**Soutenance stage :**

**WEEK 1 :**

Déroulement du stage :

Tâche 1 : installer les outils nécessaires au développement

Tâche 2 : choix de framework,

Back end : spring boot,

Front end : angular ou react.

**Day 1** : recontrer l’équipe, mon tuteur/manager, et présentation de l’entreprise et de son domaine d’activité. Installation de l’environnement web, maven, jdk, intellij, tomcat, configuration des variables d’environnements pour jdk et jre et maven . Cloner le projet principal depuis gitlab. Déployer le serveur et avoir accès à la page web à refaire. Problème de proxy qui bloque accès au site apache.maven pour importer les plugins et le dependencies maven.

**Day 2** : réussir à déployer le serveur à travers tomcat, et avoir accès à la page web. Toujours quelques problèmes du proxy. Solution, copier collier les plugins et les dépendences nécessaires existants depuis un autre post.

**Day 3** : phase d’installation de l’environnement web terminée et réussite. Passage à l’étape suivante : choisir quelque framework utilisé pour le développement. Création d’une présentation powerpoint qui représente quelques framework front-end : angular, react, vue.js, et quelques framework back-end notamment spring boot et django.

Pour le back-end, le code source est en java, et vu les avantages du spring boot, on le choisit comme notre framework backend.

**Day 3-day4** : essayer de créer un projet demo spring boot et de le lancer depuis mon ordinateur local, proxy qui bloque toujours l’accès maven pour importer les dépendances spring boot. Recherche de solutions, j’ai essayé de configurer le proxy, de forcer l’import des dépendances, changer les fichiers .settings.xml et pom.xml, changer les configurations réseaux pour intellij, pour le fichier.m2 , le fichier settings.xml de l’utilisateur, contourner le proxy en le configurant… rien n’a abouti à une solution

**Day 5** : terminer la présentation powerpoint des frameworks à envisager, et cherche de solutions pour résoudre le problème de proxy. Reconfiguration des différents fichiers settings.xml, tjrs pas de solution. Installation de node.js et création d’un projet angular et react pour le frontend. Proxy qui bloque la création des projets.

**WEEK 2 :**

**Day 1** : présenter le powerpoint dans une salle de réunion (S.garcia, Audrey, Mohammed) et discussion sur le sujet du proxy, choix de react pour le front end, et spring boot pour le back end. Spécification des nouvelles missions : créations des maquettes pour les différents pages html et les présenter dans une autre réunion, après : choix des maquettes, création de l’api qui reliera le back end et le front end, et refonte du backend. Recherche de solution pour le problème du proxy.

Réussir la configuration du proxy, et réussir à faire marcher le projet spring boot et react.

**Phase d’installation et de configuration et de choix de framework (tâche 1 et 2 réalisées) terminée.**

Tâche 1 :se familiariser avec react (3jours-1semaine)

Tâche 2 : commencer à concevoir des maquettes pour les différents pages html !

**Day2 :** début de la création des maquettes + phase d’apprentissage et de test de react js.

On s’est rendu compte que l’ancienne api (SOAP) ne nous permettra pas de communiquer tous nos données depuis le backend vers le front end. Donc changement de tâches : faire les maquettes html/css et passage au back end pour créer une nouvelle api rest.

**Day3 :** création des maquettes html/css pour les différentes sections. Début du test spring boot !

**Day4 :** Commencer à créer l’api REST qui permettra d’envoyer les données du backend vers le front end.

**Day 5 :**  API good for users and es (quelques modifications à faire) !

**Week** 3

**Day** 1 : ajout de fonctionnalité backend (traitement du service)

Création de 3 autres maquettes front end pour le login.

Test sur spring data jpa.

Réussir à communiquer avec le back end depuis un projet front end react à l’aide de la librairie axios.

**Day2 :** travail sur le spring jpa pour mapper le reste des classes. Réunion avec les membres de l’équipe.

‘Finalisation des maquettes front end’ et planification d’une réunion pour les présenter.

**Day3 :** réunion pour présenter les maquettes.

Réussir à lire les données depuis un fichier xml pour le backend.

