Rapport de Projet

UE INF 201, Projet :
Conception et développement d'un outil pour interroger des données PMSI

Lola Denet
Université de Bordeaux - Isped
Master 2 SITIS 2021-2022

4 Octobre 2021

Table des matières

1 Introduction

Ce projet est réalisé dans le cadre de l'unité d'enseignement INF201 du master 2 SITIS (Systèmes d'Information et Technologies Informatiques pour la Santé) et qui concerne la remise à niveau en programmation.

L'objectif est de manipuler des données issues du Programme de Médicalisation des Systèmes d'Information (PMSI) qui ont été fournies et qui sont à stocker dans une base de données MySQL.

L'application doit répondre à des questions selon les scénarios élaborés. L'outil devra reposer sur une base de données gérée dans MySQL et une application développée en JAVA.

Ce court rapport a pour but de présenter la méthodologie utilisée pour réaliser l'application et son fonctionnement. Il vient en complément de la documentation réalisée.

2 Matériel et méthode

2.1 Création de la base de données

Dans un premier temps, il a été nécessaire de créer la base de données en utilisant les données et le script SQL fournis. Le script permettait de créer et de remplir les tables mais pas de créer la base. Par conséquent, quelques lignes de code ont été ajoutées (voir listing 1) au début du fichier `bd_projet.sql` afin de permettre la création et l'initialisation de la base par la simple exécution du script.

```
1 DROP DATABASE IF EXISTS bd_projet;
2 CREATE DATABASE bd_projet CHARACTER SET 'utf8';
3 USE bd_projet;
```

Listing 1 – Code SQL pour la création de la base de données

Ces quelques lignes présentées dans le listing 1 permettent d'effacer la base existante afin de la recréer proprement. La dernière ligne permet de signifier que c'est sur la base `bd_projet` que les mises à jour et les requêtes sont effectuées.

Cela permet d'obtenir la base de données présentée dans la figure 1 contenant 6 tables.

— La table `tab_ccam` et la table `tab_cim10` sont en fait des thésaurus qui répertorient les actes (CCAM) et les diagnostics (CIM10) avec leurs codes et les libellés.

— La table `tab_acte` et la table `tab_diagnostic` contiennent respectivement les actes et les diagnostics réalisés au cours d'une hospitalisation.

— La table `tab_hospitalisation` contient toutes les informations d'une hospitalisation pour un patient donné.

— La table `tab_patient` répertorie les données patient.

Cette base va pouvoir être interrogée grâce à des requêtes SQL proposées par l'application.

