

MASTER THESIS - INTERNSHIP REPORT

# **DEVELOPMENT OF A CROSS-PLATFORM APPLICATION TO MANAGE MEDICAL DATA**

**2018/02 – 2018/06**

*Axel PERIGNON*

Supervisor: Prof. C. NGUYEN, Phd

**IMATH Laboratory  
University of Toulon**

## Acknowledgements

I wish to especially thank Christian NGUYEN, my mentor, for the time he devoted to helping me write this thesis and for the opportunity he gave me when he told me about this project.

I wish to thank Cédric GALUSINSKI, director of the MN team in the IMATH lab, for his precise explanation and his will to be certain that I understood perfectly the covered subjects.

I wish to thank Joseph RAZIK, teacher-researcher at the University of Toulon, who has been an attentive academic advisor to me and advised me when working on Kivy.

## Table of Contents

Acknowledgements.....	1
I – Introduction.....	3
1.1 Research laboratory .....	3
1.2 Internship .....	3
II - Preparation.....	5
2.1 Known needs .....	5
2.2 Environment and tools available.....	5
2.3 Anticipated difficulties.....	6
2.4 Subjects to research .....	6
2.5 First weeks .....	7
III – Application interfaces.....	8
3.1 Uniformity .....	8
3.2 Kivy .....	8
IV – Server communication & security .....	11
4.1 Transferring data .....	11
4.2 Storing data .....	11
4.3 Authentication.....	13
4.4 Account management .....	13
4.5 Algorithm .....	14
V – Medical data 3D visualization .....	15
5.1 Marching Cube .....	15
5.2 OpenGL with Kivy .....	16
VI – Technical problems related to Android.....	16
6.1 Kivy tools .....	17
6.2 HTTPS.....	17
6.3 Numpy .....	17
6.4 Unicode .....	18
6.5 Cython .....	18
VII - Conclusion.....	19
7.1 Objectives achieved.....	19
7.2 Improvement prospects .....	19
7.3 Personal thoughts.....	19
Appendix.....	20
References.....	21

## I – Introduction

### 1.1 Research laboratory

The IMATH laboratory has 3 teams, the one we will stay focused on is the Numerical Modeling team (MN) that develops numerical and mathematical models applied to Engineering Science. 23 permanent members are into the following teams:

- Applied non-linear Analysis team (AA), focused on optimization by computing variation, asymptotic and functional analysis, measurement theory and partial differential equation theory applied to various problems of classical physics (mechanics, thermodynamics, electromagnetism).
- Numerical Modeling team (MN), focused on numerical modeling and simulation, mainly in fluid mechanics.
- Computer Science and Applied Algebra team (IAA), focused on discrete mathematics, coding and cryptography to solve all problems arising from computer science and more particularly information theory.

Their researchers teach in the Engineer School of SeaTech, in the Faculty of Sciences and Techniques of the University of Toulon and in the Graduate School of Higher Education Teachers (ESPE) at the Academy of Nice.

The IMATH lab belongs to the FRUMAM [\[1\]](#) research federation, a union of mathematical labs from the Marseille surroundings focused on research. The lab also belongs to the M&MoCS [\[2\]](#), an international research center for Mathematics & Mechanics of Complex Systems in Italia at the University of L'Aquila.

The director of the MN team I'm working with is Professor Cédric GALUSINSKI, their research subjects are the scientific calculation for the mechanics of continuous media, the theoretical and numerical analysis of non-linear hyperbolic systems, the Monte-Carlo and quasi Monte-Carlo methods and the theoretical and numerical analysis of kinetic models for physics.

These information are available on their website [\[3\]](#).

### 1.2 Internship

My 4 months internship at the IMATH laboratory has been focused on the development of an application that will help the lab to obtain medical data. The subject covered depends from a work made in Fluid Modeling.

The lab has been contacted by the Hospital of Toulon to fulfill a need when interpreting data obtained from a scanner. Until now they have to interpret grayscale images to determine their patient medical situation. The computation done by the IMATH lab could greatly support the medical staff in the interpretation of pictures rendered by the scanner.

Similar work has already been done in the past but until now the data analyzed came from a pre-calculated result done by the scanner and we had to extrapolate important information from pictures rather than raw data.

Now it is possible to have such data but there are no tools available to ensure:

- Anonymity of those data when they are sent to the lab from the medical staff.
- Encrypted communications when sending those data.
- Account management for the medical staff and the lab.
- Availability of those data on a secured server.
- An easy way to manage those data for an outsider to complex computer tools.

I have been interested in working on this project for my internship when Christian NGUYEN, my traineeship supervisor, explained their situation to me.

It's also important to keep in mind that the hospital administration has been proved to be difficult to stay interested in this project even though they contacted the laboratory to develop a solution. Hence we knew that it was highly possible that I would have to figure out by myself the needs of medical staff during the development of this system.

This work being in a research laboratory, it's important to think how it can be used in other similar fields with little modifications while being certain to not compromise sensitive data so this will have to be taken into account for every functionality added to the project. If needed, the system must be usable with other hospitals or even for other laboratories.

I will have two main systems to develop in this project:

- Only one application to do everything like managing admin and restricted accounts or medical data with their result and so on. The app must be available on PC (Linux) and Android without having to edit a lot of code. It must also be coded in Python.
- The medical data will be stored on a server not yet available when the internship starts. A central computer service, DSIUN, is in charge of managing this server and it will be necessary to communicate with them to specify our needs.

The MN team from the IMATH lab has developed an algorithm to simulate fluid movements which has been named BiNSi (Bifluid Navier-Stokes in incompressible flow). It will not be used directly in this software but later the computation done on the raw medical data will use BiNSi to define an easy to understand result intended for medical staff.

## II - Preparation

### 2.1 Known needs

We will need a server to exchange medical data with the app and a protocol to secure those data whether during the transmission or once stored on the server.

The same app will be used by doctors to send medical data and receive results computed by the IMATH lab but it will be also used by our researchers to manage those data, so a rights and accounts management service will have to be implemented.

