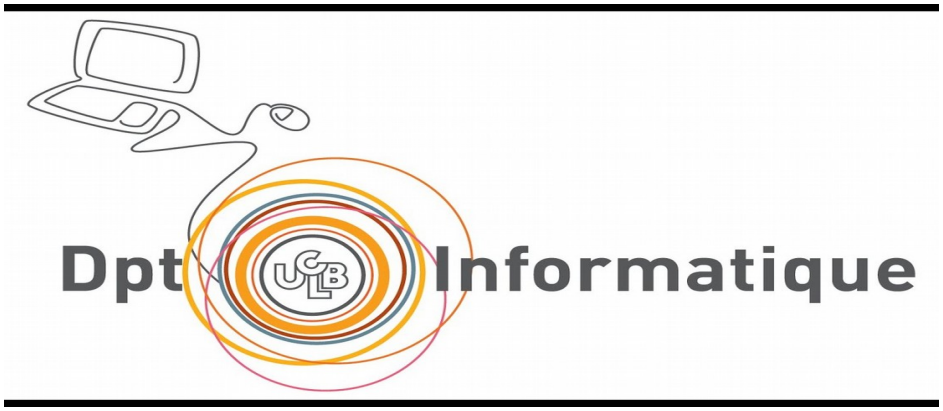


25 av. Pierre de Coubertin, 69100 Villeurbanne

Projet génie logiciel

« Sélection de CV »

Responsable de l'UE : Matthieu Moy

Chupin Pierre-Henri

Scheirlinck Clement

Description du projet

Un recruteur veut pouvoir sélectionner des candidats selon certains critères, comme par exemple leur compétence en C. Il peut alors rechercher chez les candidats, les compétences requises avec un score minimal ou une moyenne. Pour cela, nous avons à notre disposition un projet maven qui est fonctionnel et que nous avons dû faire évoluer avec le plus de fonctionnalités possibles, ainsi q'une meilleure architecture, tout en restant user-friendly.

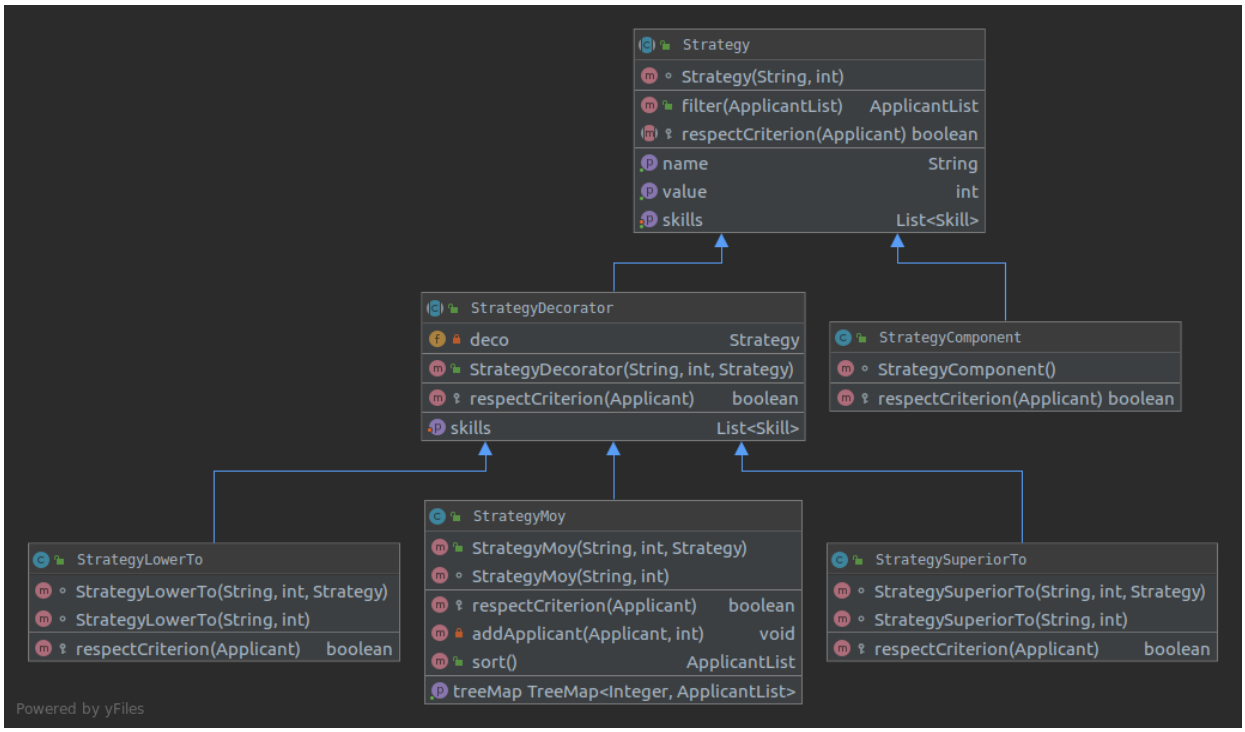
Design patterns

Nous avons appliqué différents designs patterns pendant le projet et nous allons vous en présenter quelques-uns en expliquant nos choix d'architecture. Il y a notamment:

