P\_OO



Alexandre Samuel King et Thomas Rey – CID2B

ETML

17 Semaines

Aurelie Curchod

Table des matières

[1 Spécifications 3](#_Toc98749547)

[1.1 Titre 3](#_Toc98749548)

[1.2 Description 3](#_Toc98749549)

[1.3 Prérequis 3](#_Toc98749550)

[1.4 Cahier des charges 3](#_Toc98749551)

[1.4.1 Objectifs et portée du projet (objectifs SMART) 3](#_Toc98749552)

[1.4.2 Fonctionnalités requises (du point de vue de l’utilisateur) 3](#_Toc98749553)

[1.4.3 Travail à réaliser par l'apprenti 4](#_Toc98749554)

[1.4.4 Méthodes de validation des solutions 5](#_Toc98749555)

[2 Planification Initiale 5](#_Toc98749556)

[3 Analyse 5](#_Toc98749557)

[3.1 Opportunités 5](#_Toc98749558)

[3.2 Document d’analyse et conception 5](#_Toc98749559)

[3.2.1 Maquette du projet 5](#_Toc98749560)

[3.2.2 Base de donnée 7](#_Toc98749561)

[3.3 Conception des tests 7](#_Toc98749562)

[4 Réalisation 8](#_Toc98749563)

[4.1 Dossier de Réalisation 8](#_Toc98749564)

[4.2 Modifications 8](#_Toc98749565)

[5 Tests 8](#_Toc98749566)

[5.1 Dossier des tests 8](#_Toc98749567)

[6 Conclusion 8](#_Toc98749568)

[6.1 Bilan des fonctionnalités demandées 8](#_Toc98749569)

[6.2 Bilan de la planification 8](#_Toc98749570)

[6.3 Bilan personnel 8](#_Toc98749571)

[7 Divers 8](#_Toc98749572)

[7.1 Journal de travail 8](#_Toc98749573)

[7.2 Webographie 8](#_Toc98749574)

[8 Annexes 9](#_Toc98749575)

# Spécifications

## Titre

Projet Orienté Objet – Smart Thésaurus

## Description

Le projet consiste à faire un programme qui va indexer les contenus d’un répertoire, d’un site web et un serveur FTP(optionnel). L’utilisateur devra utiliser un moteur de recherche pour afficher les résultats de l’indexation.

## Prérequis

Modules 226 et 120. Modules 326 et 318 en parallèle.

## Cahier des charges

### Objectifs et portée du projet (objectifs SMART)

Réaliser un programme informatique de qualité

* UML (classe, activité, séquence, cas d’utilisation, …)
* MCD – MLD- MPD pour les données
* Design patterns (MVC, Template, …)
* Tests unitaires et d’intégration
* Documentation pertinente

Remplir une documentation professionnelle

* Planification (à jour)
* Journal de travail
  + Format ETML, Trello ou commit log
* Pro-activité
  + Poser des questions au client
  + Faire des démonstrations
  + Utiliser un système de versioning de code (GIT)

### Fonctionnalités requises (du point de vue de l’utilisateur)

Indexation

* Sources possibles
  + K
  + Site web
  + Serveur FTP (optionnnel)
* Choix des éléments à indexer
* Fichiers textes
* Documents Office et OpenOffice
* Documents PDF
* Images (juste le nom et la référence, contenu optionnel)
* Stockage des résultats d’indexation à choix entre
* Base de données (SQLite, MariaDB, MongoDB)
* Fichier CSV/XML
* RAM
* Choix du mode de mise à jour
* Manuel
* Chaque heure

Recherche

* Saisie des critères de recherche
* Support des opérateurs + et – (comme dans Google)
* Affichage des résultats par priorité de pertinence (meilleure correspondance)
* Accès direct à la ressource originale (clic)

