

Rapport Projet IHM

Sujet :

Le but du projet était de créer une interface graphique pour interagir avec des étudiants et les assigner à un ou plusieurs groupes.

Pour cela une implémentation fonctionnelle mais non persistante d'une API a été fourni pour servir de modèle à notre implémentation qui elle devra être persistante via une base de données.

Membre du groupe :

Baptiste Nevejans (Groupe 2)

Joffrey Fouché (Groupe 2)

Victor Descamps (Groupe 3)

Organisation :

lors de la création du groupe les tache se sont d'elle même repartis ainsi:

Victor = réalisation des méthode persistante,

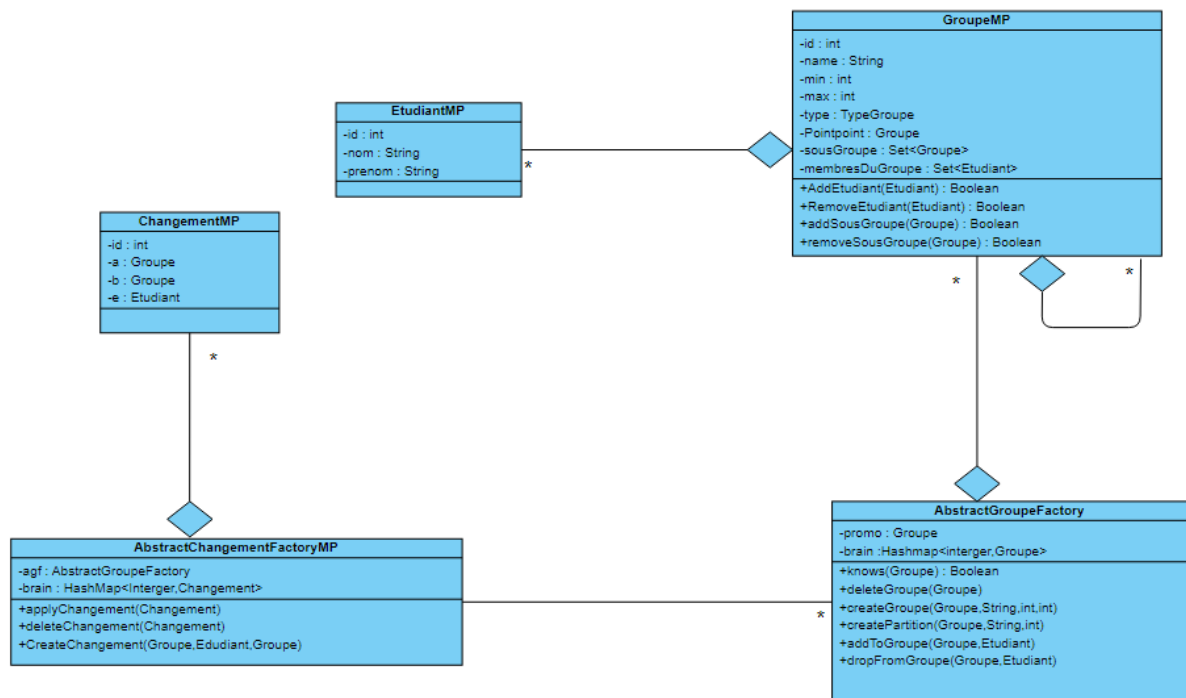
Baptiste, Joffrey = création de l'interface

et les deux créations ont été ensuite indépendantes.

modèle persistant :

Lors de la création, les classes ont été implémentées par ordre de nécessité d'utilisation

(Etudiant->Groupe->abstractGroupeFactory->Changement->AbstractChange
mentFactory)



les classes si dessus implémentent toute leurs équivalent dans l'API mais pour des raison de lisibilité cela n'est pas représenté tout comme la classe enum TypeGroupe

Jusqu'au AbstractChangementFactory je n'ai pas eu de problèmes majeurs. je réalisais mes tests via un programme simulant une utilisation (qui se trouve dans le fichier test) une fois fini j'ai testé l'API avec la classe de test fournie dans l'API est après avoir réglé quelques problèmes j'ai bien réussie à la faire fonctionner avec les méthode persistante.

Je pense sérieusement qu'avec ce que je sais maintenant je pourrais faire une implémentation fonctionnelle plus rapidement que celle-ci mais j'ai perdu trop de temps à comprendre la manière d'utiliser l'API.

