Mobile application to support nurses’ workflow



Bachelor thesis presented by

**Argentin Yvann**

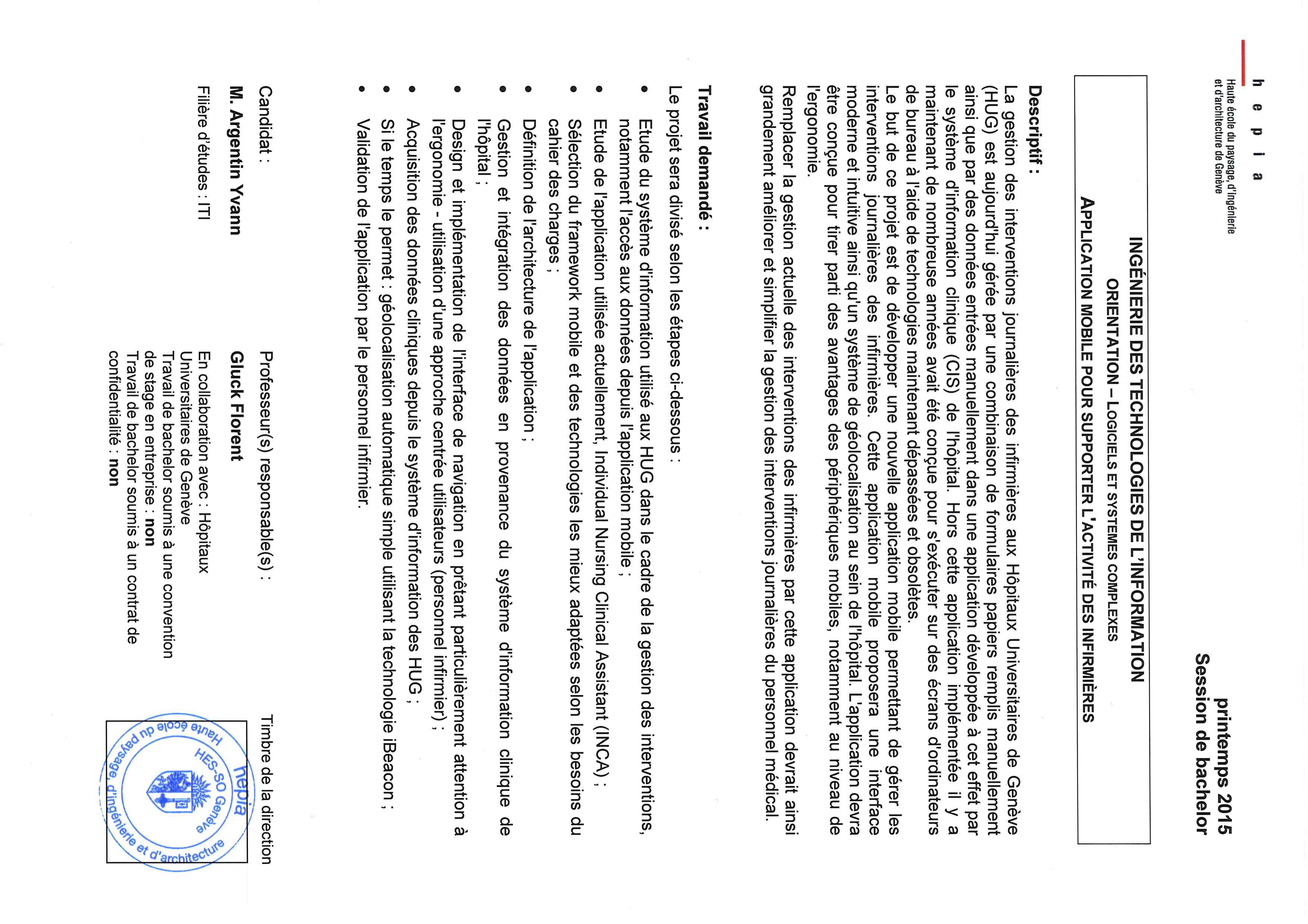
To obtain the title Bachelor of Science HES-SO in

**Information technology engineering with a specialization in**

**Software and complex systems**

***September 2015***

|  |  |
| --- | --- |
| Supervising Professor  **Florent Glück** | Supervisor at Geneva University Hospitals  **Frederic Ehrler** |



**Résumé :**

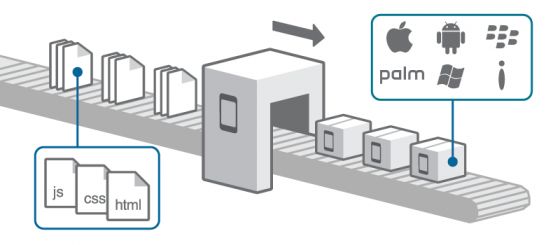
A l’ère de l’informatique actuelle, rare sont les métiers qui n’incluent pas l’intervention d’un système informatique. Le maximum d’information est informatisé et accédée par le biais d’ordinateurs ou de smartphones.

Les infirmières des HUG (Hôpitaux Universitaires de Genève) utilisent actuellement une application bureau leur permettant de consulter et imprimer les informations médicales des patients, le fait que cette application ne soit disponible uniquement sûr bureau obliges les plus de 500 infirmières qui travaillent en même temps à imprimer une grande quantité de papier chaque jour. En plus du papier, chaque opération effectuée par les infirmières se doit d’être notée et informatisée, ce qui oblige les infirmières à noter tout leur travail sur des feuilles de papier pour ensuite le retaper dans le système informatique des HUG.

Ce projet proposé par les HUG vise à développer une application mobile facilitant le travail des infirmières tout en limitant le risque d’erreurs de saisie. Ladite application mobile permettrait aux infirmières d’accéder aux informations patient et de saisir toutes les opérations effectuées lors de leur temps de travail, tout cela depuis un smartphone fournis par les Hôpitaux Universitaires de Genève

Ce travail propose une solution basée sur des technologies web permettant le développement d’une application native hybride (Android, IOS, …) produisant un résultat très similaire à une application native.

Les technologies mise en œuvre dans ce projet permettent d’encapsuler une application web dans un navigateur. Grâce à ce concept, il est possible de construire une application mobile native contenant un navigateur web qui affiche l’application web en plein écran faisant ainsi croire à l’utilisateur qu’il interagit avec une application native plutôt qu’une application web. Cette technologie (Apache Cordova) permet également d’accéder à des composants matériels des smartphones depuis une application web via un set d’API JavaScript.



: Professeur(s) responsable(s) : Timbre de la direction

**Argentin Yvann Gluck Florent**

Filière d’études :

HUG

Travail de bachelor soumis à une convention

de stage en entreprise :

Travail de bachelor soumis à un contrat de

confidentialité :

**Foreword**

Document’s structure

Before describing the structure of the document, it is important to know that this bachelor project was developed in collaboration with the Universitary Hospital of Geneva (HUG).

This thesis is separated in several parts; it starts with a description of the context of the project, in particular, what the initial issue is and why a better solution is needed. Additionally, a description of parts of the informatics architecture at HUG relevant to this project is given.

Next, I describe what technologies were used to reach the overall goal, along a description of the nurses' new workflow. This first part ends with a description of the restrictions and issues when working remotely with sensitive data and what the main challenges were.

The second part is an in depth description of all the technologies and protocols used. First, each piece of technology is described separately. Then, I explain how these technologies are used together to produce a native application for both Android and IOS**.**

The third part of the thesis describes the architecture of the application with many illustrations and schemas to help understanding every component of the application. Furthermore, a detailed description on how the HUGpatient’sdata are structured, how I accessed, manipulated and displayed them to ensure the best user experience is given.

Lastly, there will be a discussion on possible enhancements to the application, mostly in terms of user experience along a description of the issues and problems encountered. The thesis ends with concluding remarks on the whole project.

Typographic conventions used in the document

To allow an easier understanding and readability of the document, some text formatting rules are applied throughout the document:

* Names, acronyms and company names are in italic.
* Important words are in ***bold***.
* References to code and/or file/folder are written with the Courier New font.

Special thanks

I would like to start by thanking Mr. Glück Florent for his guidance and help during the entire thesis.

I would also like to thank Mr. Ehrler Frédéric for his availability, help and advice, as he was my main source of information for everything HUG related. He also provided me with a desk in their research group at the HUG.

1 Introduction 8

1.1 Context 8

1.2 IT architecture for nurses 9

1.3 Proposed solution 11

1.3.1 Existing application 11

1.3.2 Requirements 15

1.3.3 Used Technologies 16

1.4 Restrictions 16

1.5 Challenges 17

2 Framework 18

2.1 Angularjs 19

2.2 Ionic 23

2.3 Apache Cordova 27

2.4 iBeacon 30

2.5 OAuth 2.0 31

3 Application 34

3.1 Global architecture 34

3.2 File hierarchy 36

3.3 Navigation 38

3.4 Wireframes and functionalities 39

3.5 Patients data structure 45

3.6 Data transmission 46

3.7 iBeacon 46

3.8 Geolocalisation 47

4 Discussion 49

4.1 Performances 49

4.2 Issues encountered 50

4.3 Future work 51

5 Conclusions 52

6 Appendices 53

7 References 54

FIGURE 1.1 - INITIAL INCA APPLICATION 8

FIGURE 1.2 – NURSES’ CURRENT WORKFLOW 9

FIGURE 1.3 - OVERALL DPI COMMUNICATIONS 10

FIGURE 1.4 – NEW NURSES’ WORKFLOW ASSISTED BY THE INITIAL MOBILE TACTICLE INCA APPLICATION 11

FIGURE 1.5 - FIRST INCA APPLICATION'S STRUCTURE 12

FIGURE 1.6 – INITIAL INCA PROTOTYPE’S MAIN VIEW 15

FIGURE 1.7 - TOUCH GESTURE EXAMPLES 16

FIGURE 2.1 –THE STACK OF SOFTWARE TECHNOLOGIES USED IN THE PROJECT 18

FIGURE 2.2 – ANGULAR’S ARCHITECURE 19

FIGURE 2.4 – INITIAL ANGULARJS APPLICATION DATA TRANSMISSION 21

FIGURE 2.5 – DATA TRANSMISSION ON VIEW CHANGE 22

FIGURE 2.6 - UI-ROUTER AND UI-VIEW 25

FIGURE 2.7 - UI-ROUTER AND INHERITED RESOLVED DEPENDENCIES 26

FIGURE 2.8 - COMPARISON BETWEEN NGROUTE AND UI-ROUTER FOR IONIC 27

FIGURE 2.9 – APACHE CORDOVA’S LOGIC 28

FIGURE 2.10 – APACHE CORDOVA’S LOGIC DETAILS 29

FIGURE 2.11 – ESTIMOTE BEACON BREAKDOWN 30

FIGURE 2.12 – BEACONS RANGING SYSTEM 31

FIGURE 2.13 - OAUTH2 PROTOCOLE 33

FIGURE 3.1 – OVERALL HIGH LEVEL FUNCTIONNING DESCRIPTION OF THE NEW INCA APPLICATION 34

FIGURE 3.2 - OVERALL MEDIUM LEVEL FUNCTIONNING DESCRIPTION OF THE NEW INCA APPLICATION 35

FIGURE 3.3 - OVERALL DETAILLED MEDIUM LEVEL FUNCTIONNING DESCRIPTION OF THE NEW INCA APPLICATION 36

FIGURE 3.4 - FILE HIERARCHY 37

FIGURE 3.6 - UNITS STATE INTERFACE 40

FIGURE 3.8 - PATIENTS' STATE VIEW 40

FIGURE 3.9 - INTERVENTION STATE'S VIEW 41

FIGURE 3.10 - RESERVES STATE'S VIEW 41

FIGURE 3.11 - RESERVES DETAIL STATE'S VIEW 42

FIGURE 3.12 - VITALS STATE'S VIEW 43

FIGURE 3.13 - VITALS FULL HISTORIC STATE'S VIEW 44

Figure 3.14 - PATIENTS' DATA STRUCTURE 45

FIGURE 3.15 - DATA TRANSMISSION PROTOCOL 46

FIGURE 3.16 - BEACONS ORGANIZATION 47

FIGURE 3.17 - HOSPITAL ROOMS EQUIPPED WITH BEACONS 47

# Introduction

## Context

At the Geneva University Hospitals, nurses use a desktop application to help them with their everyday work as illustrated on Figure 1.1.

