et à l’utilisation de l’ORM JpaRepository. Ce dernier offre une protection intégrée contre les attaques d'injection SQL en utilisant des requêtes paramétrées et des mécanismes de liaison de paramètres sécurisés. Cela garantit que les données transmises à la couche de persistance sont traitées de manière sécurisée, réduisant ainsi les risques d'injection de données malveillantes.

Pour ce qui est de la protection contre les failles XSS (Cross-Site Scripting), nous avons mis en place des mesures de sécurité. Cela inclut la validation et l'échappement appropriés des données saisies par les utilisateurs avant de les afficher dans les pages web. En utilisant des techniques telles que l'encodage des caractères spéciaux et l'utilisation de bibliothèques de sécurité, nous réduisons considérablement les risques d'injection de code malveillant dans notre application.

## Conclusion

La protection d’une application est sûrement la partie la plus importante à mes yeux, car en dehors de protéger notre application d’un code malveillant, nous protégeons aussi nos utilisateurs et leurs données. C’est pourquoi chez SII chaque nouveau collaborateur est invité à assister à une formation d’une journée sur le développement web sécurisé. Cette formation m’a apporté beaucoup et m’a permis de mieux comprendre les différentes règles de sécurité à mettre en place pour protéger son application ainsi que ses utilisateurs.

# 11 - Recherche effectuée

## Le besoin d’information

Tout au long de mon alternance, j’ai eu de nombreuses fois recours à la recherche d’information pour m’aider à me sortir de certaines situations ou tout simplement dans le but de comprendre une nouvelle technologie sur laquelle je devais travailler. Pour cela, j’ai mis en place plusieurs méthodes pour optimiser mes recherches et surtout ne pas faire appel trop vite à mes collègues, ce qui aurait pour effet de ne pas vraiment m’apprendre à trouver les solutions à mes problèmes.

Dans un premier temps, la majeure partie de mes recherches s'est faite via Google. Ensuite, j'ai utilisé des sites de référence tels que Stack Overflow, Baeldung (spécialisé dans Spring) et la documentation des développeurs de Mozilla. J'ai également eu l'opportunité de tester le moteur de recherche Phind (anciennement Hello), dédié aux développeurs. Son fonctionnement est simple : il recherche directement les réponses sur les sites les plus importants et affiche généralement le code correspondant (provenant de Stack Overflow) directement sur la première page de recherche, ce qui permet de tester plus rapidement les solutions proposées.

Cette année a également été marquée par l'avènement des intelligences artificielles et leur accessibilité. Comme de nombreux développeurs, j'ai parfois utilisé ChatGPT, l'IA développée par OpenAI. Je l'ai principalement utilisé lors de l'ajout de tests unitaires dans l'application. ChatGPT est un outil révolutionnaire, mais il reste un outil et il est important de faire preuve de discernement dans les réponses qu'il propose. J'ai dû prendre le temps de comprendre comment l'utiliser et interagir avec lui. L'un des avantages de ce type d'outil est qu'il explique sa réponse en détail. En plus de me fournir une proposition de code, il m'indique à quoi correspond chaque ligne et pourquoi elle est utilisée. Grâce à une persévérance constante, j'ai réussi à maîtriser son utilisation et à travailler en lui fournissant les différentes erreurs que ses solutions apportaient, aboutissant ainsi à une solution appropriée.

En conclusion, la recherche d'informations a joué un rôle essentiel dans ma progression professionnelle tout au long de mon alternance. Grâce à l'utilisation de diverses ressources en ligne, de moteurs de recherche spécialisés et d'outils d'intelligence artificielle, j'ai pu développer mes compétences en résolution de problèmes et en compréhension des nouvelles technologies. Cela m'a permis de gagner en autonomie et de devenir plus efficace dans mon travail au quotidien.

## Le problème

En ce début d'année, j'ai participé à ma première mise en production du travail que nous avions réalisé sur le projet WEATHER. Pour cette occasion, un autre alternant et moi-même avons effectué la mise en production sous la supervision de notre cheffe de projet et avec l'aide d'un autre développeur expérimenté sur l'application. Cependant, malgré nos préparatifs, cette mise en production s'est avérée plus complexe que prévu. Une fois celle-ci terminée, lors de l'exploration de l'application, nous avons découvert un problème lié à la gestion des dates. En effet, nous avons constaté que certaines de nos dates avaient été modifiées pour correspondre à la date de la mise en production.

Dès que nous avons pris conscience de cette erreur, nous avons immédiatement rétabli la version précédente de l'application en activité afin de nous donner le temps de résoudre ce problème. Étant la personne ayant développé la fonctionnalité impactée, j'ai pris l'initiative de rechercher une solution. En effet, nous venions juste d'ajouter un mécanisme d'historisation des statuts, ce qui impliquait la création d'un nouveau script SQL pour ajouter ces dates dans la base de données. L'idée était également de mettre en place un jeu de données permettant d'initialiser les statuts des anciens projets, ce qui a eu pour conséquence de réécrire toutes les dates associées.

Cette situation a constitué un défi majeur, nécessitant une analyse minutieuse du problème et une compréhension approfondie du fonctionnement de l'application. J'ai travaillé en étroite collaboration avec l'équipe pour identifier l'origine de l'erreur et mettre en place les mesures correctives nécessaires. Cette expérience m'a permis de renforcer mes compétences en résolution de problèmes et de développer une meilleure compréhension de la gestion des données et de la coordination lors des mises en production.

Bien que cette mise en production ait rencontré des difficultés inattendues et engendré un problème de gestion des dates, l'équipe a réagi rapidement et efficacement pour corriger l'erreur. Cette situation a également été l'occasion pour moi de développer mes compétences techniques et de renforcer ma capacité à résoudre des problèmes complexes au sein d'un environnement de développement en constante évolution.

## La recherche

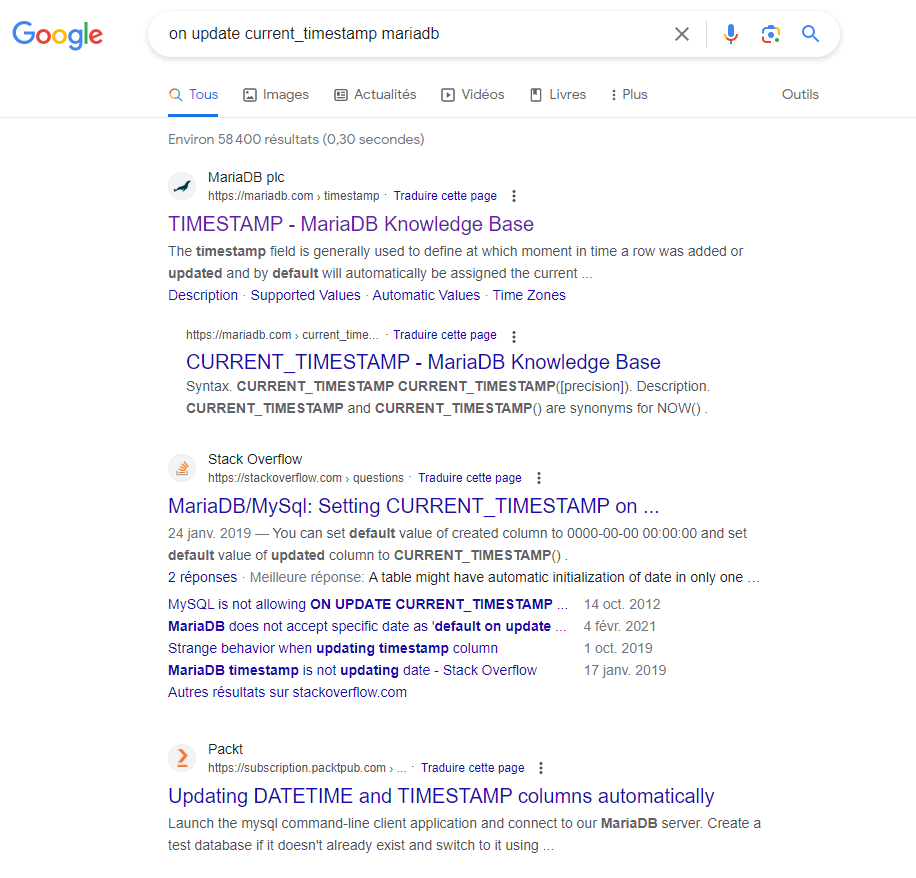
Pour résoudre ce problème, j'ai entamé mes recherches en analysant le code ainsi que la structure de la base de données afin de déterminer la cause exacte du dysfonctionnement. C'est lors de cette exploration approfondie que j'ai découvert l'origine du problème. J'ai remarqué qu'une option spécifique (On update: current TimeStamp) avait été activée par défaut sur le champ de date nouvellement créé. Je me suis demandé pourquoi cette option était activée et pourquoi je ne pouvais pas la modifier. Pour obtenir des réponses à ces questions, j'ai entrepris des recherches en ligne. J'ai simplement saisi le nom de cette option ainsi que le type de base de données que nous utilisons.