- un décorateur, pour les filtres
- un MVC, pour la base de projet

Le décorateur

Pour les stratégies, nous avons implémenté le design pattern décorateur, permettant d'appliquer de multiples filtres à une recherche. Nous avons pris cette décision afin d'éviter la multiplication de classes. Il nous est alors très facile d'ajouter un nouveau filtre, non nécessairement sur les compétences mais aussi sur les expériences du candidat. On peut alors associer les différentes stratégies que nous avons créé en mettant à chacune des valeurs différentes ce qui offre une indépendance à l'utilisateur.

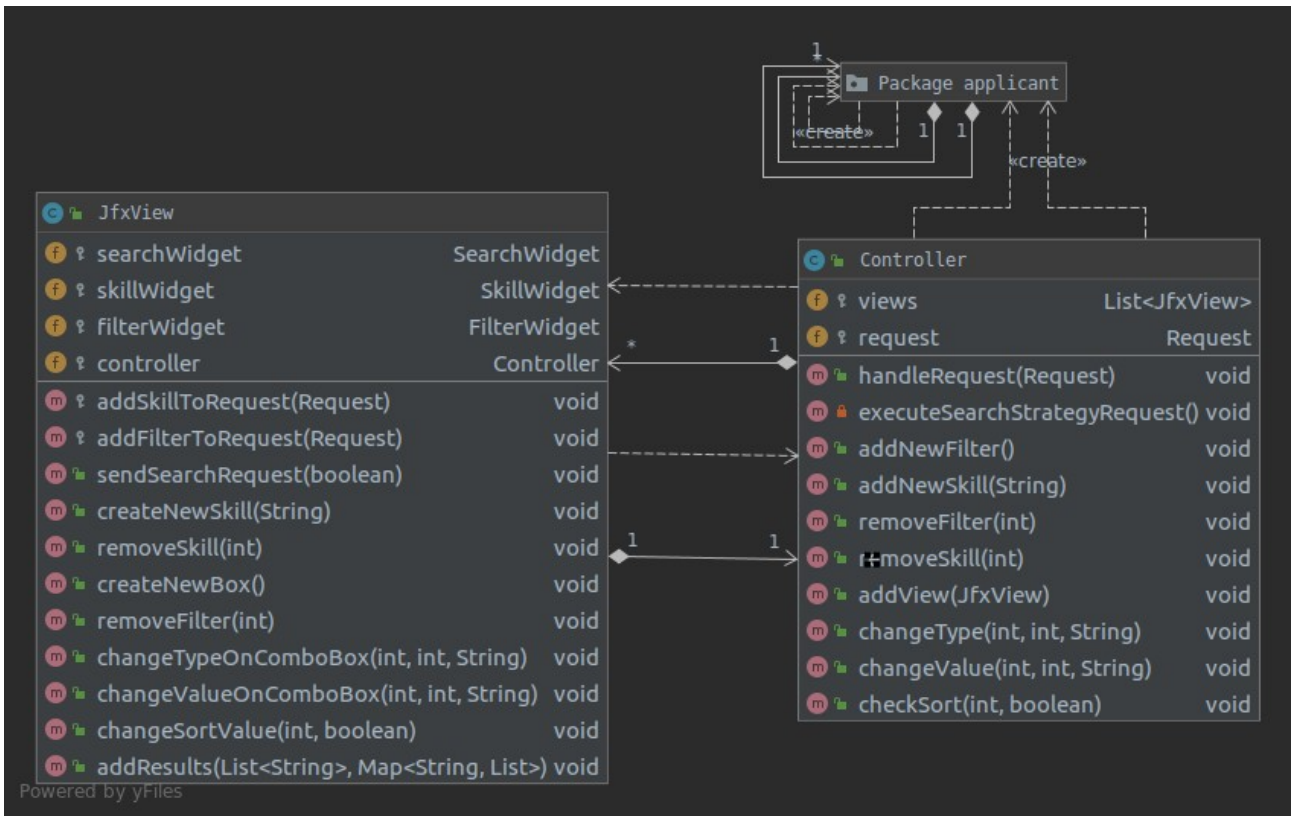


Cette fonctionnalité est applicable dans le monde du travail, car en fonction des métiers, on peut être amené à vouloir une application flexible, ne nous limitant pas à des filtres contraignants avec des valeurs fixes. Nécessitant alors plusieurs développeurs afin de rajouter les filtres d'un client, alors que dans notre cas, un seul peut très facilement en rajouter tout en prenant en compte les dizaines de combinaison.