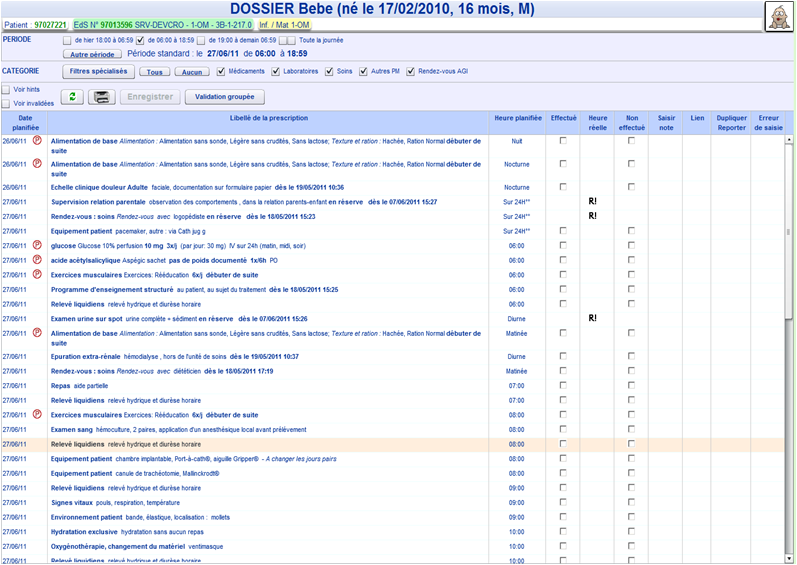


FIGURE 1.1 - INITIAL INCA APPLICATION

Nurses have complicated changing schedules when working at the hospital. Let's consider a fictional nurse and let's name him Francis**.** On a typical working day, Francis must take a set of actions illustrated on Figure 1.2 and listed below :

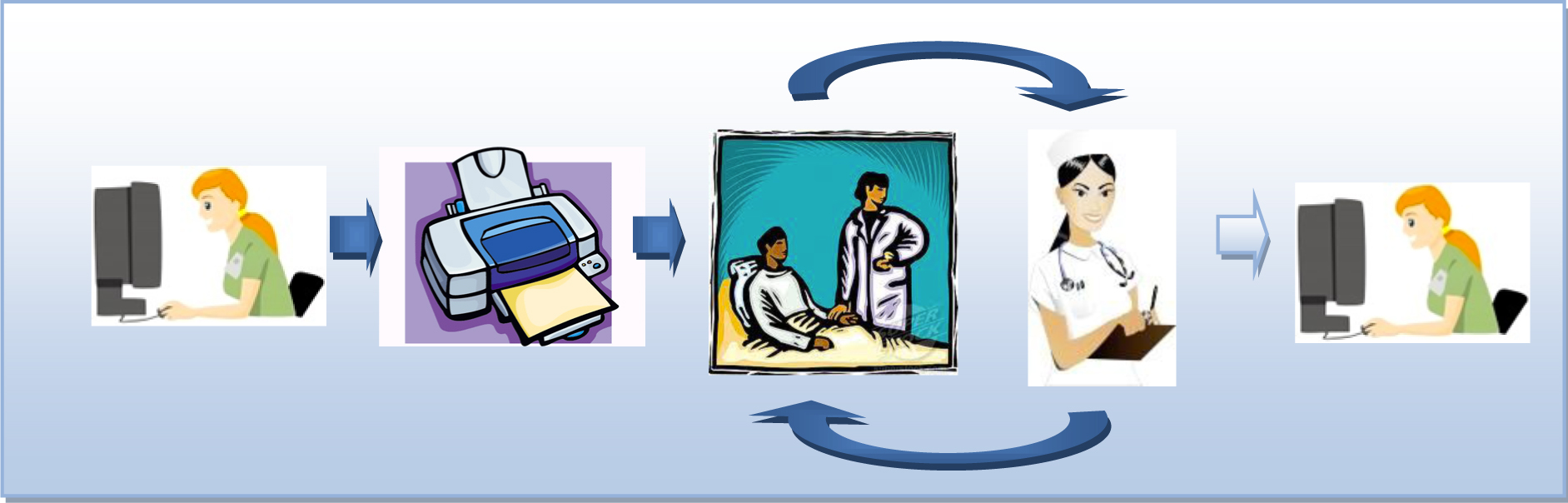


FIGURE 1.2 – NURSES’ CURRENT WORKFLOW

1. Francis logs into the desktop application and prints a sheet for every patient he has to visit today.
2. Francishas now a lot of paper with him; he chooses which paper sheets he will use shortly and visits the patients in different rooms.
3. He notes on those sheets everything he does, when he does it and how the patient reacted if necessary.
4. Once he’s finished, he goes back and takes the next sheets for the next group of patients he will visit.
5. Francishas a lot of paper to take care of.
6. Francistakes every sheet one by one and rewrites their content into the computer system**.**

Taking apart the fact that this is an old fashion way of working, this workflow presents some very important issues that should be fixed.

Francis and his co-workers have different way of writing, sometimes they’re tired and just don’t write as well as when they started working. They don’t watch the exact time for every intervention they perform, which means data can be inaccurate and in the medical field, accuracy is paramount.

In addition, Franciscan mistype or misread something and enter wrong data into the system. There’s a double risk for errors and that puts a lot of pressure on Francis’.

The hospital desires something that would increase the accuracy of both measures and typing/reading, allowing almost real-time data synchronization for all the working employees on a medium much less volatile than paper.

## IT architecture for nurses

The hospital hosts many servers storing not only patient’s data, but also data related to the staff, such as nurses. In the case of our nurse Francis, after connecting to the system, he receives a list of all the patients he must visit during his current shift.

To read data from the Hospital’s servers, Francis needs to use a trusted machine that will connect to a proxy service in order to ask for the “Dossier Patient Intégré” / “Integrated Patient File” (DPI). The whole procedure is shown on Figure 1.3.

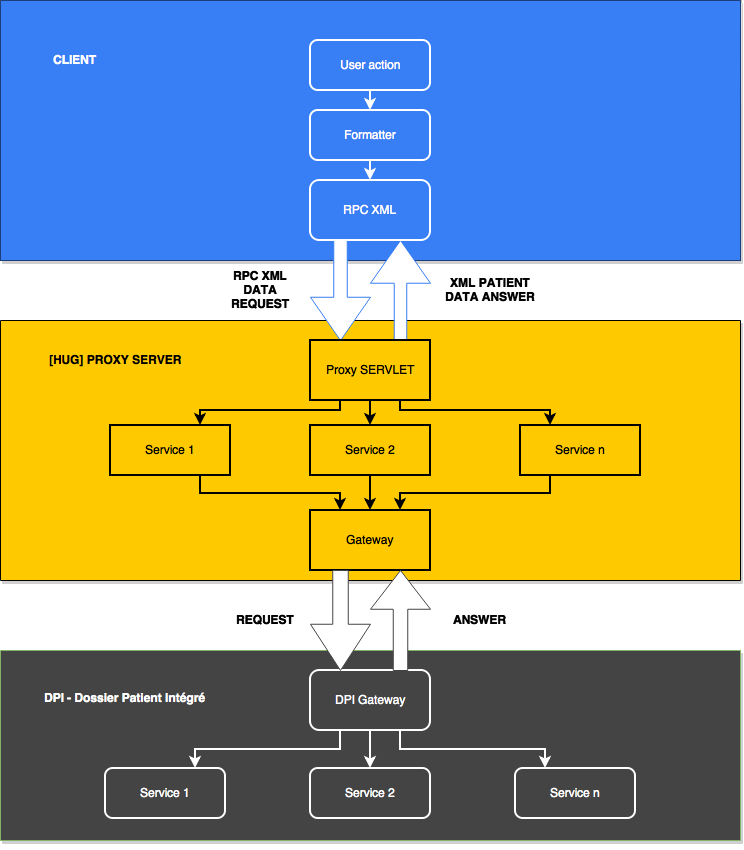


FIGURE 1.3 - OVERALL DPI COMMUNICATIONS

To get access to the proxy, the user must be authenticated on HUG’s authentications servers to get a token allowing the application to query the proxy through remoteprocedurecall(RPC) XML**.**

## Proposed solution

### Existing application

#### Introduction

Mr. Frederic Ehrler and his team have developed a prototype of mobile application to help enhance the nurses’ workflow as illustrated in Figure 1.4 called project individual nursing care assistant (INCA). They narrowed the number of steps down to two through the help of a smartphone application.

It allows Francis(the nurse)to visualize all the interventions he must perform during his shift, validate them, take vitals measurements in real time and enter them into the system. Such an application would greatly ease the work of *Francis*, by allowing him to bypass the previous mandatory extra steps of printing – carrying all patients’ sheet with him – taking notes – and entering it again into the system.

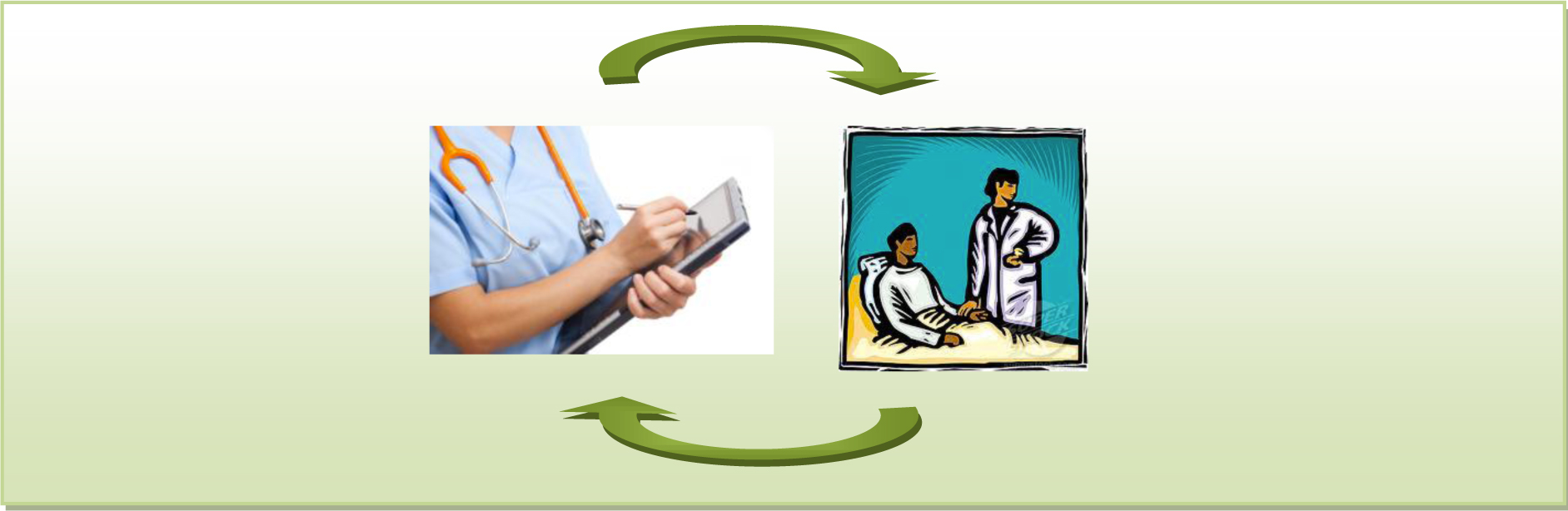


FIGURE 1.4 – NEW NURSES’ WORKFLOW ASSISTED BY THE INITIAL MOBILE TACTICLE INCA APPLICATION

#### Processus

The overall application is client centered and assumes a nurse takes care of one patient at a time. Thus, all the interventions are organized by patient. Patients are grouped by room and rooms are grouped in units. The aim of this structure is to allow fast and intuitive navigation from one patient to another.