The medical staff needs to anonymize their patients' data but they must also easily find back the patient associated to any data sent.

The results received have to be visible in the app, just downloading the file is not enough. We may have two kinds of data, text and a 3D scene set with isosurfaces.

### 2.2 Environment and tools available

For this internship, I was asked to develop the application in Python with Kivy. Kivy is a convenient framework to develop interfaces and to port an app to Android.

The service managing the server for our lab is the DSIUN (it is a department of the University of Toulon). At the start of the internship, they were installing new servers so nothing was available yet and we couldn't know exactly what tools would be available. That is why we chose to use the most common technologies.

We knew the DSIUN would set an access to the data we store on the server using NFS but after some weeks it appeared that it would be only with static IPs, which they need to register beforehand. It would have been incompatible with our will to give access to these data through the application for anyone with an account. Also, on Android, mounting a NFS folder requires root rights on the device so even with static IPs it would have been a problem since the app must be easy to use and install. Consequently, we decided to not give our users direct access to NFS folders and not knowing what services would be installed on the server we chose a common service, the application will interact with the server through Php to handle files and user rights.

Thus a Php server with MySQL will be available. If it cannot be installed on the same server where the data are stored we will still be able to mount an NFS folder on the Php server to access the data. HTTPS protocol will be available to crypt data exchanges.

While waiting the DSIUN availability, I installed a Wamp server on my computer on Windows 7 to implement data transfer using Php and MySQL.

The application was tested and developed on Ubuntu 16 with the IDE Pycharm.

To manage the project evolution I used Git. The projects (server and application) are hosted on Bitbucket. Since the server runs on Windows and the app on Linux, I used two Git GUIs: Sourcetree (Windows) and gitEye (Linux).

I was asked to use OpenGL to setup 3D scenes with Python. Kivy uses OpenGL ES 2.0 and gives us access to it.

The Android development is done using a Motorola G5 on Android7, it is important to precise that the G series of Motorola has a 32 bit Android installed even though the processor is 64 bit.

### **2.3 Anticipated difficulties**

Python 3 is favored over Python 2 but we expect difficulties to port the app to Android. Kivy's documentation says that Python 3 support is in Beta.

My mentor explained me that there have been problems with Numpy and Android. After some research I discover that Numpy on Android should work on Python 2 but could fail with Python 3 on Android. I will have to study why and determine what can be done to overcome this problem.

I never used Kivy so I will have to study this framework, some delay in the development should be expected the first weeks.

The DSIUN and the Hospital of Toulon will be really difficult to contact. It will be complicated to obtain the wanted medical data and setting the server could take several months.

When discussing this internship subject with my mentor, we both knew he wouldn't be easily available the first months hence I have to plan my work as much as I can without having to wait for an answer to continue.

### **2.4 Subjects to research**

When I will have to secure data, HTTPS protocol will be use for the transmission which is already available within Python, but hashing data or storing encrypted data to decrypt later is not something I can do simply. There is a lot of way to secure such data, I will have to research the last algorithm available and adapt it to this situation.

The interfaces development will be done with a framework I never used, Kivy, so I will have to research how it works. The 3D visualization of human organs on OpenGL ES 2.0 through Kivy will require the study of the Marching Cube algorithm. This algorithm is used to obtain a 3D object compatible with OpenGL. Most results we obtain within the medical field are isosurfaces and need to be processed to be visible in OpenGL. Sometime ago, I worked on Marching Cube with my mentor so I already know how it works but it was in C, not in Python, and since running it can be time consuming it's important to research what has been done with this algorithm since then.

## 2.5 First weeks

Given this project subject and the fact that I'm alone working on it, it was deemed not necessary to plan everything. There is no division of labor, determining what must be done at one step to be able to work on the next step is enough.

Hence I subdivided the workload in simple steps, here is an example:

To set and test the server I need to develop the app enough to authenticate.

Interfaces development needed to set a login and password.

Determine what an account is, what information should be stored on the database.

Once this step done, all the other functions are ready to implement and test separately. If I don't feel confident in my choices for a functionality I can mail my mentor to ask for advices and can still continue to work on others. This way I won't have to wait for an answer and I can keep a good productivity.

The only plan made was on the first month, it will be possible to interact more with my mentor after this time so I need to have basic functionalities ready to decide with him how we will plan the rest of my internship.

Week 1-2:

Set the Php/MySQL server on Windows.

Study Kivy and use it to create the Login screen.

Create accounts database and implement communication between the app and server.

Decide what authentication method to use and implement it.

Week 3-4:

Create a Screen in the app for administrators to manage any accounts (creation, deletion, password reset).

Implement Php functions related to account management.

Set the explorer screen and the server side to get the content of a folder on the server and simple interactions, like the upload and download of files.

Week 5-6:

Decide what other file managing functions are necessary and implement them.

Now there are enough actions possible with the app to test it on Android, so Android portability to implement.

After the first month and half my mentor should have enough time to discuss what should be changed and what must be done.

The rest of the planning depends of this discussion and thus was not planned from the start.



## III – Application interfaces

### 3.1 Uniformity

The app is developed to work as well on PC as on mobile/tablet devices.

The HMI on a PC implies we can easily do two different actions on any component with the mouse left or right click while on a tactile device we are left with one action, touching a component.

That's why I chose to develop interfaces with buttons to click on when the action is simple (cancel, validate, open, etc). For more delicate use like selecting multiple files then perform an action on them, since there is no keyboard to do a combination the user will have to press a toggle button to lock the selection to do, this way when clicking on objects rather than opening files or moving to another folder we will just do a selection. The selection state activated will give the user access to other commands like delete, cut/paste, download.

This toggle solution has been chosen to keep the same interface for both platforms. Now that I know how will look the screens on our app, I can develop them then implement and test the communication between the app and the server.

