

Développement d'une application de gestion de cabinet médical

Tanguy Lallemand et Jonathan Cruard

December 6, 2018

1 Analyse du problème et solution

1.1 Analyse du problème

Ce projet vise au développement d'une application de gestion d'un cabinet médical. Il devra être capable de gérer l'ensemble des données du cabinet et notamment les données associées aux patients, aux docteurs mais aussi aux rendez-vous et aux prescriptions faites par les médecins aux patients. L'application devra être développée en C++ en suivant le paradigme de l'orientation objet.

1.2 Présentation de la solution

Pour développer cette application une analyse du problème a été faite et la solution a été élaborée sous la forme d'un diagramme de classe fournis ci-dessous. Il rassemble l'ensemble des classes utilisées dans ce projet. À ces classes s'ajoute un main et un fichier main_functions ajoutant diverse procédure utiles permettant une plus grande modularité du code.

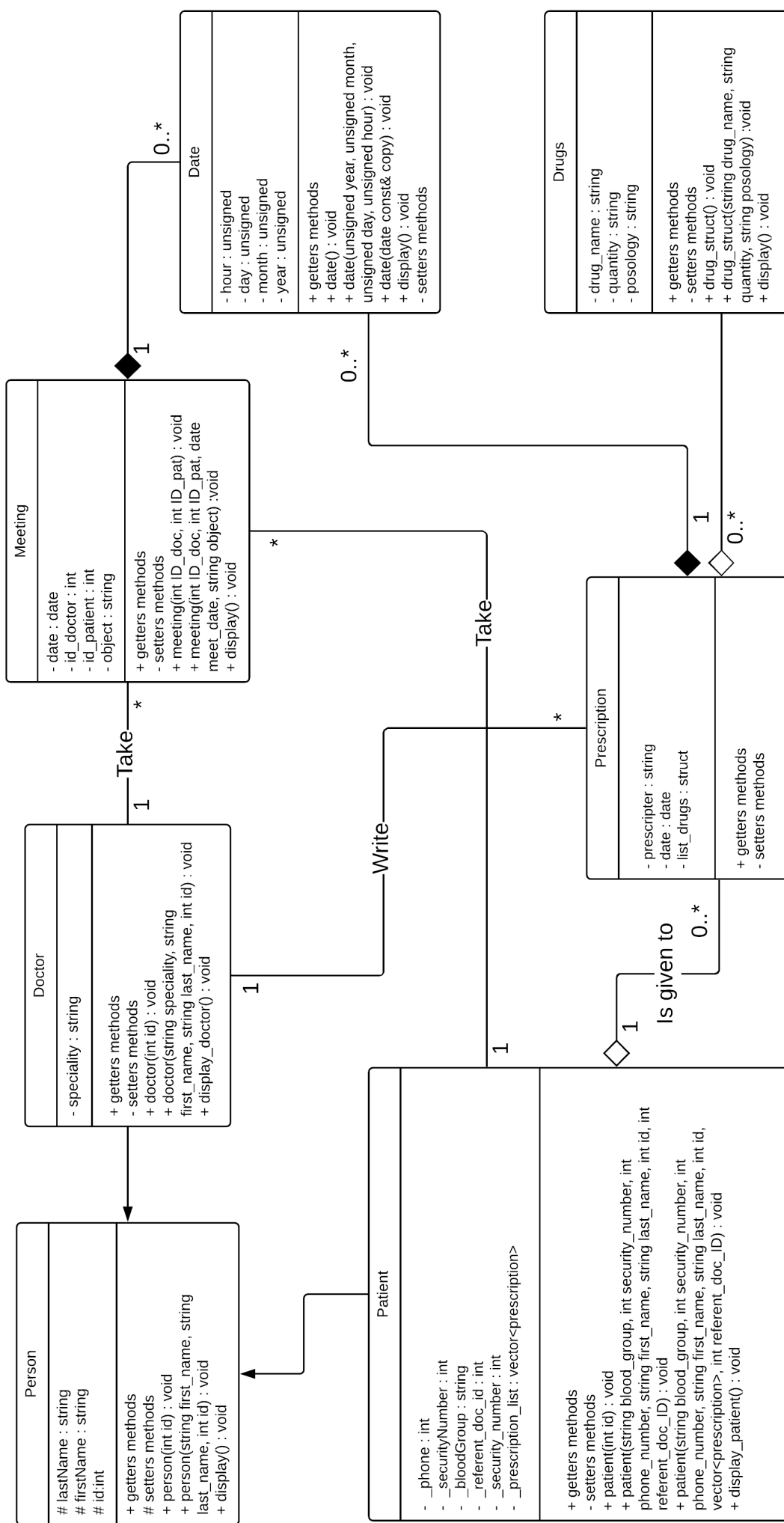


Figure 1: Diagramme de classe

Ainsi la classe prescription est composé d'une date (une prescription ne peut exister sans date). De plus, prescription est une agrégation de Drugs puisqu'une prescription peut être faite sans y ajouter de drugs. La classe meetings est composé de date puisque un rendez vous sans date ne peut exister. Les classes patients et docteur héritent de la classe personne, limitant ainsi le développement dans ces deux classes et permettant de leur faire partager une partie de leurs méthodes.