The application’s navigation system is shown in French, as the application is in French, on Figure 1.5.

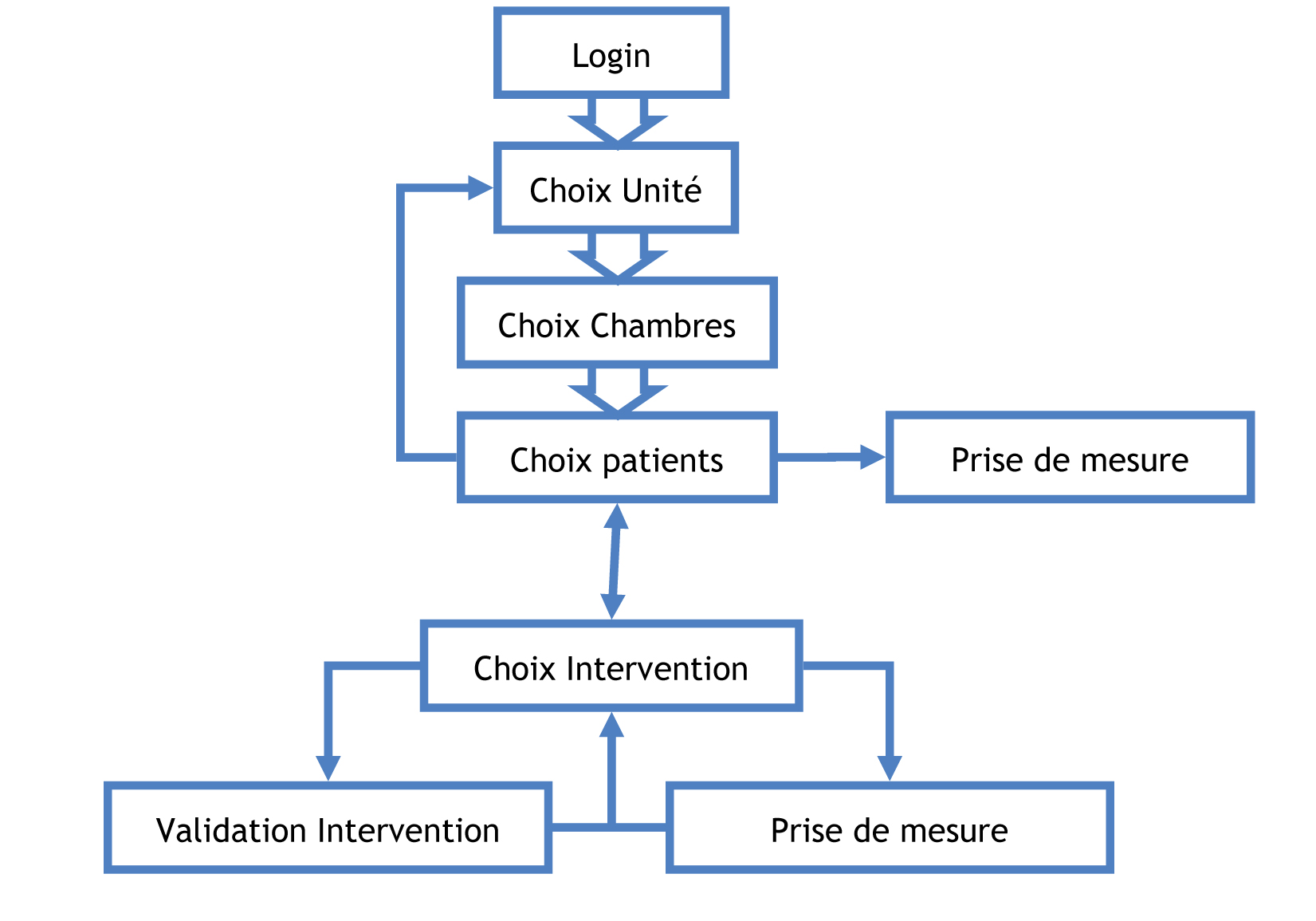


FIGURE 1.5 - FIRST INCA APPLICATION'S STRUCTURE

Every block in the figure represents an interface or a view.

**Login:** User logs in and is identified.  
**Units:** Once identified, the user chooses which unit he wants to work with.  
**Rooms:** Nurses are not responsible for the whole unit, thus they can choose the room they want to work with.   
**Patients:** The nurse chooses the patient he/she wants to help.  
**Interventions:** Once a patient is selected, a list of all interventions is displayed; the nurse can then choose which one to perform first.   
**Intervention validation:** Depending on the intervention type, a simple or more complete validation will be required.  
**Measurements:** Some interventions require measures to be taken; they will be entered in real time in the application.

#### Terminology

Interventions

All the interventions are described in a separated file and organized hierarchically which eases the displaying process. There are 22 "high level" interventions categories. They are listed below in French given the application's language is French:

* Alimentation
* Cognition-Perception
* Communication
* Développement – Concept de soi
* Environnement socio-familial
* Sommeil-Repos
* Spiritualités
* Gestion de la santé
* Hygiène
* Mobilisation
* Peau & Téguments
* Respiration
* Thermo-neurorégulation
* Enseignement
* Examens
* Bilans
* Surveillances
* Traitements
* PRESCO
* Equipement
* Rendez-vous

Vital signs

Vital signs represent all measurements done on the patients in order to know his/her health state.

All measures are taken in person, which is why it’s very important that nurses can enter those measures right after they’ve been performed.

Plenty of different measures can be taken, such as (again, in French):

|  |  |
| --- | --- |
| Le pouls   * Rythme   + Non précisé   + Irrégulier   + Régulier   + Autre * Lieu   + Non précisé   + Radial gauche   + Radial droit   + Rétro-malléolaire gauche   + Rétro-malléolaire droit   + Pédieux gauche   + Pédieux droit   + Fémoral gauche   + Fémoral droit   + Poplité gauche   + Poplité droit   + Cardiaque   + Autre   La température   * Lieu   + Non précisé   + Rectal   + Buccal   + Axillaire   + Inguinale   + Tympanique   + Cutané   + Autre   Saturométrie   * Fio2 * Status * Air ambiant * Peak-flow   Selles   * Status   + Non précisé   + Absence de selles   + Présence de selles * Consistance   + Non précisée   + Dures   + Liquide   + Moulées   + Fausse diarrhée / fécalome * Volume   + Non précisé   + Petit   + Moyen   + Grand | La tension   * Prise   + Non précisé   + Debout   + Couché   + Assis   + Après effort   + Autre * Lieu   + Non précisé   + Bras gauche   + Bras droit   + Jambe gauche   + Jambe droite   + Autre   Fréquence respiratoire   * Rythme   + Non précisé   + Régulier   + Irrégulier   + Apnée   + Cheynes-Stoke   + Kusmmaul   + Autre * Observation   + Non précisé   + Dyspnée   + Tachypnée   + Sibilance   + Tirage   + Balancement thoraco-abdominal   + Paradoxal   + Stridor   + Autre |

Indice de la douleur  
Poids  
Taille  
Bilan

#### Application

The initial INCA prototype is an Android native application developed by Mr. Ehrler and his team. The application’s main view is a view listing all the interventions a nurse has to perform on a patient during her/his shift as illustrated on Figure 1.6.

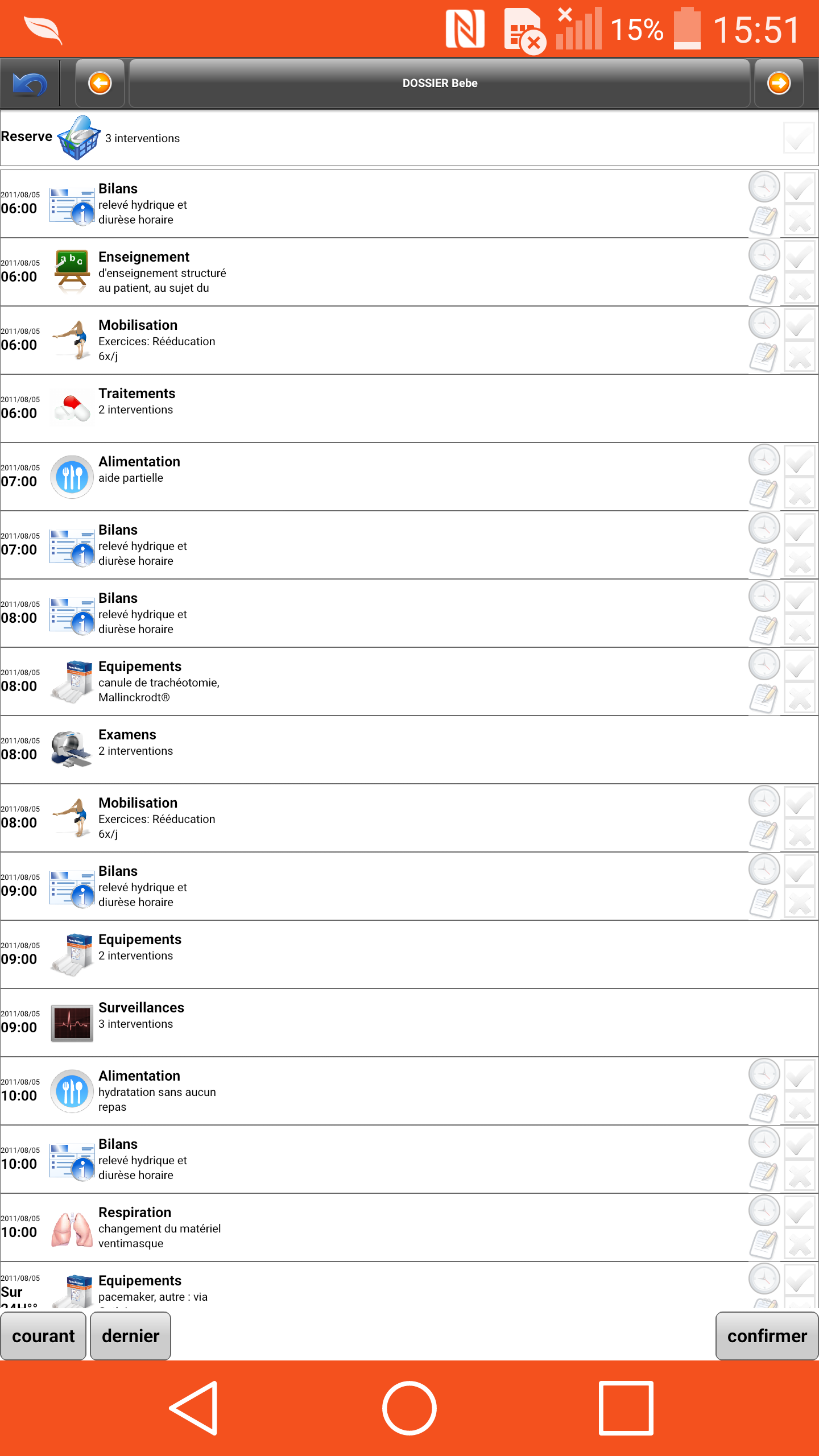


FIGURE 1.6 – INITIAL INCA PROTOTYPE’S MAIN VIEW

### Requirements

The new INCA application should be as easy as possible to use for nurses. It should take in consideration all the current steps of their workflow and try to make the whole process much more efficient.

Building a mobile application comes with a set of challenges, as the user interactions (some of which are illustrated on Figure 1.7) are different than on a desktop application.

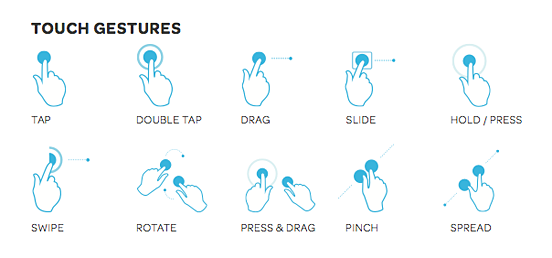


FIGURE 1.7 - TOUCH GESTURE EXAMPLES

The interface of the new INCA application has been redesigned to take advantage of the touch gestures offered by mobile devices in order to provide simple and intuitive navigation.

