

# Rapport du projet de téléinformatique

Filière : RT2

Groupe : 2

Réalisé par : Ben Ghazeil Hana  
Chihi Sofiene  
Ayadi Yassine

Année universitaire : 2019-2020

# Introduction

Ce rapport s'installe dans le cadre du projet de travaux pratiques en téléinformatique pour mettre en emploi la programmation des sockets dans un réseau local et l'interaction d'un serveur avec une base de données.

Le sujet de notre projet est la gestion de données pour un cabinet de médecin. Les applications résultantes de ce projet permettent au médecin (administrateur) d'organiser l'ensemble des données relatives à ses patients dans une base de données facilitant ainsi l'accès à l'information. Quant à l'application Client, elle est conçue pour être utilisée par les assistantes pour effectuer plusieurs opérations tel que l'ajout d'un nouveau patient, des recherches dans la base et la consultation du calendrier des rendez-vous.

En passant à l'ordre de développement choisi, les tables de la base de données ont été mises en place en premier suivi par la création des interfaces graphiques des deux applications ainsi que la connexion entre le client et le serveur. Ensuite on a lié la base de données au serveur et on a établi les requêtes et les interactions client-serveur pour répondre aux besoins des éléments de l'interface.

# Conception et aspects techniques

Cette partie a pour objectif de présenter la solution conceptuelle proposée.

## 1. Conception de la base de données :

```
mysql> describe patient ;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| id     | int unsigned | NO   | PRI | NULL    | auto_increment |
| nom    | varchar(40)  | NO   |     | NULL    |                |
| prenom | varchar(40)  | NO   |     | NULL    |                |
| numeroTel | char(8)    | YES  |     | NULL    |                |
| email  | varchar(50)  | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> describe facture;
+-----+-----+-----+-----+-----+-----+
| Field      | Type              | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| code       | int unsigned      | NO   | PRI | NULL    | auto_increment |
| total      | smallint unsigned | YES  |     | NULL    |                |
| paye       | smallint unsigned | YES  |     | NULL    |                |
| fin        | date              | YES  |     | NULL    |                |
| debut      | date              | YES  |     | NULL    |                |
| id_patient | int unsigned      | YES  | MUL | NULL    |                |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> describe seance;
+-----+-----+-----+-----+-----+-----+
| Field          | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| temps         | datetime  | NO   | PRI | NULL    |        |
| patient_consulte | int unsigned | NO   | MUL | NULL    |        |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> show tables
-> ;
+-----+
| Tables_in_cabinet |
+-----+
| facture            |
| patient            |
| seance             |
+-----+
3 rows in set (0.12 sec)

mysql>
```

## 2. Interaction entre le serveur et le client :

On a utilisé les sockets en java à partir des classes « `java.net.Socket` » et « `java.net.ServerSocket` » pour établir une connexion locale entre le client et le serveur.

Pour la réception et l'envoi de données on a utilisé des objets de type `DataInputStream` et `DataOutputStream` qui fournissent les méthodes `writeUTF (ch)` et `readUTF ()` permettant le transfert des chaînes de caractère via les sockets

Les méthodes importantes utilisées :

- `Public void send (String ch)` : envoi une chaîne de caractères par le socket
- `Public void receive ()` : reçoit une chaîne de caractères du socket
- `Public void activate ()` : crée un `ServerSocket` et accepte la demande des clients pour la connexion
- `Public void connecti (String ip)` : crée un `Socket` de client et demande d'établir une connexion avec le serveur.

## 3. Exécution des requêtes SQL selon les boutons de l'interface graphique :

Le serveur distingue entre les différents boutons par un id envoyé en premier du client vers le serveur (l'envoi se fait dans la fonction `setOnAction` de chaque élément). Selon cet id, la méthode appropriée est appelée du côté serveur pour exécuter une requête SQL et puis le résultat est envoyé au client qui s'occupe de l'affichage.

