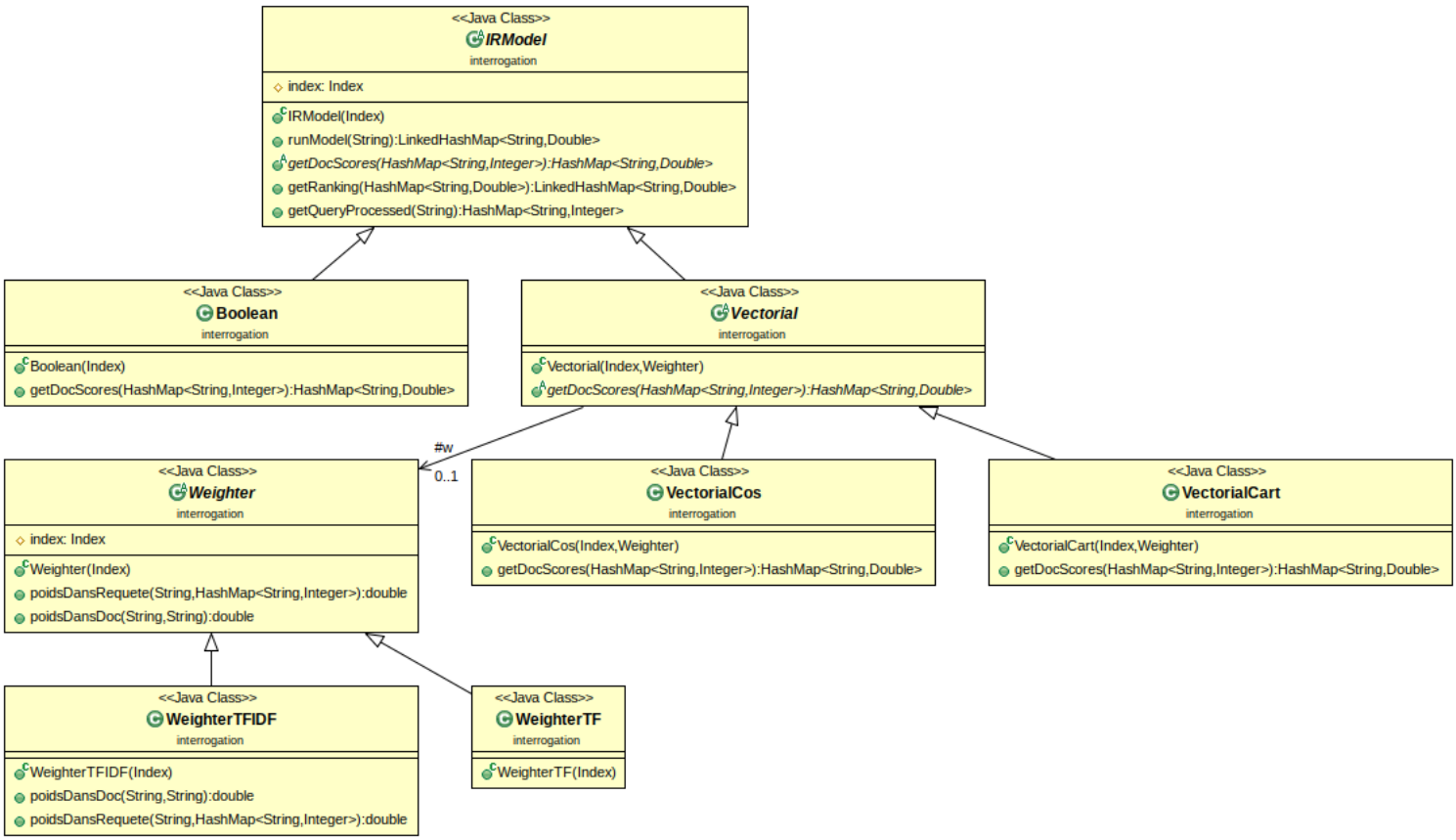
Rapport de projet  LU2IN002

# Moteur de recherche

Gabriel NOGUERO
Lina VARELLA CONTI

# Modélisation de l’architecture de l’étape d’interrogation



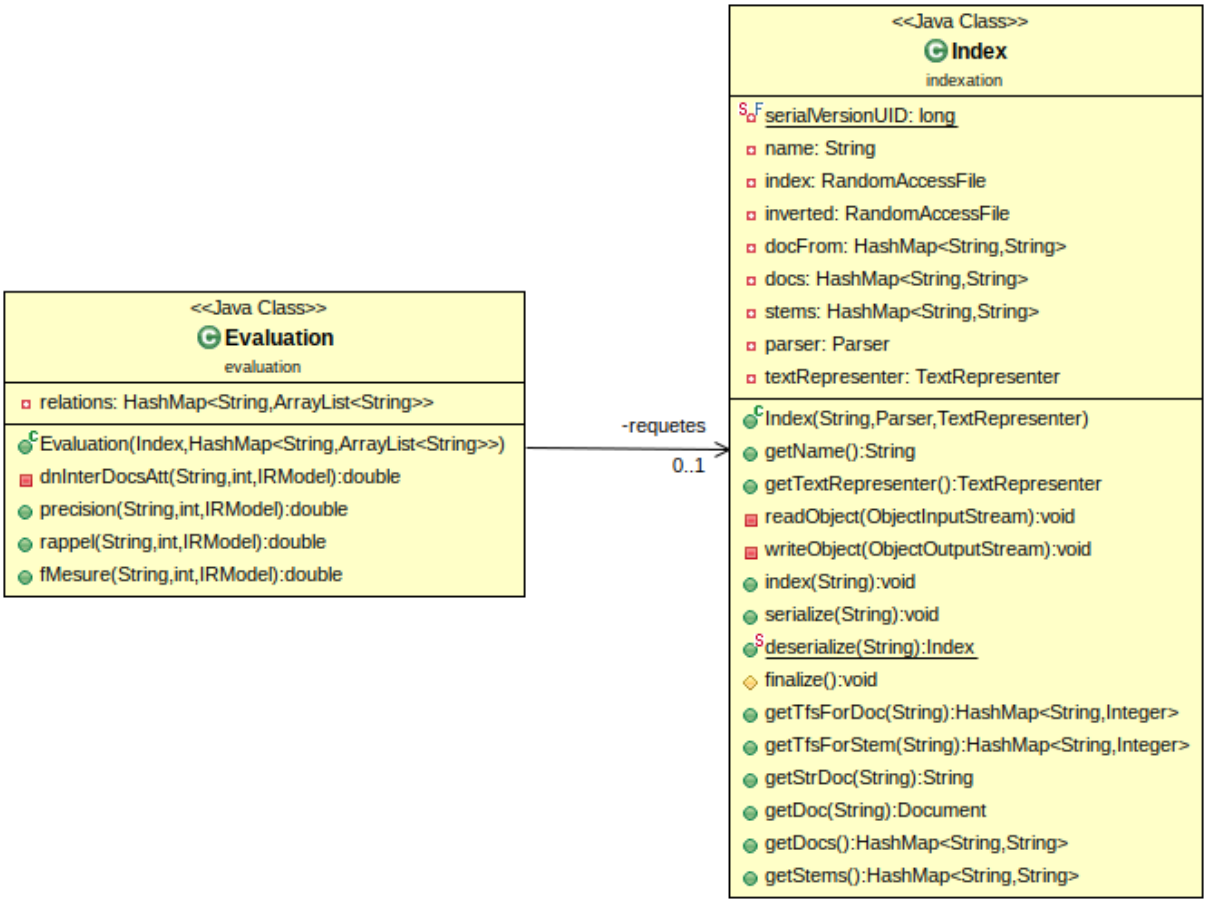

Pour nos choix de modélisation de l’étape d’interrogation, nous nous sommes surtout inspirés du main fourni et des classes qui y étaient présentes.

Le fait de séparer le calcul du poids et le calcul du score pour les modèles vectoriels – en construisant un objet *Vectorial* prenant comme paramètre un *Weighter* – permet de créer dans un main toutes les combinaisons possibles entre le produit cartésien et le cosinus et les fréquences TF et TFIDF sans avoir à coder individuellement une classe pour chaque combinaison.

