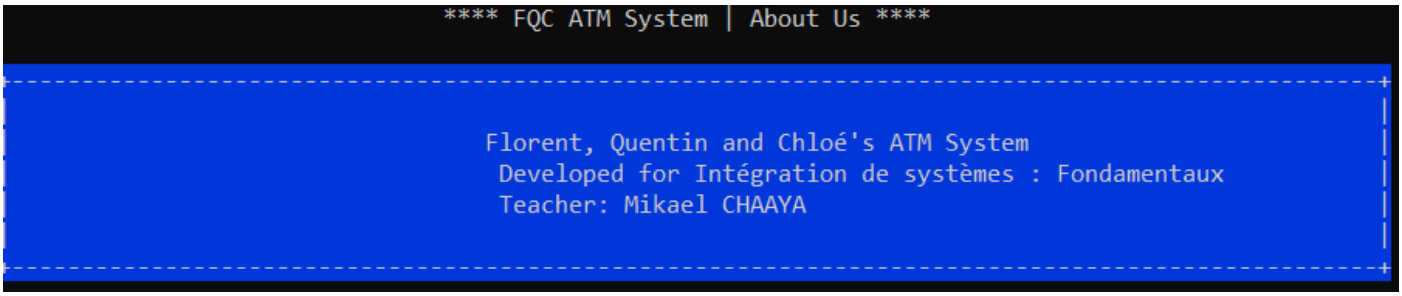
THE BANK PROJECT – C#

QUENTIN MOUREY

CHLOÉ STÉPHAN

FLORENT VALLMAJO



I – Our database

Our database is stored in the file "database.sqlite3" in the following directory: ATM\bin\Debug\net5.0. We used the DB browser for SQLite app on our computers to visualize the rows and the tables as we worked through the project.

Our database has 4 main tables.

- **"clients"**

This table stores the clients and the important information about each of them. It stores the name of the client, the PIN, the main currency, and the amount of money the client has on their bank account. It also stores information for the admin: if the user is blocked or not, the number of tries this client has tried to connect unsuccessfully in a row.

	id	GUID	FirstName	LastName	PIN	MainCurrency	isBlocked	nbrTries	moneyAmount
	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	D79F6137-93FA-44A0-82AB-...	Chloé	Stéphan	1234	Euro	0	0	100
2	2	D6748644-6013-4037-...	Florent	Vallmajo	5678	United States Dollar	0	2	15000
3	3	4E21955A-7707-4F65-...	Quentin	Mourey	9101	Australian Dollar	0	0	65
4	5	ABCD	Chloe	Polo	1234	Euro	0	0	109
5	10	DF8D82A822EF364D935A1C66AE86...	Hello	World	7896	Euro	1	0	0

```
CREATE TABLE "clients" (
    "id"    INTEGER UNIQUE,
    "GUID"  TEXT UNIQUE,
    "FirstName" TEXT,
    "LastName"  TEXT,
    "PIN"   INTEGER,
    "MainCurrency"  TEXT,
    "isBlocked" INTEGER,
    "nbrTries"  INTEGER,
    "moneyAmount"   INTEGER,
    PRIMARY KEY("id" AUTOINCREMENT)
);
```

We have a few people in it already. We manually entered one : the client with the GUID 'ABCD' and PIN '1234' as it was easier for us to work with through coding the project.

- **"currencies"**

This table stores the currencies associated to each client.

	GUID	currency
	Filtre	Filtre
1	ABCD	Euro
2	ABCD	Yen
3	D79F6137-93FA-44A0-82AB-...	Euro
4	D79F6137-93FA-44A0-82AB-...	Australian Dollar
5	D6748644-6013-4037-...	Dominican peso
6	D6748644-6013-4037-...	South Korean Won
7	ABCDEFG	Euro

```
CREATE TABLE "currencies" (
    "GUID"  TEXT,
    "currency"  TEXT
);
```

- **"messages"**

This stores the messages that the client leaves for the admin. It holds the message, the GUID of the client that left the message and the option for the admin to say whether is has been read yet or not.

	id	contenu	GUIDClient	wasRead
	Filtre	Filtre	Filtre	Filtre
1	1	Wrong main currency	ABCD	1
2	2	Account details not precise enough	ABCD	0

```
CREATE TABLE "messages" (
    "id"    INTEGER,
    "contenu"   TEXT,
    "GUIDClient"    TEXT,
    "wasRead"   INTEGER,
    PRIMARY KEY("id" AUTOINCREMENT)
);
```

