
Seekfox

Moteur de Recherche & Interface Utilisateur Avancées



Auteurs :

TRUILLET Clément
COMBELLES Etienne
DELAUZUN Remi
EL ABDAOUI Oualid
GAMBA Gaël
BIZET Raphaël

Clients :

FERRANÉ Isabelle
PINQUIER Julien
VANDERSTRAETEN Julien
TRUILLET Philippe
ESTADIEU Geneviève
De BONNEVAL Agnan
CHAUDET Christelle

Note de Confidentialité

La consultation de ce document est limitée aux enseignants et élèves de l'UPSSITECH.

Dès lors, toute consultation par un tiers doit se faire sous accord préalable des auteurs du document susnommés par la suite, soit par leurs noms, soit sous le nom de l'entreprise SeekFox.

Remerciements

Nous tenons à remercier Mme Ferrané qui nous a beaucoup aidé lors de ce projet, ainsi que M. Vanderstraeten et M. Pinquier qui nous ont eux aussi orientés au niveau de la gestion du projet et de son organisation. Enfin et pas des moindres remerciements particuliers à Mme Estadieu pour ses conseils dans le domaine de la communication.

Table des Matières

Note de Confidentialité	2
Remerciements	2
Introduction	4
Présentation de l'équipe	4
Objet du document	4
Résumé du cahier des charges	5
Etat au lancement	7
Organisation	8
Planning	8
Répartition des tâches	9
Au sein de l'entreprise	9
Au sein de ce projet	9
Moteur de Recherche Avancé	10
Recherche Complexe	10
Interface utilisateur	12
Liaison C-Java	14
Choix Techniques	15
Outils utilisés	15
Bugs connus	16
Conclusions	17
Conclusions personnelles	17
<i>Clément</i>	17
<i>Étienne</i>	17
<i>Gaël</i>	18
<i>Oualid</i>	18
<i>Raphaël</i>	18
<i>Rémi</i>	19
Conclusion de l'équipe	19

Introduction

Présentation de l'équipe

L'équipe Seekfox est composée de 6 membres. 5 membres originaux, présents lors de la création du projet, ainsi qu'un 6ème, que nous avons pu recruter à la suite de notre premier rendu, qui nous a permis de débloquer des fonds. Nous fonctionnons sur le principe de l'autonomie, par la création d'équipes pour chaque tâches, en faisant des compte rendus réguliers de l'avancement de chacune de ces équipes. Au fur et à mesure de l'évolution du projet, les rôles sont arrangés régulièrement pour attribuer à chacun ce qui lui correspond le mieux.

Objet du document

Ce document est le rapport de la deuxième partie du projet Fil Rouge effectuée par notre groupe dans le cadre de la formation de première année de la filière SRI de l'école d'ingénieurs UPSSITECH. Il fait suite au dossier de spécifications rendu en mars 2020, et vise à suivre le développement du projet depuis cette date.

Vous y trouverez un résumé des choix que nous avons fait pour le mener à terme, entre la répartition des tâches initiale et le produit fini présenté en même temps que ce document, en passant par quelques explications du code source et les inévitables difficultés rencontrées.

Résumé du cahier des charges

L'objectif technique consiste à rendre le moteur de recherche facile d'utilisation, et à le doter de fonctionnalités avancées. On pourra utiliser l'approche MVC (*figure 1*) ou d'autres *design patterns* pour doter notre application d'une interface graphique permettant l'interaction avec différents moteurs

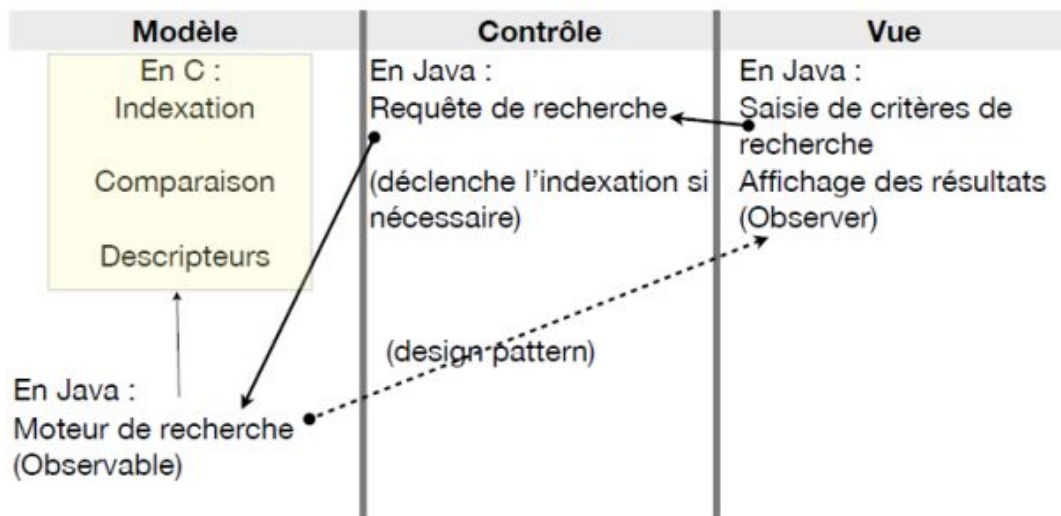


Figure 1 : Modèle MVC (image issue du cahier des charges)

On reprendra la première partie du projet, pour y implémenter cette interface graphique plus complète et des fonctions de recherches plus poussées. Cette partie du projet étant codée en UNIX et C, il faudra alors la lier avec la partie Java, et cela en utilisant une des différentes techniques prévues à cet effet (JNA, JNI, Sérialisation).