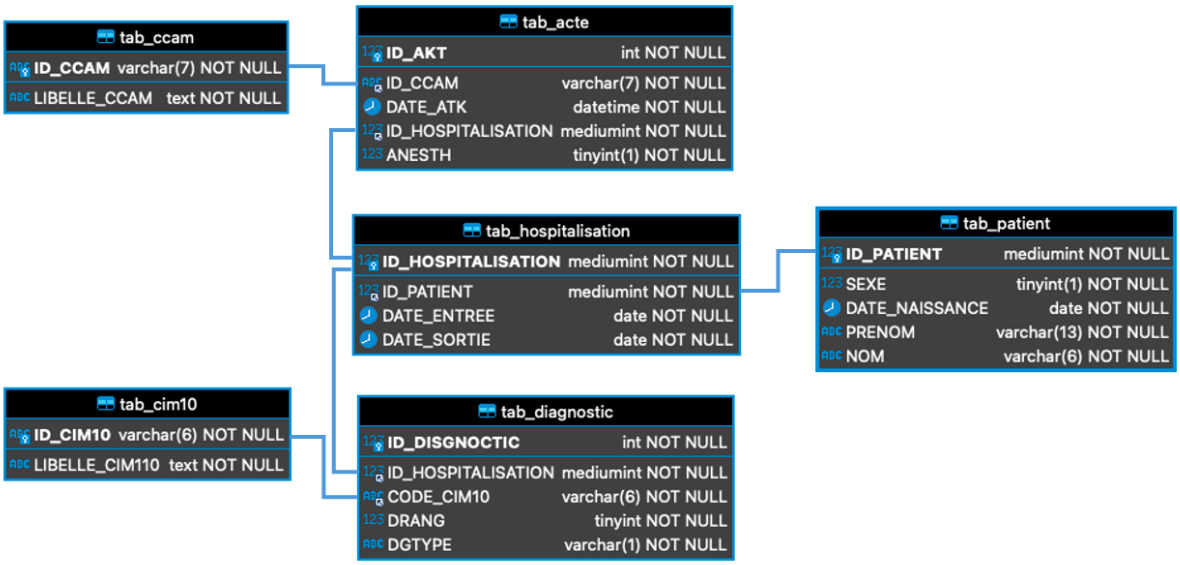

FIGURE 1 – **Base de données**.

Cette figure montre la base de données utilisée pour ce projet. Les rectangles correspondent aux différentes tables avec le nom de la table, la colonne constituant la clé primaire et le nom des différentes autres colonnes. Les traits bleus montrent les relations entre les tables. Ce sont les liens entre les clés primaires et les clés étrangères qui permettent ainsi de faire des jointures.

2.2 Création de l'application

L'environnement de développement intégré (IDE) Eclipse a été utilisé pour réaliser cette application ainsi que la librairie JDBC qui permet de communiquer avec une base de données MySQL. GitHub a servi de gestionnaire de version et permet d'héberger le code source. Overleaf et le langage LaTeX ont été utilisé pour établir ce rapport.

L'application a été développée en Java et les requêtes écrites en SQL. Elle est composée de trois classes présentées dans la figure 2.

La classe `Program` constitue la classe principale. Elle est le point d'entrée dans l'application et permet l'interaction avec l'utilisateur. La classe `DataBase` permet l'accès à la base de données et la classe `Query` de créer les requêtes.

La méthodologie suivie a été de concevoir un programme qui permet à l'utilisateur de saisir lui-même les paramètres de connexion à la base de données. Cela est rendu possible par la classe `DataBase`. Ensuite, différentes requêtes sont créées. Ces requêtes seront construites en prenant en compte les choix de l'utilisateur grâce à la classe `Query`. Toutes les interactions avec l'utilisateur se font grâce à la classe `Program`.

Les propriétés et les méthodes ne seront pas détaillées ici car ce serait redondant avec la documentation élaborée et mise en ligne [1].

Le projet peut être téléchargé en ligne [2] depuis GitHub. Le *repository* peut aussi être cloné directement avec la commande présentée dans le listing 2.

1. Documentation en ligne : `https://lola-denet.github.io/ProjetSITIS_POO_Lola_Denet_VEclipse/`
2. `https://github.com/lola-denet/ProjetSITIS_POO_Lola_Denet_VEclipse`

