Notice du micro service de notation

Auteur :
Guillaume de Bigault de Granrut

6 juillet 2020

Table des matières

Introduction

Ce micro service permet de calculer tous les résultats nécessaires au fonctionnement de la solution proposée par Serpico. Ce service, totalement indépendant du reste de la solution, prends en entrée un json avec les données nécessaires, et renvoie en sortie un autre json contenant les résultats calculés. Ce service permet la jockerisation. Ce service est développé en python.

I. - Les résulats calculés

Cette appli calcule des résultats pour des critères, des phases (stage), et des activités. Il y a deux types de résultats :

- Les résultats individuels : ce sont les résultats réalisés par une entité, que ce soit un user, une team, ou même une phase. Ils se décompose en deux types de valeurs :
  - Les résultats weighted : les résultats pondérés par le poids des notés et des notants, selon utilisation dans les formules.
  - Les résultats equal : les résultats non pondérés, ce qui équivaut à utiliser une pondération de 1.

  Dans un souci de simplification, les formules seront détaillées une seule fois, avec la pondération.
- Les résultats globaux : c'est des moyennes sur les différents critères, ainsi que quelque autres paramètres calculés. Ils se décomposent en 4 cas (plus en compant les valeurs avec un maximum théorique) :
  - Les valeurs sur les user : moyennes calculées sur les user, et non les team. Elles se décomposent également en weighted et equal.
  - Les valeurs sur les team : moyennes calculées sur les team, et pas les user. Elles se décomposent également en weighted et equal.

  Dans un souci de simplification et de lisibilité, les formules seront détaillées que dans un seul cas (pondéré). Il suffira de remplacer par les populations correspondantes.

1. Traitement de la jockerisation

Le moyen le plus simple de traiter la jockerisation est simplement de ne rien renseigner quand une note est manquante, l'application bouclera sur les notes renseignées, et fera les calculs de pondérations en conséquence. Le seul défaut est de devoir calculer la somme des poids pour chaque entité, ce qui est peu couteux. Pour le calcul des résultats globaux, on utilise les moyennes comme avant, car changer les formules aurait peu de sens, et apporterait uniquement de la complexité pour rien, et ne causerait que des problèmes.

II. - Les paramètres en entrée

Ce micro service prend en paramètre un json qui contient toutes les informations nécessaires pour calculer tous les résultats détaillés précédamment. Voici ci dessous les paramètres nécessaires :

1. Pour les critères

```
{
    "userWeights": {
        "id_user": value,
    },
    "teams":{
        "id_team":[
            "id_participant",
        ]
    }
    "teamWeight":{
        "id_team": value,
    }
    // Not yet implemented
    "legitimity":{
            "id_graded":{
                "grader":value,
            }
        }

    "criterias": {
        "criteria_id": {
            "lowerbound": value,
            "upperbound": value,
            "user_grades": {
                "grader_id": {
                    "graded_id": value // note du graded, donnee par le grader
                    "graded_id2" : value
                }
            }
            "team_grades"{
                "grader_id": {
                    "team_id": value // note de la team, donnee par le grader (un
                        user)
                    "team_id2" : value
                }
            }
        }
  }
}
```

est ce que c'est assez clair, ou est ce que je dois détailler plus ?

2. Poue les phases et activités

```
{
    "globalData":{
        "weight" :
        "averageUsersWeightedResult" : value,
        "averageUserslEqualResult." : value,
        "averageTeamWeightedResult" : value,
        "averageTeamEqualResult" : value,
        //RelativeResult
        "averageUsersWeightedRelativeResult" : value,
        "averageUserslEqualRelativeResult" : value,
        "averageTeamWeightedRelativeResult" : value,
        "averageTeamEqualRelativeResult" : value,
        //StdDev
        "averageUsersWeightedStdDev" : value,
        "averageUsersEqualStdDev" : value,
        "averageTeamWeightedStdDev" : value,
        "averageTeamEqualStdDev" : value,
        //MaxStdDev
        "maxUsersWeightedStdDev" : value,
        "maxUsersEqualStdDev" : value,
        "maxTeamWeightedStdDev" : value,
        "maxTeamEqualStdDev" : value,
        //Inertie
        "usersWeightedInertia" : value,
        "usersEqualInertia" : value,
        "teamWeightedInertia" : value,
        "teamEqualInertia" : value,
        //MaxInertie
        "maxUsersWeightedInertia" : value,
        "maxUsersEqualInertia" : value,
        "maxTeamWeightedInertia" : value,
        "maxTeamEqualInertia" : value,
        //Ratio
        "usersWeightedRatio" : value,
        "usersEqualRatio" : value,
        "teamWeightedRatio": value,
        "teamEqualRatio" : value,

        "user" :{
            "id_participant":{
                "weightedResult": value,
                "equalResult":value,
                "weightedRelativeResult":value,
                "equalRelativeResult":value,
                "weightedStdDev":value,
                "equalStdDev":value,
                "weightedRatioStdDev": value,
                "equalRatioStdDev": value
            }
        }
        "teams":{
            "id_participant":{
                "weightedResult": value,
                "equalResult": value,
```

```
                "weightedRelativeResult": value,
                "equalRelativeResult": value,
                "weightedStdDev": value,
                "equalStdDev": value,
                "weightedRatioStdDev": value,
                "equalRatioStdDev":value
            }
        }
    }
}
```