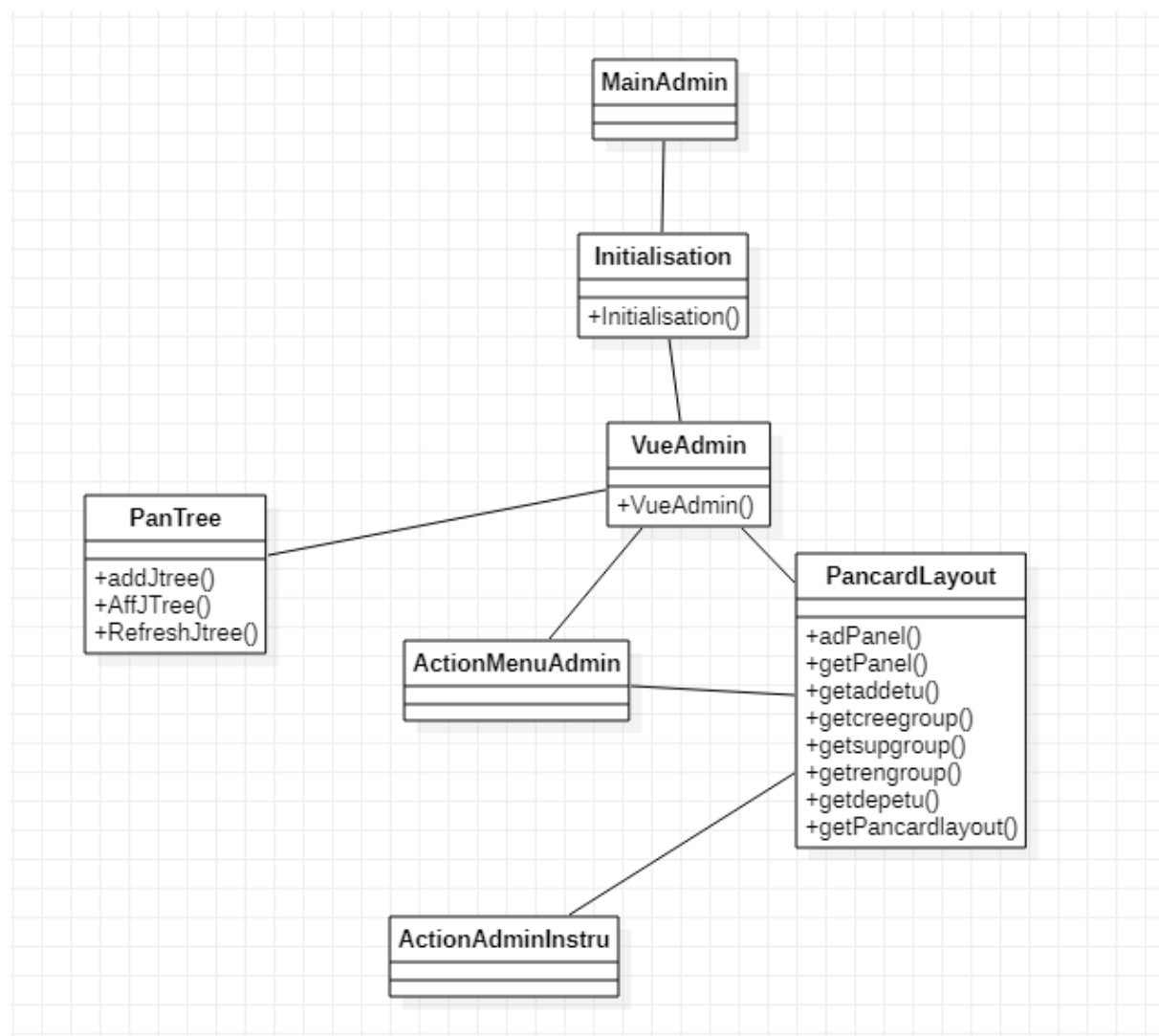
Administrateur

On a fait un modèle commun à chaque interface. On affiche l'utilisateur en haut. Admin possède plusieurs boutons, chaque bouton modifie le panneau en bas à droite, tous ces boutons sont placés à droite pour respecter la structure en F.

Chaque bouton valider lance une fonction qui vient de l'API.

Par exemple creegroupe récupère le nom d'un groupe, le nom du nouveau groupe a créé, si ce groupe doit être un groupe ou une partition, le nombre minimum d'étudiant, le nombre maximum d'étudiant et enfin le nombre de partition si il doit y en avoir. Puis il doit normalement lancé la fonction crategroupe ou createpartition de l'api mais ne le fait pas car il en manque un string. Par contre la fonction ajouter et déplacer étudiant fonctionne.