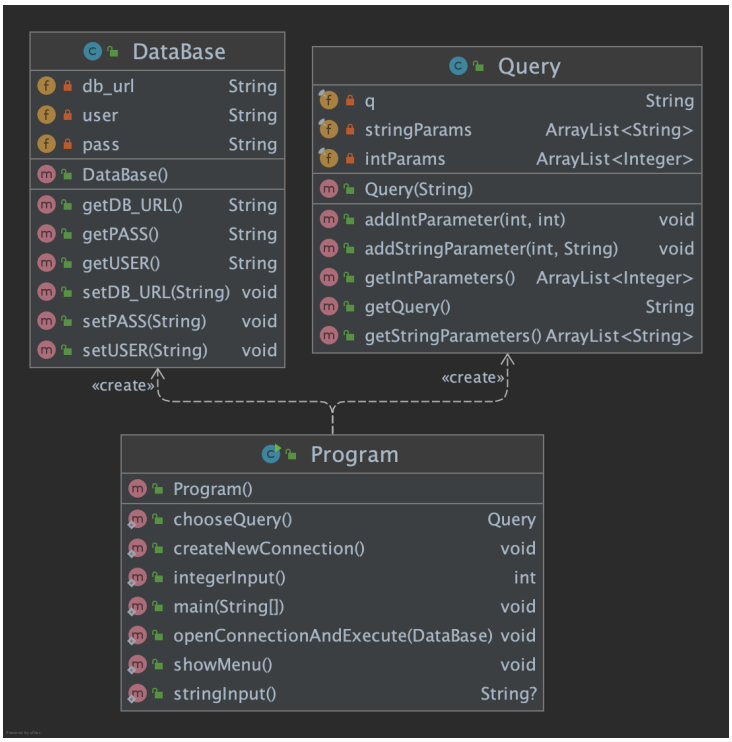

FIGURE 2 – **Diagramme de classes.**

```
$ git clone https://github.com/lola-denet/
   ProjetSITIS_POO_Lola_Denet_VEclipse.git
```

Listing 2 – Ligne de commande pour cloner le *repository*

Un fichier `README.md` présent dans le projet a été réalisé afin d'expliquer le fonctionnement de l'application. Il est possible de lancer l'application en exécutant directement le fichier `Program.jar` avec la ligne de commande présentée dans le listing 3. Il est nécessaire de se placer dans le répertoire contenant le fichier pour exécuter la commande ou d'indiquer le chemin d'accès au fichier.

```
$ java -jar Program.jar
```

Listing 3 – Ligne de commande pour exécuter l'application en utilisant le `.jar`

L'environnement Eclipse peut aussi être utilisé pour compiler et exécuter le programme simplement. Les fichiers binaires ont aussi été mis en ligne pour faciliter l'exécution et éviter à l'utilisateur de passer par l'étape de compilation. C'est alors la ligne de commande présentée dans le listing 4 qui permet de l'exécuter (si l'utilisateur est placé dans le répertoire `bin`).

```
$ java Program
```

Listing 4 – Ligne de commande pour exécuter l'application en utilisant le `.class`

3 Résultats

Le programme implémenté donne l'opportunité à l'utilisateur d'accéder facilement à la base de données et de choisir parmi les requêtes proposées.

Il s'exécute en ligne de commande et l'utilisateur visualise le résultat directement dans le terminal.

Tout d'abord, le choix a été fait de permettre à l'utilisateur de modifier les paramètres de connexion à la base de données. Il peut ainsi saisir lui-même l'hôte, le port, le nom d'utilisateur, le mot de passe et le nom de la base. Cela est présenté dans les figures 3 et 4.

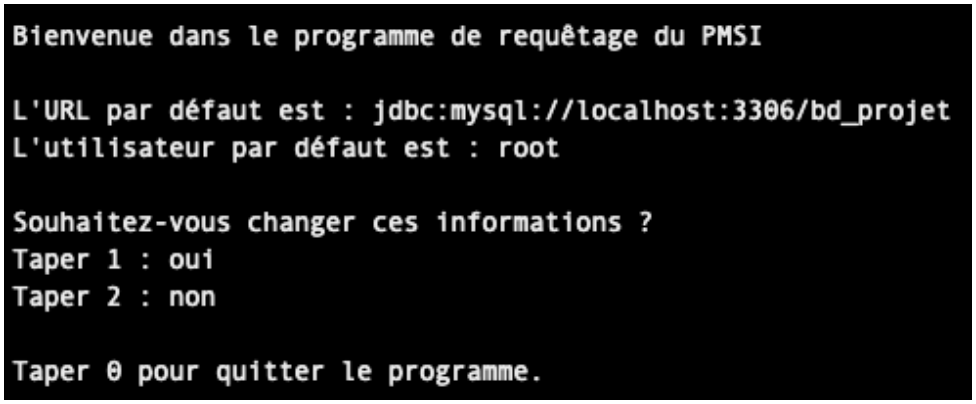

FIGURE 3 – **Menu utilisateur à l'exécution du programme.**

La figure 3 montre l'affichage à l'exécution du programme. Celui-ci laisse alors le choix à l'utilisateur de changer ou non les paramètres de connexion par défaut.