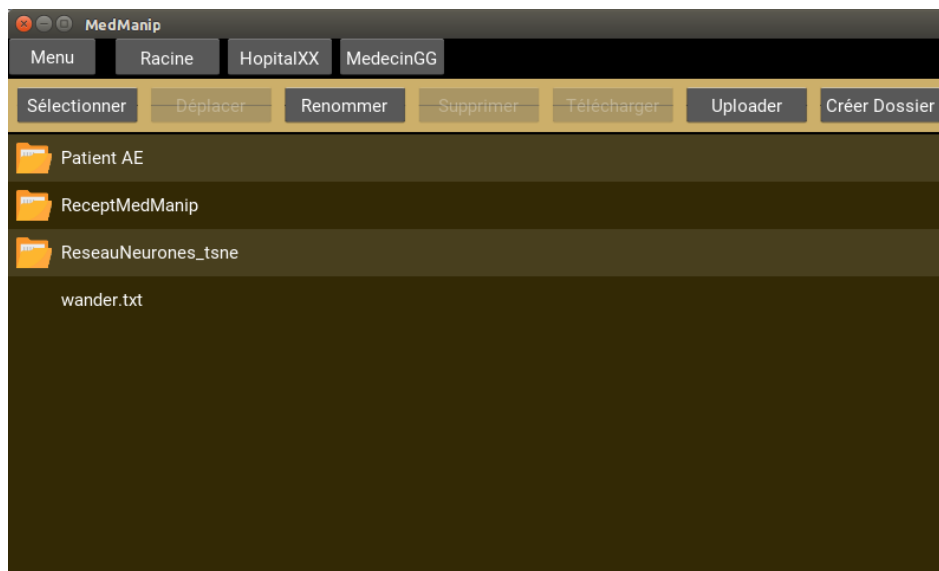


Figure 1 - Explorer screen. The select button is a toggle.

### 3.2 Kivy

#### *Hierarchy*

Since the available actions of a user are driven by the interface to which he has access, I decided to create a class for each interface. To keep the code logic as close as possible to the way the user will interact with the application, each instance of a class will create an instance of another only if an user can access to it from the related interface. For example, the following figure shows the Menu screen, from it we can go to 4 screens; Log in, Explorer, Settings and a temporary one to test 3D rendering. Thus, the instantiation of these 4 classes is done at the initialization of the Menu.

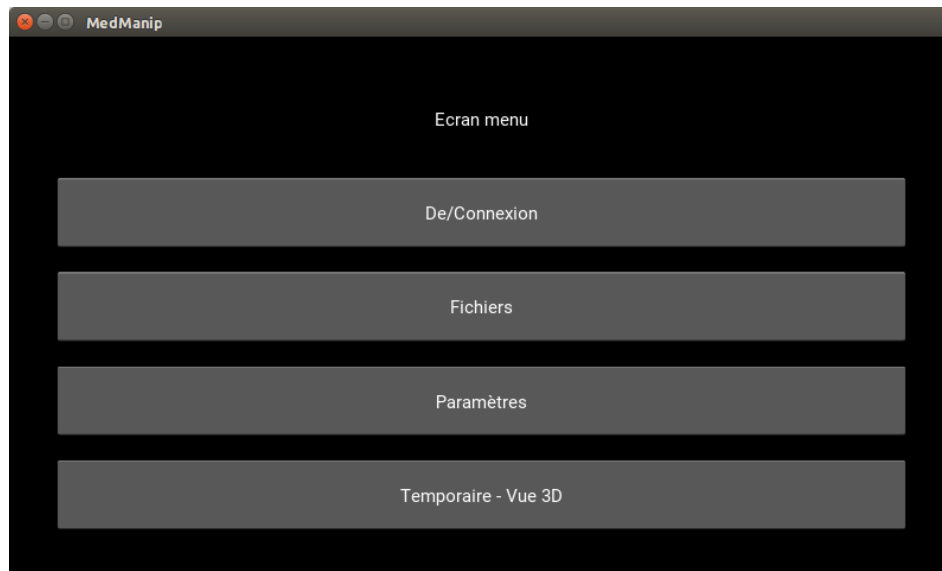


Figure 2 - Menu Screen

## Screens

Rather than having a classic view of my application with a menu bar at the top, since it's meant to be ported on a mobile, I kept the view light where each interface takes the full window plus a button to go back to its parent except for the Menu. That's why, for every class related to an interface, I chose to have them inherit from the Kivy class Screen. The Screen class is a widget intended to be used with a ScreenManager, its behavior with other widgets is the same as a RelativeLayout, by default it will take the whole available space from its parent. The ScreenManager being the class where we manage the Screens, I made my root class (VueRacine) a subclass of it.



Figure 3 - Screens and ScreenManager

An important information to know with the ScreenManager and its Screens is that even when you add a Screen widget to the manager (using add\_widget), it doesn't mean you will be able to access to the manager by calling the screen parent. Usually, the parent property of a widget is set when you

add another widget to it, the addition becoming its children [4], but with the ScreenManager there is a situation not referenced. We have to use the `add_widget` function to set a Screen as a child of ScreenManager but the ScreenManager manages its children by itself and only has one child at once, unsetting the other and revoking the parent property. Hence, if we are not careful when accessing the parent property, it could crash the application because we try to access functions or property of a None value.

### *Popup alternative*

There is a widget called Popup in Kivy which, as its name suggests, gives us a way to obtain a widget over the current view. This is especially useful to show a confirmation window. However, its use means we don't instantiate it at the application start but on demand. In our situation, the project is light enough to not be a problem but this behavior should be avoided when possible. Indeed, with a heavier content, showing a window using Popup could impact the performance due to the on-the-fly generation of its content. An alternative, implemented before knowing the widget Popup exists, is to create a float widget above the others and to instantiate it at the application start. To hide it, we only need to move it far off screen rather than destroying it as it is the case for Popup. The downside of this method is that, since we don't want a delay when showing this widget, its content stays loaded in the RAM for every similar widget which can end taking up a lot of space in the memory.

### *Relative scale*

There are various ways to set a widget size in Kivy, we can specify it as a percentage of its parent widget (the root widget is the application window) or use a unit. The available units are:

Pixels (px), not a physical size since it depends on our screen pixels size (the pixel density, not the screen definition).

