

**Le logiciel « Personnes d’intérêt »**

***Document confidentiel***

**Version : 1.0**

Si la plupart des connaissances d’une personne sont mortes ou disparues, ne vaut-il pas la peine d’investiguer sur elle?

Est-ce que la personne est dangereuse, ou est-elle en danger?

## Le logiciel « Personnes d’intérêt »

Le système « Personnes d’intérêt » permet de conserver des informations sur des personnes à considérer comme dangereuses, ou en danger.

Les fonctionnalités principales du système sont :

* Ajouter, modifier, supprimer des fiches sur des personnes
* Lier des personnes entre elles (ex : Fred connait Marc-André)
* Faire des recherches dans les fiches présentes dans le système
* Faire de l’analyse sur les données existantes.



## Votre mandat

Le logiciel est pratiquement terminé. L’interface graphique ainsi que la logique d’affaires sont déjà programmées, il ne manque que l’implémentation de la gestion et la persistance des données.

Votre travail consiste donc à terminer le logiciel en complétant la classe PersonDAO. La JavaDoc présente décrit ce qui doit être fait pour chaque méthode. Il devrait y avoir peu (ou pas) de modification sur le reste des classes.

# Informations supplémentaires

* L’application doit être codée à partir de l’application/squelette fourni.
* L’application doit utiliser avec BerkeleyDB pour les photos, et soit MongoDB ou Neo4j pour le reste.
* Le logiciel doit fonctionner sur le poste de travail du professeur sans installation de librairies supplémentaires
* Le projet doit être fait en équipe de 2 au maximum
* Chaque jour de retard entraîne 10% de pénalité sur la note de ce travail
* Si vous utilisez du code pris sur Internet, veuillez citer la source en commentaire, autrement cela pourrait être considéré comme du plagiat.
* Il est conseillé d’utiliser Git pour faire la gestion du code source.
* Même s’il n’y a pas beaucoup de données dans vos bases de données, créez les index aux bons endroits. Vous n’avez pas à créer les index en Java, seulement dans les consoles des bases de données (Compass et/ou neo4j browser)
* Programmez clairement (pas de code inutile, en commentaire ou mal indenté).

### README.md

Un fichier README.md doit être remis avec votre projet. Celui-ci doit contenir :

1- Le nom des coéquipiers

2- Quelles bases de données seront utilisées, et pour quel type d’information. La justification du choix des bases de données est très importante (entre 100 et 250 mots).

3- Quelle seront les collections ou libellés utilisés?

4- Où sont les index, ordre (asc/desc) + type (unique ou pas) ?

### Squelette du programme

Une base du programme a déjà été conçue. Celle-ci intègre déjà l’interface graphique et possède les librairies pour communiquer avec BerkeleyDB (*embedded*), Neo4j et MongoDB.

## Échéancier et livrables

Le projet (le code de l’application complète et le fichier README.md) doit être remis sur le U:/FTHERIAULT/DBA. Le logiciel et ses fonctionnalités doivent être présentés à l’enseignant sur son poste de travail lors du dernier cours de la session.

## Grille de correction générale

|  |  |
| --- | --- |
|  |  |
| README.txt   * Noms des coéquipiers * Justification du choix des BD (environ 250 mots)   + Quelles bases de données sont utilisés et pourquoi ?   + Quels sont les collections/libellés utilisées (s’il y a lieu) ? * Emplacement des index, leur type et l’ordre (s’il y a lieu) | % 10 |
| Sauvegarde des fiches   * Mise à jour * Insertion | % 20 |
| Liste des fiches   * Filtre et avec/sans image * Limitation des résultats | % 15 |
| Suppression | % 5 |
| Ratio libre/total | % 5 |
| Nombre de photos | % 5 |
| Nombre de fiches | % 5 |
| Personne la plus jeune | % 5 |
| La prochaine cible | % 10 |
| L’âge moyen des personnes | % 5 |
| Suppression totale | % 5 |
| Divers   * Qualité de la programmation et de la solution implémentée, etc. | % 10 |
|  |  |
| Total | **% 100** |