Grâce à mes recherches, j'ai pu recueillir des informations précieuses sur cette option particulière et son fonctionnement dans le contexte de notre base de données. J'ai consulté la documentation officielle du système de gestion de base de données et exploré des forums de discussion spécialisés tels que Stack Overflow. Ces ressources m'ont permis de mieux comprendre pourquoi l'option était configurée de cette manière par défaut et comment elle pouvait affecter les dates en cas de mise à jour.

De plus, j'ai également pu bénéficier de l'expérience et des connaissances de mes collègues développeurs. J'ai discuté du problème avec eux, partageant mes découvertes et échangeant des idées sur la manière de le résoudre efficacement.

En combinant mes recherches en ligne, la documentation officielle et les discussions avec mes collègues, j'ai réussi à obtenir une vision claire de l'option problématique et de son impact sur notre application. Cela m'a permis de proposer une solution appropriée pour désactiver cette option et rétablir le fonctionnement correct de la gestion des dates.

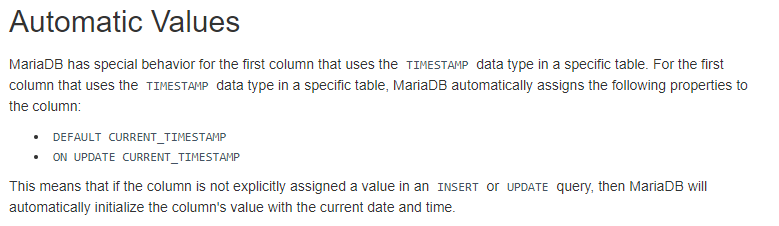
Cette recherche a été essentielle pour comprendre la cause du problème et trouver une solution adéquate. Elle a mis en évidence l'importance de mener des investigations approfondies, de consulter des ressources fiables et de partager des connaissances avec les membres de l'équipe. Ces compétences de recherche ont été précieuses tout au long de mon alternance et m'ont permis de résoudre efficacement divers problèmes techniques rencontrés dans mon travail quotidien.



## La solution

Pour résoudre ce problème, j'ai effectué des recherches approfondies et me suis tourné vers la documentation officielle de MariaDB. C'est en explorant cette ressource que j'ai finalement découvert la solution à mon souci. Au cours de ma lecture, j'ai trouvé un passage spécifique traitant précisément du problème rencontré et de la manière de le résoudre. J'ai appris que lorsqu'un attribut de type date est créé sans valeur par défaut spécifiée dans une table, MariaDB applique automatiquement la configuration « DEFAULT current\_timestamp / ON UPDATE current\_timestamp ». Cette configuration entraîne la réécriture des dates lors des mises à jour de la table.

Grâce à cette information clé, j'ai pu identifier la cause profonde du problème que nous avions rencontré. L'option par défaut « ON UPDATE current\_timestamp » était à l'origine de la modification indésirable des dates lors de la mise en production. Pour résoudre ce problème, j'ai opté pour une approche de solution consistant à désactiver cette option spécifique dans la structure de la table.

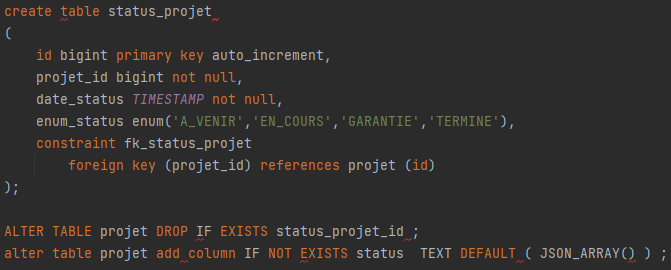


## La mise en œuvre

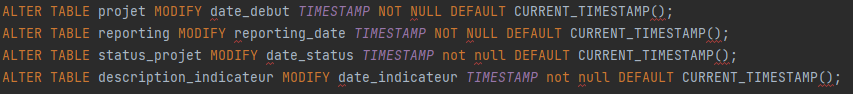
J'ai procédé à la modification de la définition de la colonne concernée en spécifiant une valeur par défaut explicite pour l'attribut de date, plutôt que de compter sur la configuration par défaut de MariaDB. En faisant cela, j'ai évité que les dates soient réécrites automatiquement lors des futures mises à jour de la table.

Après avoir apporté cette modification, j'ai effectué des tests approfondis pour m'assurer que le problème avait été résolu avec succès. J'ai vérifié les opérations de mise à jour de la table et confirmé que les dates étaient désormais préservées conformément à nos attentes.

En résumé, la solution à ce problème a été trouvée en consultant la documentation officielle de MariaDB, qui a fourni des informations cruciales sur la configuration par défaut « ON UPDATE current\_timestamp ». En modifiant la définition de la colonne et en désactivant cette option, j'ai pu résoudre efficacement le dysfonctionnement lié à la réécriture des dates lors des mises à jour. Cette approche a permis de restaurer la gestion correcte des dates dans notre application.



*Script de base de la création de la table status\_projet*



*Nouveau script avec les modifications pour toutes les tables impactées*

# 13 - Conclusion

En conclusion, mon expérience en tant qu'alternant chez SII a été extrêmement enrichissante et me confirme dans ma reconversion professionnelle entreprise il y a deux ans. Cette année passée au sein de SII m'a apporté énormément sur les plans personnel et professionnel. Cependant, le départ de ma tutrice, Mme HALLARD, qui a marqué la fin du projet WEATHER et son développement, a suscité une certaine déception. J'ai grandement apprécié travailler au sein de cette équipe et sur ce projet qui m'a offert une grande liberté de création.

SII a su me soutenir de manière exceptionnelle, que ce soit en termes de formations pour améliorer mes compétences ou en me confiant des projets stimulants qui m'ont permis d'apprendre de nouvelles technologies. Depuis le mois de mars, j'ai rejoint une nouvelle équipe de TMA travaillant sur un projet pour la Croix-Rouge Française, ce qui représente une source de motivation considérable pour moi.