Centimeter (cm), millimeters (mm) and inches (in) which are dependent of the physical size of our screen. If I set a 2 cm button, it will always be 2cm tall whether the screen is a 4 inches mobile phone or a 2 meters wide TV screen.

Density-independent Pixels (dp), an abstract unit based on the physical density of the screen. It's similar to the pixel unit but it won't be dependent of the screen pixels density, hence it's a unit similar to the previous physical ones but based on pixels, not meters or inches. This kind of pixel density related unit has become a must be replacement nowadays to the pixel unit. Since the appearance of Retina Display by Apple, we can observe a normalization of high-density screens for mobile platforms, so we can no longer develop interfaces assuming 30 pixels will occupy the same space in reality for any device [5].

Two other units exist in Kivy but they are not used for widgets, they are font oriented. Following the pattern of px and dp, respectively we have point (pt), a 1/72 of an inch, and scale-independent pixels (sp) to set the font size. The sp unit doesn't only adapt the size given a pixel density, it also takes into consideration the user's font size preference [6].

## IV – Server communication & security

### 4.1 Transferring data

First, our data must not be readable if the communication is intercepted so the HTTPS protocol will be used for everything.

There won't be a lot of apps communicating with the server at once so even though the server's load is slightly more important than with HTTP, it's still an affordable choice. When downloading files we will most likely be in a situation where a few big files will be downloaded over a lot of small ones. The Handshake [7] occurring in the HTTPS protocol is mostly the only thing increasing the bandwidth so it's better to pack all the small files as one archive if there are a lot of them. As long as we are in a situation with files sized over 500 KB, it is worth to download them individually. The time lost in compressing those files in one archive would be greater than the time necessary to transfer those small packets.

To port an application on Android, it will be necessary to enumerate every foreign packages used in a requirements section of Buildozer, a Kivy tool. Buildozer will use another tool to adapt those packages if necessary but a simple pip install is often not enough to install them. If there is no recipe for a package I will have to create one, which can take a lot of time. That is the main reason why I avoid importing packages when I can in this project.

To keep as few modules as possible in my code, I chose to use the `UrlRequest` [8] function from Kivy to connect to the server in HTTPS rather than import one easier to use. The advantage of writing smaller and cleaner code possible with other packages is not worth the effort of trying to adapt a recipe to have it working on Android. Also, those packages offer always more than just an HTTPS connection to a server, meaning I would add to the app a whole package only for one function when I can already perform the same action with the base package, Kivy. I think it's not a good habit to act this way, if I do this every time I think a function is too heavy I will quickly have an application way bigger than it should be. For this project it wouldn't be such a problem but when developing an app intended to a public online store, when there are 2 similar apps people will often go to the lightest one.

`UrlRequest` is a function using the `httplib` module from Python 2 which has been renamed to `http.client` in Python 3. SSL communications are necessary to use HTTPS and as specified in the documentation [9] "HTTPS support is only available if the socket module was compiled with SSL support", which can become a problem with Android.

### 4.2 Storing data

We wish to keep the personal data stored as succinct as possible. A login and password are required but the name and surname are facultative. Since this application provides a restricted service, the medical staff should be known by the laboratory member so storing personal information such as their address, mail and phone number is seen as redundant.

The database stores personal information on each patient referenced by the doctors associated to uploaded medical data. Those data must be protected, so an anonymization process has to be developed. Still, the doctor will use the result we will provide to treat his patients, hence he must be

able to identify anonymized data. A random number is assigned to these files while the personal data are encrypted with a symmetric encryption algorithm (blowfish) to avoid any leak in case of hacking on the database. The symmetric part will ensure the possibility for the doctor to deanonymize the data he uploaded.

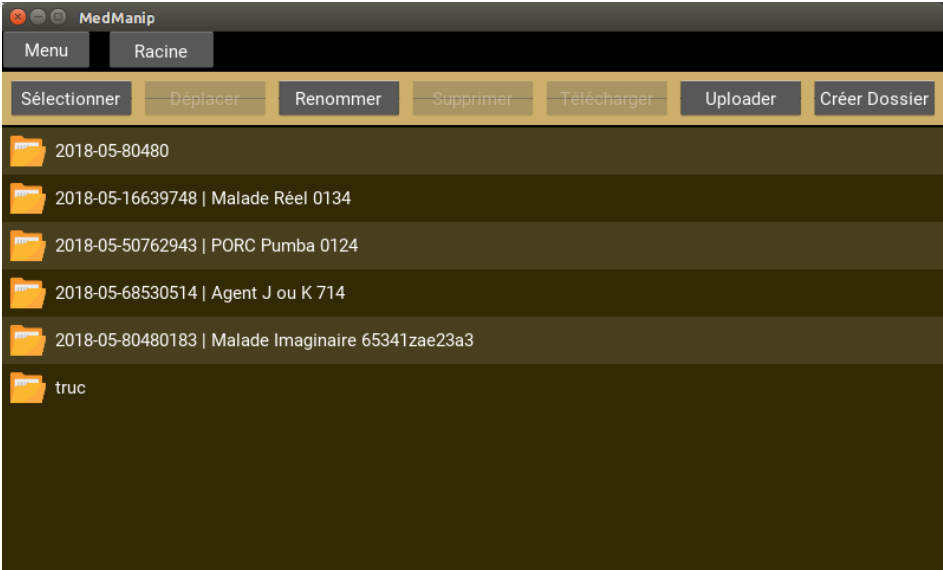
The other sensible data stored in the database is the account password. The password will be sent through HTTPS to the server but a hash will be stored. An asymmetric encryption algorithm, bcrypt, based on blowfish will be used with the Php function `password_hash()` [10] which uses, by default, the latest secured algorithm available. Hence upgrading the Php server should suffice to keep the security updated. The choice of an asymmetric encryption algorithm is because we don't wish, for anyone, to be able to find the original password from the encrypted version. The hash is the most common procedure to store passwords securely.