### Used Technologies

The new application for the INCA project has been developed using the following technologies :

* Angularjs
* Ionic
* Apache Cordova
* HTML 5
* CSS 3
* Javascript
* iBeacon
* **BLE** (Bluetooth low energy)

## Restrictions

Working with the HUGbrings important restrictions in terms of security since medical data are private and sensible.

Some of the main restrictions are listed below explaining each time what is the restriction and in which case it is limiting the efficiency of the application.

No patient’s data should be stored in the devices used

This restriction can be a problem if a nurse validates an action and there’s no network available, the data needs to be saved in memory waiting for the network to come back on. It also can be a problem if the phone turns off or the application is relayed in the background. All of this event are possible and should be considered and thought through in order to build a safe and stable mobile application.

Data can only be accessed from a trusted computer within the HUG’s network

The project had to be developed using fake sample patients’ data which feels less real.



## Challenges

Because the project is directly related to an external entity (HUG) represented by Mr**.** Ehrler, a lot of client - developer communication challenges appeared, such as:

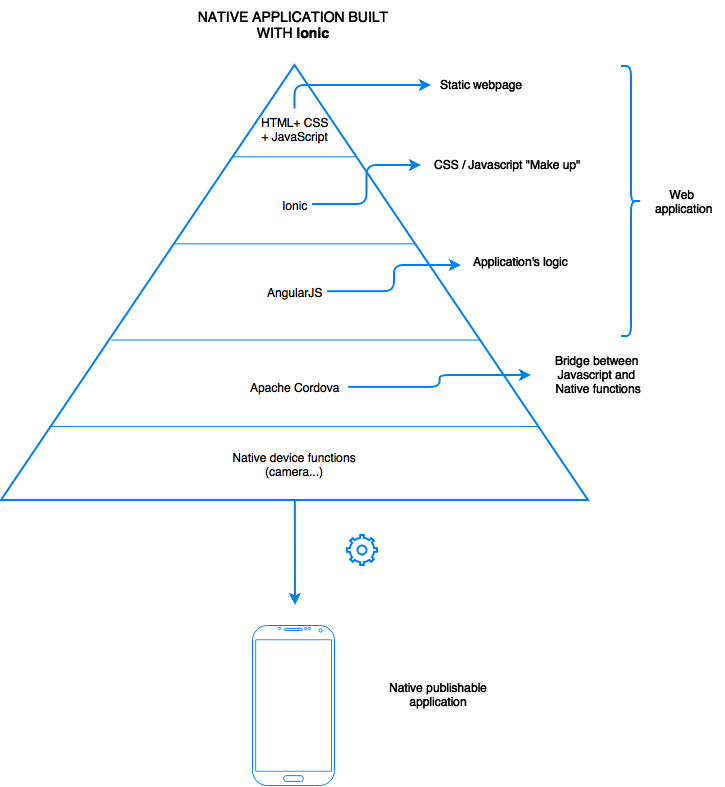
* Understanding the exact needs of the client, communicating on a regular basis to check the project’s progress.
* Rework on some features to make them more adapted or more user friendly depending on what the nurses are use to.

Moreover, the project is managing medical data, which is very sensitive and private.  
In consequence, the whole project's data source was based on sample development data coming from a single XML file provided by M. Ehrler. The data stored in the file is an exact copy of the structure of real HUG data. Additionally, all interactions with HUG’s servers and proxy were put aside because of restrictions on patient’s data access from the HUG.

The main goal was to have a working usable application with an emphasize on ergonomics and usability, validated by nurses and doctors. Only then, would it be connected to a real stream of data coming from actual patients.

On top of that, medical data can only be accessed from within the HUG’s network which means part of the work had to be done in their building.

# Framework

The INCA project makes use of different technologies and framework in order to produce a native application for both Android and IOS. The goal of using these various technologies is to bring a good user experience (UX). The Figure 2.1 illustrates the stack of different technologies used in the project.  


HTML, CSS and JavaScript are used to produce static web pages; the role of *ionic* is to add the native application user interface (UI) feel and behavior. Ionic’s core is AngularJS, which main purpose is building Rich Internet Applications (RIA).

Apache Cordova wraps it all in a web view that fills the whole screen and runs in a native container and exposes some native device function through an Application Programming Interface (API) for the web application to use.

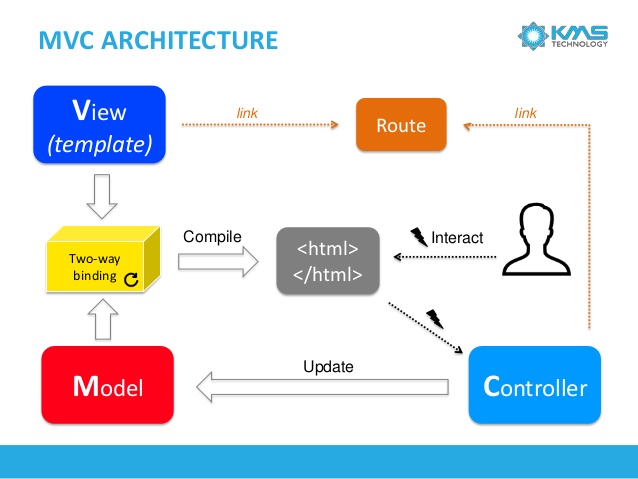
FIGURE 2.1 –THE STACK OF SOFTWARE TECHNOLOGIES USED IN THE PROJECT

## Angularjs

Angularjs**,** also called Angular**,** is a *100%* JavaScript*, 100% client side* framework compatible with both *desktop and mobile browsers*.  
Its goal is to simplify the development and testing of single page applications (SPA) by providing a framework for the model view controller (MVC**)** architecture, along with components commonly used in rich Internet applications (RIA). It’s lightweight, fast, and uses an extremely simple syntax (at least compared to Backbone or Knockout2 popular JavaScript libraries). Similarly to jQuery, it has a large number of 3rd party plugins and extensions. Angular comes with jqLite (a lightweight simplified version of jQuery) for accessing the Document Object Model (DOM) when necessary. However, jQuerycan still be loaded if needed.

Angular’s architecture

As illustrated in Figure 2.2 Angular is based on the MVC model.



2.2

The user interacts with the interface, triggering events that can be caught by the Controller. The Controller contains the application logic and processes data received by the user. If the Controller updates the Model, the two-way data binding features allows the view to be updated automatically. Routes are the way to navigate between views/controllers, very much like a regular link would change page on a website.

Two-way data binding

Two-way data binding is a very useful feature Angular provides that allows the model and the view to be constantly in synchronization.

As illustrated on Figure 2.3, when a template (html page) loads, Angular goes through the Linking phase where it creates a list of entry for every scope variable in the DOM and for some directives like ng-repeat (similar to foreach loop).

Once the template is loaded and everything is compiled to render the view, the user will begin to interact with the interface. Every interaction, even a keystroke on an input field, will trigger the browser events-loop.  
Angular extends the browser default event-loop to add something called an “angular context” to it.

Consequently, when an event is triggered and can be handled by an “angular context”, it will call for the Digest loop represented on Figure 2.3.

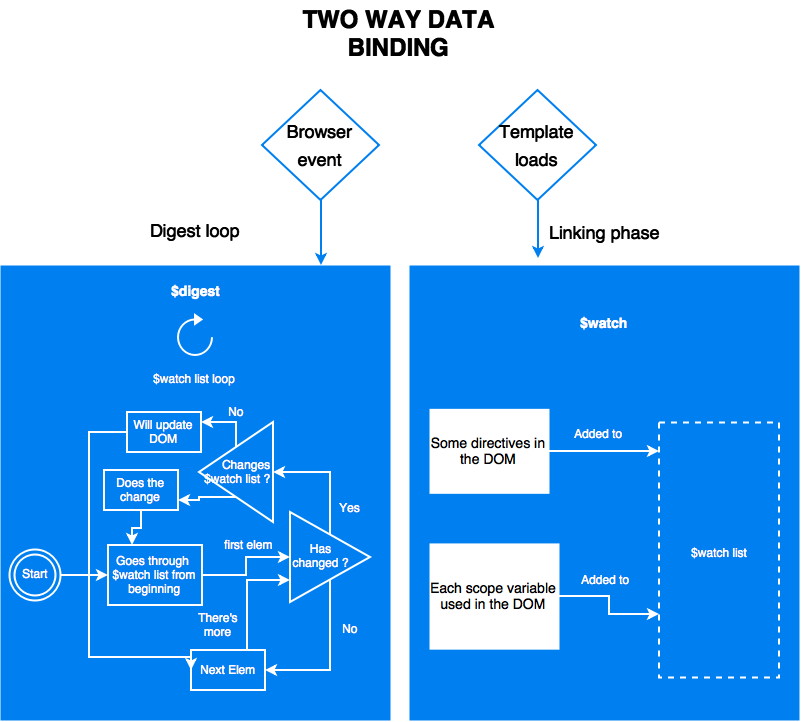


FIGURE 2.3 – TWO-WAY DATA BINDING WITH ANGULAR

Data transmission model

When first loading an Angular single page application (SPA) , server – client communications occur as illustrated on Figure 2.4

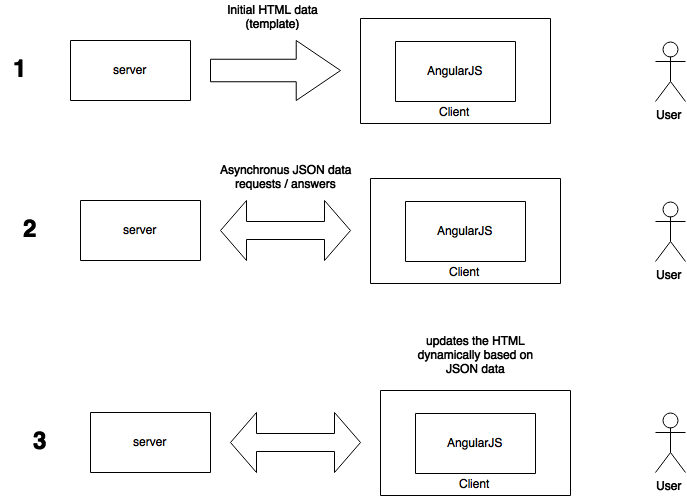


FIGURE 2.4 – INITIAL ANGULARJS APPLICATION DATA TRANSMISSION

This web model offers the following benefits:

* The server doesn’t have to send out the same set of HTML/css over and over again, which saves bandwidth consumption. Part of the HTML is sent when the view is changed only, as explained in the Figure 2.2.
* The client receives the HTML right away, without any prior execution time (usually seen with php), which reduces server workload.

The communication is a little bit different when the user switches from a view to another. Angular needs to load the new html template that fits the new view as illustrated on Figure 2.5.

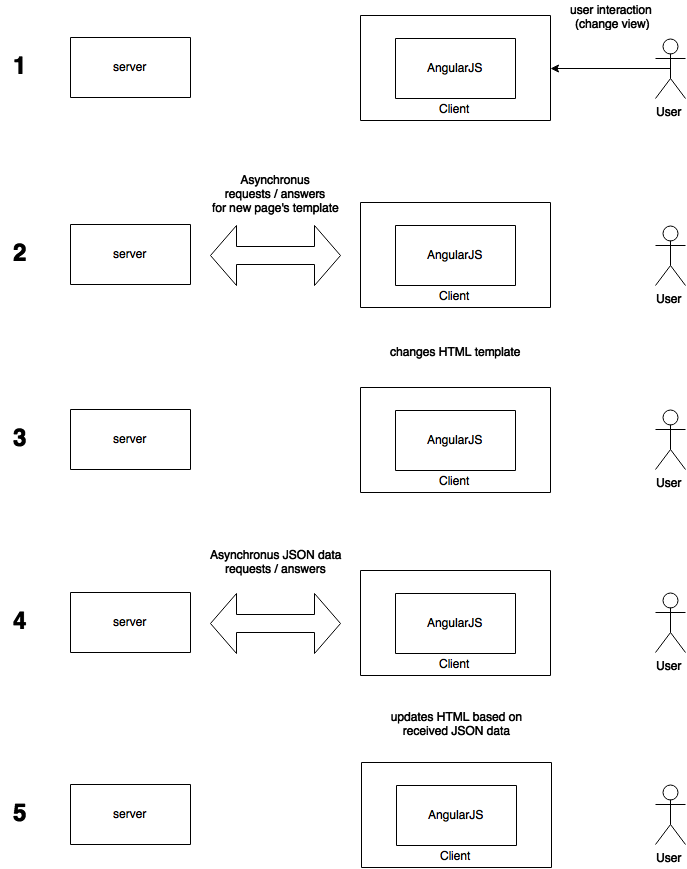


FIGURE 2.5 – DATA TRANSMISSION ON VIEW CHANGE

This web model fits right in the HTML5, AngularJS, and thin-serversstack (HAT).  
The HAT stack moves all the application logic into the web browser, letting the server handle data related operations such as sharing and storage.