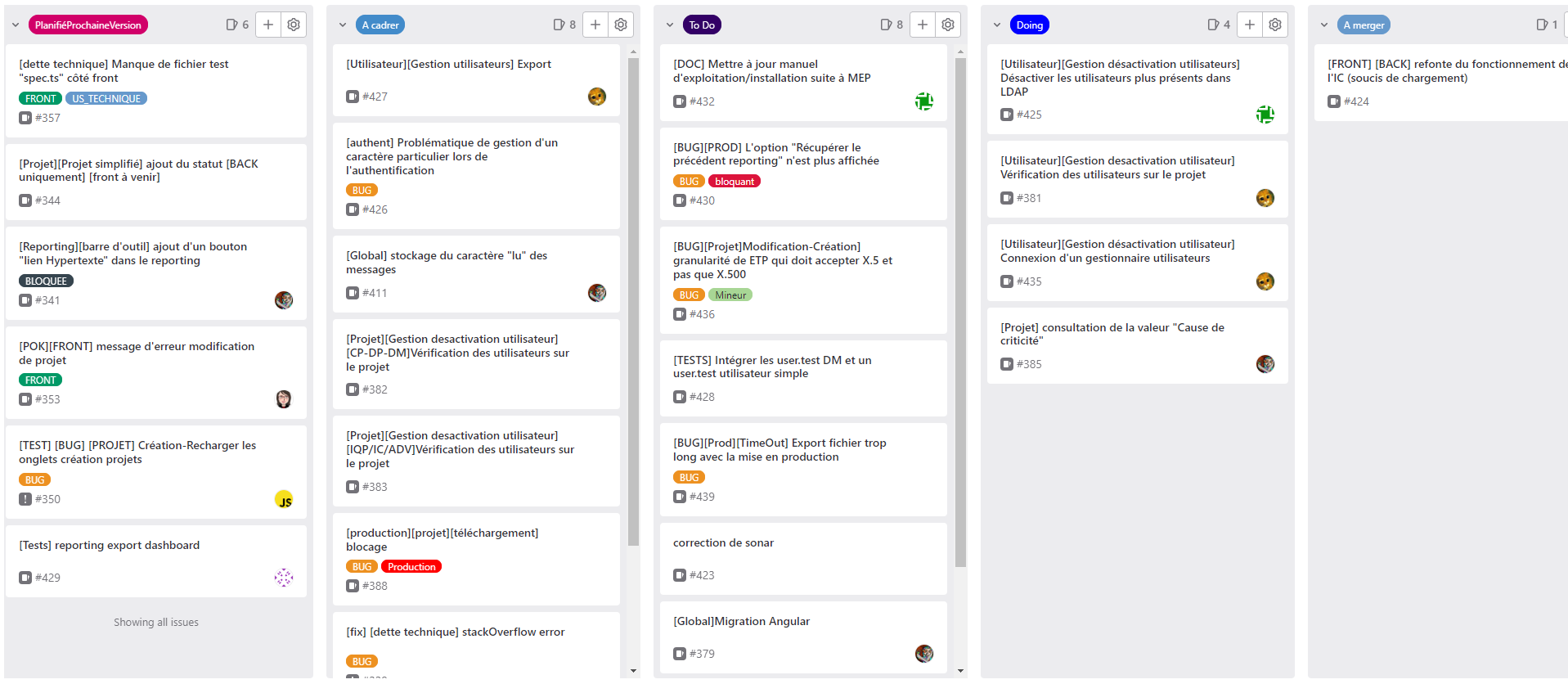
Grâce à mon implication dans ces différents projets, mes compétences en Spring et en Angular se sont considérablement renforcées. J'ai également pu acquérir une expérience précieuse en travaillant au sein d'une véritable équipe Agile. Certains développements ont été plus complexes que d'autres, mais ces défis m'ont permis de me surpasser et de progresser en tant que développeur.

Je tiens à exprimer ma gratitude envers toute l'équipe de SII pour leur soutien constant, leur expertise partagée et leur collaboration fructueuse. Je suis convaincu que les compétences et l'expérience acquises au cours de cette alternance seront des atouts précieux pour ma carrière future.

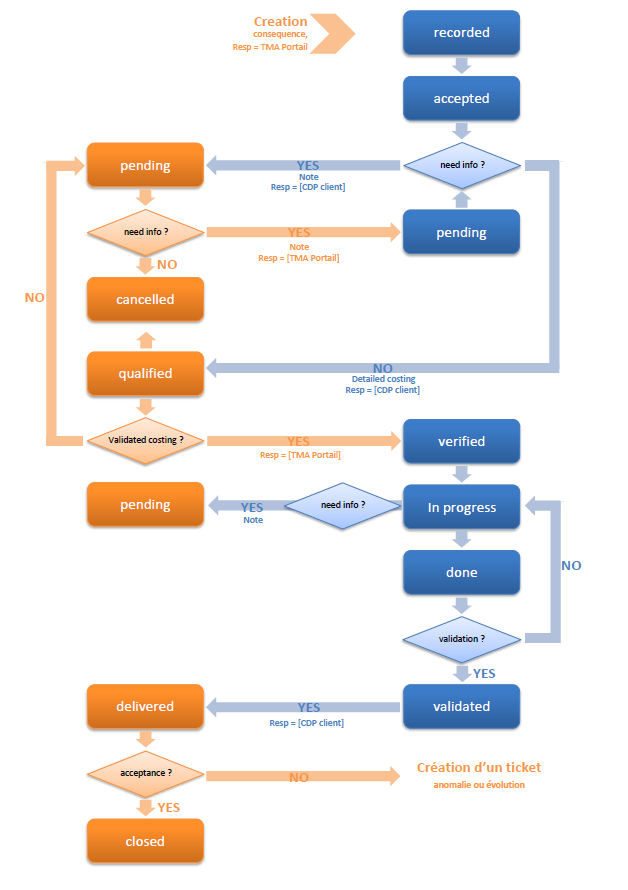
En somme, cette expérience chez SII m'a offert une opportunité inestimable de me former et de me développer professionnellement. Je suis reconnaissant d'avoir pu contribuer à des projets significatifs tout en élargissant mes horizons technologiques. Je suis désormais plus confiant et prêt à relever de nouveaux défis avec passion et détermination.