Documentation

* Installation (y.c. metériel requis)
* Utilisation

### Travail à réaliser par l'apprenti

1. Créer et maintenir une planification et un journal de travail selon le modèle ETML
2. Concevoir le logiciel à l’aide de
3. Schémas UML (Classe, activité, cas d’utilisation, séquence)
4. Réflexions sur le fonctionnement du logiciel -> Schémas et choix pertinents
5. Petits programmes brouillon pour valider/invalider les idées
6. Implémenter le logiciel à l’aide de C#
7. En appliquant le paradigme OO
8. En utilisant les patterns appropriés (template, factory, …)
9. En protégeant le code avec des tests unitaires
10. En validant la qualité du produit global avec des tests d’intégration
11. En réalisant un installateur
12. Rédiger la documentation
13. Organisationnelle :
14. Planning et (journal au format ETML, Trello ou commit log)
15. Bilans
16. Planification (différence prévu / réalité)
17. Fonctionnalités (effectué / en cours /non réalisé)
18. Personnels (qu’avez-vous appris)
19. Technique
20. Schémas/explications pour les développeurs
21. Rapports de tests
22. D’installation et d’utilisation

### Méthodes de validation des solutions

Nous allons faire des tests unitaires et d’intégration pour protéger le bon fonctionnement du code.

# Planification Initiale

Le projet commence le lundi 10 janvier 2022 et fini le 23 mai 2022. Nous avons des relâches du 21 février au 28 et des vacances de Pâque du 18 avril au 1 mai. Le projet durera donc au total 17 semaines. Nous devrons nous occuper en premier de la documentation et des schémas UML et des maquettes, pour ensuite finir avec le code

# Analyse

## Opportunités

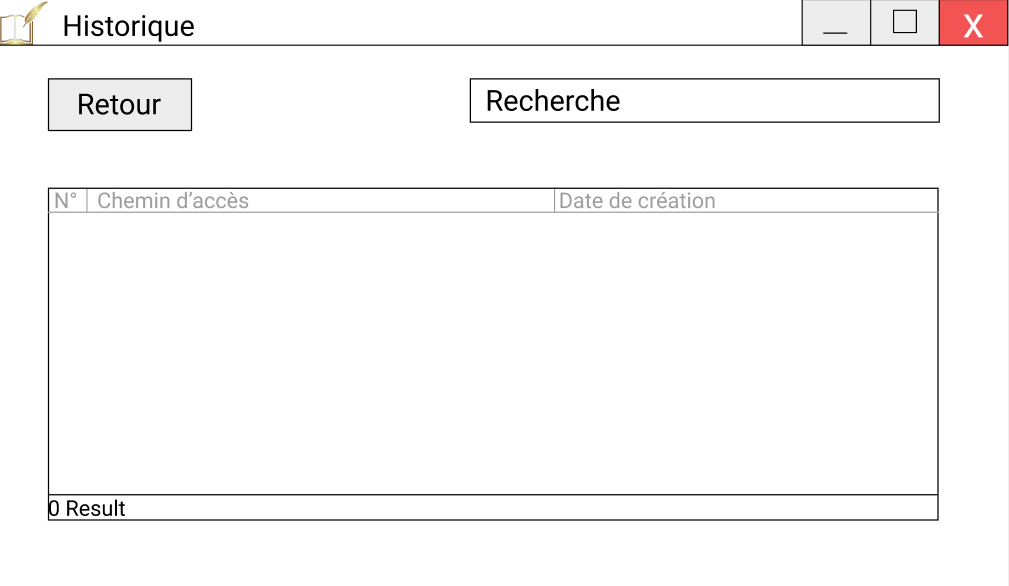
Pour ce projet, on devra s’améliorer au niveau des schémas UML. Nous devons aussi faire des recherches sur la gestion de fichier plus poussé et un apprentissage sur les design pattern.

## Document d’analyse et conception

### Maquette du projet

Le projet ressemblera à ça.





On pourra choisir le répertoire et le chemin d’accès.