2 Notice utilisateur du logiciel

2.1 Instructions de compilation

Pour compiler ce logiciel il est nécessaire d'avoir un compilateur de standard c++11 (g++ 4.7 au minimum). On exécute alors les lignes suivantes à la racine du projet:

```
cd cabinet_medical
make
```

Si une recompilation est nécessaire il faut exécuter les lignes suivantes à la racine du projet:

```
cd cabinet_medical
make clean
make
```

Pour lancer le programme :

```
./gestion_cabinet.exe
```

La compilation est gérée par un Makefile permettant ainsi de modifier plus facilement les paramètres. À l'exécution du logiciel celui-ci commence par importer les données sauvegardées. Celles-ci sont stockées au format Json dans le dossier data, dans le fichier data.json. Le projet fournit un jeu de données d'exemple. Si l'importation s'est terminée correctement un menu s'affiche. Celui-ci est le cœur du logiciel et permet d'accéder à toutes ses fonctionnalités. Il est ainsi possible d'interagir avec lui en écrivant le chiffre correspondant à l'option choisie:

- Menu des patients permettant d'ajouter des patients, d'en sélectionner un pour interagir avec lui.
- Menu des docteurs permettant d'ajouter des docteurs, d'en sélectionner un pour interagir avec lui.
- Agenda qui permet d'ajouter des rendez vous ou voir ceux déjà posés.
- Sauvegarder et quitter permettant d'écrire l'ensemble des données dans le fichier data.json et de quitter le programme. Celles-ci seront rechargées automatiquement au prochain démarrage.

3 Présentation des résultats du programme

Voici le déroulement type du programme présenté sous la forme d'une série de capture d'écran clés placées en annexe. Ainsi ce programme permet de gérer l'ensemble des opérations demandées par le cahier des charges. Nous n'avons pas relevé de bugs non corrigés.

4 Conclusion

4.1 Problèmes rencontrés

Différents problèmes ont été rencontrés lors de la réalisation de ce projet. Tout d'abord la programmation objet demande une rigueur et une approche différente de nos habitudes en programmation. En effet, le principe d'encapsulation, les différents constructeurs demandent un temps d'adaptation. Néanmoins, passé ce temps, cette approche présente de nombreux avantages même pour un projet relativement court comme celui-ci.

De plus, la sauvegarde des données et leur récupération a posé quelques problèmes. En effet nous avons choisis de passer par le format Json, un format très utilisé (et donc utilisable au sein de projets variés), facilement lisible tel quel et dont la représentation imbriquée des données correspondait parfaitement à nos besoins. Pour accélérer le déploiement de cette technologie nous avons utilisé la bibliothèque json.cpp[1], une bibliothèque permettant l'écriture et la lecture de fichiers json. Afin de rendre notre projet plus exportable nous avons utilisé les sources fusionnées permettant l'importation directement dans les fichiers de notre projet d'un fichier .cpp et de deux .h. Mais la fusion des fichiers de la librairie n'a pas été simple à mettre en place. De plus bien qu'une documentation existe pour cette librairie nous avons mis du temps à la prendre en main. De plus certaines de nos structures étaient imbriquées sur de nombreux étages rendant difficile leur écriture et leur lecture.

Pour finir nous n'avons peut-être pas suffisamment bien pensé l'architecture initiale de notre projet ce qui nous a obligé à rajouter régulièrement de nouveaux accesseurs répondant au mieux à nos besoins. ce processus a ralenti notre développement.

4.2 Améliorations possibles

Bien que le cahier des charges soit respecté de nombreuses améliorations sont possible afin de rendre ce programme plus utilisable.