# Annexes

Annexe 1 : Board de git Lab

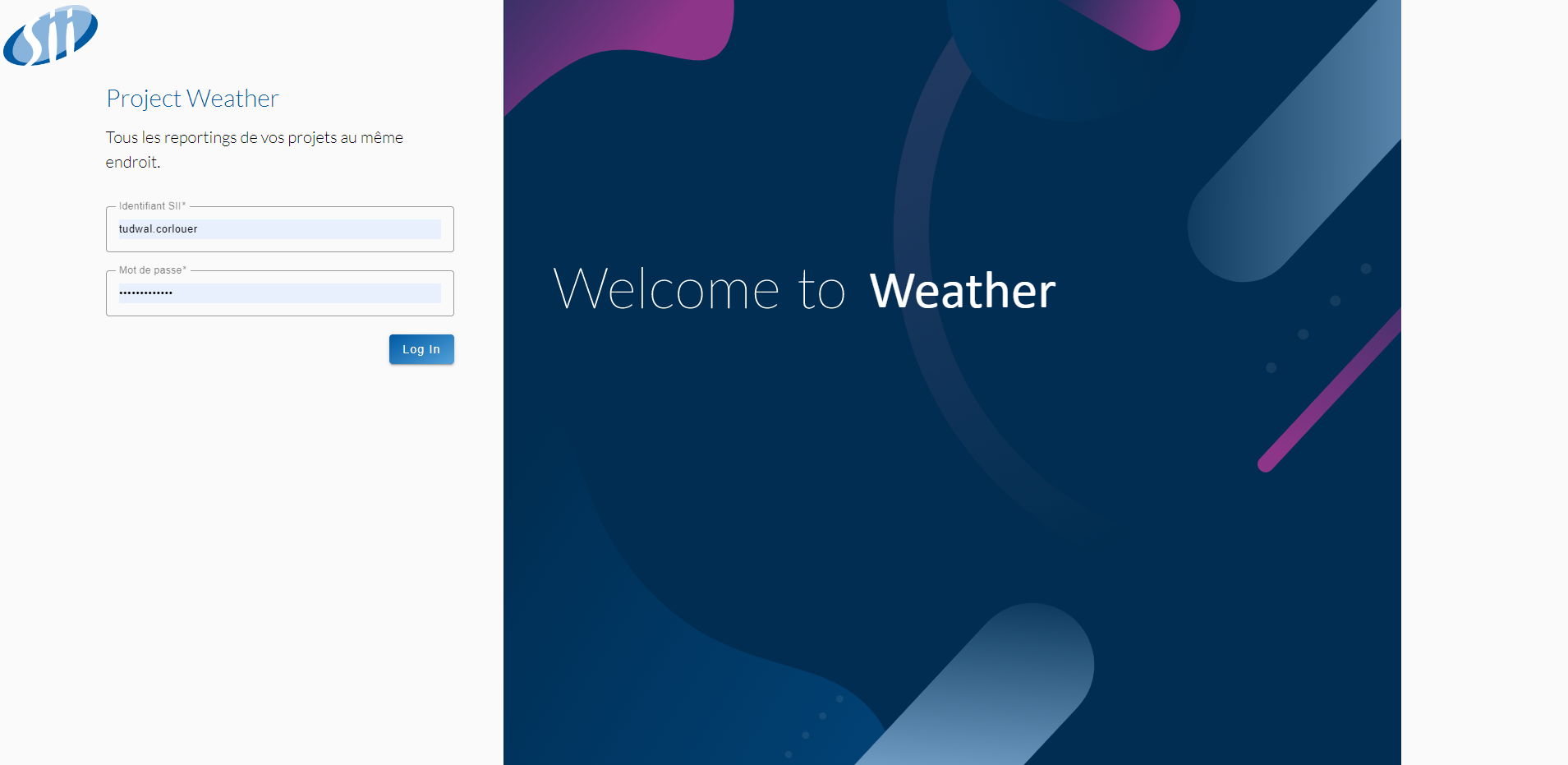


Annexe 2 : Workflow Dev&Co

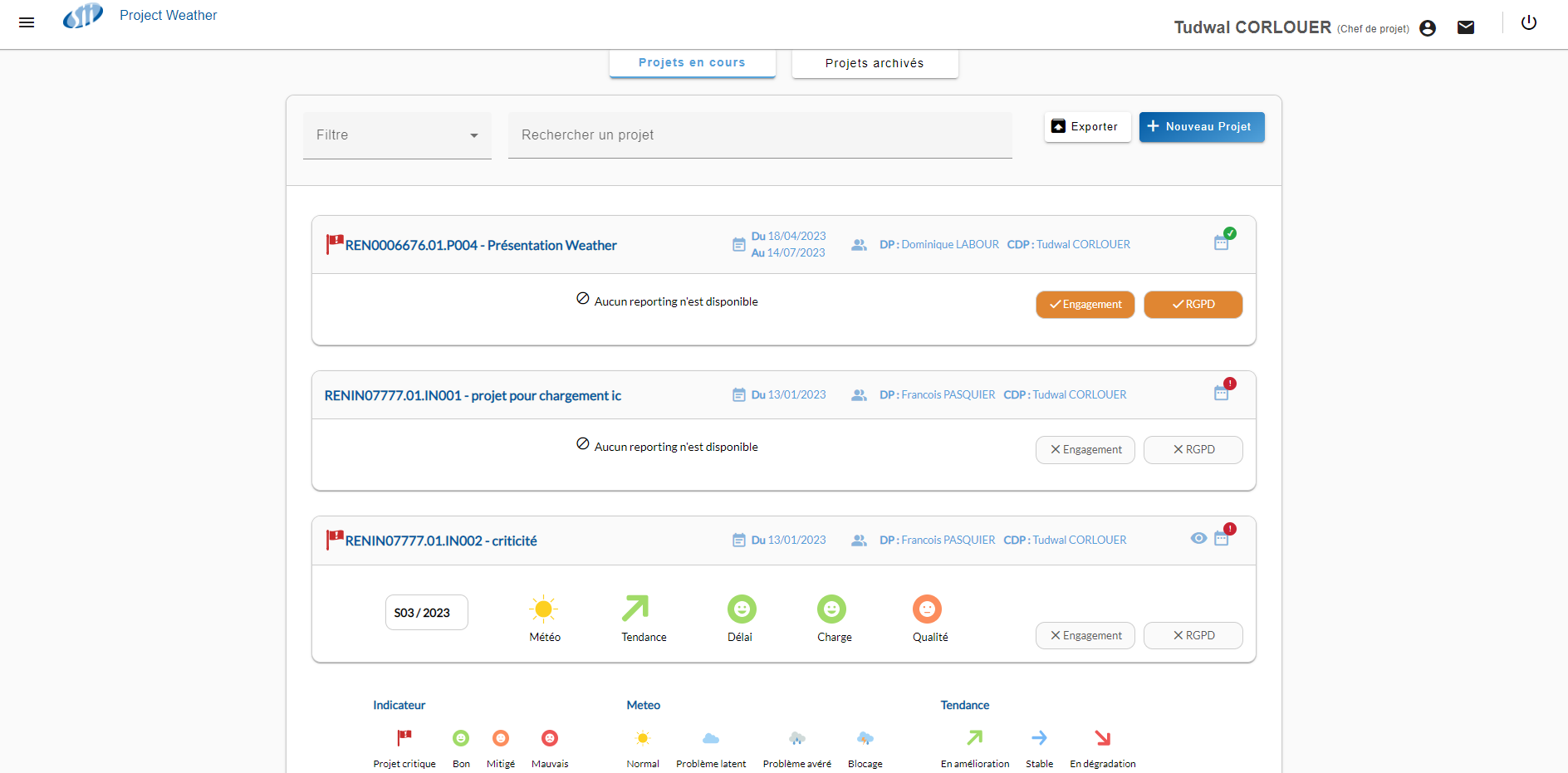