- **"transactions"**

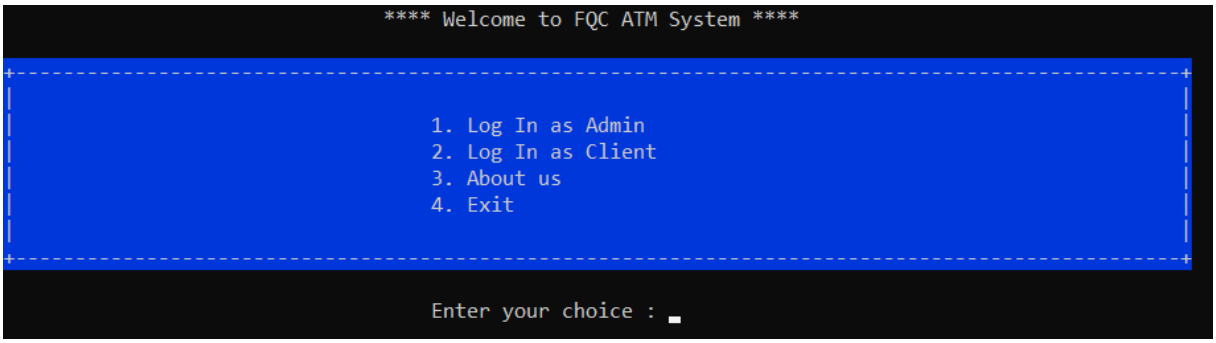
This table stores the transactions made by clients. It lets us know the type of the transaction, the date when it happened, the GUID of the client that did it, the amount of money involved and the currency,

which is the main currency of the client at the time of the transaction and then it also let us know if it was verified by an admin or not.

	id	type	date	GUID	amount	currency	isVerified
	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	Deposit	2021-11-06 15:57:15.3154294	ABCD	70	Euro	0
2	2	Deposit	2021-11-06 16:27:08.1732282	ABCD	87	Euro	1
3	3	Withdraw	2021-11-06 16:28:30.2790877	ABCD	87	Euro	1

```sql
CREATE TABLE "transactions" (
    "id"    INTEGER,
    "type"  TEXT,
    "date"  TEXT,
    "GUID"  TEXT,
    "amount"    INTEGER,
    "currency"  TEXT,
    "isVerified"    INTEGER,
    PRIMARY KEY("id" AUTOINCREMENT)
);
```

When our project is started, we are first facing a basic menu:



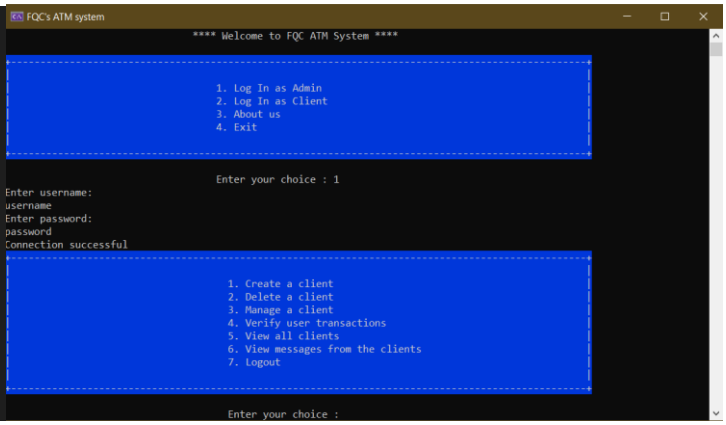
Let's log in as an admin !

II – An admin and their functionalities

An admin first logs in by sing the username : 'username' and the password 'password'. It is hardcoded in the code and an admin gets to try to log in as many times as they want.

```
private void LogInAdmin()
{
    Console.WriteLine("Enter username:");
    string usernameInput = Console.ReadLine();
    Console.WriteLine("Enter password:");
    string passwordInput = Console.ReadLine();

    if (usernameInput == "username" && passwordInput == "password")
    {
        Console.WriteLine("Connection successful");
        AdminMenu();
    }
    else
    {
        Console.WriteLine("Please Try again");
        LogInAdmin();
    }
}
```



An admin can then do different things, such as create or delete clients, manage a client, verify the client's transactions, view the messages, etc.