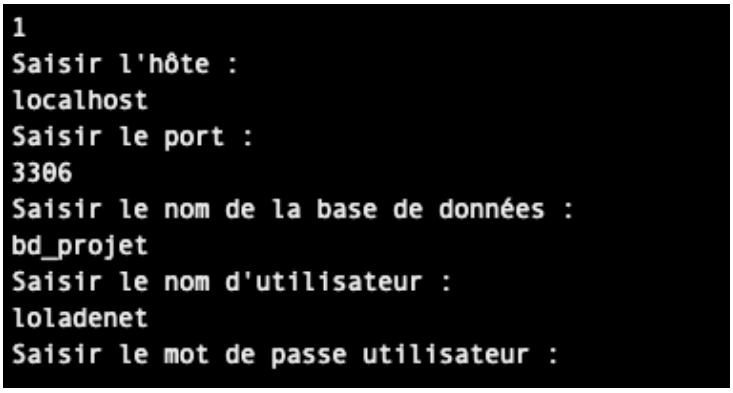

FIGURE 4 – **Menu permettant de changer les paramètres de connexion.**

La figure 4 montre l'affichage lorsque l'utilisateur fait le choix 1 (changer les paramètres de connexion). Différentes demandes apparaissent alors et permettent à l'utilisateur de saisir les informations requises.

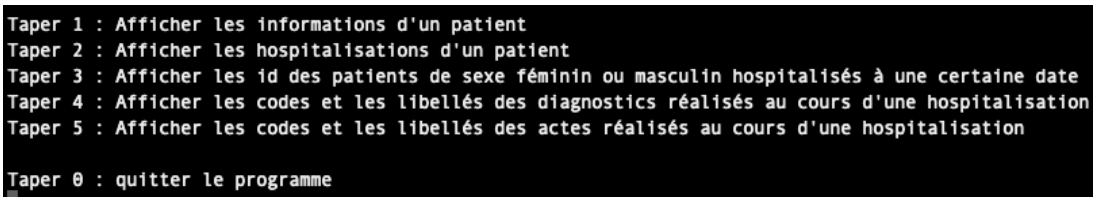

FIGURE 5 – **Menu permettant de choisir une requête.**

La figure 5 montre le menu des requêtes. L'utilisateur peut alors choisir parmi elles. Elles sont au nombre de 5. Le choix de ces requêtes a été fait pour montrer que les interactions avec les différentes tables fonctionnent (sans jointure, avec une ou plusieurs jointures) et qu'il est possible de prendre en compte les saisies utilisateur (avec un ou plusieurs paramètres). Elles reçoivent toutes un ou deux paramètres qui permettent de personnaliser les requêtes. L'application permet cependant de recueillir un nombre supérieur de paramètres (elle ne se limite pas à deux).

Les figures 6, 7, 8, 9 et 10 montrent l'exécution de chaque requête [3].

```
Taper 1 : Afficher les informations d'un patient
Taper 2 : Afficher les hospitalisations d'un patient
Taper 3 : Afficher les id des patients de sexe féminin ou masculin hospitalisés à une certaine date
Taper 4 : Afficher les codes et les libellés des diagnostics réalisés au cours d'une hospitalisation
Taper 5 : Afficher les codes et les libellés des actes réalisés au cours d'une hospitalisation

Taper 0 : quitter le programme
1
Entrer l'id du patient (6 chiffres) :
930012

Connexion réussie

ID_PATIENT : 930012
SEXE : 2
DATE_NAISSANCE : 2005-02-08
PRENOM : CHRISTIANE
NOM : POORIN
```

FIGURE 6 – **Requête 1.**

```
Taper 1 : Afficher les informations d'un patient
Taper 2 : Afficher les hospitalisations d'un patient
Taper 3 : Afficher les id des patients de sexe féminin ou masculin hospitalisés à une certaine date
Taper 4 : Afficher les codes et les libellés des diagnostics réalisés au cours d'une hospitalisation
Taper 5 : Afficher les codes et les libellés des actes réalisés au cours d'une hospitalisation

Taper 0 : quitter le programme
2
Entrer l'id du patient (6 chiffres):
930012

Connexion réussie

ID_HOSPITALISATION : 704968
ID_PATIENT : 930012
DATE_ENTREE : 2005-02-08
DATE_SORTIE : 2005-02-12
```