Il faudra pouvoir lancer une recherche avec comme choix de critères :

- Un mot clé;
- Une plage de couleurs dominantes;
- Un enregistrement audio;
- Un fichier.

Le résultat à obtenir sera la liste des fichiers contenant plus qu'un certain seuil d'itérations du critère, ou des fichiers les plus ressemblants dans le cas d'une recherche par fichier.

La recherche d'un de ces critères est qualifiée de recherche simple. Il existe également une recherche complexe pour la recherche par mot clef, constituée de plusieurs mots clefs accompagnés d'une polarité (+) pour la présence du critère dans le résultat, et (-) pour son absence.

L'interface doit intégrer une représentation visuelle du traitement de la recherche. Une requête lancée sur un moteur peut être lancée sur un autre, en parallèle ou séparément. Les résultats obtenus pourront être comparés ou fusionnés. Il faudra ajouter deux modes de fonctionnement : mode ouvert ou mode fermé. Le choix de ces modes déterminera si les nouveaux fichiers ajoutés lors de l'utilisation du moteur de recherche seront directement indexés et pris en compte ou non. L'indexation de documents doit par ailleurs être transparente pour l'utilisateur.

Deux types d'utilisateurs seront précisés, *l'utilisateur standard* et *l'administrateur*. Seul l'administrateur aura accès aux réglages des seuils et aux paramètres d'indexation, et il faudra représenter cette séparation de possibilités sur l'interface.

Lors de l'affichage des résultats, on les classera par ordre de pertinence, et il devra être possible de les afficher sous différents modes (liste de liens cliquables, icônes ...), ainsi que de définir combien de fichiers seront affichés à la fois. Lors du clic sur un document, il faudra permettre la consultation du document, selon son type. Toute erreur doit être notifiée à l'utilisateur de manière appropriée. La sauvegarde des résultats d'une requête sera possible.

Etat au lancement

Toute la partie recherche simple est déjà implémentée dans la première version du moteur, excepté la recherche par couleur dominante, qui sera simulée. Les différents types d'utilisateurs étaient aussi déjà inclus.

