Mobile application to support nurses’ workflow



Bachelor thesis presented by

**Argentin Yvann**

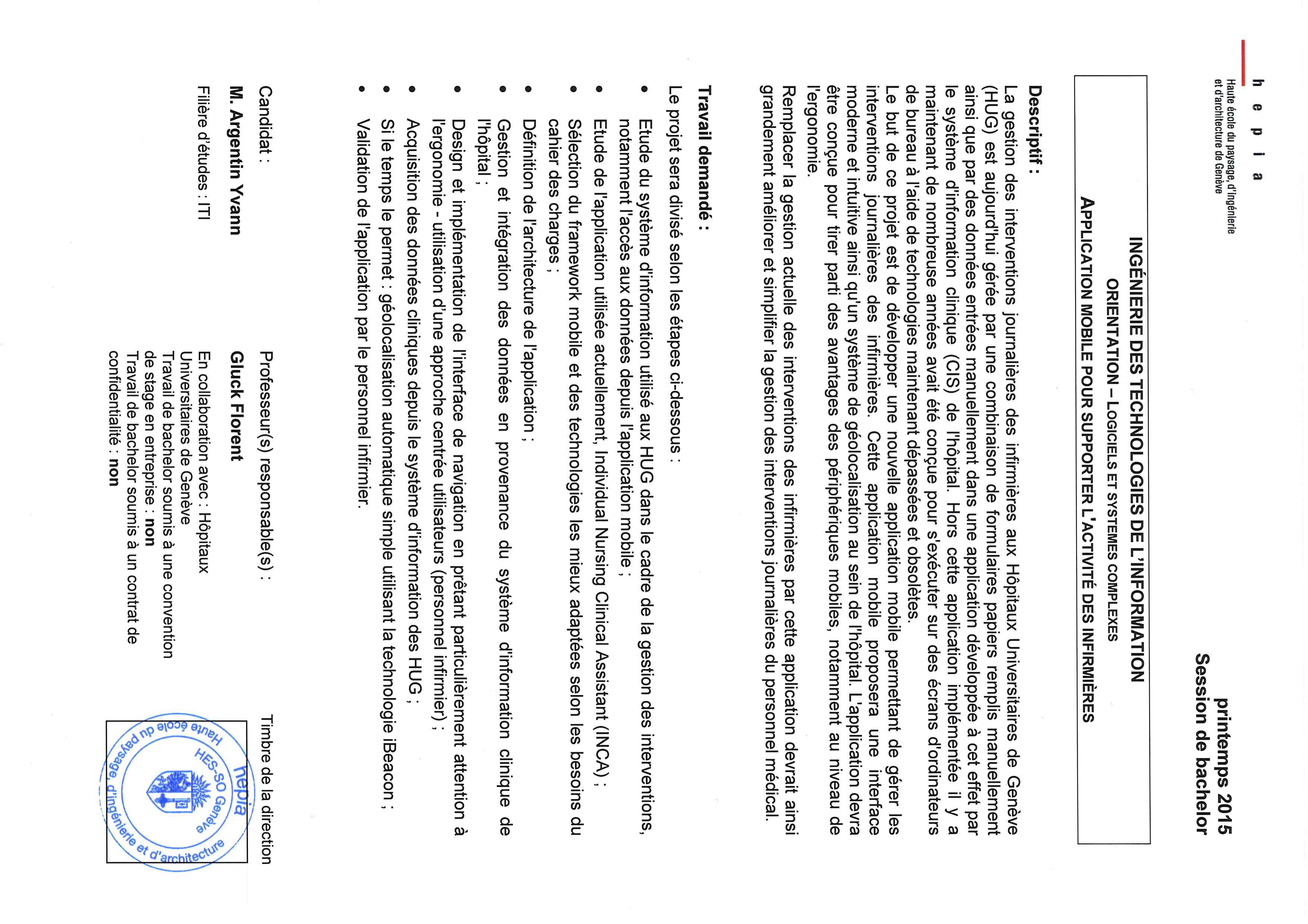
To obtain the title Bachelor of Science HES-SO in

**Information technology engineering with a specialization in**

**Software and complex systems**

***September 2015***

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



**Foreword**

Document’s structure

This thesis was developed with the help of the Universitary Hospital of Geneva (HUG).

This thesis will start by a context description, what was the initial issue, how it was solved up until now and why a better solution was needed. It will also explain how the current informatics architecture of the HUG works.

It will then explain what technologies will be used to reach the overall goal and describe the new nurses workflow. This first part will end by the description of all the restrictions and issues working in a distant place with sensitive data imply and what were the main challenges coming along with it.

The second part will be an in depth description of all the technologies and protocols I’ve used. How they work alone and together to produce a “native app” on both Android and IOS**.**

The third part of this thesis will describe the application in itself; it will contain a lot of illustrations and schemas to help understanding exactly what everything does. It will also describe in detail how the HUGdata are structured and how I accessed, manipulated and displayed them to ensure the best user experience (UI).

It will then end with the possible enhancing mostly in terms of user experience (UX).  
It will talk about the issues and problems encountered during the creation process and finish with the conclusions drawn out of the entire project.

Text formatting/Manual of style

To allow an easier understanding and readability of the document some text formatting rules will be applied:

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

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 advices, as he was my main source of information for the entire HUG related questions. He also provided me with a desk within the HUGmainframe.