Résoudre quelques erreurs de parsing

Conception de 6 nouvelles maquettes front end et les envoyer par mail à l’équipe pour un vote.

**Day4 :** Choix de la maquette pour la page login après quelques modifications !

Test sur React js. Composants/props/..

**WEEK 4 :**

Days2/3/4/5 : création du projet react et implementation des 4 requettes http get/post/delete/patch en utilisant axios !

Création de 2 maquettes pour les utilisateurs/systèmes élémentaires !

**WEEK 5 :**

Finalisation des pages users et es sur react et présenter ça à l’équipe.

**Day 1** : terminer la logique des requettes http pour users et es !

Ajout du css !

Finir une première version pour les pages users/es/admins (3/7)

Rebasculement vers le backend pour filtrer les données remontées à notre frontend!

**Day 2 :** Ajout de quelques fonctionnalités dans le projet react, qui permettent d’ordonner (croissant/décroissant) les données selon la colonne choisie par l’utilisateur.

Début d’implémentation du processus d’authentification et d’autorisation (spring boot security) ! puis après le filtrage des données !

**DAY 3 /4/5**

Implémentation de la sécurité en utilisant spring security jwt ! json web tokens.

Nécessité de créer une nouvelle base de données !

Sécurité Bonne ! cryptage des clefs secrètes !

Tâches suivantes : relier cela à notre frontend React.js

**WEEK 6 : 24/04/2023**

Tâches : 3 roles for users :Admin project and admin entity and utilisateur, with different access level !

2 roles implemented with 3 pages each !

**DAY1 :** travail sur le frontend pour lier ce qu’on a créé en backend !

**DAY2 :** finaliser le compte admin, ajout du logout et lier les pages aux titres !

**DAY3 :** ajout du compte utilisateur et bind all pages !

**DAY4 :** work on es page for users !

**DAY5 :** filter data coming from the backend, and rendering different views depending on the user’s role. add jwt test frontend side.

WEEK 7 : 01/05/2023

**DAY1 :** terminer l’implementation des 3 niveaux d’accès, et tous rooter en react !

Reduire le biolerplate code !

Minimiser le hardcoding des urls ! et le rendre dynamique selon le type de l’authorité ! optimisation du code

Problème de performance ! react : il faut viser le composant qui change pour ne pas faire un render de toute la page ! Bien utiliser react hook effect !

React window à envisager pour optimiser la performance de l’appli !

Ajout d’un admin entity controller coté backend ! et manipulation des jquery spring data jpa !

**DAY2/3/4 :** Finalisation de la page es pour les utilisateurs, et finalisation des pages user/admin/es pour les admins entity !

Finalisation de la page Role reviewer !

Custom jquery pour filter les données selon l’utilisateur connecté !

Ajout de class state, pour les états de chaque système, chaque système à 1 ou plusieurs états !, et d’une class assign pour assigner chaque system à un ou plusieurs utilisateurs, avec des enums et implémentation des relations entre tables (jointure) en spring data jpa (hibernate) ! résultat => une class qui modélise tous les systems avec tous les données dedans ! id, libelle, type, service, liste des states (json etat :en\_cours, indice :A, …), liste des utilisateurs (json, id : 120 , user\_nni :zakaria, userRight :operateur..), après quand un utilisateur login, on récupère son identifiant et son entité, et on ne lui envoie seulement les es qui ont la meme entité, et dans la liste des assigns, on trouve une cléf user\_nni avec comme id, l’id récupérée depuis le login !

Pour les admins entity, ils voient tous les es de la même entité peu importe les actions, et les admins project voient tout !

Basculement vers le frontend pour récupérer ses données json, et les displayer à l’utilisateur, création de composants qui gère ses données, et des composants filtres qui permettent de filtrer ses données selon les choix de l’utilisateur, type, libelle, type de droit,

Ajout du css pour rendre les pages plus agréable, utilisation des useState pour changer le nom de la classe de nos div, pour garder des effets d’une manière permanente !

Manipulation des requêtes sql sur la base de données postgres, pour tester des fonctionnalités !