Annexe 3 : Maquettes et vue de l’application

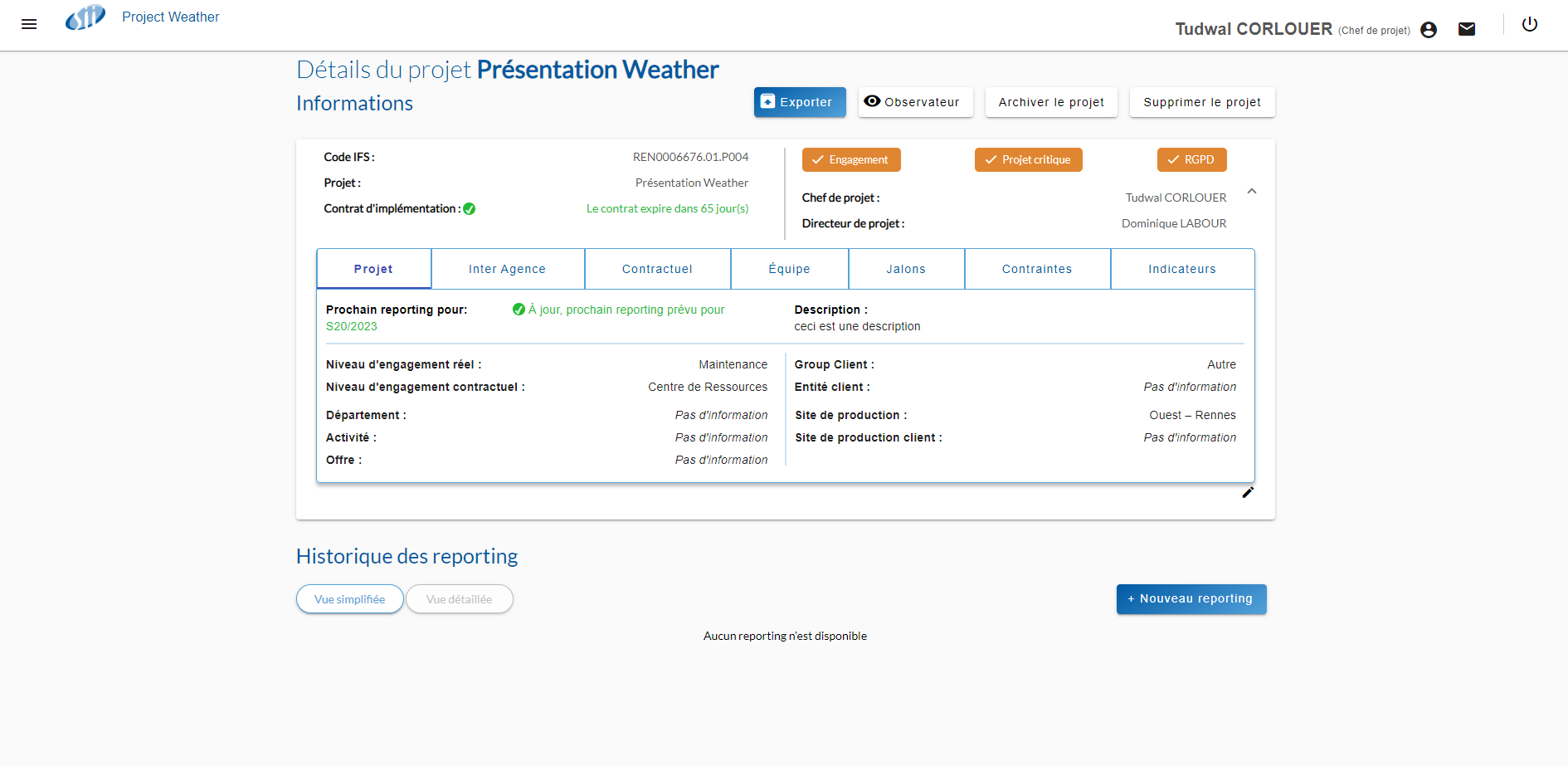
Page de connexion



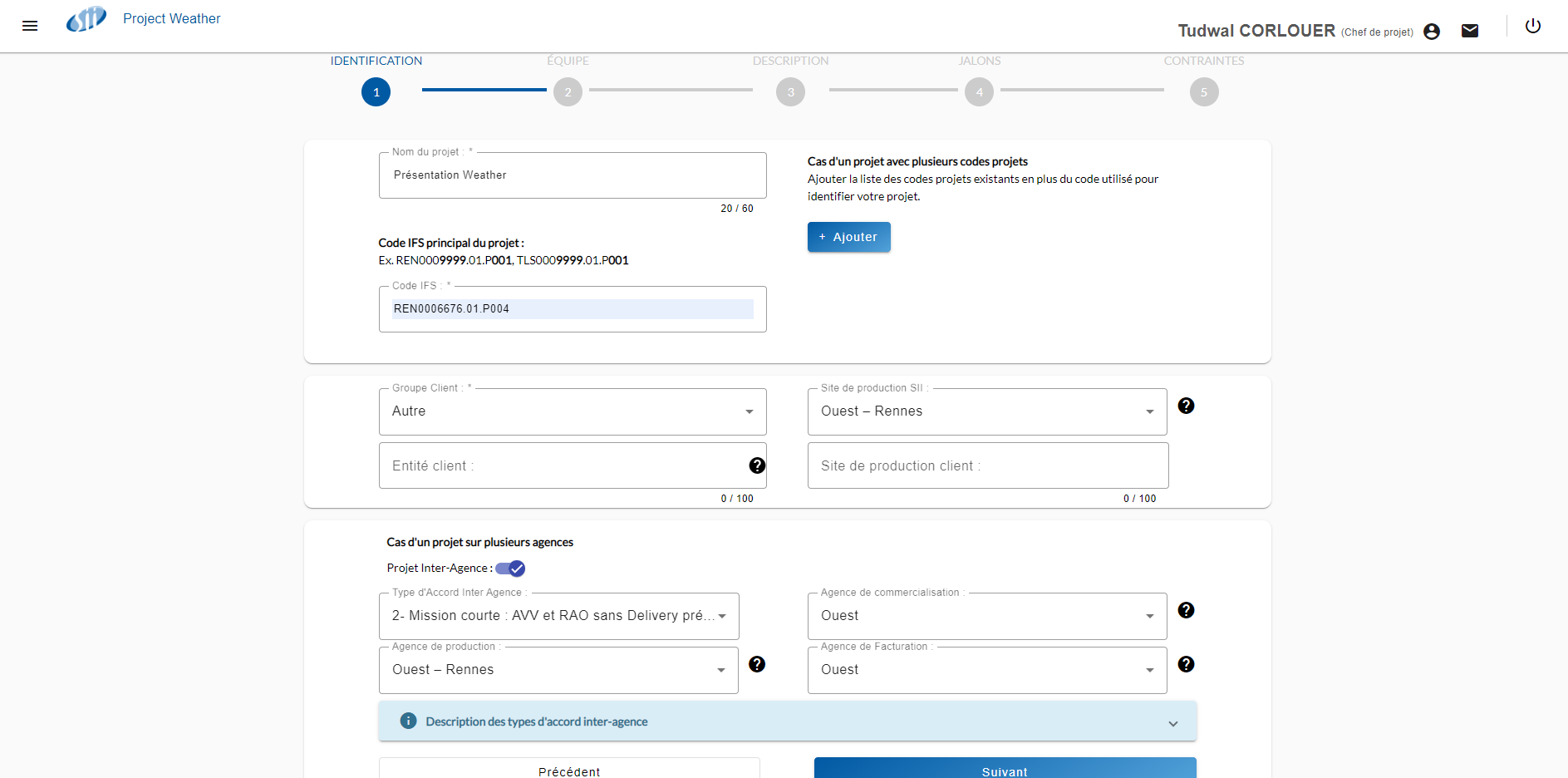
Page d’accueil contenant la liste des projets