1 Introduction 7

1.1 Context 7

1.2 Hospital architecture 8

1.3 Proposed solution 10

1.1.1. Existing application 10

1.1.2. Requirements 14

1.1.3. Technologies 15

1.4 Restrictions 15

1.5 Challenges 16

2 Framework 17

2.1 Angularjs 18

2.2 Ionic 22

2.3 Apache Cordova 26

2.4 iBeacon 29

2.5 OAuth 2.0 30

3 Application 32

3.1 Architecture 32

3.2 Navigation 33

3.3 Wireframes and functionalities 33

3.4 Patients data structure 33

3.5 Handling of patients’ data 35

3.6 Data transmission 35

3.7 iBeacon 35

3.8 Geolocalisation 35

4 Discussion 36

4.1 Performances 36

4.2 Issues encountered 36

4.3 Future work 36

5 Conclusions 37

6 Appendices 38

7 References 39

FIGURE 1.1 – NURSES’ CURRENT WORKFLOW 7

FIGURE 1.2 - OVERALL DPI COMMUNICATIONS 9

FIGURE 1.3 – NEW NURSES’ WORKFLOW 10

FIGURE 1.4 - FIRST INCA APPLICATION STRUCTURE 11

FIGURE 1.5 - TOUCH GESTURE EXAMPLES 15

FIGURE 2.1 – TECHNOLOGIES WORKING TOGETHER 17

FIGURE 2.2 – ANGULAR’S ARCHITECURE 18

FIGURE 2.3 – TWO-WAY DATA BINDING WITH ANGULAR 19

FIGURE 2.4 – INITIAL ANGULARJS APPLICATION DATA TRANSMISSION 20

FIGURE 2.5 – DATA TRANSMISSION ON VIEW CHANGE 21

FIGURE 2.6 - UI-ROUTER AND UI-VIEW 24

FIGURE 2.7 - UI-ROUTER AND INHERITED RESOLVED DEPENDENCIES 25

FIGURE 2.8 – APACHE CORDOVA’S LOGIC 27

FIGURE 2.9 – APACHE CORDOVA’S LOGIC DETAILS 28

FIGURE 2.10 – ESTIMOTE BEACON BREAKDOWN 29

FIGURE 2.11 – BEACONS RANGING SYSTEM 30

FIGURE 2.12 – OAUTH2 PROTOCOLE 31

FIGURE 3.1 – IONIC’S MAIN FOLDER 33

FIGURE 3.2 – MAIN ANGULAR FOLDER 34

FIGURE 3.3 – ANGULAR SCRIPTS FOLDER 34

# Introduction

## Context

Currently, the nurses dispose of a desktop application that helps them with their everyday work as represented on the Figure 1.1.

Nurses have complicated changing schedules, when they get to the hospital; For the sake of example, the nurse will be called Francis**.**

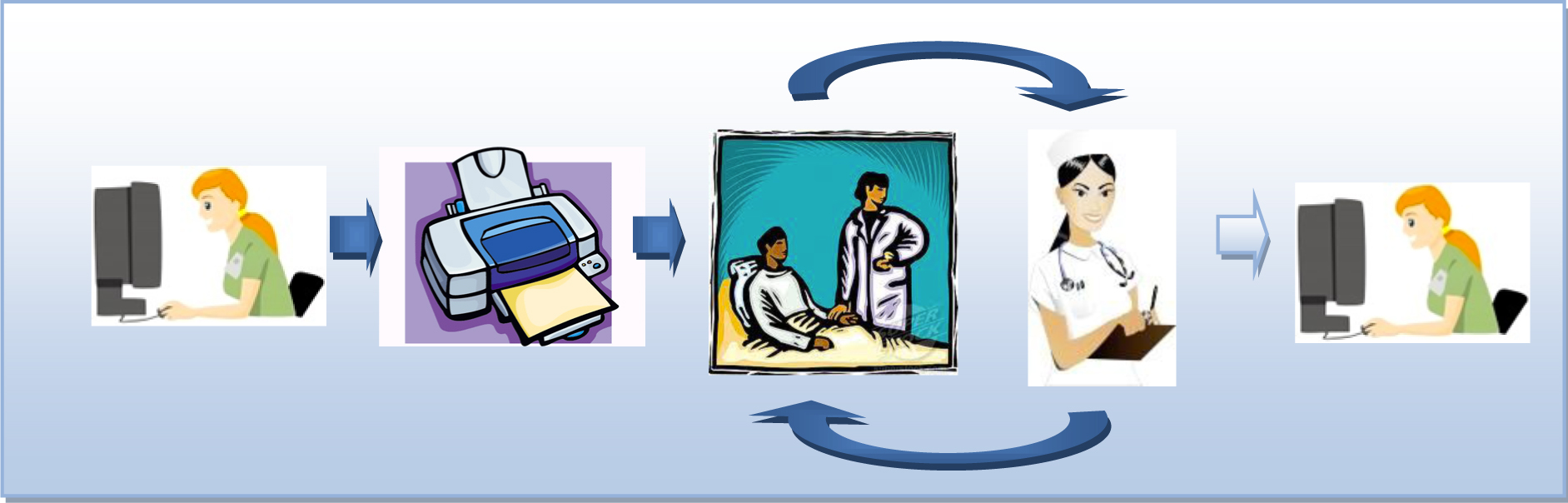


FIGURE 1.1 – NURSES’ CURRENT WORKFLOW

1. Francis logs into the desktop application and prints a paper sheet for every patient he has to visit today.
2. Francishas now a lot of paper with him, he chooses the one he will use shortly and go visit 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 done he goes back and take 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 paper one by one and rewrites it’s content into the 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 start working. They don’t watch the exact time for every intervention they perform, which means data, can be not accurate and in the medical field, accuracy is primordial.

Adding to that, Franciscan mistype or misread something and enter false data into the system!