La majeure partie des nouvelles implémentations se répartira entre la liaison entre les deux parties de code, la recherche complexe et l'interface graphique (la figure 2 donne un aperçu de la première interface textuelle).

```
root@Periclès:~/1A SRI/Projet Fil Rouge/PRF1/seekfox_part1# make
Loading ...
dos2unix: converting file bin/readRequete.sh to Unix format ...

-----RUN-----

Seekfox

1\- Recherche
2\- Administration
3\- Informations
4\- Quitter Seekfox
1

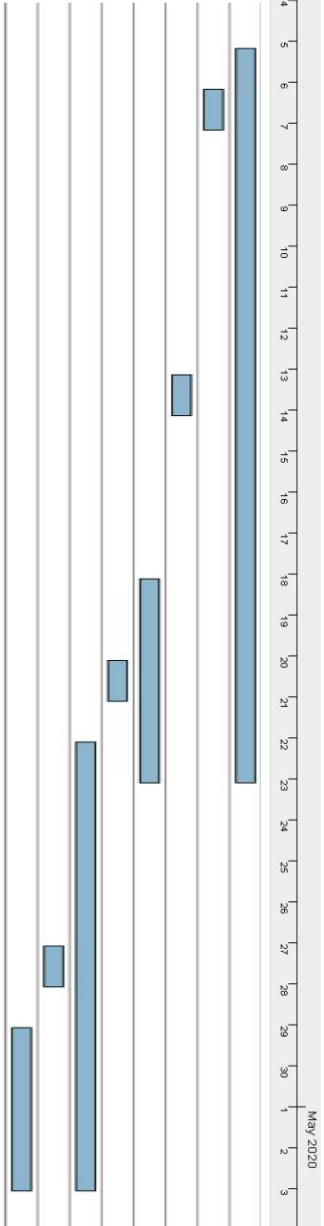
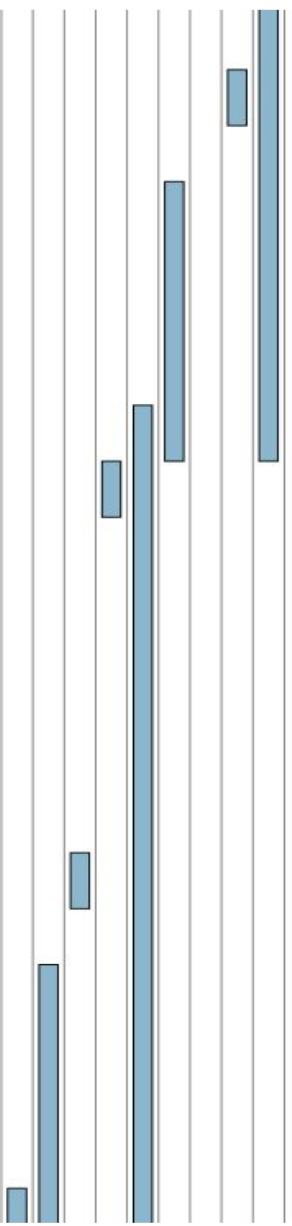
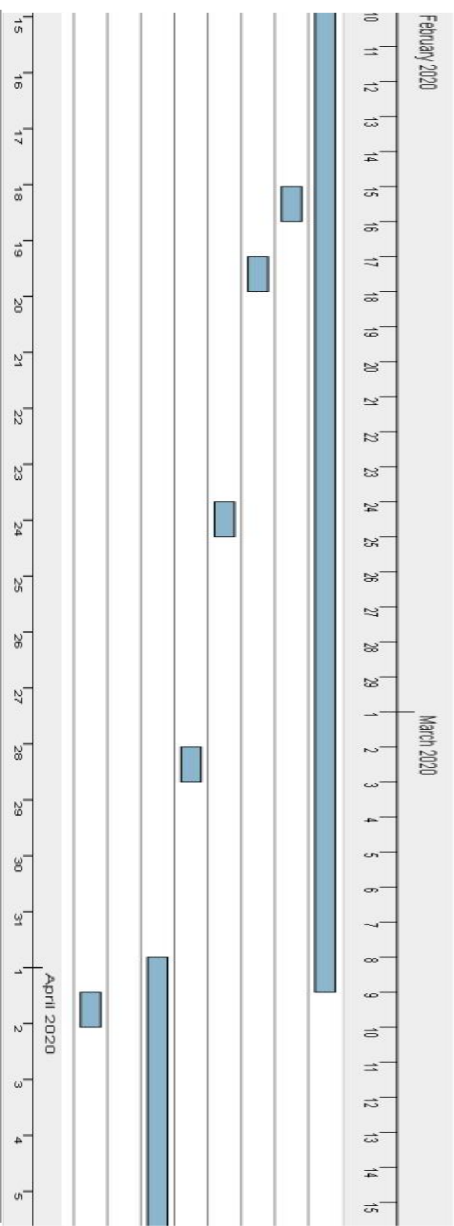
=== RECHERCHE ===
-1/- Retour
0/- Recherche par mot-clef
1/- corpus-fi.bin
2/- jingle-fi.bin
3/- 42.bmp
4/- 51.bmp
5/- 01.jpg
6/- 01-LoremIpsum.xml
```

Figure 2 : Aperçu de l'interface du moteur de la partie 1

Name	Begin date	End date
• Dossier de spécifications	10/02/20	08/03/20
• Réunion lecture du cahier des charges	15/02/20	15/02/20
• Réunion spécifications	17/02/20	17/02/20
• Réunion hebdomadaire avancement	24/02/20	24/02/20
• Réunion hebdomadaire distribution des tâches	02/03/20	02/03/20
• Dépôt v1 moteur avancé	08/03/20	22/03/20
• Réunion hebdomadaire	16/03/20	16/03/20
• Réunion hebdomadaire	09/03/20	09/03/20

• Dépôt v1 moteur avancé	08/03/20	22/03/20
• Réunion hebdomadaire	16/03/20	16/03/20
• Réunion hebdomadaire	09/03/20	09/03/20
• Mise en Commun	18/03/20	22/03/20
• Dépôt version Béta	22/03/20	05/04/20
• Réunion hebdomadaire	23/03/20	23/03/20
• Réunion hebdomadaire	30/03/20	30/03/20
• Mise en Commun	01/04/20	05/04/20
• Dépôt présentation	05/04/20	22/04/20

• Dépôt présentation	05/04/20	22/04/20
• Réunion hebdomadaire	06/04/20	06/04/20
• Réunion hebdomadaire	13/04/20	13/04/20
• Mise en Commun	18/04/20	22/04/20
• Réunion hebdomadaire	20/04/20	20/04/20
• Dépôt version finale	22/04/20	02/05/20
• Réunion hebdomadaire	27/04/20	27/04/20
• Mise en Commun	29/04/20	02/05/20

