Semaine 1 : 28/09 – 02/10

* Correction des dates, et du typeahead (ouverture, et non fermeture des choix)
* Ajout d'une colonne dans le tableau de résultats pour indiquer les status (FINISHED, STARTED, …) avec des icones.

Semaine 2: 12/10 – 16/10

* Filtres des recherches pour renvoyer toutes les instances qui n'ont pas d'instance parente, pour limiter les résultats, et avoir une meilleure hiérarchie.
* Changement des noms Utils -> Util
* OF-core : jeux de tests pour mon appli, envoie de message dans la queue
* Tri des résultats par status (FINISHED, STARTED, ERROR, …)
  + Problème : tri sur la page en court, et pas sur tous les résultats du tableau !
  + Résolution : change gridApi, pour spécifier l'action à faire, lorsque l'on clique sur une colonne pour la trier. Je relance la requête pour trier directement tout les éléments et ça rempli le tableau seulement après.
* Ajout du niveau des priorités pour les status, apres la collecte des messages, pour l'envoie dans ElasticSearch
* Tri des status par leur priorité : petit probleme: le ng grid ne rafraichi pas les directives de status. Resolution : watcher sur la cellule de la colonne, et si elle change, on met a jour l'icon, la couleur et le tooltip.
* Commit change
* Debut d'association de of core et of front. Je dois lancer elasticsearch et of core (collecte) depuis of front, des le lancement du serveur tomcat.

Semaine 3: 26/10 – 30/10

* Association des deux projets, avec leurs dépendances maven. Test pour voir si tout fonctionne bien. OK.
* Lancement d'Elasticsearch au démarrage de tomcat. Un elasticsearch embarqué. Ajout d'un plugin java, et modification pour qu'il s'adapte a nos besoins….. OK
  + Problème : je ne sais pas ou est instancié ES.
    - Résolution: Enfaite il se trouve sur l'"url-pattern" de la servlet que le lance, je devais ajouter à "/elasticsearch", "/\*" a la fin, pour pouvoir explorer elasticsearch, et pouvoir faire des requête dessus.
  + Problème : accès interdit, via CrsfFilters, pour les requêtes de type POST,
    - Idée de résolution: creation du CsrfFilter perso pour qu'il accepte les requetes POST. Definition du csrf filter dans spring security avec <security:csrf request-matcher-ref="maclass"/>
    - Fonctionne -> ajout des données déjà creer sur le serveur, pour les integrer dans le dossier spécifié
* Resolution des erreurs au demarrage de tomcat. Ex : log4j qui n'a pas acces au fichier de log
* Ameliorer l'outil de versionning de documents. 2 logiques de versionning proposé, a moi de choisir l'une d'elle.
  + Logique 1 : Comparer le nouveau document (avant insertion), avec le document actif. Puis ajouter dans la base de versionning, que les champs qui diffère entre les 2. Et inserer le nouveau documents comme actif.
  + Logique 2 : Inserer dans la base de versionning le document actif au complet, puis mettre le nouveau document comme actif. Et faire tournée une routine qui va comparer les champs et supprimer ceux qui sont identiques.
  + Reflexion -> logique 2, compliqué et surchage de boucle pour comparé dans la base de versionning ce qui a changer etc.
  + J'ai choisi la logique 1, apres avoir trouver une api java qui compare 2 objets, et renvoie les differences entre les 2, ce qui permet de savoir ce qui a reelement changer, et nous permettra d'ajouter directement dans la base de versionning que les champs qui ont été modifié.
  + Test de l'api java Javers qui compare 2 objet et renvoie les champs qui diffèrent.
  + Implémentation de Javers dans le projet