There’s a double risk for errors and that puts a lot of pressure onto Francis’ back.

The hospital wants something that will increase the accuracy of both measures and typing/reading, allow almost real-time data sync for all the working employees and less volatile than paper.

## Hospital architecture

The hospital has several servers delivering data chunk relative to the employee asking for it, in the case of Francis, he receives a list of all the patients he has to visit during his current shift.

To read data from the Hospital’s servers, Francis needs to use a trusted machine that will ask a proxy’s service to ask the “Dossier Patient Intégré” / “Integrated Patient File” (DPI) as represented on Figure 1.3.

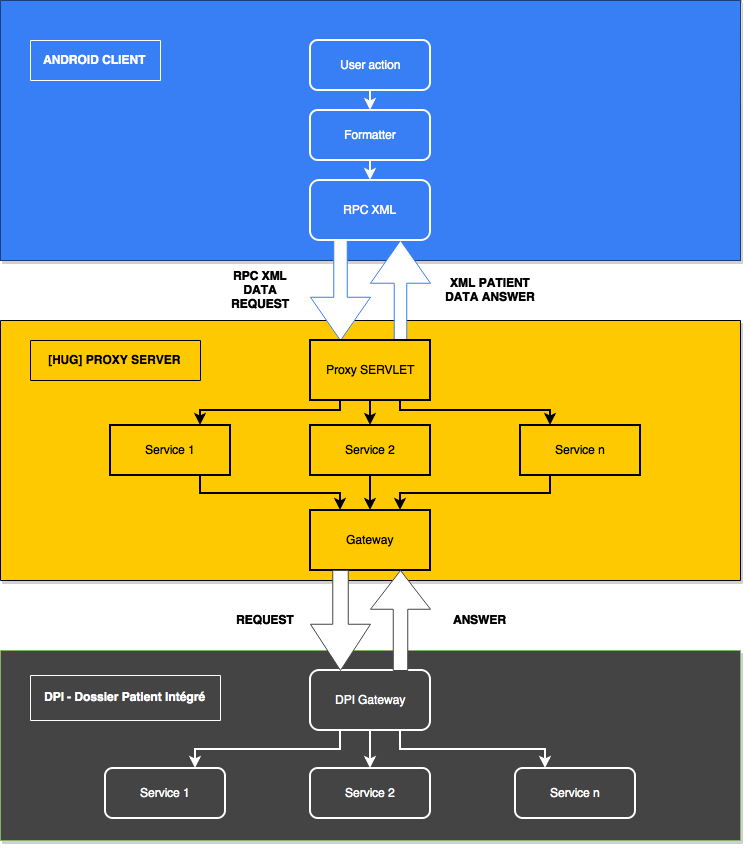


FIGURE 1.2 - OVERALL DPI COMMUNICATIONS

To get access to the proxy, the user must be authenticated on HUG’s authentications servers. Then he gets a token that allows the application to query the proxy via remoteprocedurecall(RPC) XML**.**

## Proposed solution

### Existing application

#### Introduction

Mr. Frederic Ehrler and his team have developed a prototype to help enhance the nurses’ workflow as illustrated it the Figure 1.4.

They narrowed the number of steps down to 2 by using the help of a smartphone.

It allows Francis(the nurse)to visualize all the interventions he has to 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 – entering it again into the system.

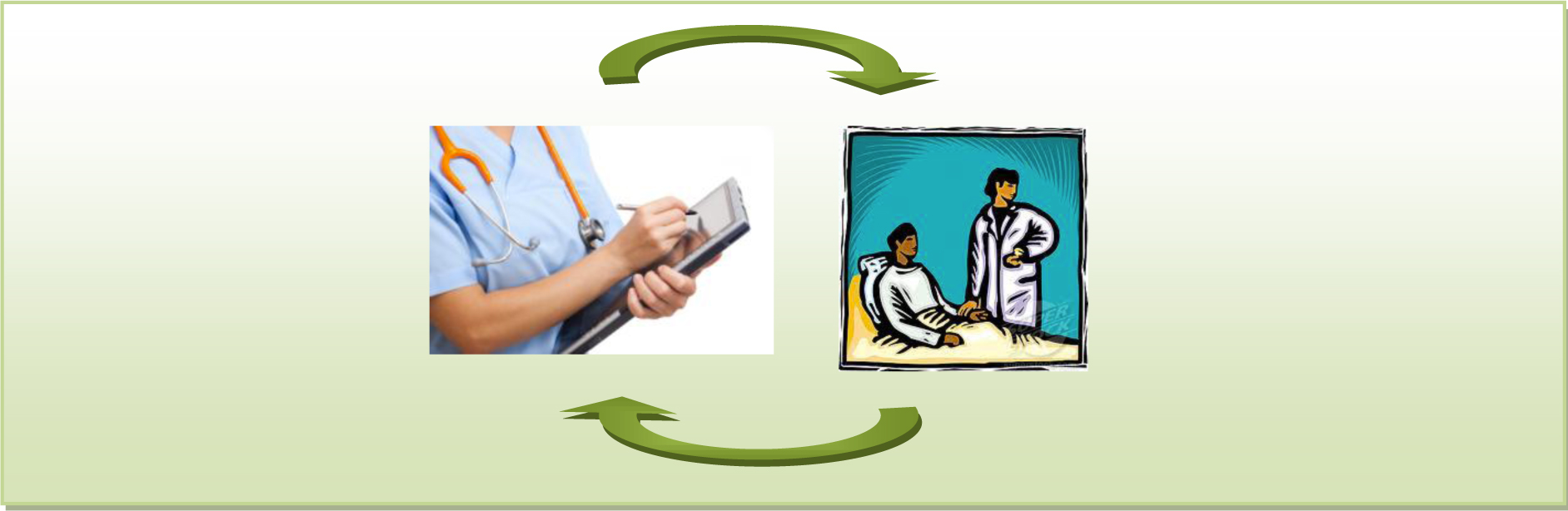


FIGURE 1.3 – NEW NURSES’ WORKFLOW

#### 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 patients, patients are regrouped by room and rooms in units. The aim is to allow fast and intuitive navigation towards one patient and another.