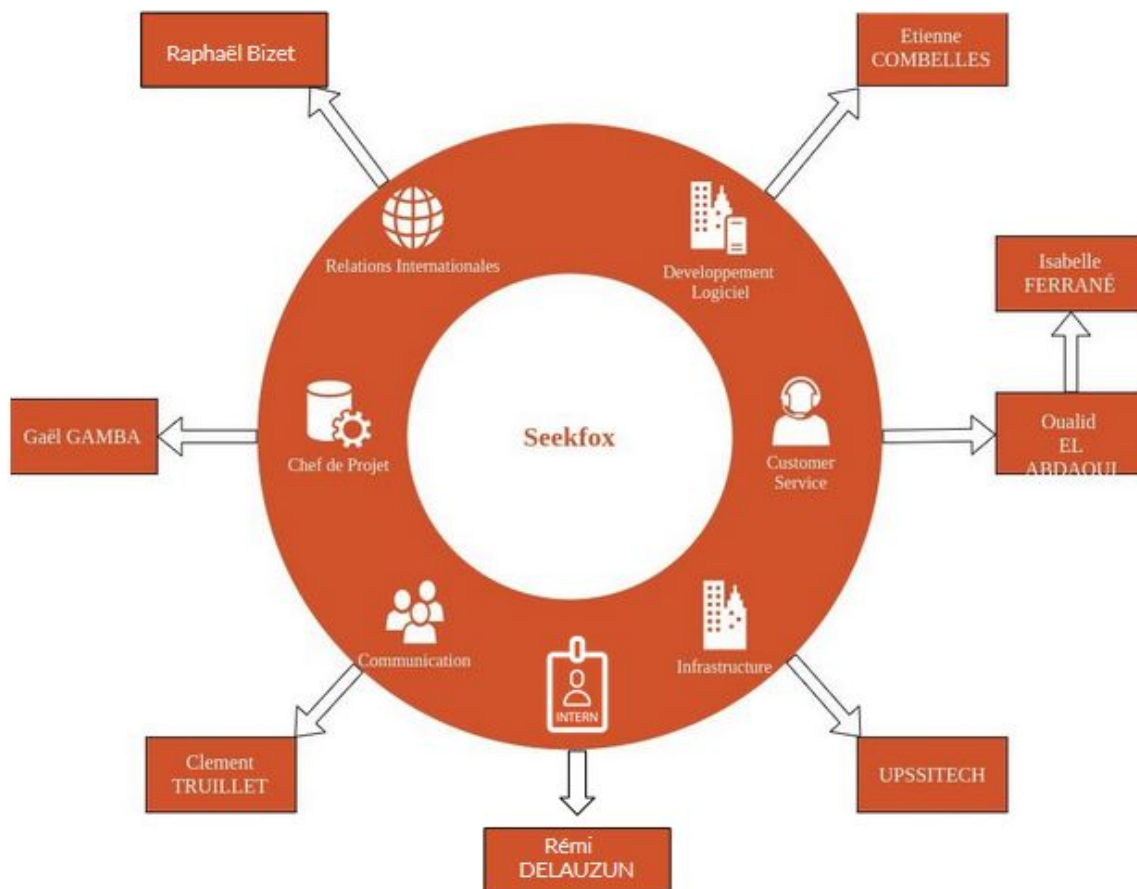


Organisation

Planning

Répartition des tâches

Au sein de l'entreprise



Au sein de ce projet

Tâche	Membres
Interface (2 personnes)	Gaël, Oualid
Liaison Java/C (2 personnes)	Clément, Etienne
Développement des recherches complexes & multi-moteur (2 personnes)	Raphaël, Rémi

Moteur de Recherche Avancé

Recherche Complexe

Rappel du principe d'une recherche / requête complexe :

Une requête complexe est constituée de plusieurs critères de recherche. Les critères de recherche peuvent avoir une polarité (+) ou (-). Dans un premier cas, le critère doit être présent dans le résultat et dans le second cas il doit en être absent. La polarité par défaut est (+). Une recherche complexe est simplement formée de plusieurs requêtes simples.

L'utilisateur va saisir une chaîne de caractère contenant les mots-clés au-devant desquels il indiquera leur polarité : (+) ou (-).

Exemple de recherche utilisateur : '+dimanche -trois'

Signification de cette requête : on désire avoir tous les livres contenant le mot-clé *dimanche* tout en enlevant ceux qui présentent le mot-clé *trois*.

Fonctionnement en trois étapes de l'algorithme de recherche complexe :

1. Segmentation de la chaîne

On fragmente la chaîne saisie par l'utilisateur pour récolter chaque mot-clé et sa polarité séparément.

2. Recherches simples

Pour chaque mot-clé recueilli précédemment, on lance une recherche simple sans se soucier du critère de polarité. Cette recherche est sauvegardée pour une prochaine utilisation.

3. Comparaison et Résultat

On vient ensuite comparer entre eux chaque résultat des recherches simples. Si un titre est en commun et si la recherche simple était à polarité (+) alors le titre du livre est préservé dans la banque finale des résultats. Sinon s'il est présent dans la banque finale et que la polarité est (-) alors il en est supprimé.

Exemple de recherche utilisateur :

Reprenons l'exemple ci-dessus : '+dimanche -trois'

En segmentant on obtient les mots-clés *dimanche* et *trois* ainsi que leur polarité (+) et (-).

Après une recherche simple on voit que le mot-clé *dimanche* est présent dans deux livres :

- *Fernando Alonso*
- *Battue par la Slovenie*

Après une autre recherche simple on voit que le mot-clé *trois* est présent dans six livres :

- *Les accidents vasculaires cerebraux*
- *Les mauvais comptes d'une élimination*
- *Carlos do Carmo*
- *La réalité virtuelle et l'informatique*
- *Agassi plie face au meilleur*
- *Battue par la Slovenie*

Ces deux mots-clés se trouvent tous deux dans le livre *Battue par la Slovenie*.