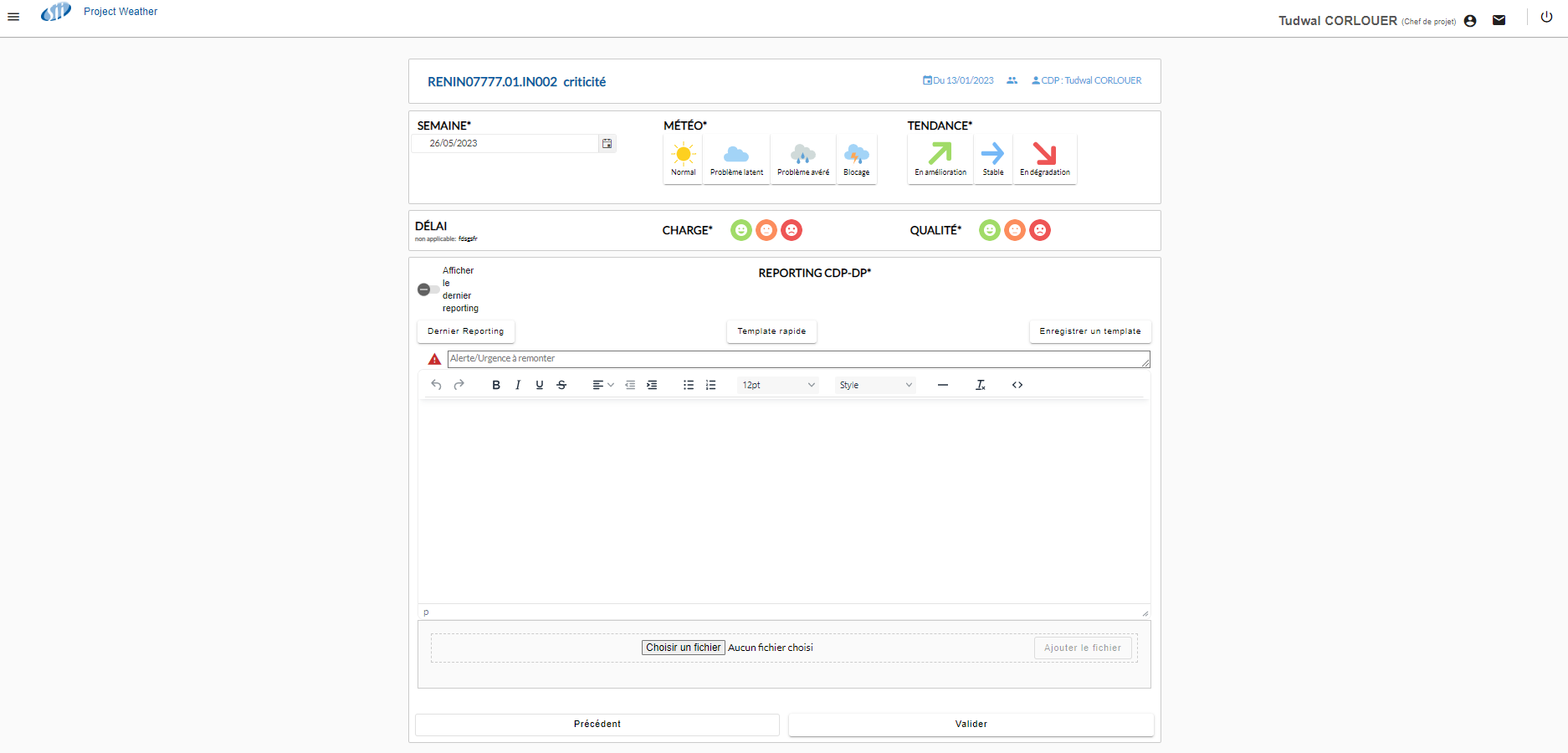
Détail d’un projet



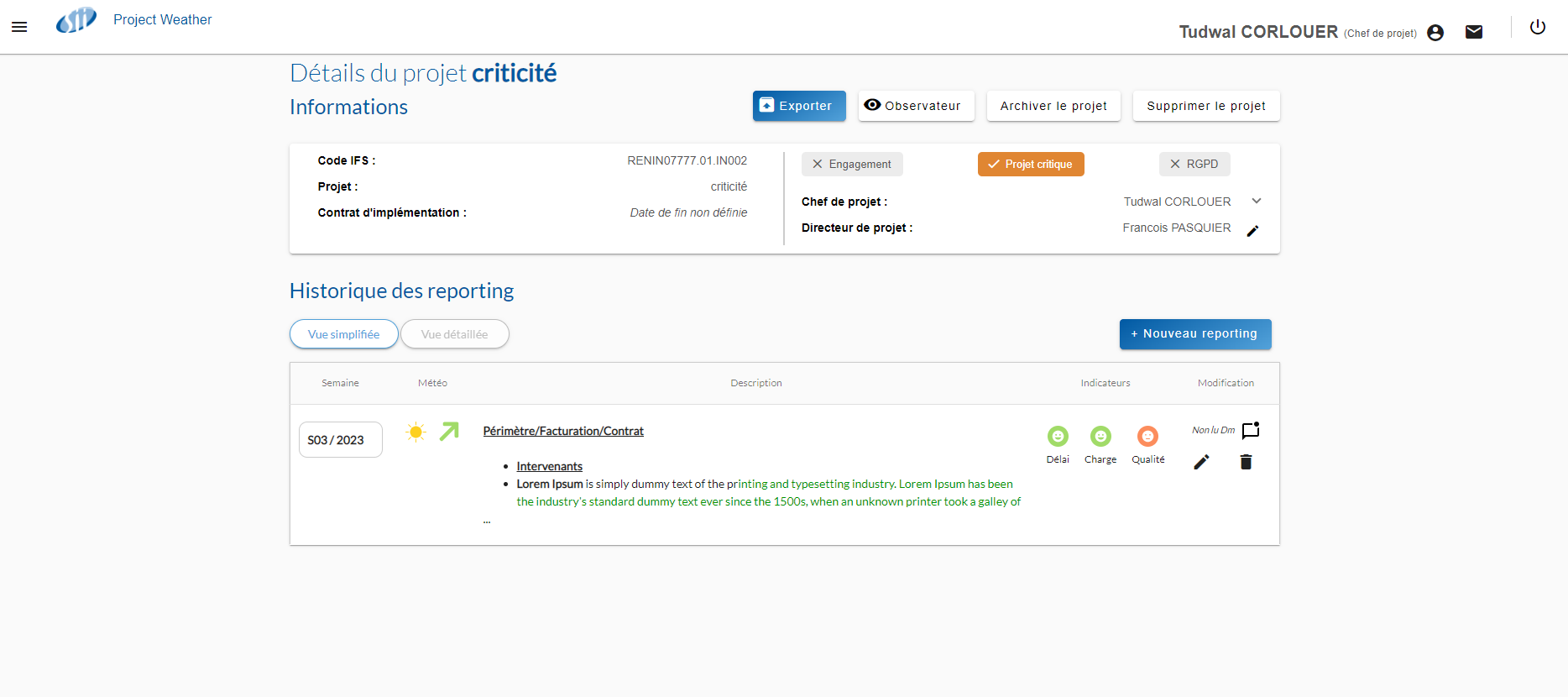
Page de création / modification de projet