The application’s navigation system is represented below on Figure 1.5.

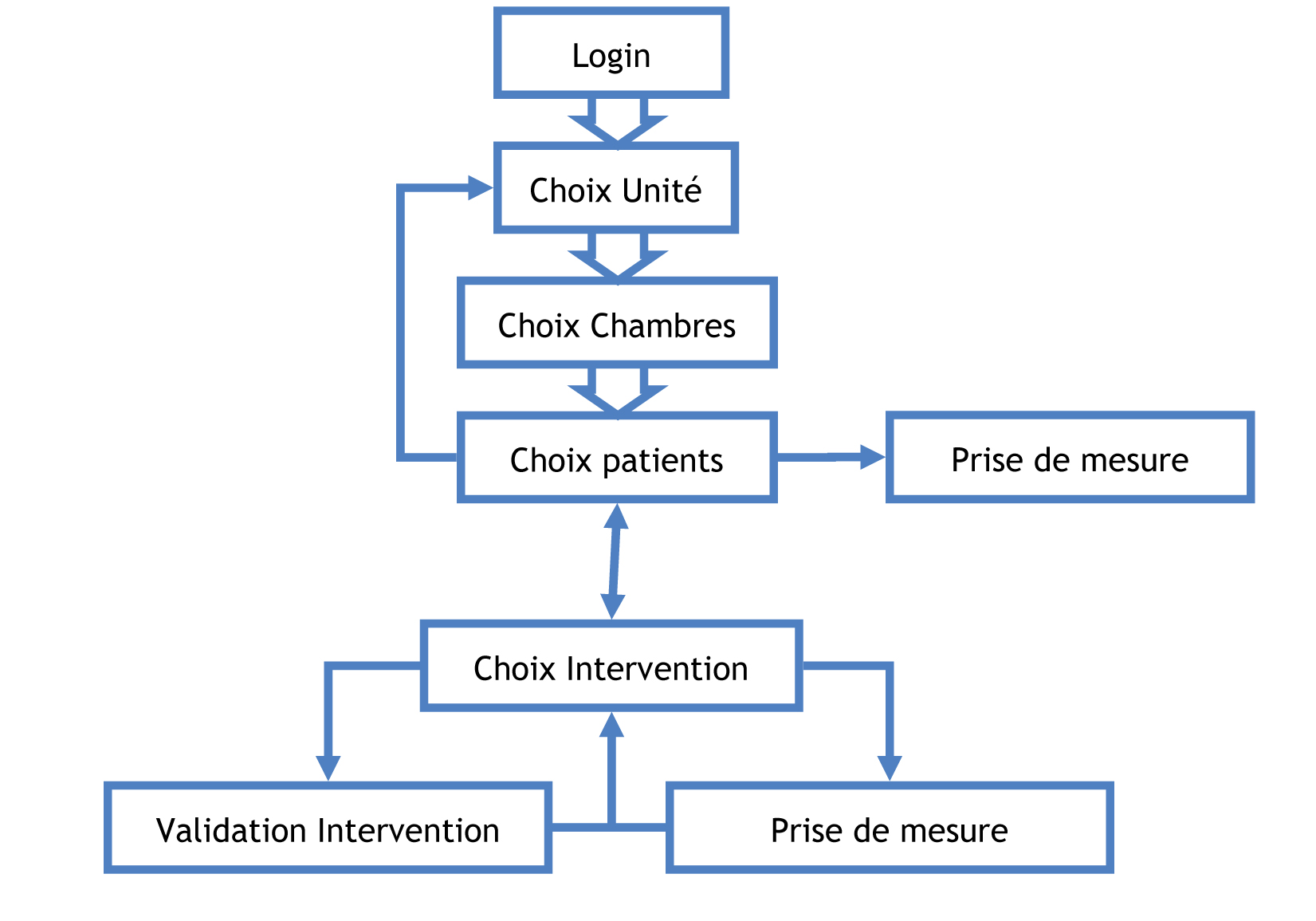


FIGURE 1.4 - FIRST INCA APPLICATION STRUCTURE

Every block represents an interface or view.

**Login:** User logs in and is identified.  
**Units:** Once identified the user choose which unit he wants to work with.  
**Rooms:** Nurses are not responsible for the whole unit so they can choose the room they want to work with.   
**Patients:** The nurse will choose the first patient and be able to switch between one and the other easily.   
**Interventions:** Once a patient is selected, a list of all the interventions will be displayed for the nurse to choose what to do first.   
**Intervention validation:** Depending on the intervention type a simple or a more complete validation will be required.  
**Measurements:** Some interventions demands 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 ease the displaying process. There are 22 categories of intervention of “high level” (they will be listed in French as the application is in 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 or 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 performed them.