Comme la polarité de *dimanche* est (+), on ajoute tous ses livres issus de la recherche simple à la banque finale des résultats. Puis on vient comparer les résultats provenant de la recherche par le mot-clé *trois*. Pour toute occurrence d'un livre de cette recherche dans la recherche par *dimanche* il en est supprimé de la banque finale des résultats du fait de la polarité (-).

La banque finale des résultats contient donc :

- *Fernando Alonso*

Remarque : Il est nécessaire de saisir une polarité (+) ou pas de polarité comme premier critère de recherche, l'absence de polarité étant considérée comme une polarité positive.

Interface utilisateur

Nous avons développé l'interface avec Processing afin de pouvoir avoir un meilleur contrôle sur ce que nous allons créer et également pour avoir un code plus simple. Bien qu'il ait fallu développer toutes les fonctions de base (boutons, sliders etc.), ce qui nous a donné du travail supplémentaire, nous avons réussi à créer une interface graphique simple d'utilisation et intuitive.

Limiter les erreurs de l'utilisateur

La conception de cette interface a été réalisée dans l'optique de limiter au maximum les sources d'erreurs de l'utilisateur, ce qui a été accompli de différentes façons, par exemple en utilisant des sliders plutôt que des zones de textes qui auraient permis à l'utilisateur d'écrire ce qu'il veut, et donc possiblement des choses erronées.

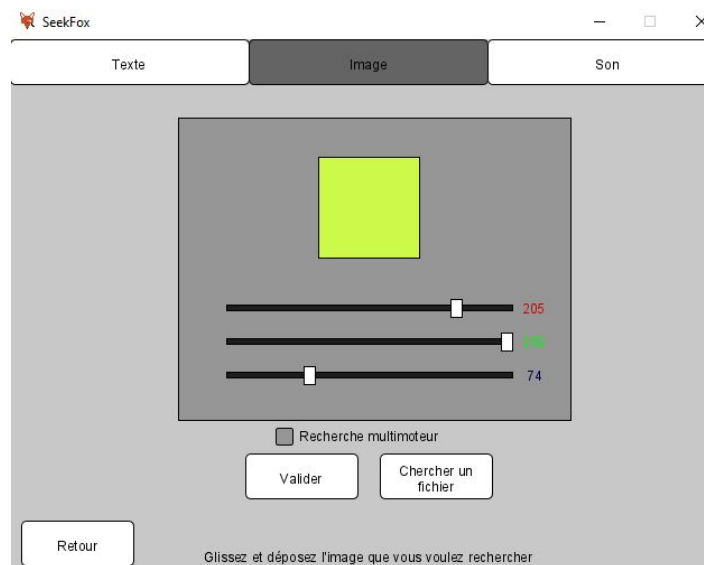


Figure 3 : Écran de configuration de la recherche par couleur dominante.

Comme vu ci-dessus (figure 3), l'utilisateur n'a pas directement accès aux valeurs de la couleur qu'il veut chercher mais il a 3 sliders avec la valeur écrite à côté et un carré de représentation de la couleur créée. Il n'est donc plus possible de créer une couleur qui n'existe pas car l'entrée est mieux contrôlée.

De plus, il est possible d'utiliser des fichiers pour la recherche, ce qui pose problème car il est très simple de faire des erreurs lors de l'écriture du chemin du fichier. Il était donc impossible de demander à l'utilisateur de le faire, nous avons donc ajouté une fonctionnalité de glisser-déposer le fichier directement sur l'interface ainsi qu'une interface de recherche de fichier (voir figure 4) qui traite directement les fichiers utilisables.

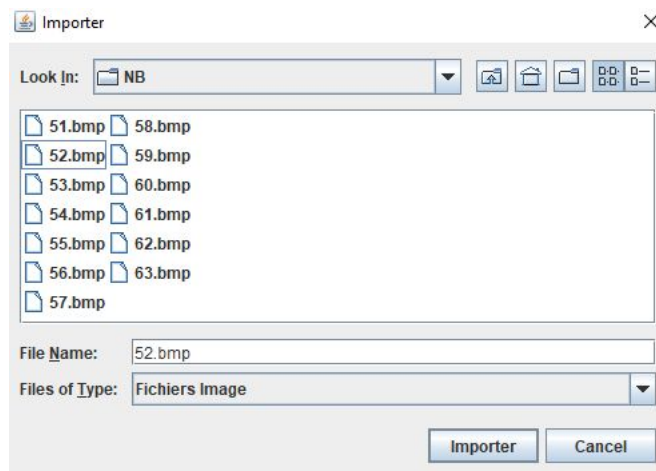


Figure 4 : Interface de sélection de fichier

Lier l'interface et le reste du projet

Avoir une bonne interface ne sert à rien si elle ne permet pas de contrôler le projet, nous avons donc dû concevoir non seulement l'interface en elle même, mais aussi un système capable de lancer les différentes fonctions du programme simplement.