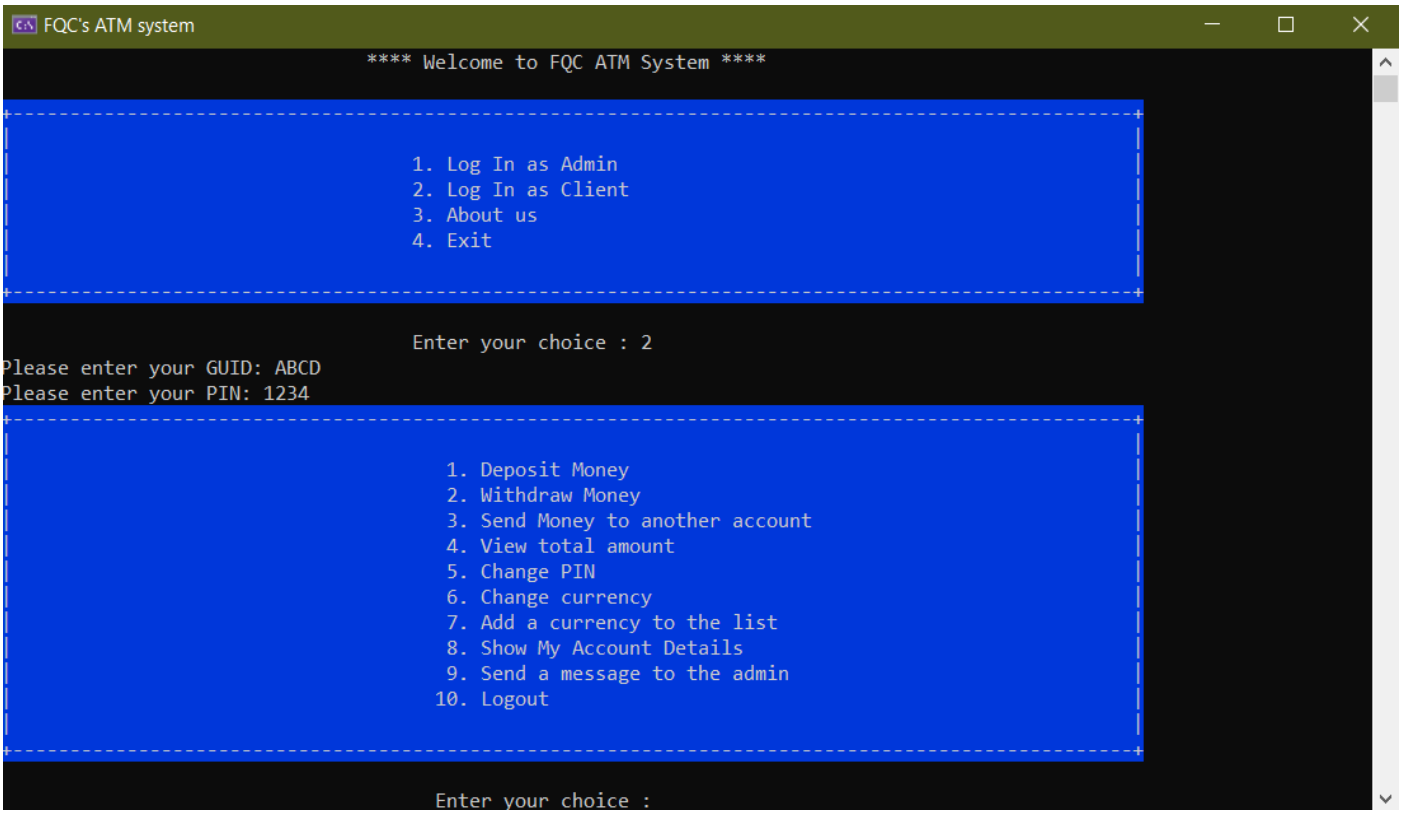
- Create a client: the admin enters the information manually for the First Name, Last Name and the main currency. The GUID and the PIN are then randomly generated, which ensures that the necessary formats are respected. We initialize the number of tries to zero and the amount of money to zero as well. The client is also created as not blocked.

- Delete a client: an admin enters the GUID of the client to delete and the client is then removed from the database. The client is removed from the "clients" and the "currencies" database, but we keep the traces in "transactions" and "messages" by keeping their GUID as we think it is important.
- Manage a client: this means resetting tries for a client, changing their PIN randomly again, add a currency for a client or block/unblock a client.
    - Resetting tries for a client means making the nbrTries 0 again.
    - Blocking a client means making isBlocked 1 and unblocking a client means making isBlocked 0.

- Verify user transactions: verifying transactions means getting a list of all the transactions yet to be verified and then entering the id of the transaction to verify. The value isVerified in the transactions table is then set to 1.
- View all clients returns the list of all the clients and their information.
- View messages from the clients is the last implementation for the admin. In this function, an admin can choose to see all the messages or only the unread ones. They can also set a message as read.

III – A client and their functionalities

A client can log in by writing their GUID and PIN.



The login function for a client is bigger than the one for the admin as more things are taken into account. A client can't log in if they are blocked. You can be blocked by the admin if the admin uses the block function. A client is also blocked once they try to connect and fail more than three times.

If the client isn't blocked and successfully logged in under three tries, the menu above appears.