Effectuer quelques changements au niveau de la couche service pour appliquer des actions sur nos données ! (Remplacement de l’user\_nni avec l’user nom et prenom ! user\_nni stocker dans une bd, et le nom et prenom dans une autre ! on fait appel à 2 interfaces ! esRepository et UserRepository)

Refactor code ! remove allMenu, and keep the menu only ! more dynamique, rendering a menu depending on the user’s role

**week 8**

**DAY1 /  /2 /3 :** Implémentation de la page role manager qui gère les roles !

Correction des bugs liées aux verbes http (requette vers le serveurs),

Lier le backend et le front end d’une manière plus efficace, tout ce qui passe au niveau backend, on le voit immédiatement coté front end,

Correction des erreurs liées aux fonctions asynchrones !

Ajouter des relations entre les tables, en ajoutant des relations, one to many, many to one dans les classes java !

Implémentation de la page qui montre les relations entre les systèmes élémentaires, modification du backend pour aboutir aux résultats souhaités !

Des problèmes dans les useState, c’est des fonctions asynchrones !!

70% projet réalisé !

Ajout du css, et ajout de plusieurs fonctionnalités qui améliore l’accessibilité de la page web, refactorer le code pour reduire le boilerplate code,

Manipulation de la base de données, psql cmd !

Utilisation du localStorage pour améliorer la performance de la page web.

Reste à ajouter du css pour améliorer la vue de la page relationEs, et de gérer le cas où on ferme le serveur !

Ajouter la possibilité de supprimer des actions que certains users ont sur des es !!

Comme d’hab, 2 réunions avec l’équipe pour discuter où on en est !

**Day4 :**

Corriger quelques bugs lier au type Optional, quand on essaie d’exécuter optional.get() avant de vérifier si l’optional existe !

Ajout du style pour la vue de la page esRelation,

Flex / grid

Ajout des opérations crud pour les roles, ainsi que les relations ! il reste update à finir pour les roles et les relations.

Centraliser l’affichage des données au cas où le serveur est arrêté !

Gérer les fonctions asynchrones notamment les use state, pour rendre l’affiche de la page dynamique.

Ajout du bussiness logique coté serveur pour aboutir aux résultats voulus.

Et implémenter la logique pour les autres utilisateurs : admin entity, utilisateur.

**WEEK 9 15 mai**

**Day 1 :** Ajout de l’option de supprimer toutes les relations d’un es, ou de supprimer une relation en particulier !

Ajout de l’option d’ajout d’une relation pour un es, ou de plusieurs relations

Travaille sur l’UI pour améliorer l’affichage, gérer encore les erreurs liées à l’optionnel. get() , qui retourne des nulls si l’optionnel n’existe pas !

Ajout de l’option de modifier un rôle précis qu’un user a sur un es !

Implémenter tous cela en frontend react avec axios

Reste à faire, d’implementer la possibilité de supprimer une seule relation et de modifier un seul droit, la partie backend est déjà implémenté et testé à l’aide de postman !

Ajout d’un filtre user des es !

**DAY 2 :** terminer ce qui rester à faire, implementer la possibilité de supprimer une relation, une action, ou toutes les relations/actions, modification des zones du service des zones input vers des listes déroulantes, traiter le cas où c’est admin entité, donc la zone service doit être par défaut l’entité de l’admin, modification des composants alltable et alltableEs.

Modifier les APIs endpoints pour qu’il convient les requests axios : point bloquant : pour une raison ou une autre, quand j’envoie le body + les params dans un request axios, le body ne se ‘map’ pas bien et le body que le backend recoie est vide ! solution envoyer des paramètres et construire le body d’après ses paramètres dans le backend

Ce qui reste à faire, faire l’update des services pour ne pas avoir des chiffres mais les vraies entités et après entamer la page TOD !

**DAY3 :**

Corriger le bug qui faisait que les services/entités des users/es devient null quand on modifie la valeur des autres champs !

Back end front end updateotherField

0 -> cnen affiche cnen send cnen

To back end

When refetching the data, he finds cnen instead of 0, and cnen is not in the conf file, so he prints null.