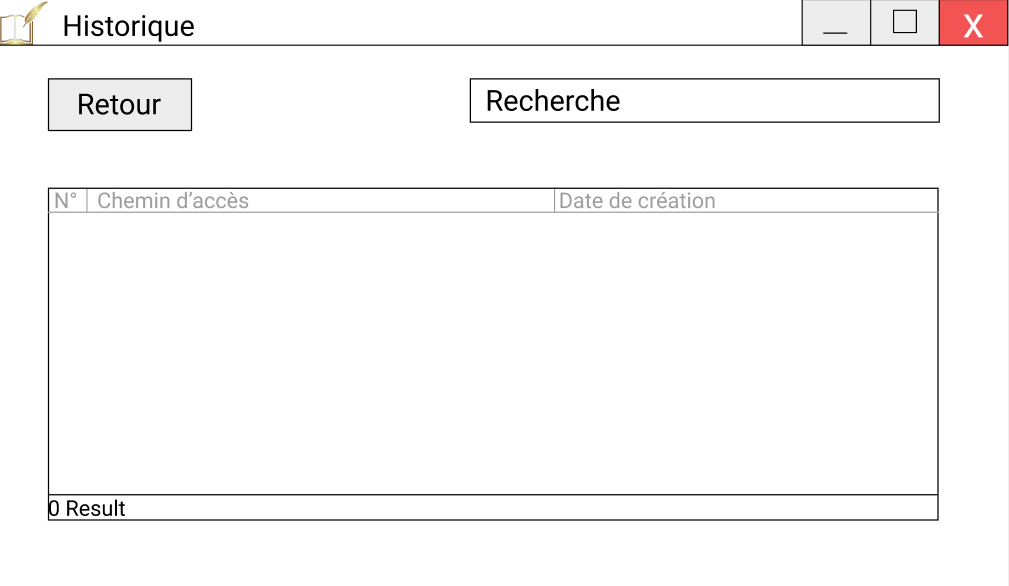
On pourra aussi choisir un dossier/ fichier grâce à la barre de recherche et ses supports des opérateurs. Il sera aussi possible de choisir le type de fichier voulu.



Après être dans un dossier, on pourra indexer le contenu du dossier. Après avoir fait cela, il sera possible d’accéder à cette indexation grâce à un historique qui nous ressort tous les indexations que nous avons faites jusque-là. De plus, il sera aussi possible d’ouvrir le fichier sélectionner directement dans l’explorateur de fichier.

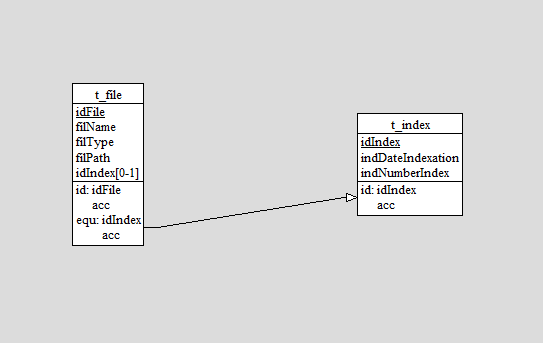


Une fois l’historique ouvert, l’utilisateur pourra sélectionner une indexation déjà crée.



### Base de donnée

Pour la base de donnée, nous avons deux tables. La premier est la table index qui va contenir tous les fichiers indexer dans un dossier. La deuxième est la table file, elle va contenir toute les caractéristiques d’un fichier ex : nom, type, chemin d’accès.



## Conception des tests

1. Flèche du dessus qui navigue entre les disques(site).
2. Barre de rechercher qui afficher les fichiers.
3. Ouverture d’un fichier quand on clique dessue.
4. Indexation correcte quand on clique sur le bouton indexer.
5. Bonne récupération de la base de donnée.

# Réalisation

## Dossier de Réalisation

Pour la réaliser de ce projet, nous avons utilisé le modèle MVC. Nous avons deux vues (une vue pour l’indexation et l’autre pour l’historique), un contrôleur et deux modèles (pareil que pour les vues, un pour l’indexation et un autre pour l’historique).