est ce que c'est assez clair, ou est ce que je dois détailler plus ?

III. - Les données en sortie

Les données en sortie auront toujours la même structure, que ce soit pour des calculs sur un critère, une phase, ou une activité. Cà permet notamment de faire une seule fonction de traitement dans l'appli principale.

```
{
    "id": {

        "averageUsersWeightedResult" : ,
        "averageUserslEqualResult." : ,
        "averageTeamWeighteResult" : ,
        "averageTeamEqualResult" : ,
        //RelativeResult
        "averageUsersWeightedRelativeResult" :,
        "averageUserslEqualRelativeResult" : ,
        "averageTeamWeighteRelativeResult" :,
        "averageTeamEqualRelativeResult" :,
        //StdDev
        "averageUsersWeightedStdDev" :,
        "averageUsersEqualStdDev" : ,
        "averageTeamWeightedStdDev" : ,
        "averageTeamEqualStdDev" : ,
        //MaxStdDev
        "maxUsersWeightedStdDev" : ,
        "maxUsersEqualStdDev" : ,
        "maxTeamWeightedStdDev" : ,
        "maxTeamEqualStdDev" : ,
        //Inertie
        "usersWeightedInertia" : ,
        "usersEqualInertia" : ,
        "teamWeightedInertia" : ,
        "teamEqualInertia" : ,
        //MaxInertie
        "maxUsersWeightedInertia" : ,
        "maxUsersEqualInertia" : ,
        "maxTeamWeightedInertia" : ,
        "maxTeamEqualInertia" : ,
        //Ratio
        "usersWeightedRatio" : ,
        "usersEqualRatio" : ,
        "teamWeightedRatio": ,
```

```
        "teamEqualRatio" : ,

        "user" :{
            "id_participant":{
                "weightedResult":,
                "equalResult":,
                "weightedRelativeResult":,
                "equalRelativeResult":,
                "weightedStdDev":,
                "equalStdDev":,
                "weightedRatioStdDev":,
                "equalRatioStdDev":
            }
        }

        "team" :{
            "id_participant":{
                "weightedResult":,
                "equalResult":,
                "weightedRelativeResult":,
                "equalRelativeResult":,
                "weightedStdDev":,
                "equalStdDev":,
                "weightedRatioStdDev":,
                "equalRatioStdDev":
            }
        }
    }
}
```

Remarque sur `"id"`. Cette valeur n'a de sens que pour les critères, car ce framework peut en traiter plusieurs en même temps. Dans le cas des stage et des activity, cette valeur vaut `"step"`. (Ne pas la changer sans mettre à jour l'appli principale).

Le choix est fait pour l'instant de ne pas regrouper les team et user, la principale raison qui a motivé ce choix est la gestion des clés. En effet, un user et une team peuvent avoir la même clé, en les séparants ainsi, on peut les renseigner avec leur vraie clé, ce qui simplifiera l'utilisation du micro service par le reste de l'appli Serpico.

IV. - ARCHITECTURE DE L'APPLI

On propose une solution orientée objet, et non pas un calcul matriciel, pour plusieurs raisons.

— Une telle matrice est creuse, et c'est pire avec la jockerisation, ce qui implique de la perte en mémoire, et en performance. Surtout, çà perd en lisibilité pour le code.

— En revanche, ce choix permet de coller à la structure de donnée, et de gagner en lisibilité, et de modulariser les calculs beaucoup plus facilement. Si jamais ils devaient être modifiés dans le futur, il suffira de juste modifier certaines fonctions, et pas tout le calcul matriciel.

Le code se décompsoe en deux parties. La première, pour le calcul par critère, et la deuxième pour le calcul par phase/activité.