- Deposit money: The first transaction that a client can make. This adds money to the total in the account and in the main currency used at the moment. The operation is added in the transactions.
- Withdraw money: A client can't withdraw more than what they currently have in their account. If the client wishes to withdraw in the correct limits, then the operation is added to the transactions.
- Send money: last type of transaction possible, one account can send money to another account and count it as a transaction.
- View total amount returns the amount of money the client currently has in their account in their main currency.
- Change PIN changes the client's PIN randomly, yet again to ensure that it is in the proper format and random.
- Change currency allows to change the client's main currency by using one in the existing list of currencies.
- Add a currency to the list allows the client to add a currency to their list of currencies.
- Show my account details returns some info on the account, not much as most of the info must be visible by the admin only.
- Send a message to the admin allows the user to communicate with the admin. Their GUID is stored in the process so that the admin knows who send the message.

III – A few other functionalities

We use a couple functions to make the interface easier on the eye and easier to use. We use a function that centers everything and makes the background blue so that the menus are nicer to use as we use a lot of them to navigate through our project.

The function About returns a couple of information on the project, nothing much but enough just in case.

```
942         /* --- MAKE CONSOLE PRETTY --- */
            51 références
943         private string AlignText(int SpacesToAdd, string Msg, string Alignment = "R")
944         {
945             if (Alignment == "L")
946                 Msg = Msg.PadLeft(SpacesToAdd + Msg.Length);
947             else
948             {
949                 Msg = Msg.PadLeft(SpacesToAdd + Msg.Length);
950                 Msg = Msg.PadRight((98 - Msg.Length) + Msg.Length);
951             }
952             return Msg;
953         }
            10 références
954         private void DrawLine()
955         {
956             Console.WriteLine("+-----------------------------------------------------------------------------------------------+");
957         }
            4 références
958         private static void Center(string message)
959         {
960             int spaces = 50 + (message.Length / 2);
961             Console.WriteLine(message.PadLeft(spaces));
962         }
963
            1 référence
964         private void About()
965         {
966             Console.Clear();
967             Center("**** FQC ATM System | About Us ****\n");
968             Console.BackgroundColor = ConsoleColor.DarkBlue;
969             DrawLine();
970             Console.WriteLine("|{0}|", AlignText(0, ""));
971             Console.WriteLine("|{0}|", AlignText(34, "Florent, Quentin and Chloé's ATM System"));
972             Console.WriteLine("|{0}|", AlignText(35, "Developed for Intégration de systèmes : Fondamentaux"));
973             Console.WriteLine("|{0}|", AlignText(35, "Teacher: Mikael CHAAYA"));
974             Console.WriteLine("|{0}|", AlignText(0, ""));
975             DrawLine();
976             Console.BackgroundColor = ConsoleColor.Black;
977         }
978
```

Finally, a very simple Main as we initialize everything before.

```
        0 références
        static void Main(string[] args)
        {
            Program obj = new Program();
            obj.MainMenu();
        }
    }
}
```

III – Functionalities to add later to optimize

There are a couple of things that we wish we could have added to fully complete this project. We unfortunately didn't complete the call to the API to get the currencies changes. Thus, changing between currencies isn't optimal as we don't have the latest news. Thus, we have the update in the tables, but the currency doesn't really change, which means that this part isn't as functional as it could be.

We also think it would be better if we broke everything down between different files, such as one for the admin, one for the client and the rest to ease the navigation between parts while coding.

We also don't have a json backup in case our sqlite database fails. We also reckon that our inputs aren't as safe as they could be, so we would definitely work on that as well.

Finally, we would also try to optimize the code as much as possible. Right now, our project works but when we try to do a few actions in a row, it crashes easily. We recommend trying each command on its own and coming back to the main log in menu often to clear everything up to ensure that it works.

IV – An overall look of the implementations we succeeded

Your software user will have one of 2 types: Admin or Client ✔

An admin has the following functionalities:

- Create a client ✔
- Manage a client (unblock, block, change pin, reset tries, delete client….) ✔
- Verify user transactions (optional) ✔
- View a list of all users ✔

An admin will only have a hard coded username and password. ✔

A client have the following functionalities:

- View GUID and credentials (all except pin!) ✔
- View total amount in preferred currency ✔
- Retrieve money from currency ✔
- Add money to currency ✔
- Change pin ✔
- Exchange between currencies (not optimal)
- Transfer between client (optional) ✔
- Leave message for admin(optional) ✔

As for the client, he will have the following:

- A GUID (use guid library in c#), GUID is a global unique id ✔
- A first name and last name, not unique ✔

- A pin ✔
- List of currencies and amount in each currency (use list) ✔
- Main currency ✔
- Any other important fields for the smooth run of the software ✔

As the exchange between currencies isn't optimal because we didn't connect to the API, we computed all the optional propositions to compensate. We are overall very happy with what we computed !

Thank you 😊