The screenshot shows the 'Gestion du compte' section of the MedManip application. It features a table with five rows of patient data. Each row includes a patient ID, a role (Malade, PORC, éléphant, Agent, Malade), a status (Réel, Pumba, Trompé, J ou K, Imaginaire), a date (2018-05), and a unique identifier. A trash icon is present at the end of each row. The interface also includes buttons for 'Changement de mot de passe', 'Ajout de patient', and 'Liste des patients'.

Paramètres		Gestion du compte		
Changement de mot de passe		Ajout de patient		Liste des patients
0134	Malade	Réel	2018-05-16639748	
0124	PORC	Pumba	2018-05-50762943	
	éléphant	Trompé	2018-05-56830225	
714	Agent	J ou K	2018-05-68530514	
65341zae23a3	Malade	Imaginaire	2018-05-80480183	

Figure 4 - Anonymized patients



The screenshot shows the 'Menu' section of the MedManip application. It displays a list of files that have been decrypted for doctors. The files are organized into folders, each containing a date and a list of patient identifiers. The interface includes buttons for 'Sélectionner', 'Déplacer', 'Renommer', 'Supprimer', 'Télécharger', 'Uploader', and 'Créer Dossier'.

Menu		Racine
Sélectionner	Déplacer	Renommer
Supprimer	Télécharger	Uploader
Créer Dossier		
	2018-05-80480	
	2018-05-16639748   Malade Réel 0134	
	2018-05-50762943   PORC Pumba 0124	
	2018-05-68530514   Agent J ou K 714	
	2018-05-80480183   Malade Imaginaire 65341zae23a3	
	truc	

Figure 5 - Anonymized stored data decrypted for doctors.

### 4.3 Authentication

To send the password in plain text at every action, even if it's encrypted with the HTTPS protocol, is definitely not a good idea. That's why we chose to use a token delivered and updated regularly by the server. Once an authentication with the password is done, a secured random hash is created in Php (using `openssl_random_pseudo_bytes` [11]).

The app asks, at a fixed interval, a new token to the server. If the server doesn't get this request after a time longer than the fixed interval, then the token is cleared from the database and the client will have to log in once more.

There is a lot of function in Php to generate pseudo-random strings or number but most of them are not secured. Clear explanations are often available in those functions comments on the Php documentation pages. It appears that most of them have an uneven random distribution making them a wrong choice as a salt generator in cryptology. That's the reason why I used a Php function from the OpenSSL module, a reputable cryptography library. This module is included by default since Php 4.0.5 [12].

### 4.4 Account management

It is necessary to give access to more actions to researchers than to doctors, such as deleting files, downloading files from any doctor and others. Thus, 2 kind of accounts have been made on the database, "admin" and "limited" referring to their rights. An admin account can act on any other account, they are made for researchers and they can create new accounts, reset passwords, lock accounts and delete accounts.

The lock account feature is meant to avoid security issues. We wouldn't want a new user to use the login of an account used previously by someone else but forgotten because it was deleted rather than lock. The non deletion of this former user files or data recovery mistake due to a backup would give access to personal data from another doctor, hence the implementation of a lock functionality.

Still the deletion feature has been implemented in case of mistake when creating a new account, it's not meant to be used to get rid of an old account. The validation step doesn't ask confirmation for this feature but rather asks the user to type the login of the account he wishes to delete, I believe it's a better solution to ensure the user know what he is doing, indeed it sometimes happen that we click the wrong button on a confirmation window due to panic or lag from the application hence a confirmation by typing is less risky.

Gestion des comptes					
/\ Supprimer un compte		---		Liste des comptes	Création de comptes
ABC12	HUBERT	Cousin	Limité	Modifier le mot de passe	Compte activé
ADMIN18	Mon nom		Administrateur	Modifier le mot de passe	Compte activé
AGAIN	let's	try	Administrateur	Modifier le mot de passe	Compte activé
ARBRE	fruit	thé	Administrateur	Modifier le mot de passe	Compte activé
AZRTZZ			Limité	Modifier le mot de passe	Compte activé
AZRTZZAAZA			Limité	Modifier le mot de passe	Compte activé
AZRTZZAAZARRAZT			Limité	Modifier le mot de passe	Compte activé
AZRTZZAAZAZRAZT			Limité	Modifier le mot de passe	Compte activé
RTZZAAZAZARRAZT			Limité	Modifier le mot de passe	Compte activé

**Figure 6 - Account management**

#### 4.5 Algorithm

Blowfish [13] is a symmetric cryptographic block cipher designed in 1993. It successfully resists to every kind of attack known. The only way to break an encryption is to enumerate every possible key, which takes much more time than for the AES because of the S-Boxes initialization.

Blowfish initialization takes more time than AES because its S-Boxes are key dependent. It needs to generate 4 KB of encrypted data every time the key is updated, which takes a really short time (a few milliseconds at worst) so it's not a problem for a standard user. However, in a key-exhaustion attack situation, having this delay for each key generated will greatly increase the time needed to decrypt the encrypted data, improving significantly the security compared to the AES.

## V – Medical data 3D visualization

### 5.1 Marching Cube

The Marching Cube algorithm was developed by Cline and Lorensen in 1987. Its purpose is to extract a polygonal mesh from an isosurfaces described in a 3D domain but a full explanation of this algorithm would take too much time.

This algorithm relies on the occurrence of similar cases by symmetry when going through the isosurface. At each step, a cube moves in the 3D field and the values collected at the cube 8 vertices determine if each vertex is inside or outside the object described by the isosurface. To define the object mesh, the only cases we are interested are when the cube location is on the object surface, which means at least one of the vertices has a value different from the others. This way there are  $2^8$  different situations but we can see similar ones when rotating the cube, as such we end with only 15 different cases as shown in the following figure.