FIGURE 7 – **Requête 2.**

3. Le résultat de la requête 3 n'est qu'un extrait car le nombre de résultats est important est prendrait trop de place ici

```
Taper 1 : Afficher les informations d'un patient
Taper 2 : Afficher les hospitalisations d'un patient
Taper 3 : Afficher les id des patients de sexe féminin ou masculin hospitalisés à une certaine date
Taper 4 : Afficher les codes et les libellés des diagnostics réalisés au cours d'une hospitalisation
Taper 5 : Afficher les codes et les libellés des actes réalisés au cours d'une hospitalisation

Taper 0 : quitter le programme
3
Entrer le sexe du patient (1 ou 2) :
2
Entrer la date d'hospitalisation (au format AAAA-MM-JJ) :
2004-11-11

Connexion réussie

ID_PATIENT : 930012
ID_PATIENT : 928504
ID_PATIENT : 928044
ID_PATIENT : 927998
ID_PATIENT : 927226
ID_PATIENT : 926589
ID_PATIENT : 925388
```

FIGURE 8 – **Requête 3.**

```
Taper 1 : Afficher les informations d'un patient
Taper 2 : Afficher les hospitalisations d'un patient
Taper 3 : Afficher les id des patients de sexe féminin ou masculin hospitalisés à une certaine date
Taper 4 : Afficher les codes et les libellés des diagnostics réalisés au cours d'une hospitalisation
Taper 5 : Afficher les codes et les libellés des actes réalisés au cours d'une hospitalisation

Taper 0 : quitter le programme
4
Entrer l'id de l'hospitalisation :
503699

Connexion réussie

ID_CIM10 : E119
LIBELLE_CIM110 : DIABETE NON INSULINO-DEPENDANT (FLORIDE) (SUCRE) (DNID)
ID_CIM10 : G819
LIBELLE_CIM110 : HEMIPARESIE SAI
ID_CIM10 : N19
LIBELLE_CIM110 : INSUFFISANCE RENALE SAI
ID_CIM10 : R471
LIBELLE_CIM110 : DYSARTHRIE
ID_CIM10 : A499
LIBELLE_CIM110 : INFECTION BACTERIENNE GERME NON PRECISE
ID_CIM10 : I252
LIBELLE_CIM110 : INFARCTUS ANCIEN DU MYOCARDE (ASYMPTOMATIQUE)
ID_CIM10 : Z850
LIBELLE_CIM110 : ANTECEDENTS PERSONNELS DE TUMEUR MALIGNE DU COLON
ID_CIM10 : R410
LIBELLE_CIM110 : DESORIENTATION TEMPORO-SPATIALE (NON PSYCHOGÈNE)
ID_CIM10 : F059
LIBELLE_CIM110 : CONFUSION AIGUE
ID_CIM10 : J80
LIBELLE_CIM110 : SYNDROME DE DETRESSE RESPIRATOIRE AIGU (SDRA) (U1101)
ID_CIM10 : Z018
LIBELLE_CIM110 : BIOPSIE DE CONTROLE : REIN (RENALE)
ID_CIM10 : Z466
LIBELLE_CIM110 : SONDE URINAIRE (SEJOUR POUR MISE EN PLACE, REAJUSTEMENT, SURVEILLANCE...)
ID_CIM10 : Z018
LIBELLE_CIM110 : BIOPSIE DE CONTROLE : REIN (RENALE)
```