To solve it, we need to check what the frontend is sending to the server, if its 0 /1/2/../7 we send the corresponding value,

If it s already a value cnen/cnepe/.. we look for the key corresponding to the value cnen, and change the value sent from the frontend to its key value cnen -> 0.

Implémenter presque tous les cas d’utilisation, on ne peut pas modifier le type après création, on ne peut pas modifier les services des admins, mais on peut pour les utilisateurs, faire une liste déroulante, le admin choisit cnen par exemple mais la valeur envoyée au serveur c’est 0, la clé du cnen.

Ajout de page publication, et début de son implémentation avant la page tod, puisqu’elle est plus facile !

Push au dépôt gitlab.

Correction de pas mal de petit bug au niveau de l’affichage (désactiver le bon input selon la colonne...) des bugs au niveau du backend ! ajout de pas mal de test de vérification !

Rendre le css dynamque selon la taille de l’écran : media query.

Un premier prototype pour les pages Users/entity/esrelation/es/Rolemanager et rolereviewer est finit.

Ajout d’un composant loader, pour remplacer le Loading… en attendant la réponse du serveur (axios asynchrone functions !)



Je commence la page Publication !

**WEEK 10 : 22/05**

**DAY 1 :**

Ajout du style sur la page esRelation, changer les boutons en carte.

J’ai décidé d’implémenter la page tod avant la page publication, puisqu’elle est plus complexe.

Implémentation du backend pour la page tod :

Ajout du package alloc, bfe, icfunction, mui, signal, ste. Chaque package contient au moins 3 classes (controller + service + entity) et une interface (repository). (Une potentielle Class pour la clé primaire composée) implémentation des opérations crud pour tous ses packages, sauf l’update pour signal et alloc.

Ce qu’il faudra faire par la suite est de basculer vers le frontend et de récupérer ses données et de les afficher ! avec l’implémentation des UI qui permettent de réaliser les opérations CRUD !

**Day2 :**

Ajout des classes \_x, \_u ainsi leurs repository pour les packages bfe, mui, signal, ste !

Mapper toutes les tables dans des classes java.

On passe au front end pour récupérer tous cela !

Travaille sur la page tod : css menu et submenu.

Terminer une première maquette et commencer à implémenter les requettes fetch vers le serveur !

Problème le paramètre passé en argument dans axios.get n’est pas bien transmis au serveur !

**DAY3 :**

Réglage du page TOD, et ajout de rest api endpoint pour gérer les différents composants, mui mui-x et mui-u.

Réglage de l’affichage et des states

Présentation du travail à l’équipe edf qui venait visiter corys.

Leurs remarques :

