**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 statuts (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 statuts, après la collecte des messages, pour l'envoie dans ElasticSearch
* Tri des statuts par leur priorité : petit problème: le ng-grid ne rafraichi pas les directives de statuts. Résolution : watcher sur la cellule de la colonne, et si elle change, on met a jour l'icone, la couleur et le tooltip.
* Commit change
* Début d'association d'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: création du CrsfFilters perso pour qu'il accepte les requêtes POST. Definition du csrf filter dans spring security avec <security:csrf request-matcher-ref="maclass"/>
    - Fonctionne -> ajout des données déjà créer sur le serveur, pour les intégrer dans le dossier spécifié
* Résolution des erreurs au démarrage de tomcat. Ex : log4j qui n'a pas accès au fichier de log
* Améliorer l'outil de versionning de documents. 2 logiques de versionning proposé, à 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 insérer le nouveau document comme actif.
  + Logique 2 : Insérer 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.
  + Réflexion -> logique 2, compliqué et surcharge de boucle pour comparé dans la base de versionning ce qui a changer etc.
  + J'ai choisi la logique 1, après avoir trouvé une API java qui compare 2 objets, et renvoie les différences entre les 2, ce qui permet de savoir ce qui a réellement 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
    - Pas beaucoup de doc, difficile d'accéder aux propriétés et à leur valeur (celle qui ont été modifié)

**Semaine 4**

09/11 – 13/11

* TODO: (BOA) modifier l'interface Kibana, pour ajouter des colonnes au tableau, et donc modifier la requête qui renvoie les résultats.
* DONE: ajouter les colonnes c'est bon, il manque de modifier la requête sur elasticsearch pour ajouter les données que je veux pour les mettre dans mes nouvelles colonnes.
* IN PROGRESS: créer nouvelle requête pour avoir le compte d'erreur uniquement
* DONE: requête trouver, et insérer.
* IN PROGRESS: insérer les données dans le tableau de résultats
* DONE: insertion des données dans le tableau
* TODO: lister les applications sans que ce soit possible de les filtrer
* Je refais la requête pour ajouter les erreurs et succès, elle s'exécute juste avant d'insérer les données dans le tableau.
* Pas facile, bcp d'erreur de promise (asynchronisme), avec les \_.each. J'ai tout repris, la logique c'est : kibana charge et compte les noms d'applications (nombre max de 10), et seulement apres qu'il ait fini, je refait une requete, qui va chercher le nombre d'erreur et le nombre de succes de cette application.
* Kibana calcul une ligne supplémentaire "Other values", avec le nombre restant de données (etant donnée qu'il y en a plus de 10), cependant sur mes colonnes que j'ai ajouter, rien n'était indiqué, donc j'ai créer une nouvelle requete qui s'execute avant le traitement des données et qui me renvoie le nombre maximum d'erreur et de success, et puis au moment du traitement des données, des qu'il insert un nombre d'erreur (ou de succes), il increment une variable de cette somme, pour qu'a la fin du traitement on ai le total d'erreur que l'on a afficher (et de succes), et ensuite j'insert a coté de "Other values" le nombre total d'erreur – la somme trouvé pour avoir le restant.
* OBJECTIF semaine prochaine : Faire un tableau FIXE avec que les noms d'applications, un bouton d'action qui ajoute des filtres pour les autres graphes, MAIS lui ne change PAS !
* Idee : créer un style (ex : tablefixe), avec une requete spécifique a integrer dans le code (voir panel/terms/module.js et modifier module.html pour faire un tableau spécifique avec les bouton etc. Cool ☺

**Semaine 5**

23/11 – 27/11

* f