FIGURE 9 – **Requête 4.**

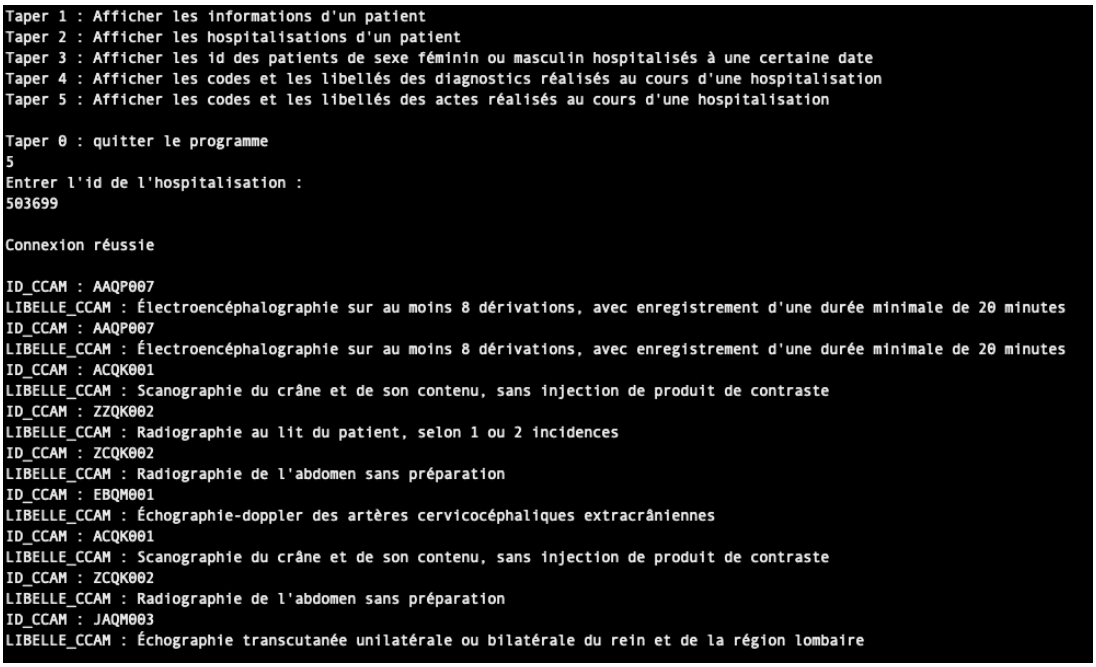

FIGURE 10 – **Requête 5.**

4 Conclusion

Pour conclure, ce projet a permis de concevoir une application donnant la possibilité à un utilisateur d'accéder simplement à une base de données et d'effectuer des requêtes. Le programme lui donne l'opportunité de changer les paramètres de connexion de façon à limiter les contraintes de gestion de la base par l'utilisateur (par exemple, il n'est pas obligé d'interroger la base en étant utilisateur *root*).

Les requêtes proposées permettent de valider le fonctionnement de l'application et démontrent que toutes les tables sont accessibles, qu'il est possible d'effectuer une ou plusieurs jointures et de recevoir un ou plusieurs paramètres. Cela montre aussi que l'utilisateur n'a pas besoin de relancer le programme pour changer de requête.

Cette version de démonstration pourra être facilement enrichie par de nouvelles requêtes grâce à la classe `Query` qui permet un formatage aisé. De plus, pour chaque nouvelle requête, il faudra ajouter une entrée dans le menu afin de la rendre visible pour l'utilisateur. C'est la fonction `chooseQuery()` qui permet cela. Aucune autre interaction avec le code source n'est nécessaire. Cela est présenté en annexe A.

Appendices

A Ajouter ou modifier des requêtes

Le listing A.1 montre le code de la fonction `chooseQuery()`. Des commentaires (en vert) ont été ajoutés afin de pointer les différentes étapes importantes à suivre pour modifier, ajouter ou supprimer une requête.

Premièrement, il est nécessaire de créer l'objet `Query` qui est construit avec une chaîne de caractères. Cette chaîne correspond à la requête SQL. Les paramètres récupérés de la saisie utilisateur sont remplacés par des " ?" dans la requête.

Ensuite, il est nécessaire d'ajouter la requête au menu pour que l'utilisateur puisse la voir sur l'interface.