À fin d'illustrer ce mécanisme, on prend l'exemple suivant :

Pour ajouter un patient (après avoir rempli les informations obligatoires pour un ajout), l'utilisateur appuie sur le bouton « ADD ». Ainsi l'id 'add' unique pour ce bouton est envoyé au serveur suivi des données du patient. La méthode `runOP (String op)` -coté serveur- sélectionne selon cet id la méthode `insert ()` qui reçoit les données puis exécute les requêtes de l'ajout d'un nouvel patient ainsi que la création de sa facture.

Quelques fonctions coté serveur interagissant avec la base de données :

`Public void select ()` : recherche le patient par son id (`getID ()`). Elle fournit les informations personnelles du patient ainsi que les détails de sa facture et son historique des rendez-vous si ce patient existe.

`Public void insert ()` : reçoit les données du nouvel patient et l'ajoute dans la table « patient » puis crée une facture vide ayant son id comme clé étrangère.

`Public Int getID ()` : reçoit un nom, un prénom et un numéro de téléphone et fournit l'identifiant unique du patient s'il existe sinon elle retourne 0.

`Public void delete ()` : assure la suppression de tout trace du patient, en supprimant toutes ses séances de la table « seance » puis sa facture pour enfin le supprimant de la table « patient ».

`Public void schedule ()` : reçoit une date et envoie au client toutes les séances pour ce jour ainsi que le nom, le prénom et le numéro de téléphone du patient consulté à chaque seance.

`Public void update ()` : cette méthode modifie la valeur d'une colonne (déterminée par l'élément sélectionné par l'utilisateur) pour le patient cherché.