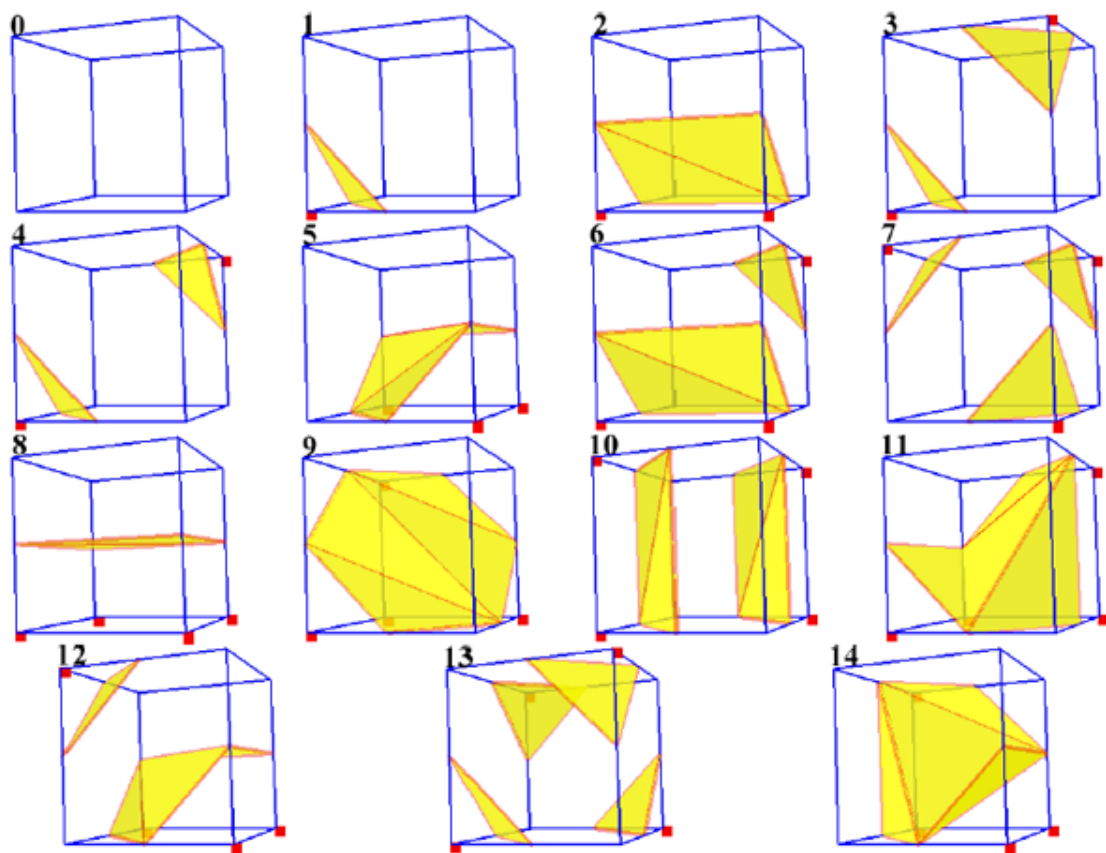


Figure 7 - Original 15 cases of Marching Cubes

The Lewiner version [\[14\]](#) corrects face ambiguities happening for some identified cases, thus we can decline 32 geometrically situations from the 15 original cases. I used an implementation of the Lewiner algorithm in Python freely available in the scikit-image toolbox on Github [\[15\]](#). A Cython module of Lewiner is included and has to be kept to greatly improve this algorithm performance, it calls back and forth from and to C or C++ code natively at any point.



## 5.2 OpenGL with Kivy

Kivy offers a Python module wrapping OpenGL commands. Unfortunately, not every command has been wrapped and the documentation is really scarce.

OpenGL is used in this project to visualize human organs. It needs the mesh made with Marching Cube but for now it is only implemented to ensure having it working on Python and Android. Indeed, the new kind of medical files from the hospital are still not available, hence we cannot set it not knowing how the isosurfaces will be organized.

To greatly increase the framerate when using OpenGL with Kivy, I used the module Framebuffer [16] (fbo) from Kivy. It is associated with the Canvas to render a texture generated by OpenGL and manages the viewport by itself. Since Kivy already uses OpenGL to generate everything we see on the screen, it's helpful to use the Fbo to avoid having to manage the graphic context. Indeed, if we are not careful with the context, by example, setting the viewport of our drawing could change the whole view rather than the one we wish for since OpenGL is used by Kivy for every widget.

It was not necessary for this project but I still developed the scene rotation during the rendering following a method often observed in video game development. The delta time is the time spent between the last frame and the current frame, in Kivy it's available through the Clock module and called the frametime. By taking into account this value to modify the rotation angle at each step, I can ensure the same rotation speed on any device. The step being linked to the framerate, we can easily conclude that if we rotate the scene by a fixed value, then the rotation speed becomes a factor of the framerate, which we don't want. If we don't implement the delta-time, when we run the application on an old device, the low framerate will show an object rotating really slowly while a newer device will make it rotate too fast. We can even imagine what would happen in the next decade with devices presenting a way higher framerate than nowadays, clicking on a button to slightly rotate the scene would make it rotate too fast to have it how we wish.