Tant q une action exite on ne peut pas supprimer user

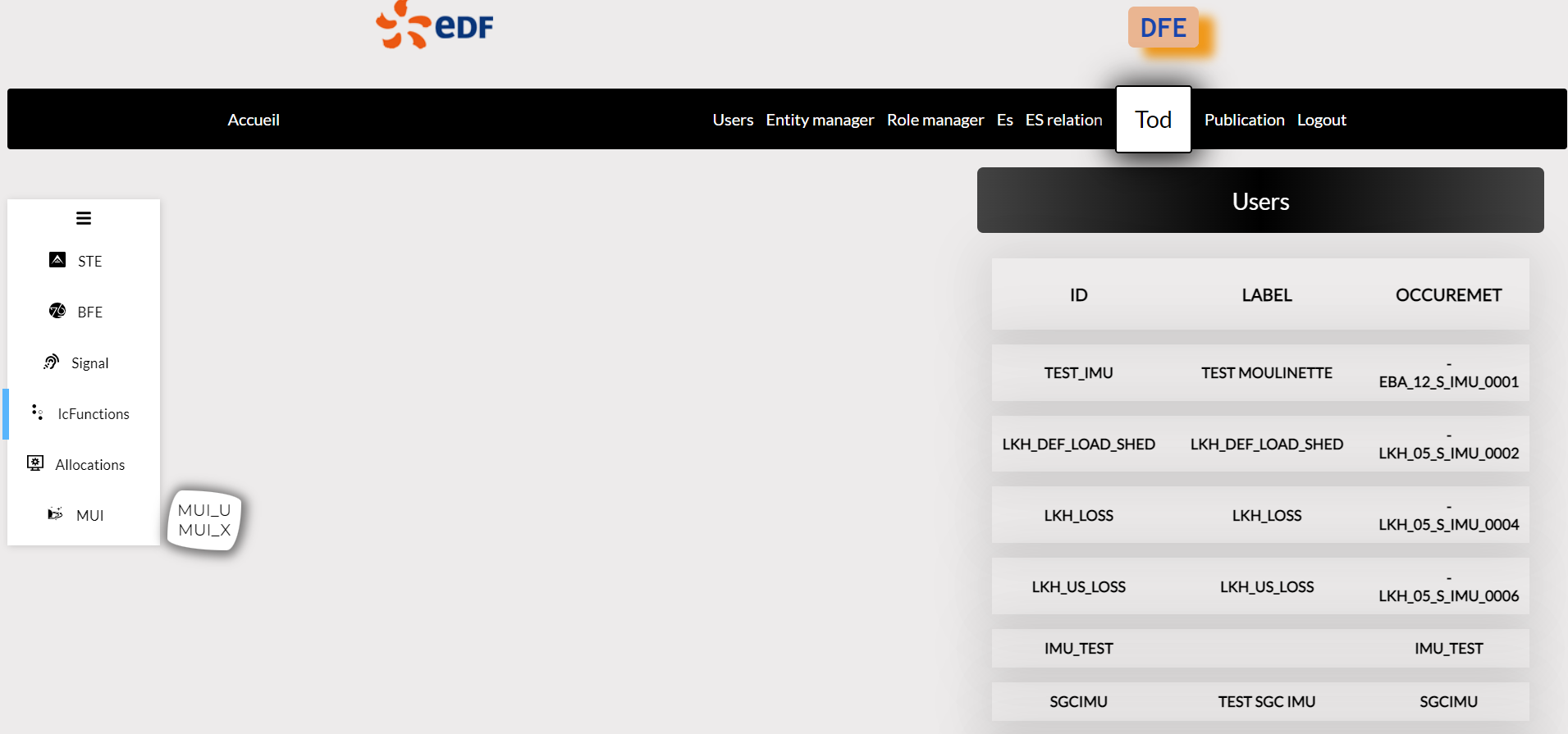
Superuser

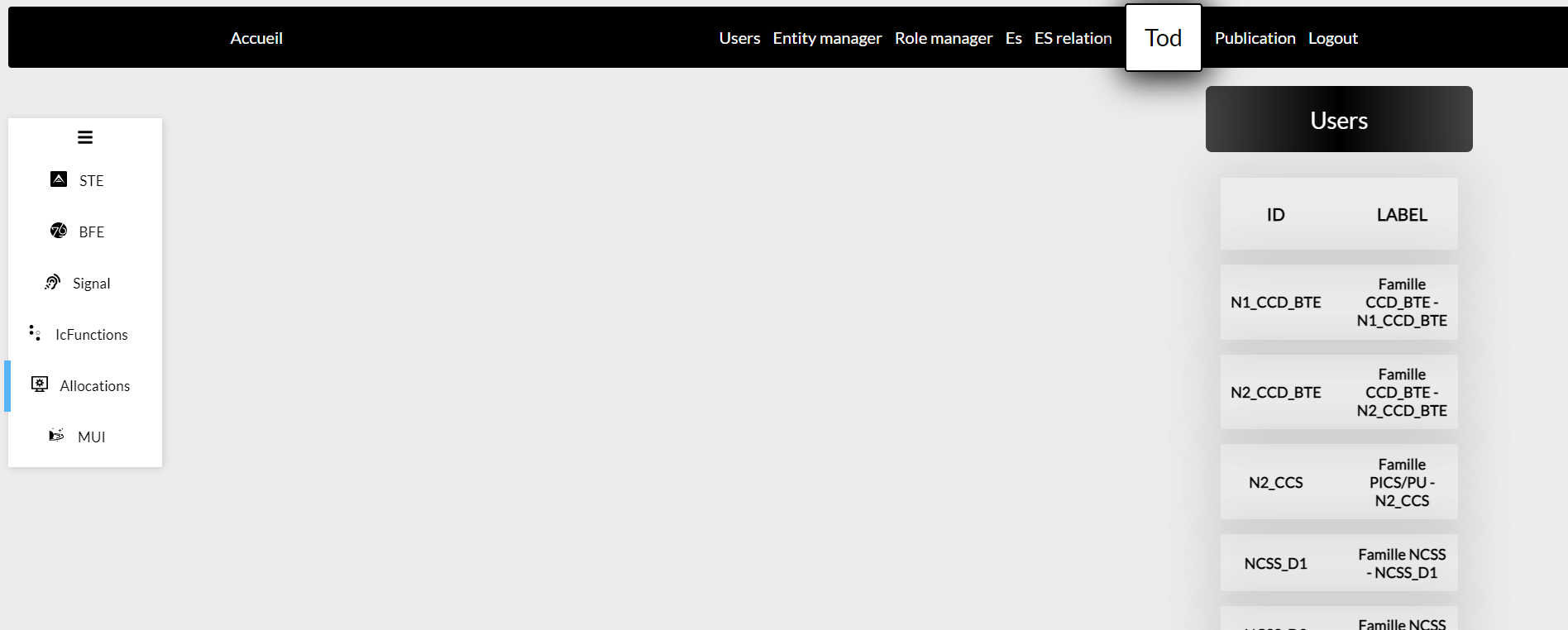
Tod filter etat coherent

Réunion avec le graphic designer du corys.

**DAY4 :**

Affichage des données sur la page tod, selon le composant choisi (mui, ste, ..) dans un tableau !