## Langages utilisés pour la réalisation de ce projet :

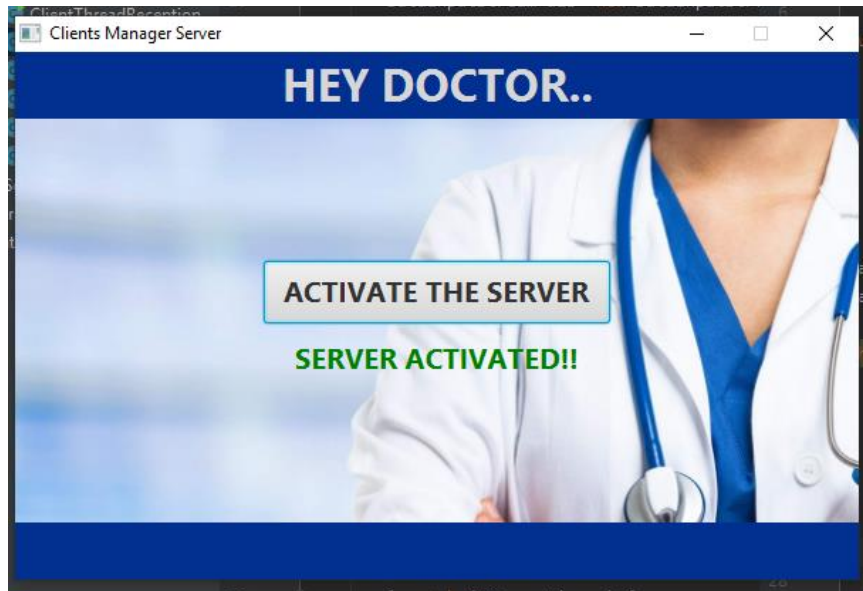
- Java  
Framework : JavaFx
- SQL

## Logiciels utilisés :

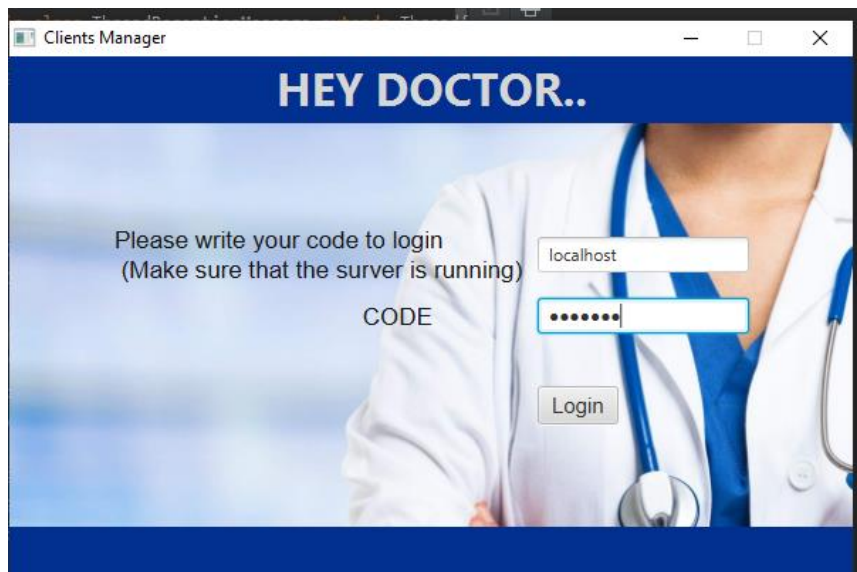
- MySQL
- JDK 8.0.1
- IntelliJ IDEA

## 4- Quelques captures d'écrans de l'exécution des applications :

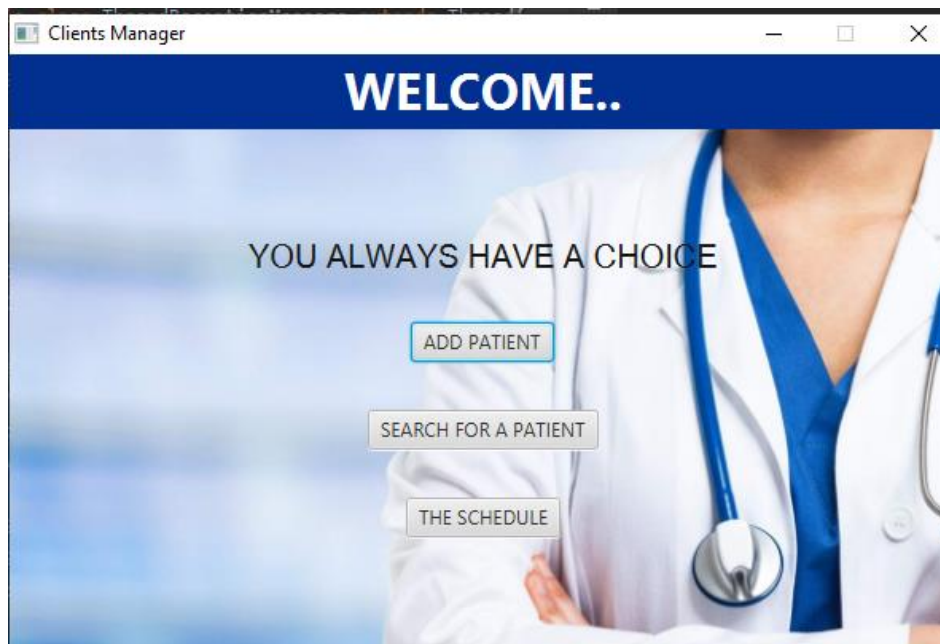
En ouvrant l'application serveur on a une page où il suffit de cliquer sur le bouton pour activer le serveur :



Pour l'application client, on a une page de login où on doit mettre le mot de passe et l'adresse ip de la machine qui contient le serveur pour le connecter après avoir activé le serveur :