There are plenty of different measures that can be taken:

|  |  |
| --- | --- |
| 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

Measures values

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Affiché | | Possible | | Incr | T unit | T range | Unité |
|  | Min | Max | Min | max |  |  |  |  |
| Pouls | 40 | 160 | 1 |  | 1 | 1 m | 10 m |  |
| Tension diastol | 50 | 250 | 0 | 350 | 1 | 1 m | 10 m |  |
| Tension systole | 50 | 200 | 0 | 250 |  | 1 m | 10 m |  |
| Température | 36 | 41 | 26 | 42 | 0.1 | 5 m | 50 m |  |
| Fréquence cardiaque | 30 | 220 |  |  |  | 1 m |  |  |
| Saturométrie | 80 | 100 | 40 | 100 | 1 |  |  |  |
| FI02 |  |  |  |  |  |  |  |  |
| Peak-flow |  |  |  |  |  |  |  |  |
| Selles |  |  |  |  |  |  |  |  |
| Indice de la douleur | 0 | 0 | 10 | 10 | 1 |  |  |  |
| Poids |  |  |  |  |  |  |  |  |
| Taille |  |  |  |  |  |  |  |  |
| Bilan |  |  |  |  |  |  |  |  |
| Glycémie |  |  |  |  |  |  |  |  |

#### Application

### Requirements

The application needs to 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 it more efficient.

Building a mobile application comes with a set of challenges, as the user interactions (some of which are illustrated on Figure 1.6) are different than on desktop.

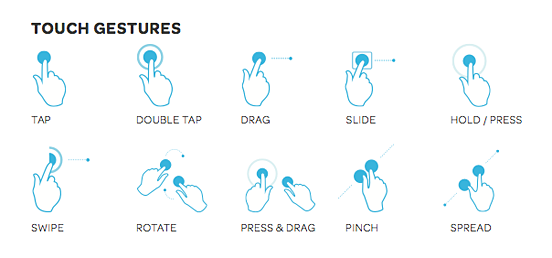


FIGURE 1.5 - TOUCH GESTURE EXAMPLES

Some interface of the application have been redesigned to take advantage of the touch gesture possibilities a mobile device offers while still providing simple and intuitive navigation.

### Technologies

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

## Restrictions

Working with the HUGbrings some restrictions, as medical data are private and sensible.

Some of the main restrictions are:

|  |  |
| --- | --- |
| **Restrictions** | **Occurring case** |
| No patient data should be stored in the devices used. | * Patient validates an action when no network * Phone turns off * Application relayed in background |
| Data can only be accessed from a trusted computer within the HUG’s mainframe | * Project has to be developed with sample development data |

## Challenges

The project is directly related to a client (HUG) represented by Mr**.** Ehrler thus bringing a lot of challenges on the client – developer relation. 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.  
That situation forced the whole project to be based on sample development data coming from a single XML file. Even if the file is an exact copy of the structure of real data, all the interactions with HUG’s servers and proxy were put aside.

The main scope was to have a working usable application, validate it in front of the nurses and doctors and hospital administration and then only connect it to a real stream of data from actual patients.

On top of that, medical data can only be accessed from within the mainframe of the HUGwhich means part of the work had to be done there.

# Framework

The INCA project makes use of different technologies and framework in order to produce native application both on Android and IOS.

All those technologies work together in order to bring a good user experience (UX).  
The Figure 2.1 illustrates how they’re stacked up.  
HTML, CSS and JavaScript are used to produce the static web pages; ionic comes around it to add the native application user interface (UI) feel and behavior. Ionic’s core is AngularJS, which is perfect to build Rich Internet Applications (RIA).

Apache Cordova wraps it all in a web view that occupies the complete 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.

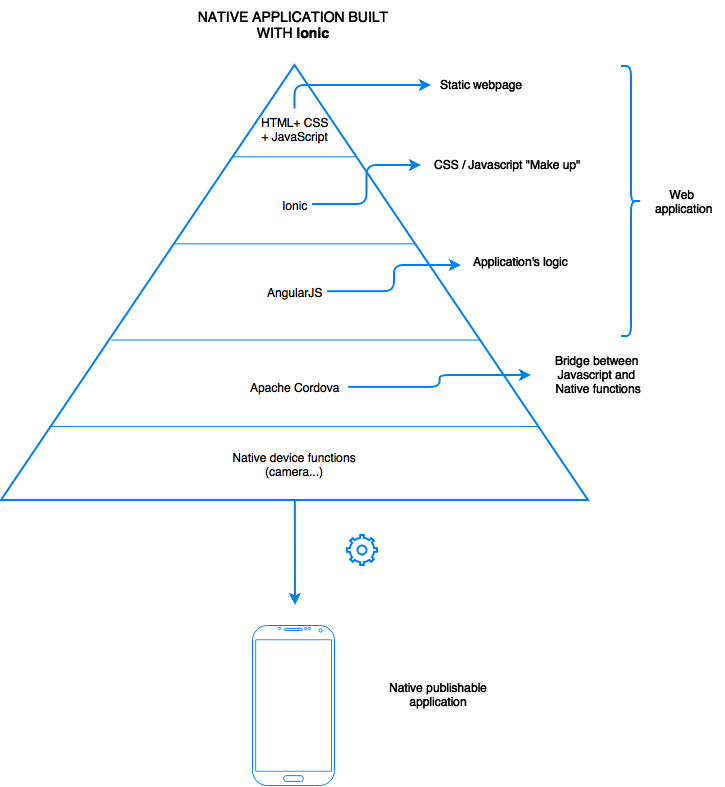


FIGURE 2.1 – TECHNOLOGIES WORKING TOGETHER

## 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 when a need for accessing the Document Object Model (DOM). However, jQuerycan still be loaded if needed.

Angular’s architecture

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

The user interacts with the interface firing events that can be caught by the Controller.  
The Controller contains the application logic and treats 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.