1. Le module de calcul par critère

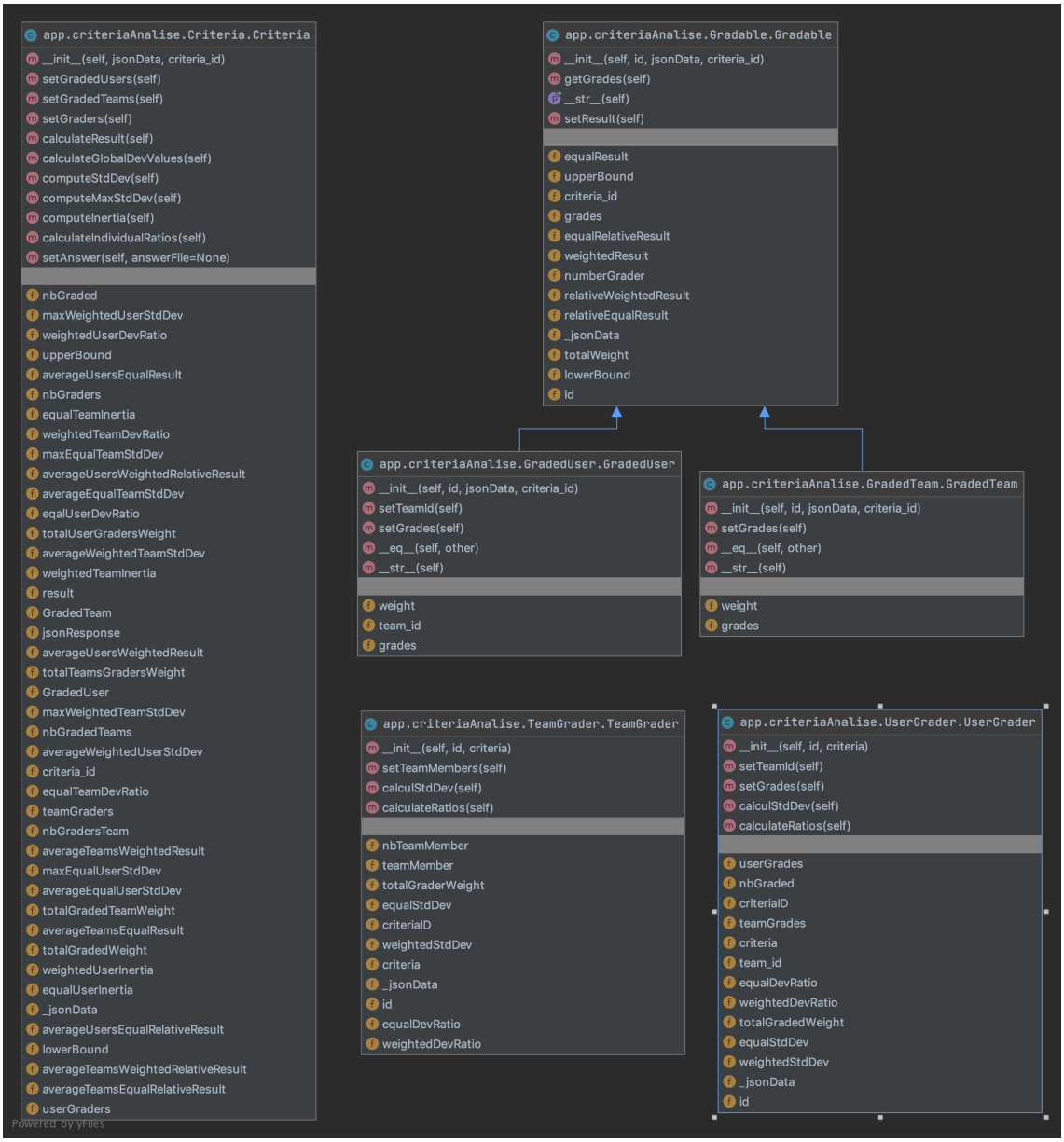

FIGURE 1 – Diagramme de classe de la partie calcul par critère

2. Le module de calcul par phase ou activité

Cette partie a été finalement faite en fonctionnel et non pas en objet, car ce n'était pas nécessaire au vu de la simplicité de cette partie.

V. - Remarques liées à l'utilisation par l'appli Serpico

Il a été décidé de séparer complètement les résultats liés à l'évaluation de phase et de user. Ainsi, ce service est appelé deux fois dans le cas ou une phase évalue et la phase et les user. Dans le cas ou on appelle le framework quand la phase est évaluée, elle est considérée comme un user passif, avec un poids de 1, et un id de -1 (le poids n'importe peu, dans ce cas la phase est le seul utilisateur noté).

VI. - Mise en prod et utilisation

Le code est déployé sur le premier serveur de Serpico, voir le Readme pour plus de détails.