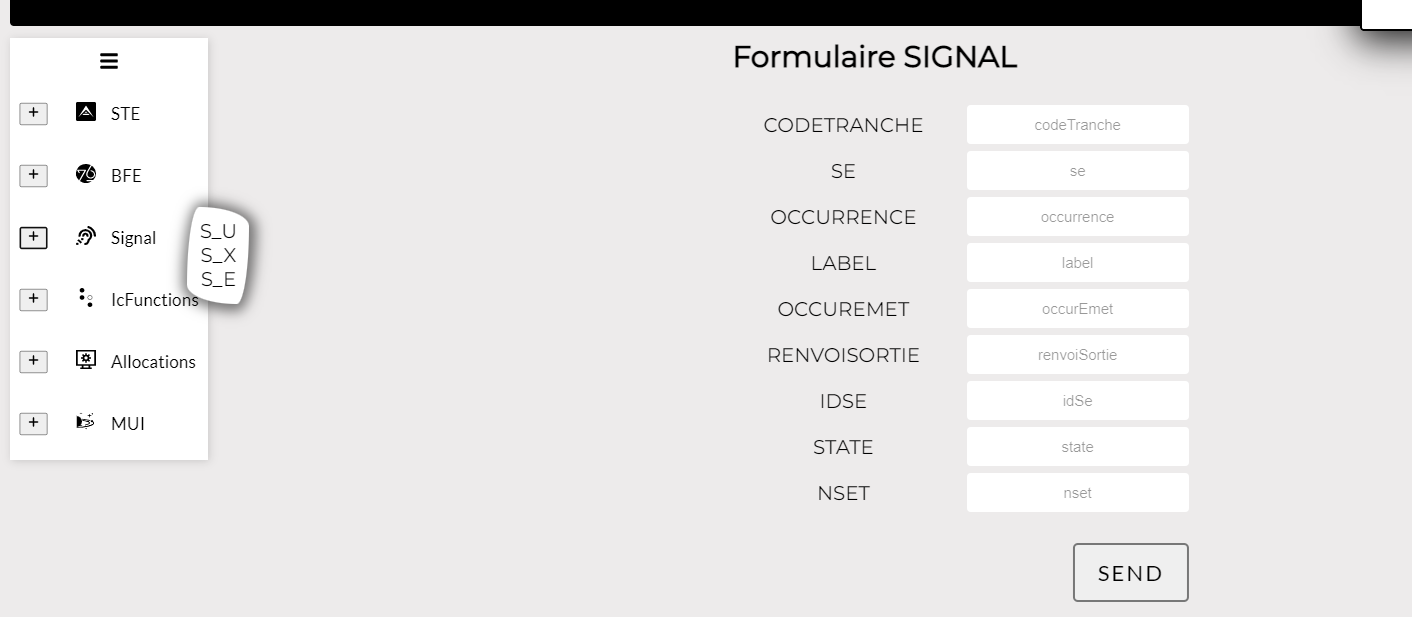


Il reste à filtrer les données (ne pas afficher tous les champs) et implémenter les opérations crud !

**DAY5 :**

Ajout d’un composant qui permet d’ajouter des données selon le composant !





Modification du css ! ajout de contraintes sur la longeur des <td> avec max width, scroll, ..

**WEEK 11 : 30/05**

**DAY1 :**

Finaliser les maquettes de la page tod sur figma, et implémenter ça en changeant le css !

Mettre à jour la totalité des composants react, en utilisant react awesome icon.

**DAY2 :**

Quelques modifications sur la page esRelation !

Post + delete good for all pages in tod.

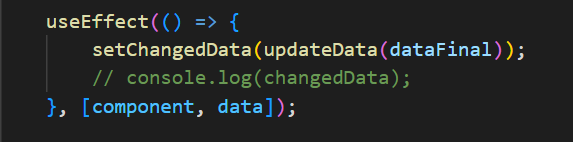
Modification du layout de la page tod !

Mise en page des données.

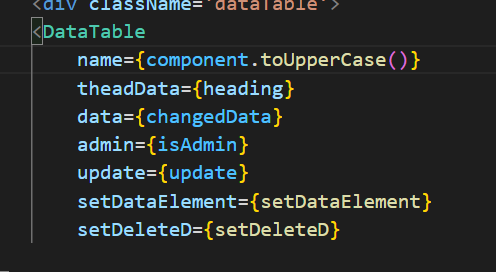
**DAY3 :**

Concaténation des colonnes GOOD !



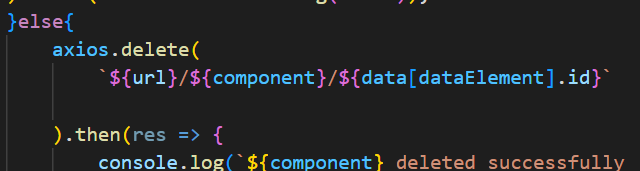


Ajout d’une fonction qui permet de changer les données d’entrées du composant datatable !



On lance l update avec un useEffect qui se trigger quand on change data ou composant !!

Après on fait une seule modif sur le composant dataTable pour au lieu d’envoyer la ligne quand on veut supprimer un élément, il envoie l’indice de la ligne pour pouvoir récupérer la ligne entière depuis data et non pas changedData !!



Correction de bug lier à la page esrelation avec les admins entity

**DAY4/5**

**+**

**WEEK 12 : 06/23**

**DAY1**

Page tod terminée !

Get/post/delete/put good for all 15 tables !

With some specifics regarding every one !

You can update only label for some table, for others you can update more fields, and for others you update all fields !

All good for Admin project, admin entity, and utilisateur !

**Day3**

Modification de types composant, et quelques modifs du reste des composants !

Mettre en place des mesures OWASP !

(voir document word owasp)

**DAY4/5**

Grand avancement au niveau de la mise en forme ! homogénéiser toutes les pages ! css !

Corriger des bugs !

Implémentation de la logique qui permet de changer le state des systèmes

**WEEK 13 : 12/06**

**DAY1**

Implémentation de la logique derrière les états des systèmes !