Advantages

Since Web 2.0, asynchronous JavaScript and XML (AJAX) became popular, Frameworks like jQuery added dynamic behavior to pages.

Unfortunately, the result was a lot of duplicated logic code on the server and client sides.  
Older browsers required the server to be the master, thus forcing the browser side logic to be written progressively as browsers improved.

In 2013, almost all users were using recent browsers with good implementations of HTML5 and JavaScript**,** meaning developers could afford putting the client as the master and the server as its servant.

This meant a revolution in the application development process.  
Previously, four versions of an application had to be implemented:

* Web
* Mobileweb
* IOS
* Android

Suddenly, developers could write browser-applications with almost half the code used for traditional server/client applications. Moreover, such applications could run on every platform or device.

HTML5, responsive design and application logic worked seamlessly together on the web, mobile web and even IOS / Android via frameworks like Apache Cordova.

Angular-like frameworks allow for a clean client-side application architecture, which helps in building much larger applications. Consequently, messy jQuery and CSS hacks to coordinate with the server-side code are avoided. These Frameworks scales much better since the server is less solicited. Moreover, the server side code can almost be totally avoided by using services such as mongoLab or Firebase- backend-as-a-service (BAAS).

## Ionic

Ionic is an open source UI framework based on AngularJS.It helps building hybrid native mobile applications by providing developers with a large set of customs HTML tags and CSS classes that allow for fast interface building.  
Moreover, the way navigation is handled by AngurlarJS makes a web application feel and look like a native mobile application.

Ionic is heavily optimized for touch devices, allowing touch events and gestures recognition, but isn’t suited to desktop use. It comes with a lot of CSS components and a JavaScript UILibrarythat allows fast interface building and brings a feel of native application.

Gestures and events

Gestures and events are handled by $ionicGesture**,** an Angular service implementing different methods like

* on(eventType,callback,$element,options) **–** add an event listener
* off(gesture,eventType,callback) **–** remove an event listener

The different types of events and gestures are illustrated on Figure 1.7 and listed below:

* [on-hold](http://ionicframework.com/docs/api/directive/onHold/)
* [on-tap](http://ionicframework.com/docs/api/directive/onTap/)
* [on-double-tap](http://ionicframework.com/docs/api/directive/onDoubleTap/)
* [on-touch](http://ionicframework.com/docs/api/directive/onTouch/)
* [on-release](http://ionicframework.com/docs/api/directive/onRelease/)
* [on-drag](http://ionicframework.com/docs/api/directive/onDrag/)
* [on-drag-up](http://ionicframework.com/docs/api/directive/onDragUp/)
* [on-drag-right](http://ionicframework.com/docs/api/directive/onDragRight/)
* [on-drag-down](http://ionicframework.com/docs/api/directive/onDragDown/)
* [on-drag-left](http://ionicframework.com/docs/api/directive/onDragLeft/)
* [on-swipe](http://ionicframework.com/docs/api/directive/onSwipe/)
* [on-swipe-up](http://ionicframework.com/docs/api/directive/onSwipeUp/)
* [on-swipe-right](http://ionicframework.com/docs/api/directive/onSwipeRight/)
* [on-swipe-down](http://ionicframework.com/docs/api/directive/onSwipeDown/)
* [on-swipe-left](http://ionicframework.com/docs/api/directive/onSwipeLeft/)

Ui-router and application’ structure

As Ionic uses ui-router**,** a routing framework for Angular(<https://github.com/angular-ui/ui-router>)**,** developers must break down the application into different parts. Each part of an interface can be viewed as a block or as ui-router calls it, a state.

The frame is the main level state and contains several second level states that all inherit from the main level state. Those second level states can themselves contain third level states without any deep limit.

Each state is independent and can react differently depending on the interaction of the user.

The user navigates between main states and can interact with its children but never navigate from a child to another as they’re not main states.

Putting one secondary state into a main state is called nesting and it presents a lot of advantages.

Adding a dot after a state’s name tells ionic that the next state is a child.  
Another way to indicate the relation between two states is to use the parameter parent and quote the parent state’s name there.

Each template (part of a page, often HTML, that can be on another file) is loaded into a <ui-view> or a <ion-nav-view> tag within the parent state template as demonstrated on the Figure 2.6.

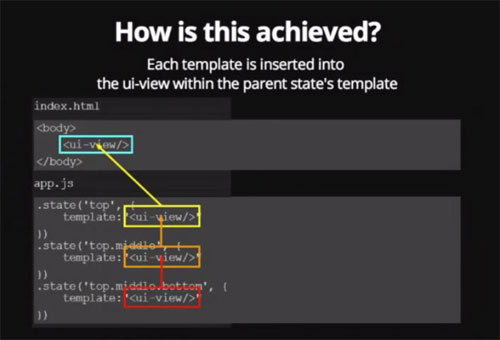


FIGURE 2.6 - UI-ROUTER AND UI-VIEW

A nested state inherits the scope and the methods of its parent (thanks to Angular).  
To make the scope inheritance work the views must be nested exactly like the states are.

Children states also inherit resolved dependencies and custom data as showed on Figure 2.7.

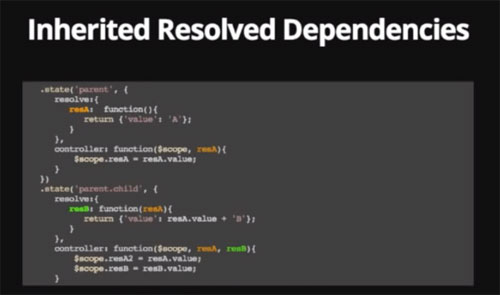


FIGURE 2.7 - UI-ROUTER AND INHERITED RESOLVED DEPENDENCIES

On this figure, something resolved in the parent state is still accessible in its child.

**What is** resolve **?**

Resolve can be used to provide the controller with content or data that is custom to the state. resolve is an optional map of dependencies which should be injected into the controller.

If any of these dependencies are promises, they will be resolved and converted to a value *before*the controller is instantiated and the $stateChangeSuccess event is fired.

**What about custom data ?**

.state('contacts.list', {

templateUrl: 'contacts.list.html',

data: {

customData1: 44,

customData2: "red"

}

}) ;

The property data can set custom data specific to a state and it’s children. It can be accessed through the variable $state.current.data.customData1. States navigation is achieved through several means.

* A custom directive ui-sref, which stands for state reference => smart anchors. It’s should be used instead of the usual href.

This allows changing the url (Uniform Ressource Locator) without changing them in the application.  
It will generate the corresponding href during compile.

Ui-sref **also accepts parameters**

<div ng-repeat contact in contacts>  
 <a ui-sref='contacts.detail({ id : contact.id})'>link 1</a>  
</div>

**urls with** ui-router

The property url exists to tie a state to a specific url.  
if state ‘sample’ has url ‘/sample’ and there’s another state called sample.child and its url is defined as ‘/child’ then the url of the child will really be /sample/child.  
This can be avoided by using the ‘^’ before the url => ‘^/child.

Angular usually uses the ngRoute service however ui-routeris more efficient in many ways.

|  |  |
| --- | --- |
| **Angular with ngRoute** | **ionic with ui.router (state machine)** |
| Flat hierarchy  To get more precise details from an initial view data display a new URI needs to be created:   * /interventions {view intervention} * /interventions/details {view details intervention}   Each URI theoretically reloads the whole view (maybe some fixed content wont change but most of the dynamic part will reload). | Nesting and inheritance To get more precise details from an initial view data display a sub-state can be created:   * /interventions – {view intervention, view interventions.details}   Only loads what needs to be changed. |
| The name of a route is it’s url. | The name of a state is an actual name |
| Routes are reachable only by their **url**. | States are reachable by their name with the command $state.go(stateName). |
| Single view using ng-view (lots of reloading). | Several views nested into each other, which means more focused reloading. |
| Only populates the current view. | A state can populate any view within its hierarchy. |

FIGURE 2.8 - COMPARISON BETWEEN NGROUTE AND UI-ROUTER FOR IONIC

## Apache Cordova

ApacheCordova is a set of device APIs for building cross platform native mobile applications using web technologies such as HTML, CSS and JavaScript**.**

One of the key feature on Apache Cordova is it’s ability to package a web application using the platform’s SDK and thus, publish it on the application store.

Another great feature Apache Cordova has is that it allows developers to access native device functionalities such as the *camera* or the *accelerometer* from JavaScript, as can be seen on Figures 2.9 and 2.10.

Cordova’s logic

Cordova works hand in hand with a web application built by the developer. For the new INCA application, this web application is powered by Ionic and Angular.

Cordova exposes a JavaScript API that allows developers to reach device natives functions.  
When a developer uses Cordova’s JavaScript API, the call goes through Cordova’s Native Libraries (Objective-C for IOS, Java for Android…) as shown in Figure 2.9 and 2.10. Given Cordova’s JavaScript libraries are cross-platform; any application can be ported to another platform or device.

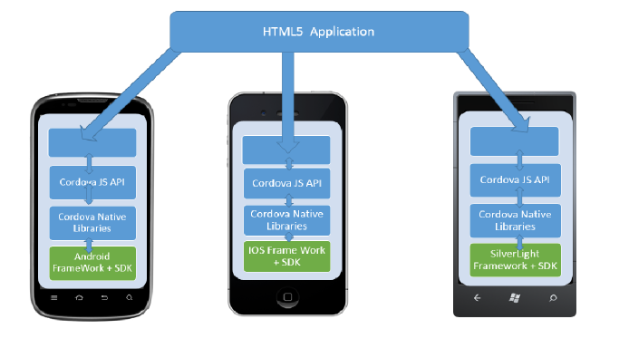


FIGURE 2.9 – APACHE CORDOVA’S LOGIC

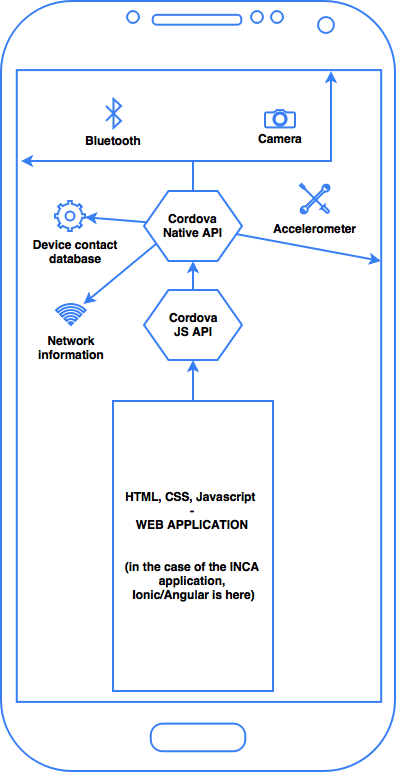


FIGURE 2.10 – APACHE CORDOVA’S LOGIC DETAILS

The UI of a Cordova Application is in reality a simple WebView embedded in a native container that fills the full screen of the device. Therefore, only the native container changes between operating systems (OS) and the WebView content remains the same.

The web application (Angular/Ionic in the case of the new INCA application) is contained within the native application, allowing everything to be packaged as a regular application using the platform’s SDK. This means it can be published on each platform’s application store.