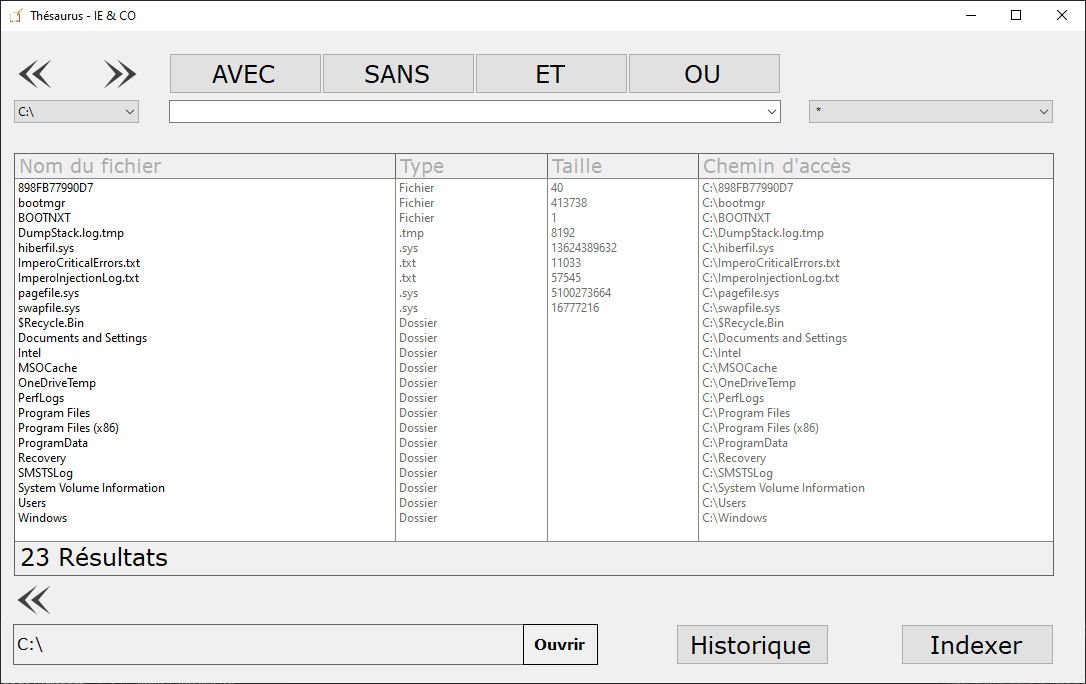
La vue pour l’indexation propose de choisir le disque du dossier/fichier à indexer ou de choisir l’URL du site à indexer, de filtrer les résultats grâce à une barre de recherche, d’ouvrir un fichier lorsque l’on double clique dessus, d’indexer les dossiers/fichiers/sites web de son choix et de se rendre sur l’historique. Celui-ci, permet d’afficher l’historique des indexations et lorsque l’on clique dessus afficher tous les fichiers/sites appartenant à cette indexation.