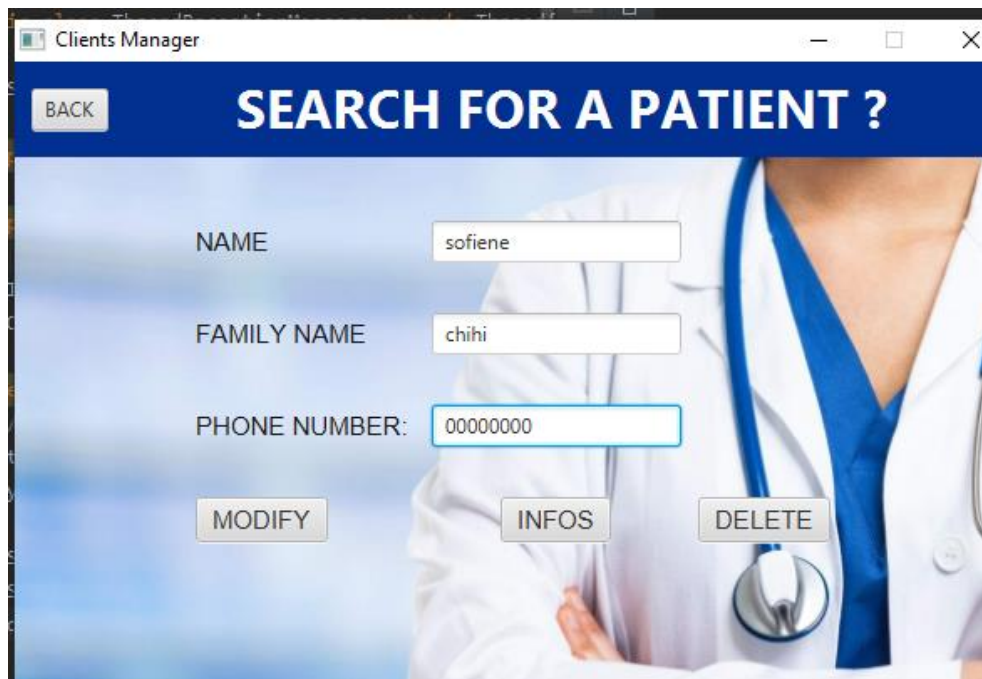
Après le login, l'utilisateur trouve une page où il a le choix entre ajouter un nouveau client (ADD CLIENT), modifier les données d'un client (MODIFY CLIENT) et chercher les séances des clients à une date bien déterminée :



En tapant sur ADD CLIENT, on trouve un formulaire à remplir d'un nouveau client, et il suffit de taper sur ADD pour ajouter ce client à la base de données.

A screenshot of a web browser window titled 'Clients Manager'. The page has a blue header with the text 'ADD CLIENT..'. Below the header is a background image of a doctor in a white coat with a blue stethoscope. On the left side, there is a 'BACK' button. The main content area contains a form with four input fields: 'NAME' (containing 'sofiene'), 'FAMILY NAME' (containing 'chihi'), 'E-MAIL' (containing 'sofienechihi@gmail.com' and marked as '(optional)'), and 'PHONE NUMBER' (containing '00000000'). At the bottom left, there is a green message that says 'Patient added successfully.'. At the bottom right, there is an 'ADD' button.

En choisissant le bouton de «SEARCH CLIENT», une nouvelle page est affichée et vous pouvez en tapant le nom, le prénom, et le numéro de téléphone du patient accéder aux informations de celui-ci, modifier ses données ou le supprimer de la base de données :