L'administrateur peut créer des groupes, déplacer des étudiants, changer le nom d'un groupe, ajouter un etudiant à un groupe et supprimer un groupe



Etudiant

Pour la partie étudiant, on observe un menu à quatre boutons. Le premier envoie sur la liste de tous les étudiants, le deuxième sur la liste de tous les groupes et le troisième sur la liste de tous les changements. On retrouve dans ces listes toutes les informations nécessaires pour chaque objet. Pour les étudiants, il y a le nom, le prénom, et l'identifiant. Pour le groupe on retrouve le nom, le type, le père, la taille, le minimum et le maximum et l'identifiant. Pour le changement, on voit le nom de l'étudiant concerné, le prénom de cet étudiant, le nom du groupe de départ et le nom du groupe d'arrivée. Il y a également sur toutes ces listes des liens. Quand on clique sur le lien dans la liste des groupes, cela nous affiche la liste des étudiants de ce groupe. De même lorsque l'on clique sur le lien dans la liste des étudiants on arrive sur la liste des groupes de cet étudiant. Dans la liste des changements on peut cliquer soit sur l'étudiant pour voir ses groupes soit sur les groupes pour voir leurs étudiants.

En plus de cela on peut faire des changements. Quand on clique sur le quatrième bouton on arrive sur une demande de changement, il faut pour cela choisir le nom dans la liste, le groupe de départ, le groupe d'arrivée et donner une explication. Quand on valide une demande de changement et que l'on retourne sur la page changement, le changement est ajouté.

The screenshot shows the 'Etudiant' application with a menu on the left containing four buttons: 'Afficher groupes', 'Afficher étudiant', 'Afficher changement groupe', and 'Faire changement groupe'. The main area displays a table titled 'Affichage des groupes'.

| Id : | Nom : | Type : | Père : | Size : | Min : | Max : | Lien : |
|------|----------------|-----------|----------------|--------|-------|-------|------------|
| 1 | BUT2 FI | ROOT | BUT2 FI | 57 | 15 | 92 | BUT2 FI |
| 2 | BUT2 FL PAR... | PARTITION | BUT2 FI | 57 | 15 | 92 | BUT2 FL... |
| 3 | TD_0 | FREE | BUT2 FL PAR... | 14 | 0 | 15 | TD_0 |
| 4 | TD_1 | FREE | BUT2 FL PAR... | 14 | 0 | 15 | TD_1 |
| 5 | TD_2 | FREE | BUT2 FL PAR... | 15 | 0 | 15 | TD_2 |
| 6 | TD_3 | FREE | BUT2 FL PAR... | 14 | 0 | 15 | TD_3 |

The screenshot shows the 'Etudiant' application with a menu on the left. The main area displays a table titled 'Affichage de tous les étudiants'.

| Id : | Nom : | Prénom : | Lien : |
|------|------------|-----------|------------|
| 1 | cesar | pyrurgus | cesar |
| 2 | denis | iramus | denis |
| 3 | marcel | castor | marcel |
| 4 | marin | eurydice | marin |
| 5 | constantin | akoni | constantin |
| 6 | donat | anakoni | donat |
| 7 | alexandre | epikalla | alexandre |
| 8 | andre | ekewaka | andre |
| 9 | renard | elikapeka | renard |

The screenshot shows the 'Etudiant' application with a menu on the left. The main area displays a table titled 'Voir les demandes de changements de groupes'.

| Id : | NomEtu : | PrénomEtu : | NomGroup... | NomGroup... | Explication : | Lien Etudia... | Lien Group... |
|------|------------|-------------|-------------|-------------|---------------|----------------|---------------|
| 2 | constantin | akoni | TD_0 | TD_1 | consta... | TD_0 | TD_1 |

The screenshot shows the 'Etudiant' application with a menu on the left. The main area displays a form titled 'Faire un changement'.


Form fields:

- Nom Prénom: [dropdown]
- Groupe de départ: [dropdown]
- Explication: [text area]
- Groupe de d'arrivée: [dropdown]
- Valider: [button]

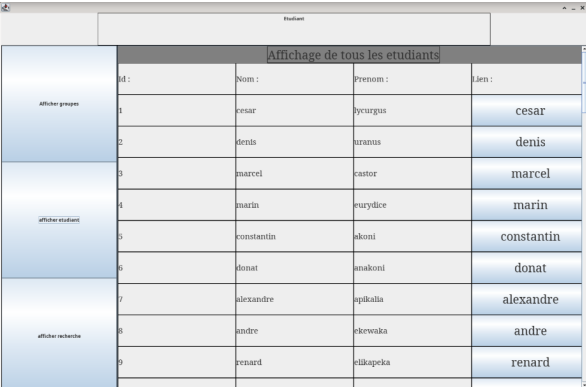
Enseignant

Pour la partie enseignant , on observe un menu à trois boutons. Le premier envoie sur la liste de tous les étudiants, le deuxième sur la liste de tous les groupes. On retrouve dans ces listes toutes les informations nécessaires pour chaque objet. Pour les étudiants, il y a le nom, le prénom, et l'identifiant. Pour le groupe on retrouve le nom, le type, le père, la taille, le minimum et le maximum et l'identifiant. Il y a également sur toutes ces listes des liens. Quand on clique sur le lien dans la liste des groupes, cela nous affiche la liste des étudiants de ce groupe. De même lorsque l'on clique sur le lien dans la liste des étudiants on arrive sur la liste des groupes de cet étudiant.