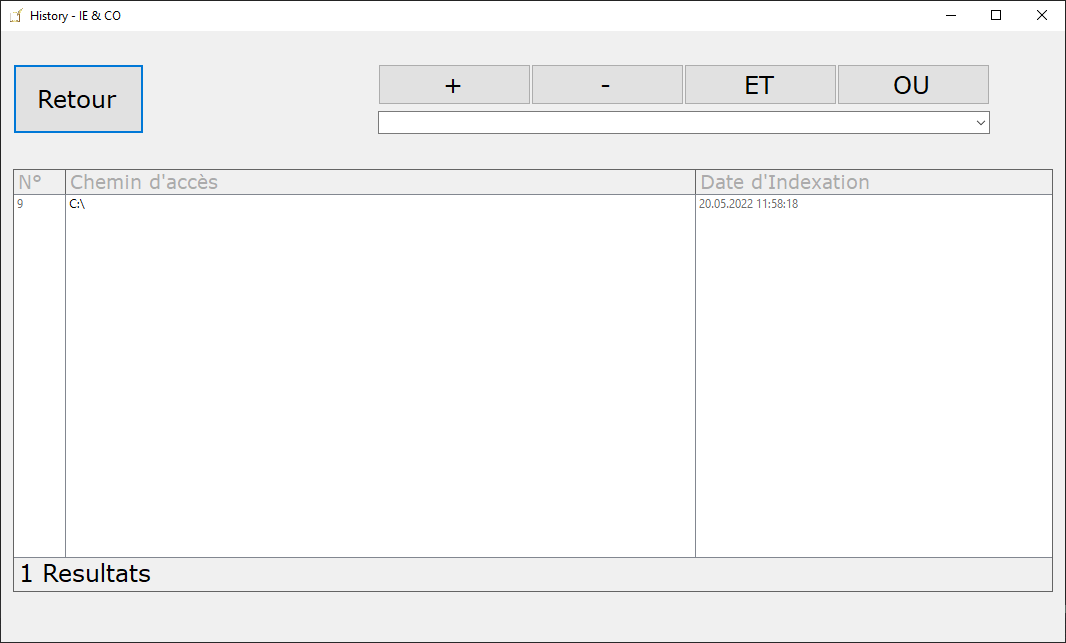
Le contrôleur sert, quant à lui, à relier la vue et son modèle.

Le modèle d’indexation, va gérer tout ce qui est mise en DB des données et indexation de dossiers/fichiers/sites web. Le modèle d’historique permet de récupérer toutes les informations nécessaires à l’affichage de celui-ci.

## Modifications

Le design de l’application à changer pour plus de facilité à comprendre l’application





# Tests

## Dossier des tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nom du test** | **Description** | **Fonctionnalité** | **Etapes** | **Résultat attendues** |
| Navigation avec les flèches | Naviguer entre les disques à l’aide du |  | 1. Cliquer sur l’une des flèches | On doit changer de disque.  Si l’index est au maximum, alors rien ne se passe |
| Barre de recherche | Afficher les dossiers/fichier quand on écrit dans la barre de recherche |  | 1. Taper le nom du dossier/fichier à rechercher | Le dossier/fichier voulu ressort lors de la fin de recherche |
| Ouverture d’un fichier | Ouvre le fichier sélectionner (fonctionne aussi avec les dossiers) |  | 1. Sélectionner le fichier voulu  2. double cliquer sur le fichier | Le fichier doit s’ouvre |
| Indexation des fichiers | Index tout le fichier qui se trouve dans un dossier de manière récursive |  | 1. Être dans le dossier dans lequel on veut indexer 2. Cliquer sur le bouton indexer | Tous les fichiers qui sont dans le dossier et les dossiers contenu par celui-ci doivent être indexer dans la DB |
| Récupération des données | Les indexations s’afficher dans la form History |  | 1. Cliquer sur le bouton historique | Toutes les indexations faites doivent être afficher |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom du test** | **Date** | **Résultat** | |
| **OK/KO** | **Description** |
| Navigation avec les flèches | 21.05.2022 | OK | Le disque courant change de disque quand on clique sur l’une des flèches |
| Barre de recherche | 21.05.2022 | OK | Le dossier/fichier voulu ressort bien quand on le recherche (Il est sensible à la casse) |
| Ouverture d’un fichier | 21.05.2022 | OK | Le fichier s’ouvre bel et bien |
| Indexation des fichiers | 21.05.2022 | OK | Tous les fichiers sont indexés dans la DB |
| Récupération des données | 21.05.2022 | OK | Les indexations sont toutes présentes |

# Conclusion

## Bilan des fonctionnalités demandées

**Indexation :**

Il est possible d’indexer des fichiers des toutes les sources possibles (tant qu’on a les droits d’accès, par exemple disque J pour les élèves) ainsi que n’importe quel site web (à noter que selon le dossier ou site choisis, le programme prendra plus ou moins de temps à indexer).

On peut indexer n’importe quel type de fichier et le stockage se fait en base de données MySQL avec Uwamp comme serveur.