Clients Manager

BACK

## SEARCH FOR A PATIENT ?

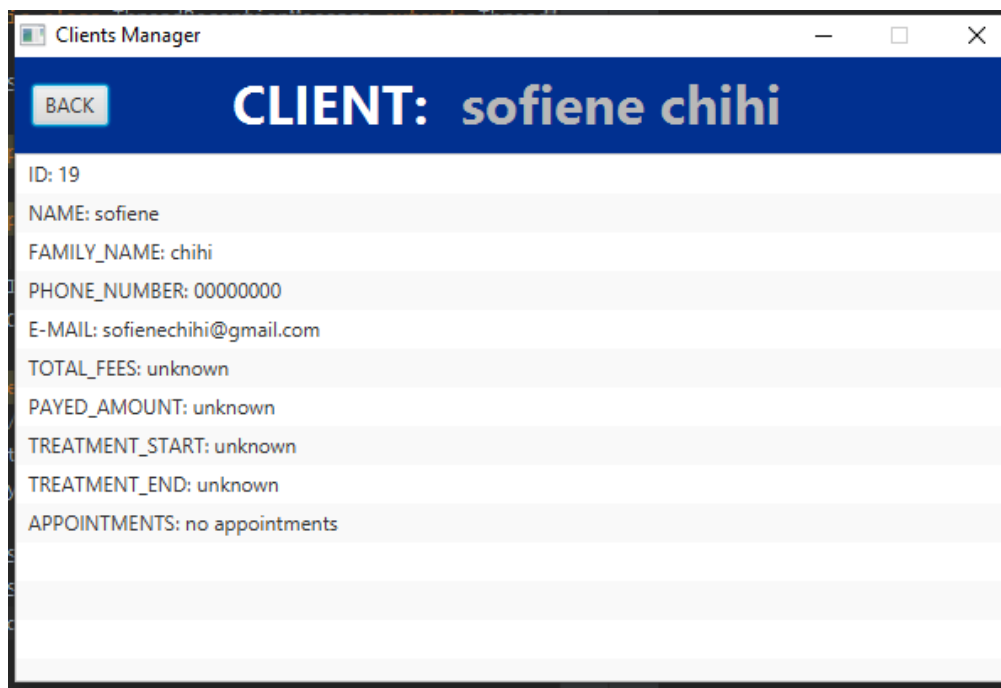
NAME:

FAMILY NAME:

PHONE NUMBER:

MODIFY    INFOS    DELETE

Si on choisit infos, on aura une page qui nous informe sur ce client :