Structure de l'interface

Pour la structure de l'interface, on a choisi une machine à état finie pour gérer les écrans (figure 5).

```
public enum ScreenName {
    MAIN,
    CONFIG,
    HISTORY,
    SEARCH_CONFIG_TXT,
    SEARCH_CONFIG_IMG,
    SEARCH_CONFIG_SND,
    LOADING,
    ADMIN_CONNECTION,
    RESULTS
}
```

Figure 5 : Liste des écrans

Chaque écran est géré par une ou plusieurs classes qui interagissent entre elles d'une manière fluide tout en limitant le nombre d'erreurs.

Le fait de suivre la structure MVC donnée dans le cahier de charges nous a amené à créer un grand nombre de classes, ce qui implique une certaine simplicité dans la structure globale du projet et une réutilisation d'une classe plusieurs fois.

Liaison C-Java

En tout premier lieu, nous étions partis sur la piste de la Sérialisation, méthode qui nous semblait plutôt simple et adaptée au projet. Mais très vite, nous nous sommes bien rendu compte que cette méthode impliquait de lancer le moteur de recherche à chaque requête et ce point ci nous posait problème (la compilation du moteur C se faisait sous Windows et ne compilait pas).

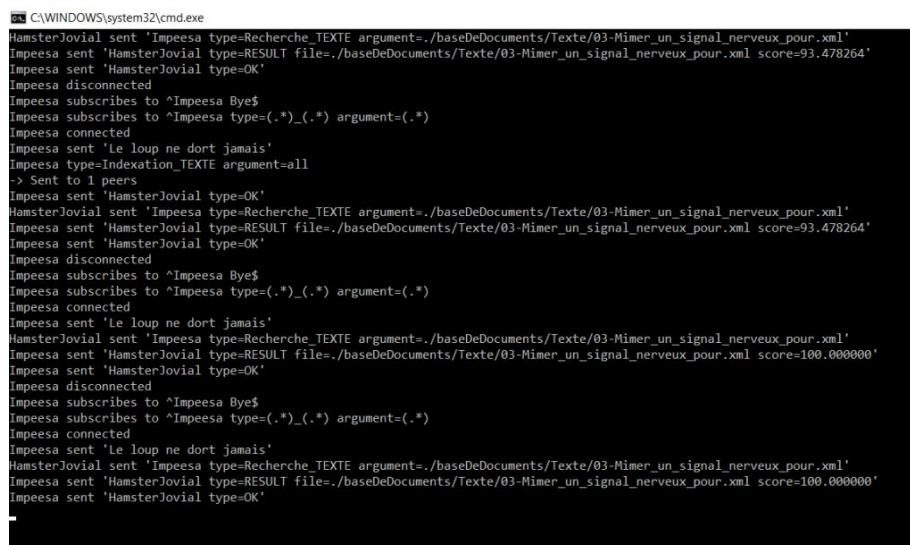
En cherchant une solution complémentaire, nous avons trouvé Ivy, qui, après très peu de modification du code permettait d'avoir deux applications (Java et C) qui parlait entre elles sur un port de communication que nous pouvions observer facilement. En bref, la solution parfaite pour utiliser nos cours de communication des systèmes et pour déboguer facilement cette partie qu'est la liaison Java/C.

Les deux applications étant en écoute permanente, il nous suffisait juste de lancer l'application C dans un environnement Unix avant l'exécution d'une recherche. Fermer les applications était beaucoup plus simple car il nous suffisait juste d'envoyer une requête à l'application C pour lui demander de se fermer.

Il a fallut ainsi créer nos propres protocoles réseaux, opération très instructive car c'était alors l'application directe de notre cours.

Enfin, pour bien différencier nos applications nous leurs avons donné des noms, ainsi le moteur C s'appelle *Impeesa* discutant avec *HamsterJovial* pour le côté Java. Lors de l'ajout d'un deuxième moteur, nous avons choisit de nommer *Akela* pour le C qui discute avec *Baloo* pour le Java.

Cette différenciation permet nous seulement une meilleur lisibilité des requêtes mais aussi de savoir qui parle à qui à tout moment (voir *figure 6*).



```
C:\WINDOWS\system32\cmd.exe
HamsterJovial sent 'Impeesa type-Recherche_TEXTE argument=./baseDeDocuments/Texte/03-Mimer_un_signal_nerveux_pour.xml'
Impeesa sent 'HamsterJovial type-RESULT file=./baseDeDocuments/Texte/03-Mimer_un_signal_nerveux_pour.xml score=93.478264'
Impeesa sent 'HamsterJovial type-OK'
Impeesa disconnected
Impeesa subscribes to ^Impeesa Bye$
Impeesa subscribes to ^Impeesa type=(.*)_(.*) argument=(.*)
Impeesa connected
Impeesa sent 'Le loup ne dort jamais'
Impeesa type-Indexation_TEXTE argument=all
-> Sent to 1 peers
Impeesa sent 'HamsterJovial type-OK'
HamsterJovial sent 'Impeesa type-Recherche_TEXTE argument=./baseDeDocuments/Texte/03-Mimer_un_signal_nerveux_pour.xml'
Impeesa sent 'HamsterJovial type-RESULT file=./baseDeDocuments/Texte/03-Mimer_un_signal_nerveux_pour.xml score=93.478264'
Impeesa sent 'HamsterJovial type-OK'
Impeesa disconnected
Impeesa subscribes to ^Impeesa Bye$
Impeesa subscribes to ^Impeesa type=(.*)_(.*) argument=(.*)
Impeesa connected
Impeesa sent 'Le loup ne dort jamais'
HamsterJovial sent 'Impeesa type-Recherche_TEXTE argument=./baseDeDocuments/Texte/03-Mimer_un_signal_nerveux_pour.xml'
Impeesa sent 'HamsterJovial type-RESULT file=./baseDeDocuments/Texte/03-Mimer_un_signal_nerveux_pour.xml score=100.000000'
Impeesa sent 'HamsterJovial type-OK'
Impeesa disconnected
Impeesa subscribes to ^Impeesa Bye$
Impeesa subscribes to ^Impeesa type=(.*)_(.*) argument=(.*)
Impeesa connected
Impeesa sent 'Le loup ne dort jamais'
HamsterJovial sent 'Impeesa type-Recherche_TEXTE argument=./baseDeDocuments/Texte/03-Mimer_un_signal_nerveux_pour.xml'
Impeesa sent 'HamsterJovial type-RESULT file=./baseDeDocuments/Texte/03-Mimer_un_signal_nerveux_pour.xml score=100.000000'
Impeesa sent 'HamsterJovial type-OK'
```