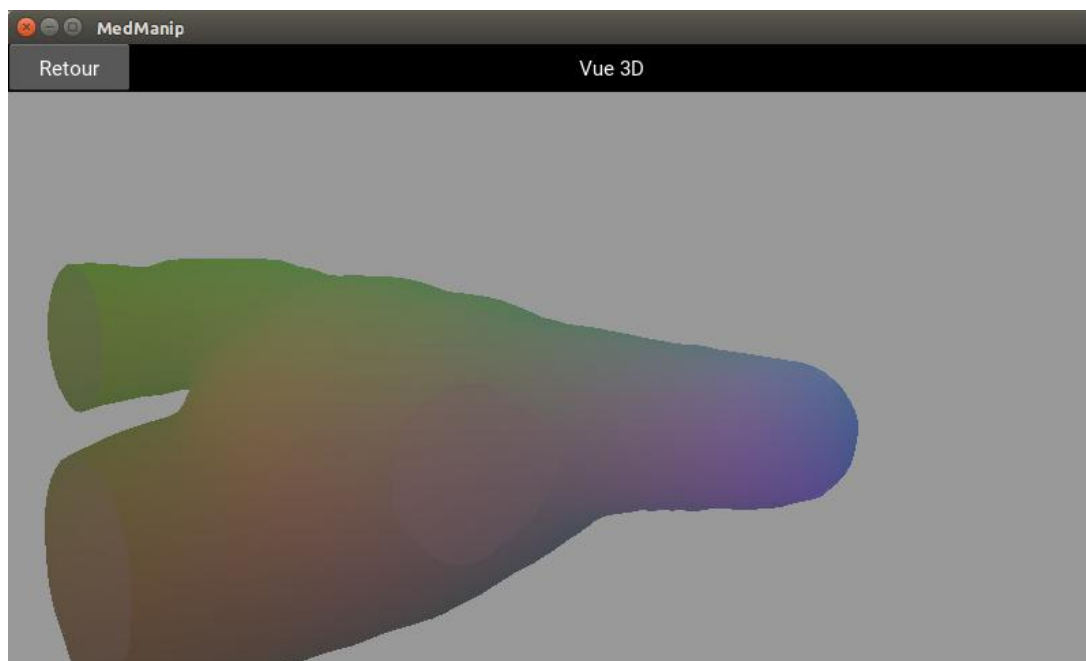


Figure 8 - Isosurface transformed into a mesh and rendered with OpenGL.

## VI – Technical problems related to Android

The application portability was not done after implementing all the functionalities but during the development. The first tests were done on the PC then on the mobile. In the end, it appears testing the portability early was a wise choice since the portability of modules in Python 3, requiring the access to mathematical libraries, presents problems not yet resolved by the Kivy team, ending with the obligation to rewrite everything to have it compatible with Python 2. It could be due to the use of the Crystax NDK, where those libraries are either not well linked or not compiled for the correct platform.

## 6.1 Kivy tools

To end with an application using the latest technologies available, everything was written in Python 3 at first even though we knew the support of Python 3 with Kivy's module P4A (Python for Android) was in Beta.

Once the authentication and basic explorer actions implemented, the app was ported to Android. Two tools were used: Buildozer [\[17\]](#), a Kivy tool used to create application packages and Crystax [\[18\]](#), a custom Android NDK (Native Development Kit) necessary to get Python 3 working on Android.

## 6.2 HTTPS

A major difficulty was when the app tried to connect to the server. On Buildozer when you need specific modules like OpenSSL you must add them to a file as requirements but in this case it didn't change anything. That's because, as specified in the Python documentation, some part of Python must be compiled with SSL support.

Indeed, Python 3 in the Crystax NDK is pre-built so to save time during the compilation, Buildozer uses it and this part didn't have the socket module with SSL support, hence HTTPS connections couldn't work.

On Kivy's GitHub, a solution [\[19\]](#) is available since January 2018 but it still isn't merged to the master branch. There are 2 important things to do; first we have to download the master branch on the computer because the patch is based on it, not on the stable, if we look at the project history. Secondly, we update it by modifying 3 files. The solution ignores the prebuilt version of Python 3 before building it and indicates where missing libraries in the OpenSSL recipe are.

## 6.3 Numpy

The second major difficulty was when asked to implement Numpy with Kivy on Android. It's working with Python 2 only since the stable version 0.6.0 released on 25 Nov 2017 so it's a recent problem [\[20\]](#). Unfortunately the flags working with Android NDK fail with Crystax's but only if I use the Python 3 in Crystax, it works if I use the Python 2 of this NDK.

It was possible to resolve this problem by going back to Python 2 but since I already had a lot of work done, I could take one week to find why Numpy doesn't work and try to find a solution. I did a lot of discoveries [\[21\]](#) and resolved most problems by adding flags but it wasn't enough. The last situation was about .so math libraries incompatible with Numpy, it could be because it wasn't built with the same compiler [\[22\]](#) used when building Numpy or because of a 32/64 bit mismatch.

Eventually we ended going back to Python 2 to have Numpy available on Android.

## 6.4 Unicode

Everything has been made to be UTF8 compatible. In Python 3, it's not a difficulty but in Python 2 it easily becomes a big problem on Android. Sending any non-ASCII character will cause a sigsegv fault, crashing the app. Hence every `print()` call must be completed with the following instruction.

```
.encode('ascii', errors='replace')
```

Since the application works correctly on Android with Python 3 as long as we don't use Numpy, rather than updating everything a new branch was made on Git for Python 2.7.

## 6.5 Cython

As said earlier, Cython is necessary to improve performance when running Marching Cube. Usually a simple pip install is enough on PC but for Android it gets much more complicated.

Cython doesn't exist within p4a, there is no recipe for it and unfortunately it means we cannot run Cython files (extension `pyx`) on real time. Thus, the module import of the Lewiner version of Marching Cube is not possible since it needs the function `pyximport`, part of Cython. The solution is to do what Cython does in real-time but beforehand. Hence, I use Buildozer to build the C library corresponding to this file. The documentation of P4A is really poor when it comes to recipe creation, it is still possible to find information on internet to make recipes but I only found some about the old recipes, which were not in Python, and correspond to the "old toolchain" version of P4A. There were missing links to libraries of the python compiled for arm processors which I could only resolve by finding them in the build folder and linking the full path from the root folder for each file. I don't either found how to specify from which folder P4A must take the Cython file nor to which folder put the compiled version. Thus, I had to move the file by myself in the correct folders and run once again the compilation to have P4A add the `.so` file generated accordingly.

I'm not proud of this round away method, I could have make it better by using Python basic package functions to move files and find the missing libraries but I think it's better to use functions from the P4A toolbox. I could have done it properly if I had enough time to read source files from P4A to discover what functions exist but this happened at the end of this internship, the best I can do for know is explaining this DIY alternative.

## VII - Conclusion

### 7.1 Objectives achieved

Every basic feature asked for has been implemented to this application. A researcher can manage accounts, creating ones for doctors. Doctors can upload and download files on a personal folder only accessible by them and researchers. An easy to use online explorer was made to manage files. Doctors see in real time the medical data they uploaded deanonymized rather than random numbers. The application can open and render isosurfaces we can rotate to have a better understanding of what was recorded, since we don't know how the files to be processed will be like it could not be implemented when clicking on a file, but since the function has been made the code can be simply edited to link a click to a download and process of the desired isosurface.