Portability

Cordova provides uniform libraries available for the following platforms:

* IOS
* Android
* Blackberry
* Windows Phone
* Palm WebOS
* Bada
* Symbian.

## iBeacon

iBeacon is a protocol now used by many vendors that was standardized by Apple at the Apple Worldwide Developers Conference in 2013iBeacon compatible hardware (beacons) are Bluetooth low energy transmitters broadcasting their identifier to nearby portable electronic devices.

The INCA project uses EstimoteBeacons. They’re powered by a small battery and feature an ARM 32-bit Cortex M0 CPU, local memory, a 2.4 GHz radio using Bluetooth 4.0 Smart (BLE), temperature and motion sensors. Figure 2.10 illustrates such an Estimote Beacons.

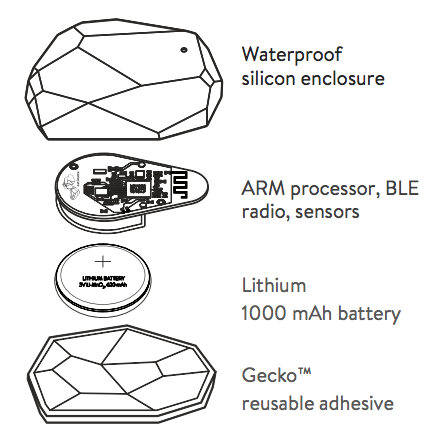


FIGURE 2.11 – ESTIMOTE BEACON BREAKDOWN

Bluetooth 4.0 LE (low energy)

Bluetooth 4.0 LE is not to be mistaken with the first versions of Bluetooth that needed pairing and never actually worked as expected.

Nokia has standardized Bluetooth 4.0 smart and it is implemented in many modern smartphones and devices.

Signal ranging and proximity measuring

As represented on Figure 2.5 EstimoteBeaconsare broadcasting their signal at a regular timeframe to all compatible devices. The maximal range of EstimoteBeacons is 70m in an interference free environment. In real world conditions, depending on obstacles, a range of 40-50m should be expected.



FIGURE 2.12 – BEACONS RANGING SYSTEM

Bluetooth 4.0 ready devices can catch the received signal and estimate the distance by measuring the received signal strength indication (RSSI). The accuracy of the distance is affected by the disturbances of the environment on the signal and the frequency (number of time it broadcast per seconds) of the broadcast signal the Beacon has been configured for.

Battery life expectancy

Estimote Beacons can last as long as six years depending on the frequency of broadcast and the range they’re configured to broadcast at.

## OAuth 2.0

OAuth 2.0 is a secured authentication protocol allowing a third party application to access a user’s data.

This operation can only happen successfully if these two conditions are met:

* The application storing the data must expose them through an OAuth2 API
* The user must allow the third party application to use his data.

The first step in setting up an OAuth 2.0 connection is to register/authorize OAuth 2.0 on the OAuth Provider. Once this step is accomplished, the user receives a “client id” and a “client secret” which work similarly as a login/password combination.

Then, the OAuth client can send a request for an authorization to a specified unique resource identifier (uri).  
In this request there are usually two parameters, which are the “client id”, and the “redirect\_uri”. The redirect uri is the uri where the custom code for OAuth is located on the client side (where the provider will send the data).

Once the authorization request is sent, the provider will prompt a dialog asking the authenticated user to say if they want to grant access on some specific data to the OAuth client.

If the user answers positively to the prompt, the OAuth provider responds with a code as a parameter on the “redirect\_uri”.  
The OAuth client then takes that code and issues a token post request to the provider on a specific uri defined by the provider containing the “client id”, the “code” received earlier and the “client secret”. This is to ensure once again that the right client is asking for the right client’s token. If all of these information are correct, the provider then sends back a temporary “access token” to the client. The token allows the OAuth client to access the provider’s API.

This procedure is illustrated on Figure 2.12.

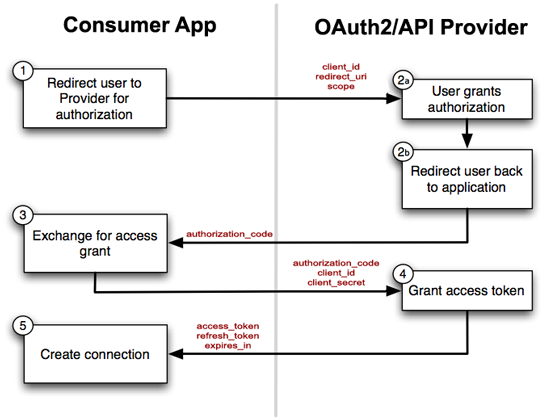


FIGURE 2.13 - OAUTH2 PROTOCOLE

Once an OAuth client possesses an authorization code for a specific client, it can ask for a temporary token at any time as long as the user doesn’t specifically revoke the right of the OAuth client to access the data.

# Application

## Global architecture

The new INCA application uses a stack of technologies that work together as a whole to produce a final mobile native application. The overall simplified functional breakdown of the new INCA application is described in the Figure 3.1 below.

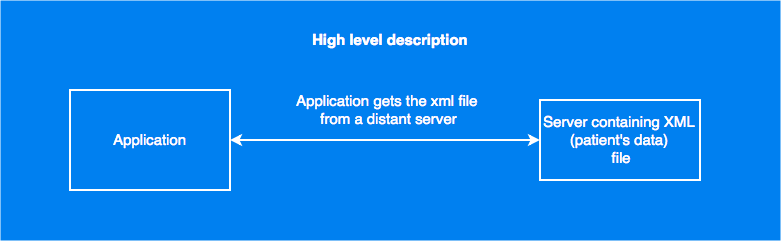


FIGURE 3.1 – OVERALL HIGH LEVEL FUNCTIONNING DESCRIPTION OF THE NEW INCA APPLICATION

The application is composed of several “layers” inside the Application block represented on Figure 3.1, those layers are represented on Figure 3.2



FIGURE 3.2 - OVERALL MEDIUM LEVEL FUNCTIONNING DESCRIPTION OF THE NEW INCA APPLICATION

Each “layers” have their own purpose and functions described in the Figure 3.3.

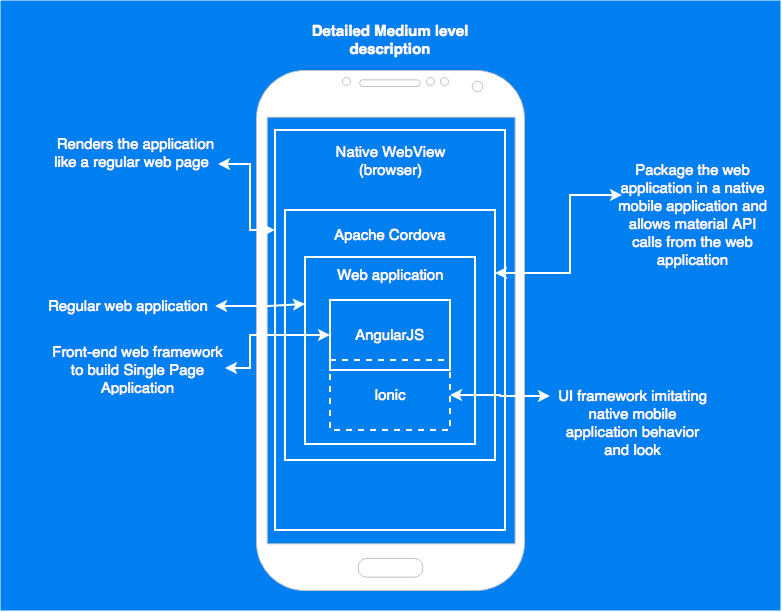


FIGURE 3.3 - OVERALL DETAILLED MEDIUM LEVEL FUNCTIONNING DESCRIPTION OF THE NEW INCA APPLICATION

## File hierarchy

The application developed during this bachelor project uses a modular structure that is many advantages for large projects development and code maintenance.

In order to help understanding the general architecture of the application, the project’s file hierarchy is introduced first in Figure 3.4.

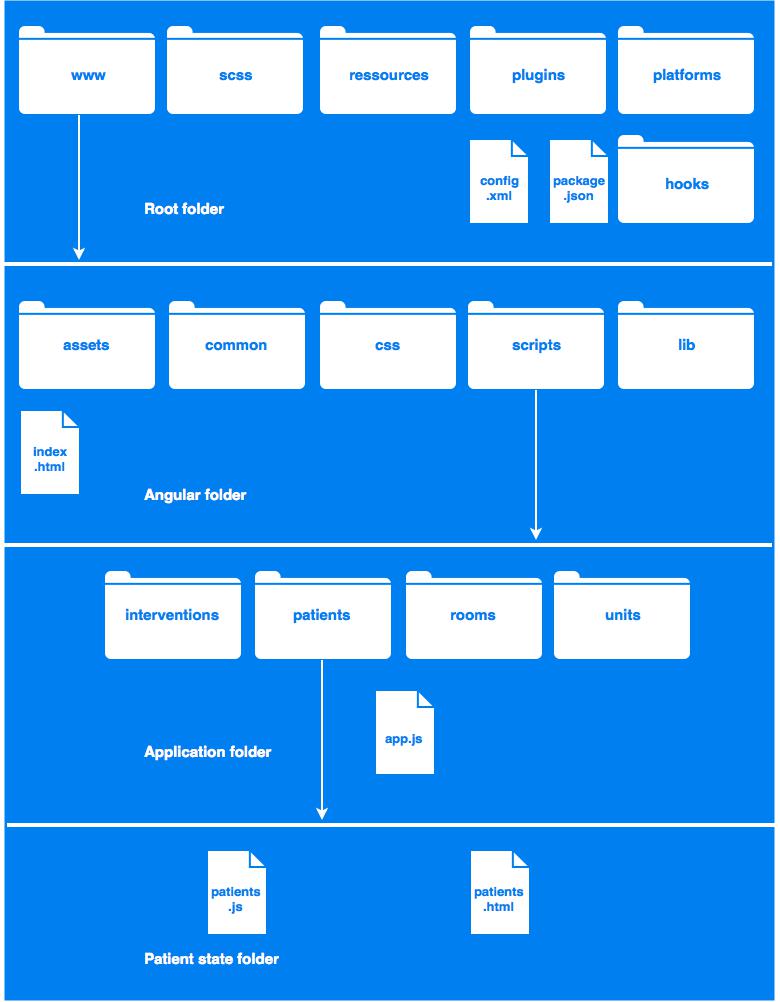


FIGURE 3.4 - FILE HIERARCHY

Each folder and file has its specific use and importance, they will be described from the root folder to the patient state folder (top to bottom):

* **hooks** => hooks are scripts that run at various times in the build process; there are thirty-two types of hooks; some examples are (before\_prepare, after\_prepare, before\_build, etc.)
* **platforms** => contains all the files useful to compile the application on different mobile platforms such as IOS and Android.
* **Plugins** => contains all the Cordova plugins added to the project.
* **Ressources** => contains the icons for the Android/IOS/… version of the app
* **Scss** => contains the Syntactically Awesome Style Sheet (SASS) file for ionic
* **www**  => contains the AngularJS application
* **config.xml** => is a Apache Cordova configuration file referencing the application’s icons, orientation, author and description.
* **package.json** => is the header of the application; it contains the modules’ versions and the name of the application
* **assets** => contains all images used by the user interface (UI).
* **common** => contains all scripts (directives, models, services) shared by the different states.
* **css** => contains cascading stylesheets**.**
* **lib** => contains all angular libraries/services used in the project.
* **index.html** => is a html container of the different states
* **interventions | patients | rooms | units** => folder containing the corresponding state’s files.
* **App.js** => is the entry point of the application, it specifies the application’s configuration. It should be kept as light as possible.
* **Patients.js** => is the controller (logic) of the patient’s state.
* **Patients.html** => is the html template of the patient’s state.

This division in logical blocks allows for an easy and fast understanding of the application’s structure and greatly reduces the risk of eternal file browsing in the application since everything needed for a state is within its own folder or in the common folder.