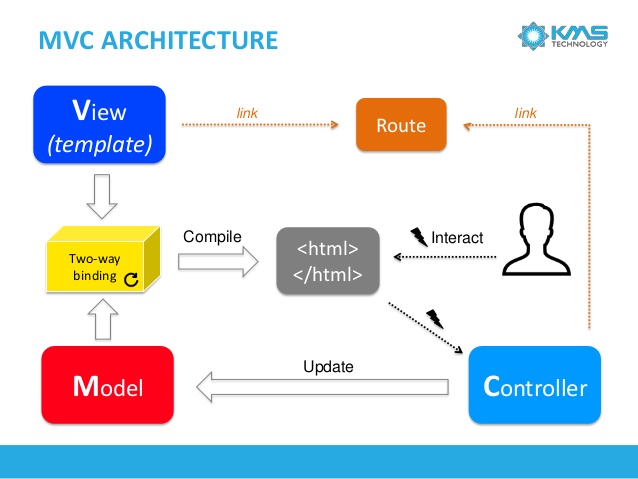


FIGURE 2.2 – ANGULAR’S ARCHITECURE

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 sync.

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

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

Thus, when an event is triggered and can be handled by “angular context”, it will call for the $digest loop.

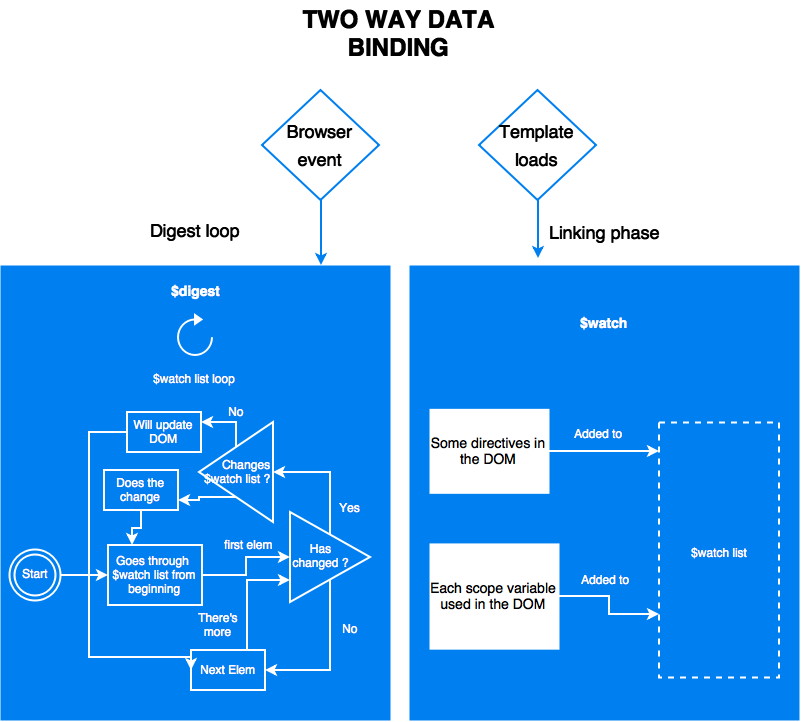


FIGURE 2.3 – TWO-WAY DATA BINDING WITH ANGULAR

Data transmission model

When first loading an Angular SPA application a few server – client communications occur as illustrated on Figure 2.4 and 2.5.

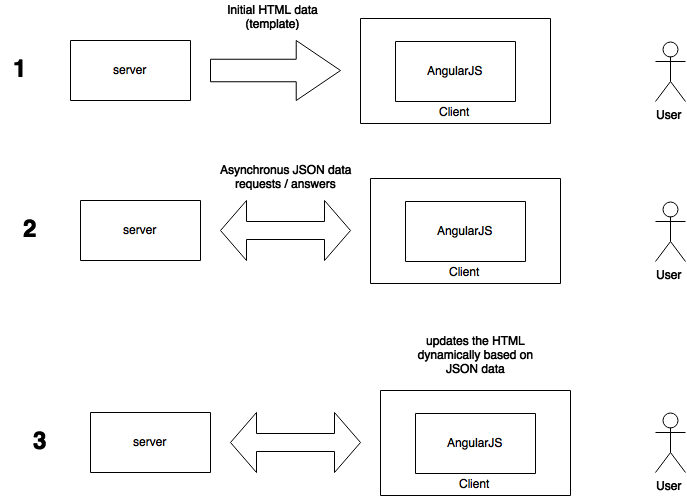


FIGURE 2.4 – INITIAL ANGULARJS APPLICATION DATA TRANSMISSION

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 execution time preceding (usually seen with php) which reduces server workload.