The server could not be set in time due to availability problem with the DSIUN but a Git with the necessary files and explanations is available to set one.

I kept in mind the desire to have this project as generic as possible when evaluating solutions to problems in case other laboratories anywhere in France would need such a tool for their researchers. I still think it stressful when I kept looking for security failure which could be exploited to break into databases of other labs that use this application.

### 7.2 Improvement prospects

As for many projects, the list of functionalities we can add to the application is never ending. In particular, there is a case where I could have to study image analysis methods. It could be interesting if, within the app, we could take a picture of a patient file with the phone's or tablet's camera and directly obtain a new entry which can be associated to medical data. It would offer a complete solution to support a doctor when consulting a patient's file but it gets too far from the initial project so it wasn't done during this internship.

An improvement to the transfer file system can be done too, for now the files are small enough to not be a great danger but there was no hash comparison implemented after the file download, it's not difficult to have it working but since it wasn't vital its priority was inferior to other functionalities, thus there was not enough time to implement it. There is no possibility to interrupt a download or upload with quitting the application, the Kivy function `UrlRequest` uses a function from Python package to get or send files over the network but there is no handle available to interrupt a given action, even though the Python function has one. It could be resolved by editing the Kivy function but any update of Kivy may end with a compatibility problem, unless we ask for this functionality to be added to the Git project.

### 7.3 Personal thoughts

I already worked in a similar field within the IMATH lab during my Master first year so working with researchers is not a first experience for me. However, I could enjoy working once again with people who desire to share their discoveries with the world but can have really different mindset while still keeping this common ideal.

I hope the application I made will prove useful for researchers and doctors, it would be even rewarding if one day it gets popular enough to be improved and merged with professional medical tools.

## Appendix

Objectives	Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-Set Php/MySQL server on Windows 7 -Study Kivy and use it to create the Login Screen -Create accounts database and implement communication between the app and server -Decide what authentication method to use and implementation															
-Create an interface in the app for administrators to manage any accounts such as creation, lock, password reset. -Implement Php functions related to account management -Research on Php behavior with UTF8															
-Set the explorer screen and the server side to get the content of a folder on the server and simple interactions, like the upload and download of files or folders with their subfolders.															
-Decide what other file managing functions are necessary and implement them. -Implement Android portability.															
-DSIUN server test, installation incomplete. -Personal server on Windows adjustments to communicate with the app on mobile. -Interfaces design correction for Android.															
-SSL correction on Python for Android -Research on the reasons why P4A fails to integrate Numpy with Python 3															
-Code adaptation from Python 3 to Python 2. -Corrections to keep UTF8 encode everywhere. -Php bug corrected on file upload.															
-Account management for doctors. -Data anonymization functions implemented. -Patients personal data management for doctors.															
-Account deletion for Administrators. -Kivy' OpenGL tests.															
-Research of Marching Cube algorithm updates and implementations on Python. -Test of skimage module integration via P4A															
-Improvement of OpenGL code, way better framerate. -Adaptation of Marching Cube algorithm using Cython with P4A. -Create camera controls and automatic adjustment of the scene based on the 3D object dimensions.															

Table 1 – Gantt Chart

## References

---

- 1 FRUMAM, a mathematical research federation: <http://frumam.cnrs-mrs.fr/>
- 2 M&MoCS, an international research center: <http://memocs.univaq.it/?lang=en>
- 3 Imath lab: <http://imath.fr/>
- 4 Parent property in Kivy: <https://kivy.org/docs/api-kivy.uix.widget.html#kivy.uix.widget.Widget.parent>
- 5 Variety of pixel density screens: <https://material.io/tools/devices/>
- 6 Metrics within Kivy: <https://kivy.org/docs/api-kivy.metrics.html>
- 7 SSL Protocol see RFC[2246]: <http://www.ietf.org/rfc/rfc2246.txt>
- 8 URLRequest in Kivy: <https://kivy.org/docs/api-kivy.network.urlrequest.html>
- 9 HTTPS in Python: <https://docs.python.org/2/library/httplib.html>
- 10 Hash with Blowfish: <http://php.net/manual/fr/function.password-hash.php>
- 11 Pseudorandom generator: <http://php.net/manual/fr/function.openssl-random-pseudo-bytes.php>
- 12 Php version for OpenSSL: <http://php.net/manual/fr/openssl.requirements.php>
- 13 Blowfish algorithm: [https://www.schneier.com/academic/archives/1994/09/description\\_of\\_a\\_new.html](https://www.schneier.com/academic/archives/1994/09/description_of_a_new.html)
- 14 Marching Cube by Lewiner: [http://thomas.lewiner.org/pdfs/marching\\_cubes\\_jgt.pdf](http://thomas.lewiner.org/pdfs/marching_cubes_jgt.pdf)
- 15 Lewiner on Github: [https://github.com/scikit-image/scikit-image/blob/master/skimage/measure/\\_marching\\_cubes\\_lewiner.py](https://github.com/scikit-image/scikit-image/blob/master/skimage/measure/_marching_cubes_lewiner.py)
- 16 Framebuffer: <https://kivy.org/docs/api-kivy.graphics.fbo.html>
- 17 Buildozer by Kivy: <https://github.com/kivy/buildozer>
- 18 Android NDK with Python 3: <https://www.crystax.net/android/ndk>
- 19 P4A OpenSSL: <https://github.com/kivy/python-for-android/pull/1217> - <https://github.com/kivy/python-for-android/pull/1195>
- 20 P4A Numpy with Python 2: <https://github.com/kivy/python-for-android/issues/1074>
- 21 P4A Numpy with Python 3: <https://github.com/kivy/python-for-android/issues/882>
- 22 Numpy build requirements: <https://docs.scipy.org/doc/numpy-1.9.2/user/install.html#fortran-abi-mismatch>