Tout d'abord dans une vision à court terme il serait possible d'ajouter une classe liste de docteurs et une autre qui serait liste de patients qui serait une agrégation de docteurs (ou de patients) ce qui permettrait de mettre en place des méthodes particulières aux besoins de gestion d'une liste de docteurs (ou patients) et notamment la gestion d'itérations dans la liste. De plus il serait intéressant de prendre la date actuelle depuis la machine pour alléger le gestion de l'agenda et des différentes dates du projet. Dans une optique similaire, il serait important de vérifier les entrées de l'utilisateur. EN effet, la plupart des entrées ne vérifient que le type afin d'éviter des problèmes de types. Néanmoins, il pourrait être intéressant de vérifier de façon plus importante les dates, les ID entrés par l'utilisateur afin d'éviter des entrées qui n'ont pas de sens. Pour finir, il pourrait être relevant d'intégrer la date de naissance des patient via un lien entre la classe patient et date. De très nombreuses fonctionnalités auraient pu être rajoutées au sein des menus tel que prise de rendez vous a partir d'un docteur etc.

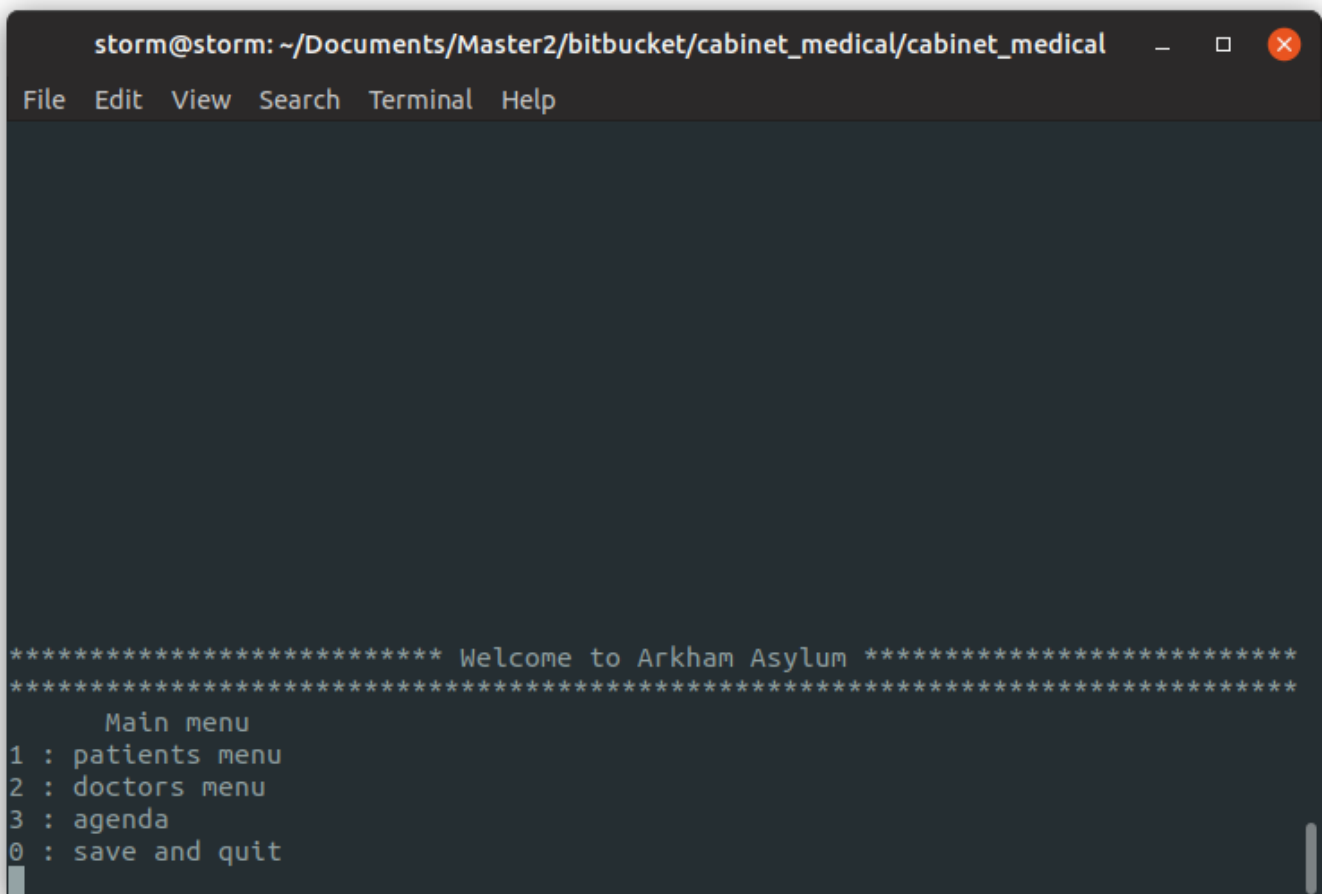
Dans un moyen terme, il serait important de permettre la suppression, ou pour le moins l'obsolescence des données. Ceci n'a pas pu être réalisé dans l'état actuel de notre projet étant donné l'approche que nous avons adopté dans la conception des ID. En effet dans l'état actuel du projet la suppression pourrait amener notre système d'ID à un comportement anormal. Pour remédier a cela il faudrait repenser ce système et implémenter un vrai système d'ID.