La mise à jour des fichiers se fait automatiquement, lorsque l’utilisateur crée un fichier dans un répertoire, celui est directement ajouté dans le programme. Ceci est dû au fait que lors de l’ouverture d’un dossier dans le programme, il récupère dynamiquement les fichiers et dossiers contenus dans ce dernier.

**Recherche :**

Il n’est pas possible de rechercher à base de opérateurs (temps estimer pour ajouter cette fonctionnalité : 4-6 périodes), cependant il est possible d’utiliser une liste d’extensions pour afficher uniquement les fichiers d’un certain type ou bien de chercher un fichier avec la barre de recherche. Celle affiche les fichiers où la correspondance est la meilleure.

Il est également possible d’ouvrir un fichier grâce à un double-clic.

**Documentation :**

Un fichier d’installation est fourni avec le programme. Celui-ci explique également comment utiliser le programme.

## Bilan personnel

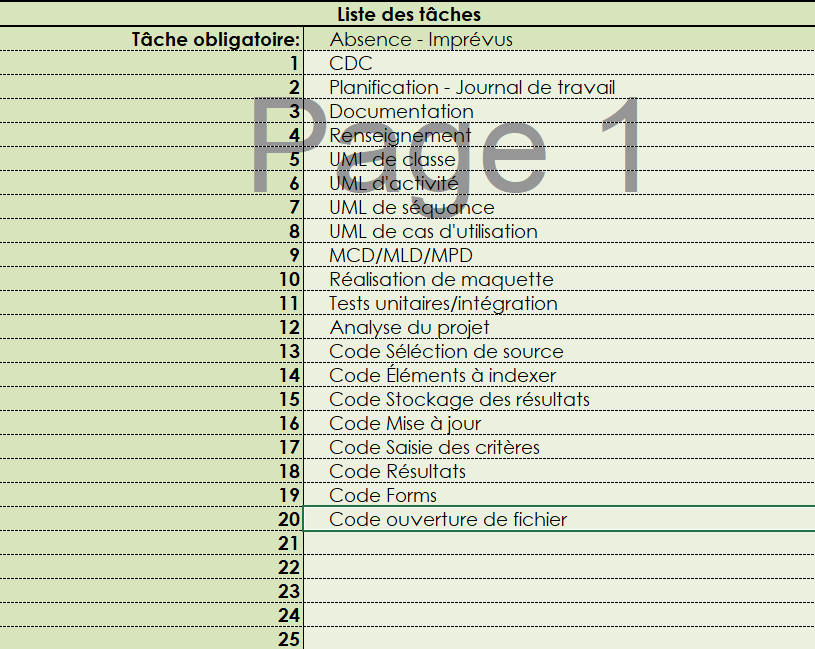
Alex : J’ai apprécié faire ce projet bien qu’au début je ne l’ai pas trouvé très clair. Le but du projet était flou et je n’ai compris qu’après quelques semaines le réel intérêt de ce dernier. De plus, j’ai trouvé qu’il n’apportait pas une bonne mise en pratique de ce que nous avions vu lors des modules fait au préalable. J’ai quand même pris du plaisir à recréer l’explorateur de fichier et réaliser l’algorithme récursif de récupération de données.

Thomas : J’ai bien aimé ce projet même je n’ai pas bien compris certaines fonctionnalités au début. Aussi, j’ai trouvé que ça n’avait pas vraiment de rapport avec les modules que nous fessions en parallèle, mise à part les schémas UML. Cependant, c’était un excellant exercice pour approfondir notre sens du travail en groupe. J’ai bien aimé faire le code en lui-même. Il y avait beaucoup de nouvelle fonctionnalité à apprendre et mettre en place.

# Divers

## Journal de travail

Le projet a duré 17 semaines de 8 période, du lundi 10 janvier au 23 mai. Nous avons déterminé ces tâches suivantes :



## Webographie

[Doc Microsoft](https://docs.microsoft.com/fr-fr/office/vba/language/reference/user-interface-help/writeline-method)

[Stack Overflow](https://stackoverflow.com/)