Enfin, il faut ajouter un `case` correspondant au numéro de la requête et permettant de récupérer les paramètres de l'utilisateur. Chaque paramètre est stocké dans l'objet de la requête. Sa position dans l'objet correspond à sa position dans la requête.

```
/**
 * This method proposes queries and allows you to choose the parameters.
 *
 * @return Query : Returns the selected query with the parameters.
 */
public static Query chooseQuery() {

    // Creates queries objects : only a string is required. This string corresponds to the SQL query. Parameters are replaced by "?".
    Query q1 = new Query("SELECT * FROM tab_patient WHERE    ID_PATIENT = ?;");

    Query q2 = new Query("SELECT * FROM tab_hospitalisation WHERE ID_PATIENT = ?;");

    Query q3 = new Query("SELECT tab_patient.ID_PATIENT FROM tab_patient INNER JOIN tab_hospitalisation ON tab_patient.ID_PATIENT WHERE SEXE = ? AND DATE_ENTREE = ?;");

    Query q4 = new Query("SELECT ID_CIM10, LIBELLE_CIM110 FROM tab_cim10 tc INNER JOIN tab_diagnostic td on tc.ID_CIM10 = td.CODE_CIM10 INNER JOIN tab_hospitalisation th on td.ID_HOSPITALISATION = th.ID_HOSPITALISATION WHERE th.ID_HOSPITALISATION = ?;");

    Query q5 = new Query("SELECT tc.ID_CCAM, LIBELLE_CCAM FROM tab_ccam tc INNER JOIN tab_acte ta on tc.ID_CCAM = ta.ID_CCAM WHERE ta.ID_HOSPITALISATION = ?;");


    // Shows queries menu
    while (true) {
        System.out.println("\nTaper 1 : Afficher les informations d'un patient");
        System.out.println("Taper 2 : Afficher les hospitalisations d'un patient");
        System.out.println("Taper 3 : Afficher les id des patients de sexe feminin ou masculin hospitalises a une certaine date");
```

```
25          System.out.println("Taper 4 : Afficher les codes et les libelles
     des diagnostics realises au cours d'une hospitalisation");
26          System.out.println("Taper 5 : Afficher les codes et les libelles
     des actes realises au cours d'une hospitalisation");
27          System.out.println("\nTaper 0 : quitter le programme");
28
29          int answer = integerInput(); //listener
30
31          int intParam;
32          int position;
33          String stringParam;
34
35          //Get the parameters entered by the user and injects its into
    the query. The position corresponds to the position of the "?" into
    the query.
36       switch (answer) {
37         case 1 -> {
38           System.out.println("Entrer l'id du patient (6 chiffres) : ");
39           intParam = integerInput();
40           position = 1;
41             q1.addIntParameter(position, intParam);
42           return q1;
43         }
44         case 2 -> {
45           System.out.println("Entrer l'id du patient (6 chiffres): ");
46           intParam = integerInput();
47           position = 1;
48           q2.addIntParameter(position, intParam);
49           return q2;
50         }
51         case 3 -> {
52           System.out.println("Entrer le sexe du patient (1 ou 2) : ");
53           intParam = integerInput();
54           position = 1;
55           q3.addIntParameter(position, intParam);
56           System.out.println("Entrer la date d'hospitalisation (au format
    AAAA-MM-JJ) : ");
57           stringParam = stringInput();
58           position = 2;
59           q3.addStringParameter(position, stringParam);
60           return q3;
61         }
62         case 4 -> {
63           System.out.println("Entrer l'id de l'hospitalisation :");
64           intParam = integerInput();
65           position = 1;
66           q4.addIntParameter(position, intParam);
67           return q4;
68         }
69         case 5 -> {
70           System.out.println("Entrer l'id de l'hospitalisation :");
71           intParam = integerInput();
72           position = 1;
73           q5.addIntParameter(position, intParam);
74           return q5;
75         }
76         case 0 -> {
77           System.out.println("\n Au revoir\n");
```

```
78          System.exit(0);
79        }
80        default -> System.out.println("Votre saisie doit faire partie des
      propositions. Recommencer");
81        }
82     }
83 }
```

Listing A.1 – Fonction `chooseQuery()` pouvant être modifiée pour ajouter des requêtes