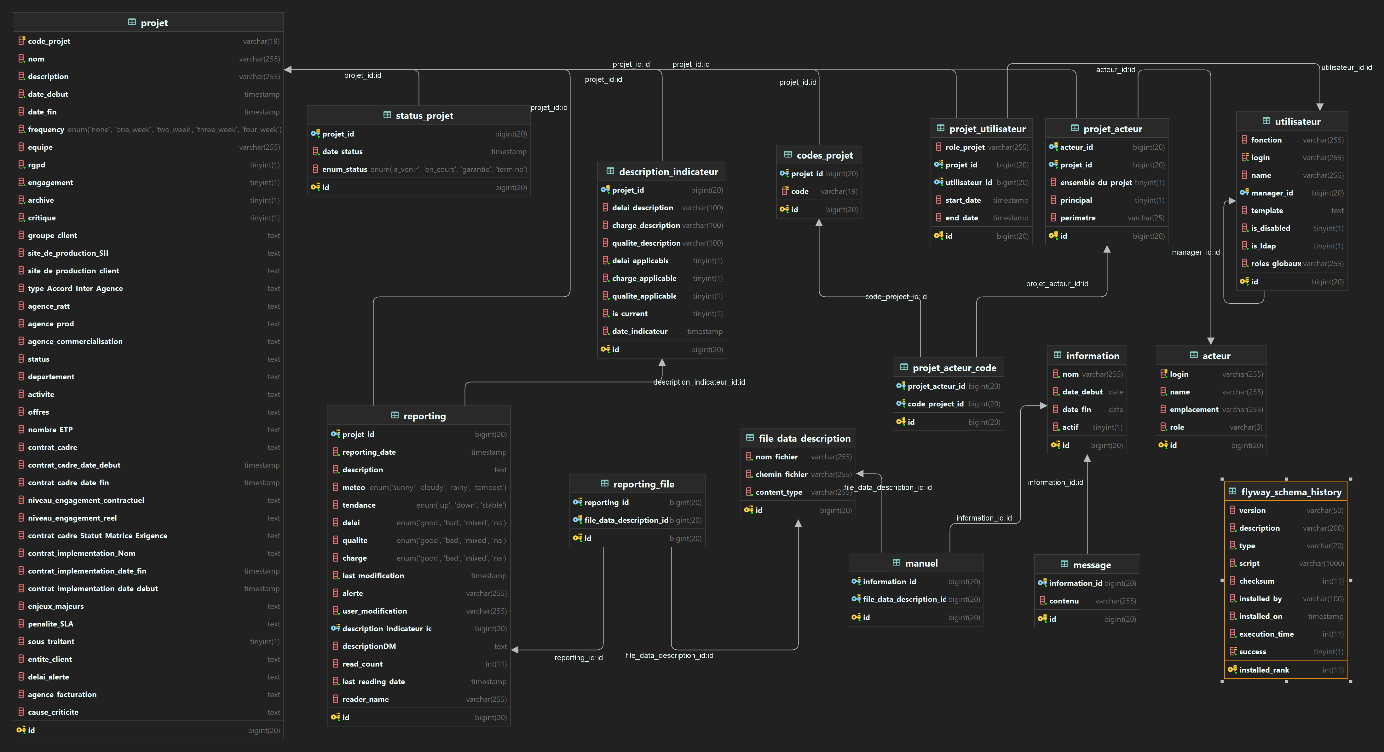
Création d’un reporting



Détail d’un projet avec un reporting

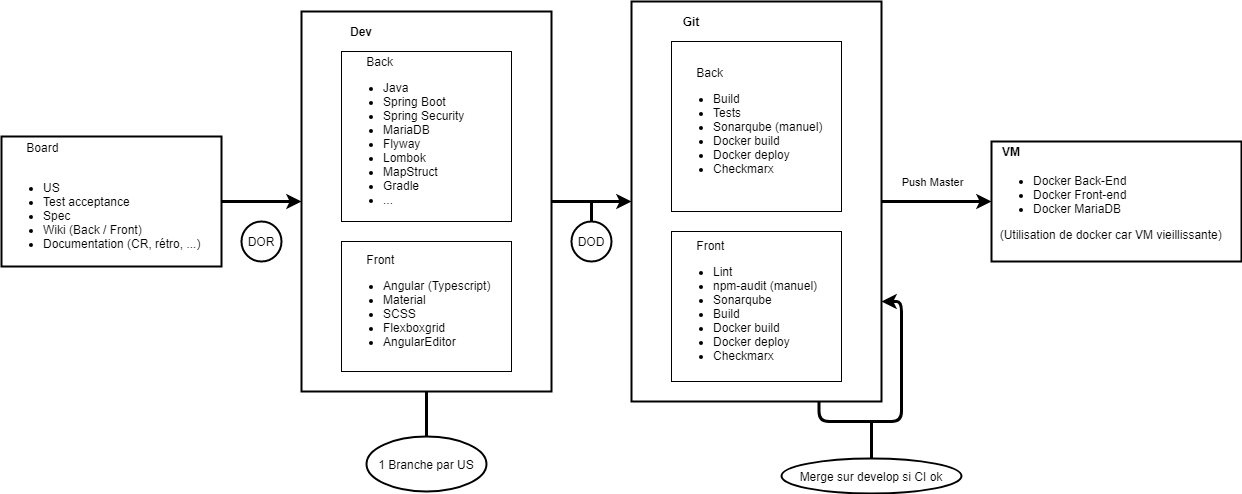


Annexe 4 : Schéma de base de données WEATHER



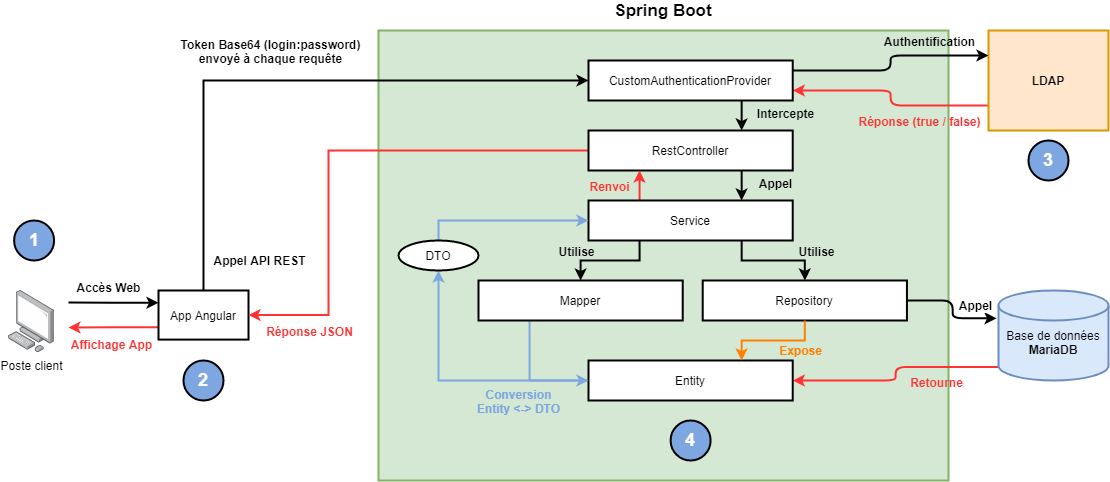
Annexe 5 :

Schéma de différentes technologies utilisées

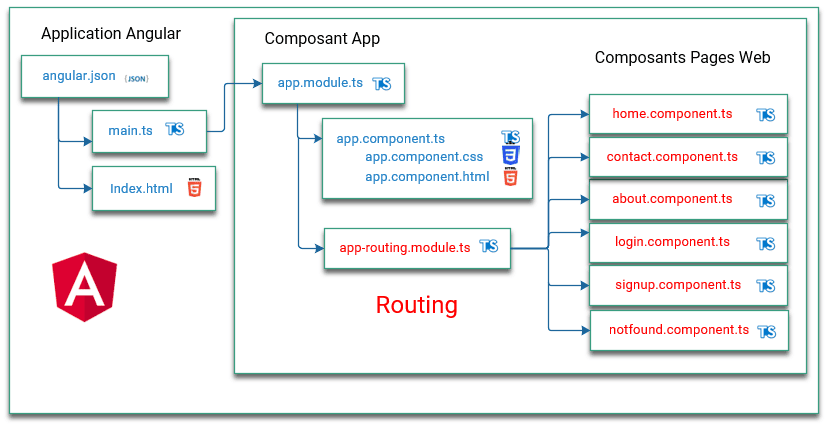


Annexe 6 :