## Navigation

The new INCA application is composed of several interfaces called states or views. Those states are represented on the Figure 3.5 as rectangles. Each rectangle represents a state / logic block. A state can contain multiple sub-states / sub-views that are accessible from the state/sub-state they’re linked to.

From the interventions state, a nurse can access:

* The “vitals” view by tapping on a “measures” intervention.
* The “reserveValidation” view by tapping on the “reserves” intervention
* The “details” view by tapping on any other intervention

From the “vitals” view the nurse can access the “full historic” view by double tapping on any graph.  
From the “reserveValidation” view, the nurse can access an historic of the reserve intervention performed on the patient.

In every view, the nurse can go back to the previous view by tapping the back button on the upper left corner of the screen.

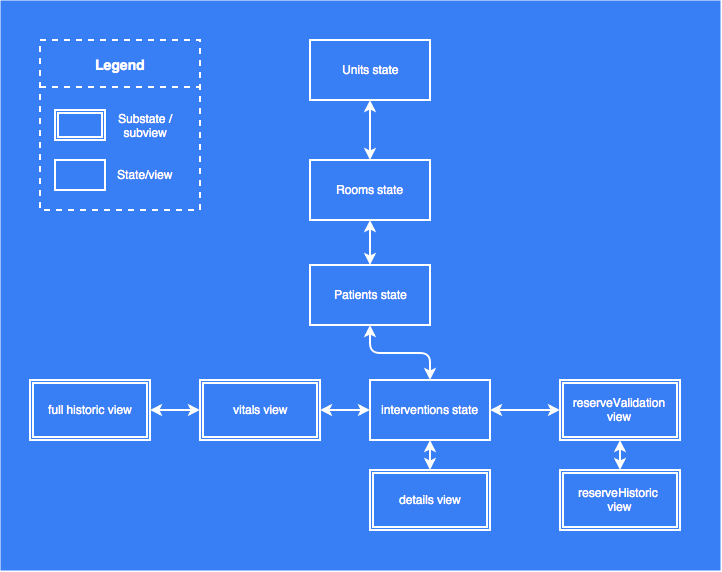
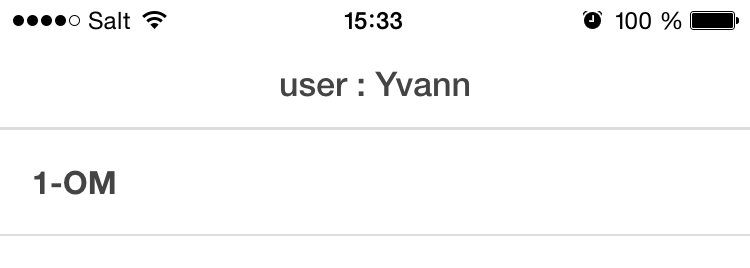


FIGURE 3.5 – INCA APPLICATION’S NAVIGATION

## Wireframes and functionalities

This section will show an illustration for each view in the new INCA application and describe their functionalities.

Units

As shown on Figure 3.6, the units state’s view is pretty simple.

The user can see her/his name written on the top bar and choose between all the units he/she has access to.

FIGURE 3.6 - UNITS STATE INTERFACE

Rooms

As shown on Figure 3.7, the room state’s view allows for a quick overview of the list of all rooms in a unit. It also displays the sex of the patients in the room.

FIGURE 3.7 – ROOMS STATE’S VIEW

Patients

As illustrated on Figure 3.8, the patients’ state view shows the following information about the patient:

* Name
* Surname
* Picture
* Age
* Sex

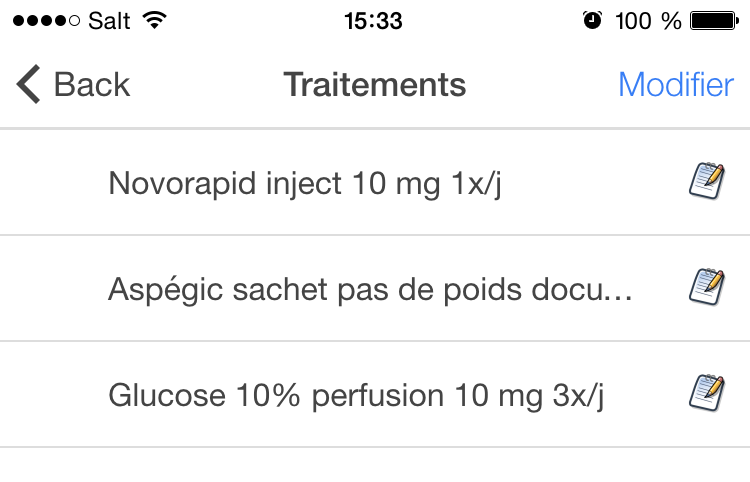
FIGURE 3.8 - PATIENTS' STATE VIEW

Interventions

As illustrated in Figure 3.9, the intervention state’s view is filed with a lot of functionalities:

* Nurses can swipe from a patient to another or tap on the arrows next to the patient’s name on the top bar.
* By tapping on the reserve intervention, nurses can access the reserve sub-state / view.
* By tapping on the notepads icons, nurses can validate an intervention and add a note before validation.
* By tapping on the intervention line similar to “Traitements 2011-08-05 06:00 2 intervention”, nurses can access a detailed view of these multiple interventions.
* By tapping on a “surveillances” intervention, nurses can access a view where they can enter new measures performed on the patient.

FIGURE 3.9 - INTERVENTION STATE'S VIEW

Reserves

AS illustrated in Figure 3.10, the reserves state’s view allows nurses to :

* Validate and attach a note on a reserve intervention by tapping on the notepad.
* Consult the history of the last reserve interventions by tapping on the intervention.

FIGURE 3.10 - RESERVES STATE'S VIEW

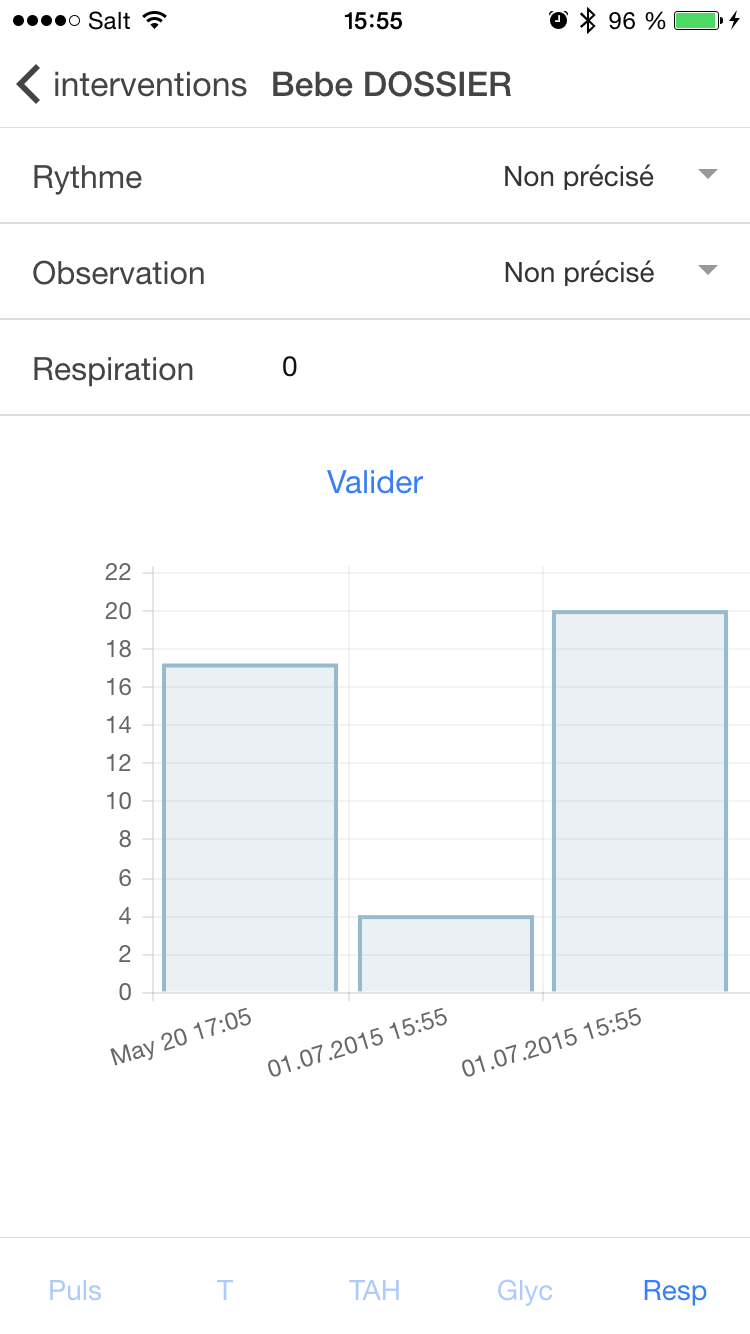
Reserves detail



As showed in Figure 3.11, the reserves details state’s view allows nurses to consult a simple historic of the last completion of the selected reserve intervention.

FIGURE 3.11 - RESERVES DETAIL STATE'S VIEW

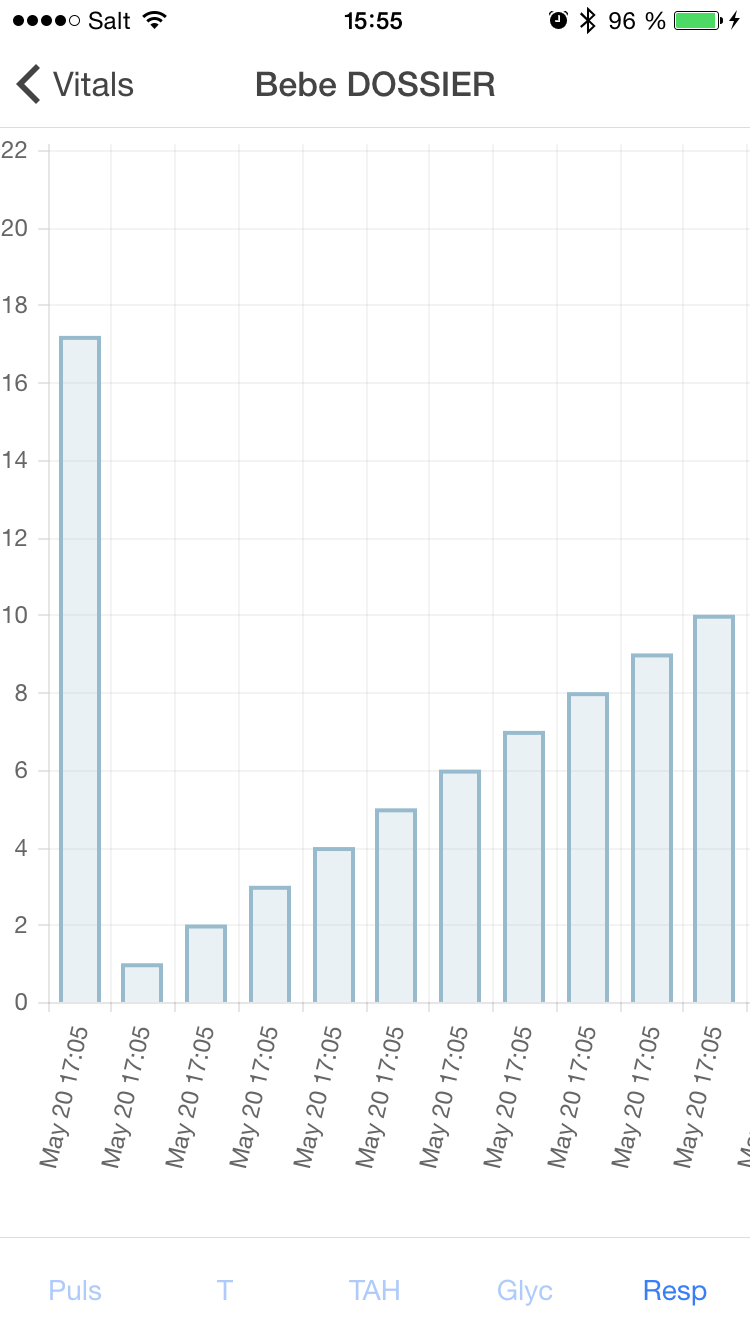
Vitals

As illustrated in Figure 3.12, the vitals state’s view allows nurses to:

* Enter new measures of the vital signs of a patient.
* Navigate across different tabs by tapping on their name.
* Access a detailed history of the measures performed on the patient by double tapping on the graphs.