Figure 6 : Port de communication Ivy en cours d'exécution

Choix Techniques

Outils utilisés

Pourquoi Processing ?

Nous avons choisi d'utiliser Processing 3 pour sa flexibilité et sa facilité d'utilisation. Nous avons également choisi de ne pas utiliser de bibliothèques pour les boutons et les sliders afin d'avoir un meilleur contrôle et d'avoir des composants parfaitement adaptés à notre utilisation. De plus, la communauté autour de Processing est impressionnante et pour chaque problème rencontré, il est très probable que quelqu'un d'autre ait eu le même, ce qui a permis de faciliter la résolution de problèmes.

Le choix était surtout entre Processing et Java Swing, mais nous nous sentions plus à l'aise avec Processing étant donné que nous avons plus travaillé sur Processing que sur Swing auparavant et les avantages de Swing (structure interne, composants déjà créés etc.) étaient finalement mineurs car le projet était d'assez petite envergure.

Choisir un fichier

Afin de choisir un fichier nous avons opté pour deux solutions : la bibliothèque *sDrop* et un élément de Java Swing *JFileChooser*. L'objectif étant de simplifier le choix du fichier en limitant les erreurs, l'option du glisser-déposer nous paraissait essentielle, et heureusement, une bibliothèque Processing existait déjà et était très facile d'utilisation. Cependant, en faisant des recherches nous avons trouvé le *JFileChooser* et il nous a semblé bon d'avoir un autre moyen de choisir un fichier si il y avait quelque problème avec le glisser-déposer.

Pourquoi Ivy ?

Le choix d'utiliser Ivy est plutôt simple, c'est une solution qui nous a été proposée par M.Truillet dans un moment du projet où la liaison Java-C par Sérialisation ne marchait pas ou marchait mal.

De plus, cette solution malgré sa faible quantité de documentation sur internet était facilement compréhensible par des exemples fournis avec.

Bugs connus

De tous les bugs que nous avons rencontré, il n'en reste qu'un seul, à notre connaissance, présent dans l'application finale.

En effet, l'application étant conçue dans un environnement unix (bash Ubuntu pour Windows) pour le C et Windows pour le Java, la synchronisation des fichiers entre ces deux environnements n'est pas instantanée.

Ainsi, pour palier à ce problème, nous avons fait le choix de pouvoir faire une recherche avec des fichiers présent seulement dans le dossier *baseDeDocuments*.

Il y a probablement d'autres bugs, dont nous n'avons pas pris connaissance. S'ils sont découverts par la suite et représentent une gêne lors de l'utilisation, ils seront rapidement patché.

Vous pouvez nous aider, par l'envoi à notre (très officielle) adresse mail :

contact@seekfox.fr d'une capture d'écran et de la description du chemin effectué pour arriver à ce résultat . Nous comptons sur vous !

Conclusions

Conclusions personnelles

“What do I do next ?” Clément

Mon retour sur cette partie du projet est plutôt positif. En effet, ayant adoré mon stage de fin de L2 qui portait sur du Java, cela a été une joie pour moi de retrouver un de mes langages préférés et d’avoir l’occasion de l’explorer encore plus.

Lors de mon stage, j’avais utilisé Java Swing pour l’interface, ainsi l’utilisation de Processing était autrement plus intéressante (j’y avais déjà touché pour ma bataille navale, TP5 de POO).

Aussi, jouer avec un bus réseau pour lier le Java et le C a été fortement intéressant où je me suis retrouvé à explorer cette partie plus que demandé en concevant des protocoles, très simples, et ainsi observer les problématiques liées à la présence de plusieurs objets sur le même port de communication lors de l’implémentation de la recherche multimoteur.

Par la même occasion, j’ai enfin pu visualiser comment une attaque *Man In The Middle* fonctionnait et à quel point c’était utile pour déboguer la liaison (ou éteindre les moteurs qui avait la fâcheuse tendance de rester allumer et de s’accumuler).