Clients Manager

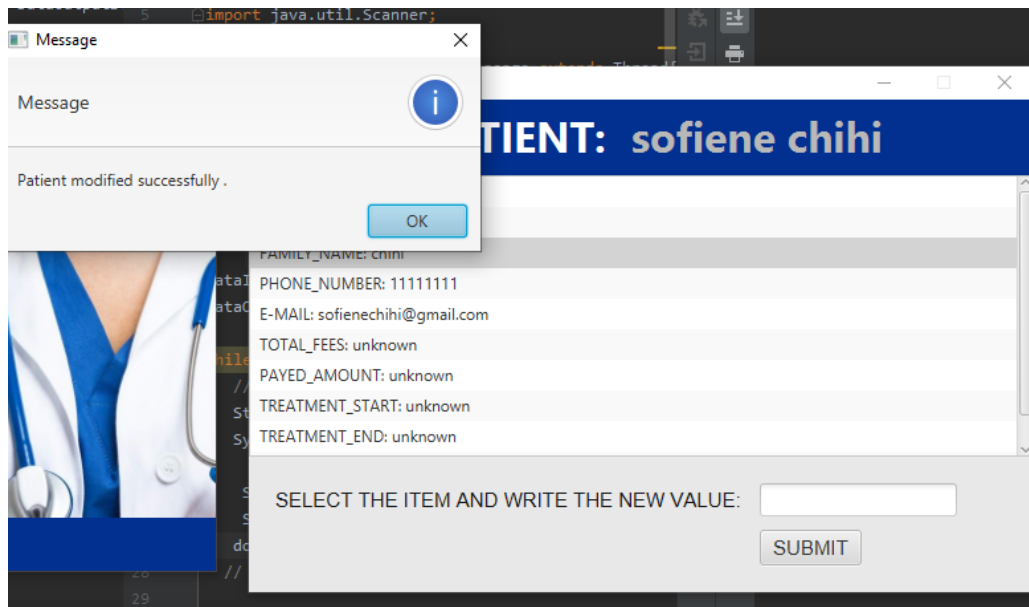
BACK

## CLIENT: sofiene chihi

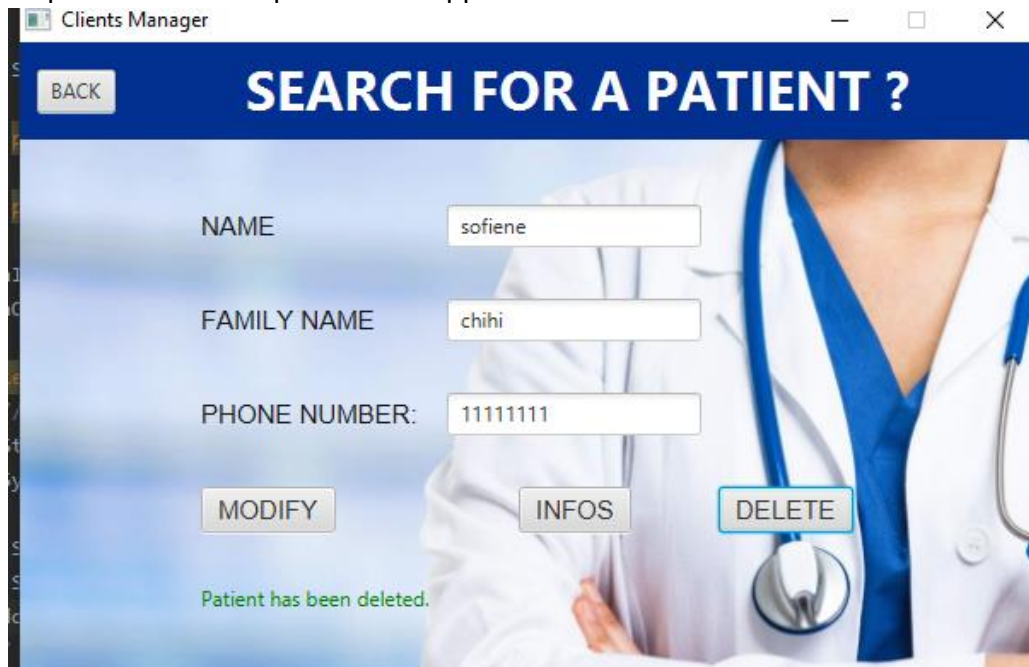
ID: 19
NAME: sofiene
FAMILY_NAME: chihi
PHONE_NUMBER: 00000000
E-MAIL: sofienechihi@gmail.com
TOTAL_FEES: unknown
PAYED_AMOUNT: unknown
TREATMENT_START: unknown
TREATMENT_END: unknown
APPOINTMENTS: no appointments



L'option « MODIFY » nous permet de modifier les infos du client :



L'option « DELETE » permet de supprimer ce client :



Et finalement, on a le bouton « SCHEDULE » qui nous permet de chercher les séances à un jour donné :

Comme pour cet exemple où le client sofienne ayant l ID 19 a une séance pour le 2017-03-24

Clients Manager

BACK

WHAT DO WE HAVE FOR..

2017-03-24

SUMBIT

sofiene

chihi

27081196

2017-03-24 08:45:00

```
mysql> select * from seance;
```

temps	ordonance	payement	patient_consulte
2017-03-24 08:45:00	VENTOLINE	60	2
2018-02-21 13:15:00	NULL	165	6
2018-02-21 17:20:00	NULL	150	5
2019-02-21 08:30:00	DAFALGAN , IMODIUM ,SPEDIFEN	250	4
2019-03-24 10:00:00	NULL	30	7
2019-03-26 10:00:00	VENTOLINE	31	3
2019-06-20 18:00:00	NULL	50	4
2019-12-06 14:30:00	NULL	50	6
2020-01-03 16:00:00	DOLIPRANE	125	8
2020-08-03 18:45:00	DOLIPRANE	0	9

# Conclusion

Ce projet s'est révélé très enrichissant dans la mesure où il a consisté en une approche concrète à l'architecture client/serveur offrant plus de sécurité ainsi qu'une administration au niveau serveur et facilitant l'évolution du réseau sans besoin de modification majeur. Il nous a permis aussi de découvrir un nouvel aspect de la programmation en Java et sa relation avec les bases de données .