Nous avons hésité entre coder les fonctions *poidsDansRequete* et *poidsDansDoc* dans la classe abstraite *Weighter* puis les redéfinir ou les coder uniquement dans les classes filles. Finalement, nous avons choisi de les implémenter dans la classe abstraite car cela nous permettait de factoriser du code. Nous avons jugé qu’il n’y avait pas de risque d’ambiguïté dans le choix de la méthode car elles ont toutes la même signature et la JVM choisit la méthode de l’instance (donc pas celle de la classe abstraite).

La même question c’est posée pour les méthodes de la classe *Vectorial.*

# Modélisation de l’architecture de l’étape d’évaluation




Pour l’étape d’évaluation nous avons fait un choix de modélisation très simple et n’avons créé qu’une seule classe. L’étape d’évaluation ne demande que de créer trois fonctions calculant la précision , le rappel et la F-mesure pour une requête et un modèle donné. Nous avons donc créé un objet « évalueur » contenant ces trois fonctions et s’appuyant sur les fichiers fournis contenant les requêtes et les documents attendus pour chacune. Nous avons de plus rajouté une méthode privée *dnInterDocsAttendus* car il est nécessaire de calculer cela dans chacune des autres fonctions. Cette méthode nous a ainsi permis de factoriser du code.

## Difficultés rencontrées et améliorations possibles

Pour ce projet, nous avons travaillé la majeure partie du temps en binôme de manière à rester sur la même ligne directrice.

Nous avons pris beaucoup de temps pour comprendre la manière d'utiliser la librairie fournie avec le sujet. Avant le message explicatif sur Piazza nous avions du mal à déterminer où devait se placer le fichier indexation.jar pour que nous puissions nous en servir. Une fois cette étape passée, la compréhension de la Javadoc fournie n'était pas si compliquée et nous avons pu avancer rapidement aux questions suivantes.
Ensuite, lors du calcul de l'IDF, nous avions des doutes sur la valeur de notre logarithme, car l'ayant fait à la calculatrice nous ne trouvions pas exactement les mêmes valeurs, cependant les résultats semblaient logiques.

La formule du *VectorialCos*, plus précisément son dénominateur, nous a laissé un peu perplexe avant votre réponse sur Piazza. Nous avions du mal à interpréter la somme des « $d_i^2$ » avec i appartenant à d et la norme des vecteurs. Même si nous avons mieux compris maintenant, nous ne sommes toujours pas tout à fait sûrs de notre calcul du score pour *VectorialCos*, notamment car pour tous les tests que nous avons effectué les valeurs de l'évaluation (précision, rappel et f-mesure) valent 0.

L'interface graphique a été compliquée puisque nous étions en totale autonomie sur un sujet jamais abordé. Nous avons cependant réussi à produire quelque chose avec les documents et des explications trouvées sur Internet. C'était une totale découverte pour nous deux.

Le projet au niveau du code des exercices un, deux et trois a été fini dans son intégralité.

Nous avons pensé à des améliorations sur l'interface graphique.
Lorsque nous lançons une requête dans notre interface et que les résultats s'affichent, si l'on veut rentrer une autre requête il faut quitter et rouvrir le moteur de recherche. Nous pourrions trouver un moyen pour que plusieurs requêtes puissent être rentrées sans avoir à quitter le moteur de recherche.

Il pourrait aussi y avoir la possibilité d'un bouton, sous la requête par exemple, qui nous permettrait de choisir le modèle que l'on souhaite utiliser.

Quand la recherche est lancée, nous pourrions aussi avoir, à titre indicatif, une case qui affiche le nombre de résultats trouvés.

Dans notre interface nous n'affichons que les dix premiers résultats pour chaque recherche, il serait donc intéressant d'ajouter un bouton qui nous donne les dix résultats suivants et ainsi de suite jusqu'à qu'il n'y ait plus de résultats.

Dans l'ensemble, ce projet était intéressant et a mis à l'épreuve notre travail d'équipe. Nous avons pu découvrir de nouveaux aspects de la programmation en Java comme la création de l'interface graphique par exemple.