FIGURE 3.12 - VITALS STATE'S VIEW

Vitals full historic

As showed in Figure 3.13, the vitals full history state’s view allows nurses to:

* visualize the different vital signs measures performed on the patient in an easy and effective way.
* Zoom in / out on the graph
* Double tap on the graph to reset the zoom.
* Navigate between vital signs categories by tapping on the tabs.
* Consult the exact value of a measure by holding a finger on a bar.

FIGURE 3.13 - VITALS FULL HISTORIC STATE'S VIEW

## Patients data structure

The patients’ data structure is composed of many information on the patients like:

* Sex
* Photo
* Age
* Name and surname
* Room number
* A list of all the interventions (medical act like blood pressure measuring or medication giving) that have to be performed on the patient during the current nurse’s shift.

All of this information is contained in a xml file and is structured as illustrated on Figure 3.14.  
The arrows represent the parent <-> children relation between two entities.

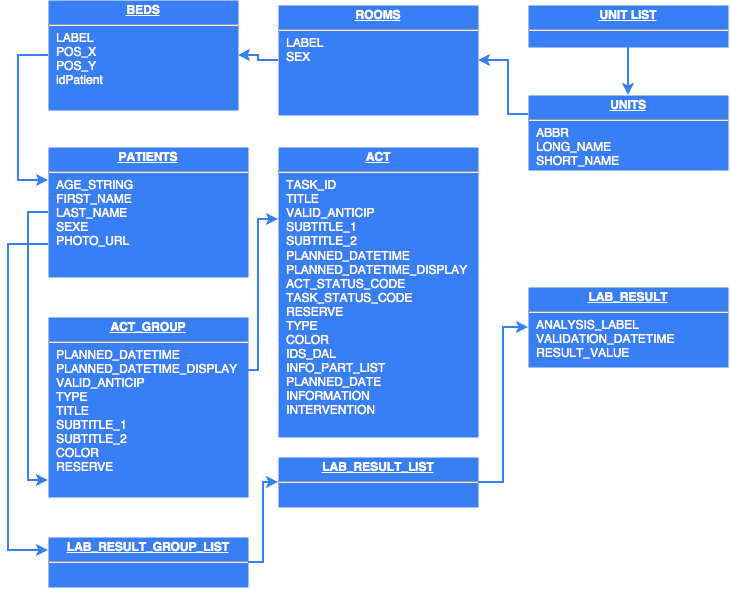


Figure 3.14 - PATIENTS' DATA STRUCTURE

## Data transmission

As patients’ data must never be stored on the mobile device, the nurses’ actions, once performed, must be sent out to the server as soon as possible.

Each time an action is validated by a nurse, an entry is added to a memory data buffer. As data are transmitted over Wi-Fi., there can be issues with the transmission. To ensure data safety, if problems are detected and the application fails to send the memory data buffer, it will try to send it again to the server every 30 seconds until it succeeds.  
This procedure is illustrated on Figure 3.14.

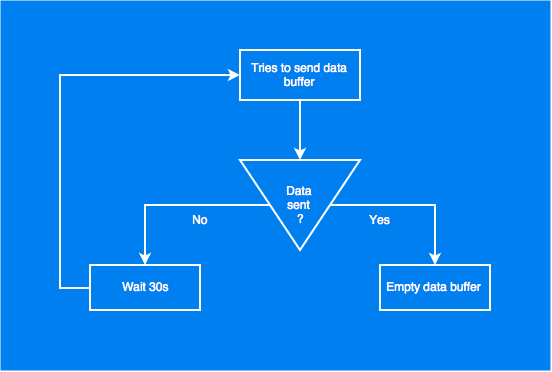


FIGURE 3.15 - DATA TRANSMISSION PROTOCOL

## iBeacon

Each beacon is configured with the same Universally Unique Identifier (UUID) and placed in a different patient’s room. The application searches for beacons with that UUID. Once found, the application receives their major and minor values. This indicates which room the device is in.  
The minor value could be used if the hospital decides to add more than one beacon per room for accuracy purposes.

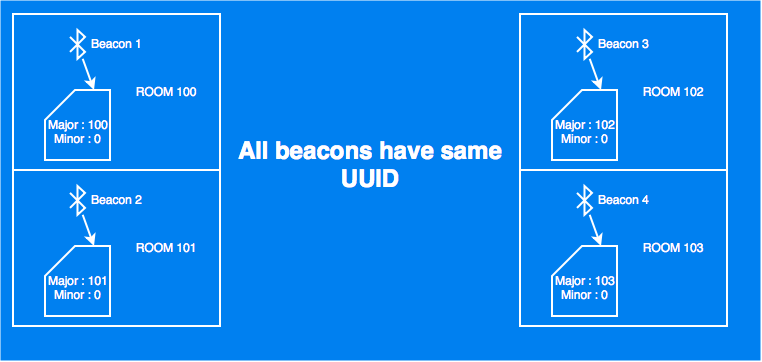


FIGURE 3.16 - BEACONS ORGANIZATION

## Geolocalisation

Estimotes beacons are used to detect the location of a device within the hospital’s building. One beacon is placed in each room broadcasting its id.

The mobile application retrieves beacons’ signals from all devices with the same predefined UUID. The geolocalisation of the room is achieved by taking the major value (room number) of the strongest iBeacon.

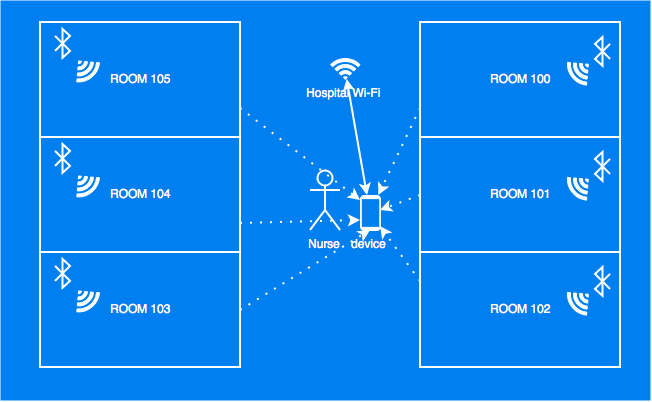


FIGURE 3.17 - HOSPITAL ROOMS EQUIPPED WITH BEACONS

Once the application has identified the nearest room, it automatically sends the data stamp to the server and switches to the view displaying the list of patients for the nearest room.

This mode of operation is called “automatic” and can be activated/deactivated in the settings view.

# Discussion

## Performances

Pros and cons of a hybrid application

Building a hybrid application instead of native presents many advantages but also some heavy cost as mentioned bellow:

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| Quick prototyping with UI frameworks like Ionic. | At least two languages have to be learned for Android and IOS. |
| A lot of libraries options for pretty much everything not related to devices’ hardware. | Cross-browser compatibility is bad, so there can be some surprises as hybrid app run in WebView (native browser). (A lot of users are on Android 2.x). |
| Managing images for multiple devices is a lot easier with web technologies than it is for an IOS or Android project. Native applications needs every image assets in different resolutions:   * ldpi (Low dots per inch, 120 pixels/inch) * mdpi (Medium dots per inch, 160 pixels/inch) * hdpi (high dots per inch, 240 pixels/inch) * xhdpi(Extra high dots per inch, 320 pixels/inch) | Hybrids application can only use one thread at the same time (the one from their WebView). Hybrid applications shouldn’t be used for big processing applications. |
| More and more native hardware functions are available to hybrid applications. | Not every native hardware functions are directly available to hybrid applications. |

FIGURE 4.1 – PROS AND CONS OF A HYBRID MOBILE APPLICATION

The decision to use or not hybrid applications should be made depending on the needs and requirements of the application, considering the pros and cons of each native and hybrid applications.

This new INCA application workflow presents two bottlenecks:

* Low hardware resource devices can have trouble running the application smoothly.
* The numerous requests coming from different nurses could overwhelm the server.

Mr Ehrler explained that nurses are going to use Samsung Galaxy note 4 smartphones to run the application. That eliminates the low hardware possible issue and ensure a smooth experience while using the application.

Five hundred requests at the same time aren’t a problem for the HUG’s servers which eliminates the second breaking point of the application workflow.

## Issues encountered

There have been many issues encountered during the application development.  
Most of them were fixed in a day or less, they were usually simple, small problems, like some libraries missing or an HTML tag at the wrong place. Some issues were more complicated to fix or simply long term issues, some were organizational and others were technical

Organizational issues

Working with two entities (hepia and HUG) brought on some challenges as well. A lot of compromise had to be made for both side to be satisfied.

Mr. Glück from hepia wanted the new application to be as close as possible to a working fully functional and usable application using real patients’ data while Mr Ehrler from the HUG tried to make compromise to help my application feel more real without touching at real patients’ data. This was a constant point of discussion and misunderstanding.  
We could have avoided that problem by defining at the beginning exactly what would be the end product and how it would work.

Many issues came from misunderstandings or a lack of communication from my end onto what exact functionalities should be implemented in the application or how they should be presented.

Technical issues

Estimotes beacons brought a lot of trouble; it took a lot of time to make the ranging and detection work with the application. The “official” plugin’s sample code available on github wouldn’t work, thus causing the whole iBeacon part of the application to be put on hold for some time.

At first, the new INCA application was structured in a very basic way separating models views and controllers in three different directories. After a few weeks of work I realized this structure wasn’t adapted for the project as it’s an application that could potentially reach the size of a big project with further coding by the HUG.

I decided to restructure everything, using another approach where each component is separated in it’s own folder containing it’s controller, view and model. This change might seem minor but it actually took a whole day to make the application work again.

## Future work

Functional improvements

In order to give the application its full potential, it should be able to retrieve and send data from/to the “dossier patient integré” (DPI) of the HUG servers.

To access the HUG infrastructure, specific actions must be performed:

* Nurses need to log in via the HUG authentication servers.
* The application must be tested, approved and validated by the hospital staff.

Upon completion of these two steps, the application will be granted access to the HUG servers, allowing live access to patients’ data.

User experience improvements

The intervention view is the application’s most used and rendered view. Its usability can be improved through the following changes and/or extensions:

* Interventions in timeframes like “Matinée”, “Diurne”, “Nocturne” should always come first when the current time matches the timeframe.   
  For instance, interventions that must be performed during “Matinée” will come first when the current time is between 6 am and 12 am. However, if the current time is 1 pm, all the “Matinée” interventions will still be ordered with the 12 am interventions and not later.
* The Reserve section should always be visible and follow the title of the view as the user scrolls down.
* The interventions should hide the icons and only reveal them if the user does a finger slide on the left on the intervention. This slide would unveil various icons for the different actions as often seen on IOS or Android applications.

# Conclusions

I learned a lot during this project. It has helped me improve my ability to work independently, planning for long term projects and organize myself to meet deadlines.

I’ve also acquired valuable knowledge. Being able to build hybrid applications is a very useful skill that I intend to use often during my future professional life.

Additionally, I know understand much better what it means and what it entice to work in a professional working environment. I wasn’t aware of the non-technical problematic a project can encounter when developed in a medical (sensitive/private) environment like the HUG. I now, can say that I’m familiar with the implications of such work and am better prepared to take on a similar challenge in a professional environment.

In conclusion, I’m happy with the result of my work. I enjoyed working on such a challenging and interesting project.

# Appendices

# References

Hopitaux Universitaires de Genève. (n.d.).

<http://angular-tips.com/blog/2013/08/watch-how-the-apply-runs-a-digest/>

http://wilk.github.io/AngularJS-Javascript-framework-for-superheroes/#9

http://wilk.github.io/AngularJS-Javascript-framework-for-superheroes/#12