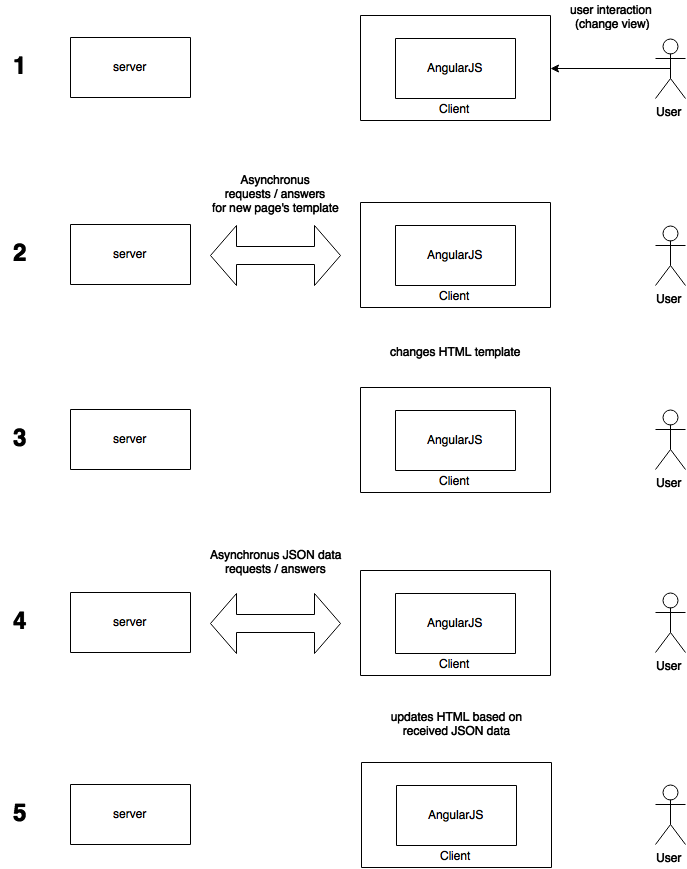


FIGURE 2.5 – DATA TRANSMISSION ON VIEW CHANGE

This web model fits right in the HTML5, AngularJS, and thin-serversstack (HAT).  
The HAT stack moves all the application logic in 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 on the server and client sides.  
Weaker 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 implementation 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 in almost half the code of the traditional server + client duplicated logic, that most users 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 the developer with a large set of customs HTML tags and CSS classes that allow fast interface building.  
Moreover, with the help of AngurlarJS, it handles navigation to make 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 for 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 1.6 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/)

They are illustrated on figure 1.6.

Ui-router and application’ structure

As Ionic uses ui-router**,** a routing framework for Angular(<https://github.com/angular-ui/ui-router>)**,** the developer 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.

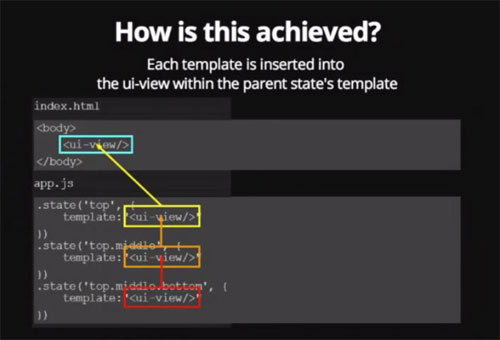


FIGURE 2.6 - UI-ROUTER AND UI-VIEW

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.

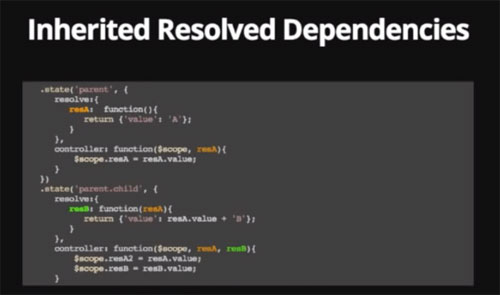


FIGURE 2.7 - UI-ROUTER AND INHERITED RESOLVED DEPENDENCIES

On this figure, something resolved in the parent state is still accessible in it’s 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. |

## 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**.** It allows the developer to access native device function such as the *camera* or *accelerometer* from JavaScript as illustrated on Figure 2.8 and 2.9.

Cordova’s logic

Cordova works in association with a web application built by the developer, in the case of the INCA application, this web application is powered by Ionic and Angular.

Cordova’s exposes a JavaScript API that allows the developer to reach device natives functions.  
When the developer uses Cordova’s JavaScript API, it transmits the call to Cordova’s Native Libraries (Objective-C for IOS, Java for Android…) and as Cordova0s libraries are consistent across all platforms, any app should be portable to other devices’ platforms.

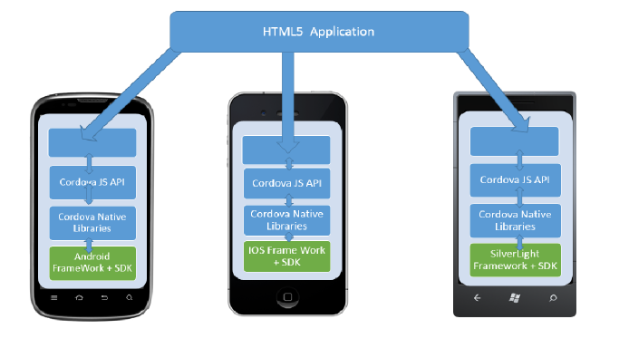


FIGURE 2.8 – APACHE CORDOVA’S LOGIC

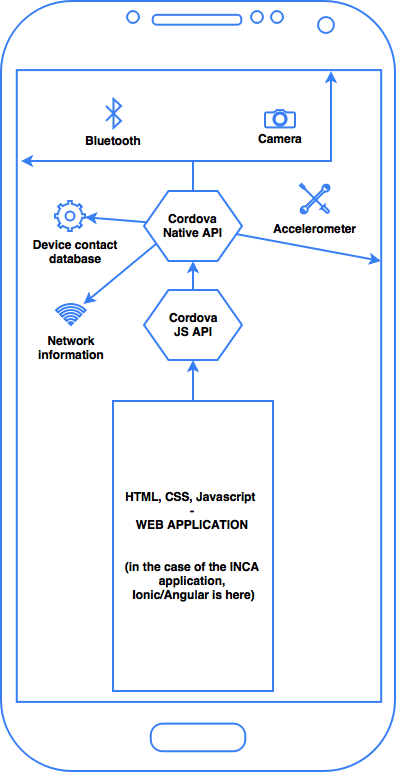


FIGURE 2.9 – APACHE CORDOVA’S LOGIC DETAILS

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. Thus, only the native container changes between operating systems (OS), the WebView content remains always the same.