Chaque système peut avoir un ou plusieurs état, INIT/ RC1/ RC2/ …

Chaque état à 4 étapes : initier, en cours, à vérifier, vérifié, et approuvé, et chaque utilisateur peut avoir une combinaison bien précise de rôle, il peut être vérificateur mais non pas approbateur, ou approbateur mais non pas opérateur, etc, et selon le rôle que l’utilisateur a sur un système et l’étape dans laquelle le système se trouve il peut faire certaines actions ! par exemple le system LJ est à l’état à vérifier, donc seuls les vérificateurs de se systèmes peuvent agir sur ce système soit en approuvant donc le système passe à l’état vérifié, et l’approbateur du système prend contrôle, soit en vérifiant le système et en refusant le passage à l’état qui suit pour des raisons comme mal fonctionnement du système/ un bug et donc le système revient à l’état en cours, et seuls les opérateurs peuvent y agir !etc, si l’état du système est à vérifier mais l’utilisateur est un approbateur donc il ne peut rien faire, jusqu’à ce que le vérificateur vérifie le système !

TO DO NEXT :

Inverser la logique derrière la page RoleManager => instead of choosing a system and adding users to it, we choose a user and add system to him.

DONE !

We can even add roles from filters !

Réunion avec le graphiste !

**DAY2/3/4/5 :**

Changement de la mise en forme de plusieurs composants, pour améliorer l’expérience utilisateur, et donc changement de la manière d’implémentation dans certaines pages !

**WEEK 14 : 19 /06**

**DAY1 :**

Quelques modifications au niveau du css et js dans certaines pages !

Implémentation de la pagination => GOOD !

**DAY2 :**

Finition de la pagination.

**DAY3 :**

Ajout d’une fonctionnalité qui filtre les données de la page tod pour tous les composants ! spring data jpa, ORM, hibernate, en manipulant ça côté front end.

**DAY3/4 :**

Travail sur la mise en forme. Le graphiste approuve la mise en œuvre actuelle.

**WEEK 14 : 26/06**

**Day 1 :**  ajout de filtre pour les autres pages selon l’entité pour la page des utilisateurs, et selon l’entité + type pour la page system élémentaires.

**WHAT IS LEFT !**

Page TOD => composant icfunction ajout des 2 jointures avec les tables input/output !

Page TOD => finalisation des formulaires ! FAIT

Page Relation => checkbox instead of select/option

Pagination => leverage the localstorage + USErefto keep the effect of the active page ! FAIT

Ajout de la page publication ! FAIT

Implémenter CSRF.

**DAY2 /3**

Travail sur les 15 formulaires ! TOUT EST BON !!

**DAY4 :**

Quelques modifs sur les formulaires, et implémenter le mécanisme qui permet de persister la page active du composant Pagination même après render !

**DAY5 :**

Ng

**WEEK 15 : 03/07**

**Day 1 :**

Ng

**Day2/3 :**

Travail sur figma pour designer 2 maquettes pour la page pub !

Choix d’un tableau.

**Day4 :**

Correction d’un bug lier à la page icfunction causé par le id qui est généré automatiquement du coup, il coincide avec les systèmes qui ont déjà cet id !

Correction d’un bug lier à la pagination : quand on choisit un filtre par label ou par type, on est censé retourné à la page 0 par default !

Début de l’implémentation de la page publication.

Fin de l’implémentation de la maquette publication.

**DAY5 :**

Présentation mi-parcours avec Monsieur Song.

Travail sur du style css.

Ng.

**WEEK 16 : 10/07**

Work on making the web app more secure !

<https://www.stackhawk.com/blog/react-csrf-protection-guide-examples-and-how-to-enable-it/>

<https://www.baeldung.com/spring-security-csrf>

<https://docs.spring.io/spring-security/reference/features/exploits/csrf.html>

<https://www.stackhawk.com/blog/react-xss-guide-examples-and-prevention/>

Apparenlty i don’t have to implement csrf protection because i don’t use cookies to authenticate but rather tokens stored in localStorage.

<https://www.baeldung.com/spring-security-csrf>

**Day2 /3/4 :**

Faire quelques logos pour la page web

**WEEK 17  : 17/07**

**Day1 /2/3 :**

Ng (angular)

Recherche devops : docker/ kubernetes pour deploiment.

Début de la rédaction du rapport !