Plus collectivement, ce projet renforce en moins l’idée qu’il est quand même plus simple de faire un projet seul mais à quel point ensemble le projet va plus loin que prévu et fonctionne à merveille lorsque chacun y trouve sa place.

“Stagiaire médaille d’or” Étienne

En ce qui me concerne, cette seconde partie du projet fil rouge a été l’occasion d’approfondir le fonctionnement d’un travail de groupe dans un contexte d’entreprise. J’ai pu me rendre compte que l’aspect organisation, communication, spécification et intégration d’un projet est crucial, et celui-ci m’a permis d’avoir un aperçu des difficultés qui peuvent en découler, comme la situation exceptionnelle de confinement que nous avons subi.

Finalement, même si je pense que mon efficacité sur cette partie a été moindre que sur la partie 1, l’expérience gagnée dans une situation de crise telle que nous l’avons vécue pourra m’être utile dans mon futur professionnel.

“Supreme Leader” Gaël

Ce projet fut ma première expérience en tant que chef de projet et même si j’ai eut quelques problèmes pour la première partie de ce projet, la seconde partie s’est mieux passée, notamment grâce au fait que nous avons fait beaucoup plus de réunions. De plus, après les quelques premières réunions, elles ont commencé à être plus structurées, plus efficaces et permettant à tout le monde de savoir quoi faire. Heureusement, j’étais entouré d’une équipe solidaire et pleine de bonne volonté.

A part ma fonction de chef de projet, j’ai aussi travaillé sur l’interface et cela m’a permis d’apprendre à créer un système ergonomique et à utiliser ce que j’ai appris en IHM.

“Still in late” Oualid

J’ai vraiment beaucoup appris dans ce projet qui m’a servit de belle initiation au langage Java et à l’importance du travail d’équipe. En effet, guidé par un digne chef d’équipe, nous avons pu réussir à atteindre notre mission.

Les nombreuses réunions m’ont pu faire prendre conscience de l’importance d’être dans le temps autant sur la programmation que sur les réunions.

“Error 404” Raphaël

J’ai commencé cette seconde mission avec des regrets sur ma gestion de la première, et l’envie renouvelée de me lancer à corps perdu dans cette nouvelle tâche, pour mesurer l’efficacité de mes nouvelles compétences. Je voulais tout d’abord me consacrer à l’interface, car cela m’avait beaucoup plu, lors des cours dédiés à Processing, mais j’ai finalement accepté la tâche de m’occuper de la recherche complexe avec Rémi.

Malheureusement, suite à l’épisode COVID-19, et après le début du confinement, je me suis retrouvé face à plusieurs soucis successifs, matériels ou de connexion, ce qui m’a fait prendre du retard, surtout dans un projet où tout passait par une communication permanente à base de github. J’ai tout de même pu reprendre le projet et je suis assez satisfait de la vitesse à laquelle j’ai pu me rendre de nouveau utile, avec une vue plus globale du projet.

J’ai aussi pendant tout le temps où j’étais opérationnel, beaucoup appris sur le Java, ce qui m’a permis de décroiser ce que j’avais appris, et de remettre tout dans son contexte. Cela me sera assurément très utile dans le futur.

“New Challenger” Rémi

Venant d'un groupe différent à l'origine, j'ai eu quelques difficultés, naturelles et passagères, à trouver ma place. Au fil des jours de travaux en équipe, j'ai finalement pu, grâce au soutien de mon équipe trouver ma place. Grâce à cette équipe soudée, unies, au travail irréprochable, guidée par un digne chef d'équipe nous avons pu mener à bien notre mission.

Ce projet a marqué un tournant dans ma manière de travailler, un tournant positif. Cela est dû notamment aux nombreuses réunions et aux travaux acharnés de l'équipe. J'ai pris à ma charge, et avec Raphaël, la partie « Requête Complexes ». Un pari ambitieux qui fut très largement récompensé par notre travail, ô combien soudé.

Malgré quelques difficultés dû aux circonstances tragiques et déplorables du COVID-19, nous avons eu, en plus, quelques fâcheux problèmes de connexion internet. Somme toute, nous avons réussi à nous organiser à distance avec l'aide de chacun. Ce fut un travail acharné. Cette conclusion marque un point positif à ce projet. Je suis fier de notre travail et de notre équipe.

Conclusion de l'équipe

Nous avons tous beaucoup appris au cours de ce projet, et nous savons que cette expérience nous servira pour tous nos projets futurs. Après un certain temps où nous resterons sur la maintenance et le développement de quelques fonctionnalités pour Seekfox, nous nous orienterons vers de nouveaux projets, toujours plus innovants, afin de faire grandir nos compétences, et notre entreprise jusqu'à peut être rivaliser avec des géants tels que Google ou Amazon. Nous espérons que vous continuerez à suivre et soutenir Seekfox, et remercions de nouveau tous ceux qui nous ont permis d'arriver jusque là ainsi que Mamie et tantine pour les tartes et le café gratuit fournis à chaque réunion (envoyés par la poste pendant le confinement, quel professionnalisme !!).

Vive la République
Vive la France
Rene Coty notre rais à nous