Dans un plus long terme, il pourrait être intéressant de se tourner vers la gestion d'une interface graphique via gtkmm et la gestion des données via sqlite3 si l'on souhaite se diriger vers la gestion d'un grand nombre de donnée.

References

[1] Baptiste Lepilleur. jsoncpp. <https://github.com/open-source-parsers/jsoncpp/>, 2007.

5 Annexe



```
storm@storm: ~/Documents/Master2/bitbucket/cabinet_medical/cabinet_medical
File Edit View Search Terminal Help

***** Welcome to Arkham Asylum *****
*****
Main menu
1 : patients menu
2 : doctors menu
3 : agenda
0 : save and quit
```

Figure 2: Menu principal du programme

```
storm@storm: ~/Documents/Master2/bitbucket/cabinet_medical/cabinet_medical
File Edit View Search Terminal Help
0 : main menu
2
*****
First Name      : Tanguy
Last Name       : Lallemant
ID              : 0
*****
First Name      : Bat
Last Name       : Man
ID              : 1
*****
First Name      : Joker
Last Name       : Joker
ID              : 2
*****
First Name      : Thanos
Last Name       : Thanos
ID              : 3
*****
First Name      : Harley
Last Name       : Queen
ID              : 4
Enter the chosen patient ID
```

Figure 3: Exemple de l’affichage de l’ensemble des patients du jeu fournis avec le projet

```
storm@storm: ~/Documents/Master2/bitbucket/cabinet_medical/cabinet_medical
File Edit View Search Terminal Help
***** Welcome to Arkham Asylum *****

Current patient
*****
First Name      : Joker
Last Name       : Joker
ID              : 2
Referent doctor ID : 1
Blood Group     : AB
Security number  : 12345785
Phone number    : 0
#####
Prescriptions
Prescriptor ID : 0
Date : 9/12/2018 Heure : 12H
Drug list :
  Drug name : Valium
  Quantity  : 12/day
  Posology  : 12

Prescriptor ID : 0
Date : 9/12/2018 Heure : 12H
Drug list :
  Drug name : Anxiolitics1
```

Figure 4: Exemple de l’affichage de l’ensemble des informations d’un patient donné

```
storm@storm: ~/Documents/Master2/bitbucket/cabinet_medical/cabinet_medical
File Edit View Search Terminal Help
0 : main menu
2
*****
First Name      : Jonathan
Last Name       : Cruard
ID              : 0
*****
First Name      : Professor
Last Name       : Xavier
ID              : 1
*****
First Name      : Clark
Last Name       : Kent
ID              : 2
*****
First Name      : Iron
Last Name       : Man
ID              : 3
*****
First Name      : Super
Last Name       : Man
ID              : 4
Enter the chosen doctor ID
```

Figure 5: Exemple de l’affichage de l’ensemble des docteurs du jeu fournis avec le projet

```
storm@storm: ~/Documents/Master2/bitbucket/cabinet_medical/cabinet_medical
File Edit View Search Terminal Help

***** Welcome to Arkham Asylum *****

Current doctor
*****
First Name      : Super
Last Name       : Man
ID              : 4
Speciality      : Laser therapist
*****

Doctor menu
1 : add new doctor
2 : choose doctor
3 : view informations
0 : main menu
```

Figure 6: Exemple de l’affichage de l’ensemble des informations d’un docteur donné

```
storm@storm: ~/Documents/Master2/bitbucket/cabinet_medical/cabinet_medical
File Edit View Search Terminal Help
#####
Date : 6/12/2018 Heure : 11H
Doctor ID : 0
Patient ID : 0
Object of the visit :
    rekt
#####
Date : 8/8/2012 Heure : 4H
Doctor ID : 1
Patient ID : 3
Object of the visit :
    Mental illness
#####
Date : 7/7/2018 Heure : 3H
Doctor ID : 1
Patient ID : 4
Object of the visit :
    A test
#####
Date : 9/7/2018 Heure : 3H
Doctor ID : 1
Patient ID : 4
Object of the visit :
    A test
```

Figure 7: Exemple de l'affichage de l'ensemble des rendez vous du cabinet