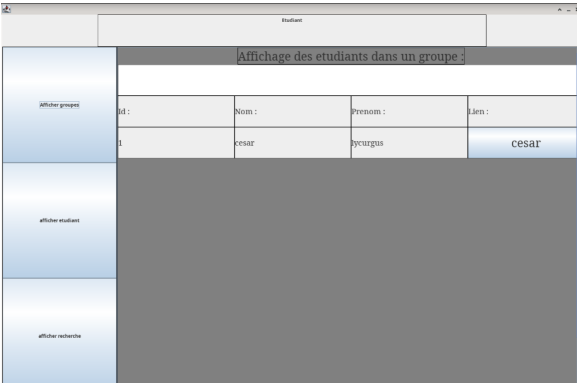
Le troisième bouton amène sur la recherche d'un étudiant en tapant les premières lettres de son nom. Elle est encore en cours de création.



| Id : | Name : | Type : | Père : | Size : | Min : | Max : | Lien : |
|------|-----------------|-----------|-----------------|--------|-------|-------|------------|
| 1 | BUT2 FI | ROOT | BUT2 FI | 57 | 15 | 92 | BUT2 FI |
| 2 | BUT2 FI, PAR... | PARTITION | BUT2 FI | 57 | 15 | 92 | BUT2 FI... |
| 3 | TD_0 | FREE | BUT2 FI, PAR... | 14 | 9 | 15 | TD_0 |
| 4 | TD_1 | FREE | BUT2 FI, PAR... | 14 | 9 | 15 | TD_1 |
| 5 | TD_2 | FREE | BUT2 FI, PAR... | 15 | 9 | 15 | TD_2 |
| 6 | TD_3 | FREE | BUT2 FI, PAR... | 14 | 9 | 15 | TD_3 |



| Id : | Nom : | Prénom : | Lien : |
|------|------------|-----------|------------|
| 1 | cesar | lycurgus | cesar |
| 2 | denis | uramus | denis |
| 3 | marcel | castor | marcel |
| 4 | marin | eurydice | marin |
| 5 | constantin | akoni | constantin |
| 6 | donat | anakoni | donat |
| 7 | alexandre | apikalla | alexandre |
| 8 | andre | okewaka | andre |
| 9 | renard | elikaepka | renard |



| Id : | Nom : | Prénom : | Lien : |
|------|-------|----------|--------|
| 1 | cesar | lycurgus | cesar |

CONCLUSION

Conclusion Personnelle:

Baptiste : Ce projet m'a permis d'apprendre beaucoup de choses notamment l'utilisation d'une API. Mais j'ai trouvé que l'utilisation d'une API pouvait nous restreindre. N'étant pas très doué en design j'ai décidé de me contenter de faire l'aspect pratique de l'interface.

Joffrey : Nous avons appris de nombreuses choses tout au long de ce projet. Comment se servir d'une API, comment faire une interface IHM, comment la lier avec une base de données, et comment travailler en équipe. C'était un projet complet qui a été compliqué.

Victor : Pour ma part j'ai eu beaucoup plus de mal avec ce projet qu'avec les autres. c'est principalement l'API qui m'a posé problème. j'ai eu du mal à comprendre le principe et même aujourd'hui ça reste un peu obscur pour moi. Le problème étant que je me cantonne presque toujours aux parties algorithmiques des projets et pour celui-ci j'ai été particulièrement perdu sachant qu'il n'y en avait presque pas à faire.

Remarque : Pour exécuter MainAdmin il faut faire:

```
-javac -cp build -d build -sourcepath "src"
src/fr/iutftbleau/projetIHM2022F12/Admin/Contrôleur/*.java -encoding utf8
-javac -cp build -d build -sourcepath "src"
src/fr/iutftbleau/projetIHM2022F12/Admin/Vue/*.java -encoding utf8
-javac -cp build -d build -sourcepath "src"
src/fr/iutftbleau/projetIHM2022F12/Admin/*.java -encoding utf8
-java -cp build fr/iutftbleau/projetIHM2022F12/Admin/MainAdmin
```

Pour exécuter MainEtu il faut faire:

```
javac -cp build -d build -sourcepath "./src"
src/fr/iutftbleau/projetIHM2022F12/Etu/Vue/*.java -encoding utf8
java -cp build fr/iutftbleau/projetIHM2022F12/Etu/Vue/MainEtu
```

Pour exécuter MainProf il faut faire:

```
javac -cp build -d build -sourcepath "./src"  
src/fr/iutfbleau/projetIHM2022FI2/Prof/Vue/*.java -encoding utf8  
java -cp build fr/iutfbleau/projetIHM2022FI2/Prof/Vue/MainProf
```