Le MVC

Afin d'avoir un code propre et évolutif, nous avons programmé un design pattern MVC, car ce dernier permet de séparer les données, la vue gérant l'interface homme machine, et la logique du contrôle. Cela permet notamment :

- D'apporter une lisibilité dans les différents parties d'un projet.
- La réutilisation de fonctionnalités pour en implémenter d'autres
- Une modification de l'une n'impactant pas forcément les autres.
- Une maintenance plus facile car plus compartimenté
- Une évolutivité du projet simple.



C’est actuellement, un des plus utilisé par les professionnels pour développer des sites, applications et bien d’autres. Dans le cadre de notre projet, il était intéressant de l’implémenter dès le début car il nous a très vite permit de gagner du temps pour développer chaque partie. Avec ce patron, nous avons facilement corrigé les erreurs. De plus, deux développeurs peuvent travailler sur la même fonctionnalité mais sur des parties du modèles différentes.
Nous avons décidé de mettre la logique métier dans le contrôleur

L'éthique

Nous avons dû développer trois stratégies imposées qui sont:

- SuperiorTo50 → sélectionne tous les candidats qui ont des scores supérieurs à 50 dans les critères demandés.
- SuperiorTo60 → sélectionne les candidats quand il dépasse le score de 60.
- Moyenne50 → sélectionne tous les candidats qui ont une moyenne supérieur à 50 avec tous les critères sélectionnés.

Nous avons dû en ajouter et nous avons pensé à ajouter comme stratégie:

- InferiorTo"value" → sélectionne tous les candidats qui ont des notes inférieures à "value" car certaines fois, avoir un candidat surqualifié coûte cher alors que l'on n'a pas besoin d'une personne aussi performante. Le problème de cette stratégie, réside dans le fait qu’il est difficile de refuser un excellent candidat. Si un humain

classait les candidats il l'aurait gardé et le salaire qu'on lui aurait proposé, serait sûrement trop bas par rapport à ses prétentions.

Pour éviter de rater les candidats intéressants, nous nous sommes dit qu'il fallait laisser l'utilisateur libre de choisir les valeurs des filtres. Il peut alors rechercher facilement:

- Les candidats qui ont plus de 45 dans les compétences demandées et inférieures à 80 tous en ayant une moyenne au minimum de 55

Nous pensons qu'un humain pourrait faire un meilleur travail car nous ne prenons pas en compte la personne derrière le CV, exemple:

- Nous ne pouvons pas vérifier s'il a menti sur les différentes expériences professionnelles
- Nous ne prenons pas en compte le casier judiciaire du candidat, ce qui pourrait être intéressant pour le recrutement dans les métiers sensibles comme transporteur de fond par exemple. Mais cela pourrait devenir peu éthique, car les utilisateur pourrait alors discriminer sur le casier judiciaire.

Les Tests

Nous avons testé manuellement l'interface graphique, afin de vérifier si la view s'affiche correctement. Il fallait vérifier que :

- La boîte de dialogue et le bouton « add skill » qui permette d’ajouter des compétences fonctionnent bien et que l’on puisse y écrire.
- Les compétences ajoutées ont une croix pour les supprimer et que cette croix supprime la bonne compétence.
- Le bouton d’ajout de filtres fonctionne correctement en faisant apparaître une liste déroulante avec les différents filtres possibles, une boîte de dialogue où l’on puisse rentrer sa valeur (seulement des chiffres) et une croix qui permet de supprimer le bon filtre. Le fait de ne pas remplir un filtre n’empêche pas la recherche.
- Le case à cocher pour trier la liste est cochable, et inversement.
- Le bouton search doit être cliquable et doit bien lancer la recherche tout en triant si la case à cocher « sort » est remplie.
- L’affichage des candidats et de leurs expériences est bien affiché correctement et que ce soit lisible.

Du coté des test unitaires, nous avons testé :

- Les différents filtres, dont le supérieur, l’inférieur, la moyenne, et plusieurs filtres ensemble. De plus, nous avons rajouté un test sur le tri s’effectuant sur la moyenne. Les test passent notamment par une abstraction du décorateur et de la stratégie.
- Les listes de candidat afin de savoir si le modèle les chargent correctement, notamment l’expérience métier. Ainsi qu’un parser des données d’un utilisateur

- La requête de recherche du contrôleur en mockant la vue afin de savoir si la requête répond
- correctement à notre recherche. Le mock permet d’éviter l’utilisation de Node de JavaFX, pour éviter une erreur de non initialisation du Toolkit.