1. Architecture de Spring

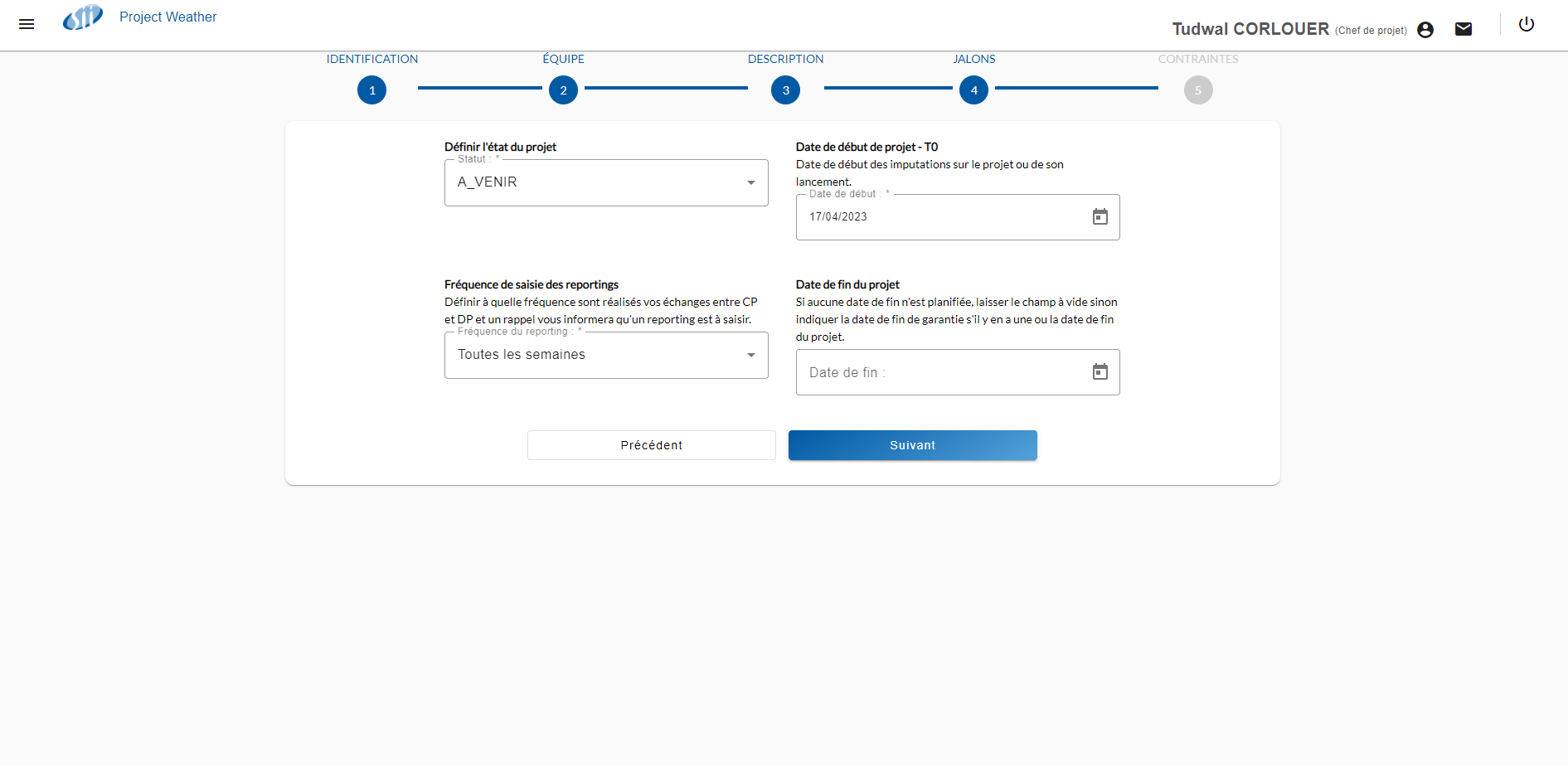


1. Architecture d’Angular

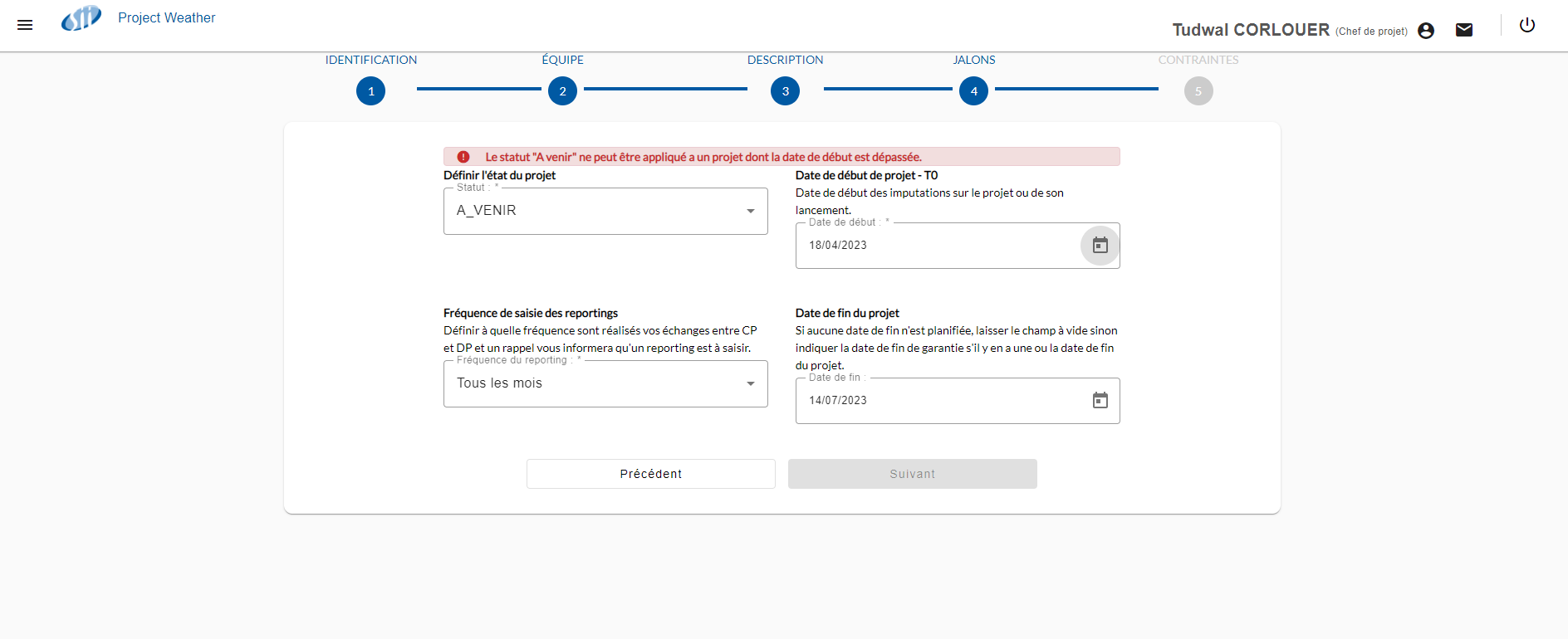


Annexe 7 :

1. Onglet Statut



1. Onglet Statut en erreur



1. Détails de l’avancement du projet