The web application (Angular/Ionic in the case of the INCA app) is hosted in the native application itself locally allowing everything to be packages as a regular app using the platform’s SDK which means it can be published on each device’s app 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 standardized by Apple at the Apple Worldwide Developers Conference in 2013 that is used by various vendors. iBeacon compatible hardware (beacons) are Bluetooth low energy transmitters broadcasting their identifier to nearby portable electronic devices.

The INCA project uses EstimoteBeacons, they’re similar to tiny computers.  
They’re powered by coins battery and contain ARM 32-bit Cortex M0 CPU, memory, a 2.4 GHz radio using Bluetooth 4.0 Smart (BLE), temperature and motion sensors as illustrated on Figure 2.9.

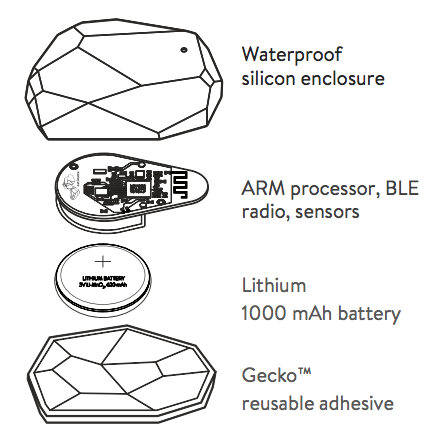


FIGURE 2.10 – ESTIMOTE BEACON BREAKDOWN

Bluetooth 4.0 smart

Bluetooth 4.0 smart 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 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 work conditions a range of 40-50m should be expected.

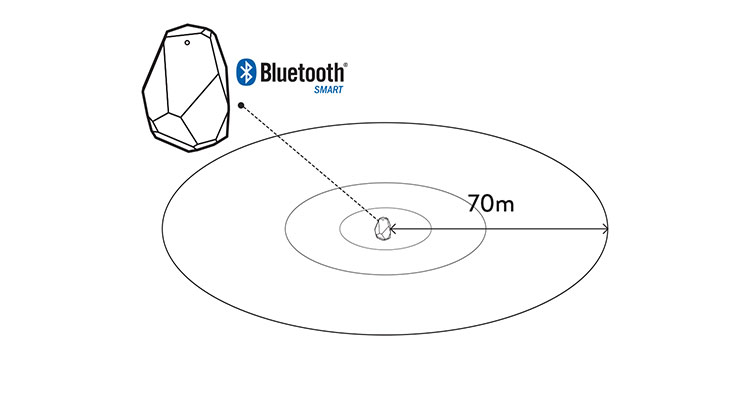


FIGURE 2.11 – BEACONS RANGING SYSTEM

Bluetooth 4.0 ready devices can catch the signal received and estimate the distance by measuring the received signal strength indication (RSSI). The accuracy of the ranging is defined by the environment disturbance on the signal and the frequency of the broadcast signal the Beacon has been configured for.

Battery life expectancy

Estimote Beacons can last as long as 4 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 some of a user’s data from another application.

This operation can be done only if two conditions are pleased:

* The application storing the data must expose them via an OAuth2 API
* The user must allow the third party application to use he’s data.

The first step is to setup OAuth 2.0 connexion is to register/authorize OAuth 2.0 on the OAuth Provider. Once that’s done, the user should get a “client id” and a “client secret”.

Then, the OAuth client can send a request for an authorization to a specified unique resource identifier (uri).  
In this request there will usually be 2 parameters, which are the “client id”, and the “redirect\_uri” which 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 he wants to grant access on some specific data to the OAuth client.

If the user answers positively to the prompt the OAuth provider will respond with a code as a parameter on the “redirect\_uri”.  
The OAuth client will then take that code and issue 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 those information are correct then the provider will send back a temporary “access token” to the client.

The token allows the OAuth client to access the provider’s API.

This procedure is illustrated below on Figure 2.12.

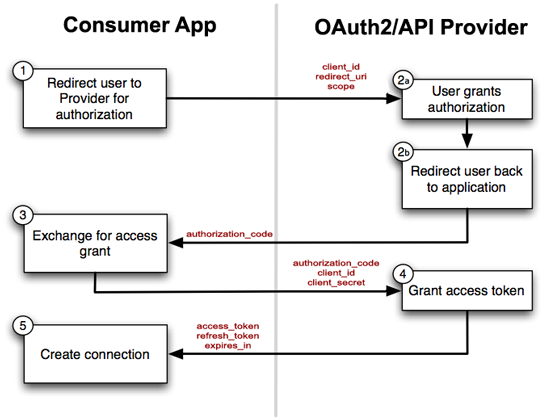


FIGURE 2.12 – OAUTH2 PROTOCOLE

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

# Application

## Architecture

The application has been built using a modular structure that has many advantages for big projects development and code maintaining.

The application uses a modular structure that brings on many advantages for big projects development and code maintaining.

As showed on Figure 3.1 there’s a lot of different files and directories in the project’s root folder.

* **hooks** => hooks are scripts that run at various times in the build process, there’s thirty-two types of hooks, some examples are (before\_prepare, after\_prepare. Before\_build…)
* **platforms** => contains all the files useful to compile the app on different mobile’s platforms like IOS or 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 cordova Apache configuration file referencing icons, orientation, author, description of the app details.
* **Ionic.project** => contains the application id and name.
* **package.json** => is the header file of the application with the version of the modules and the name of the application

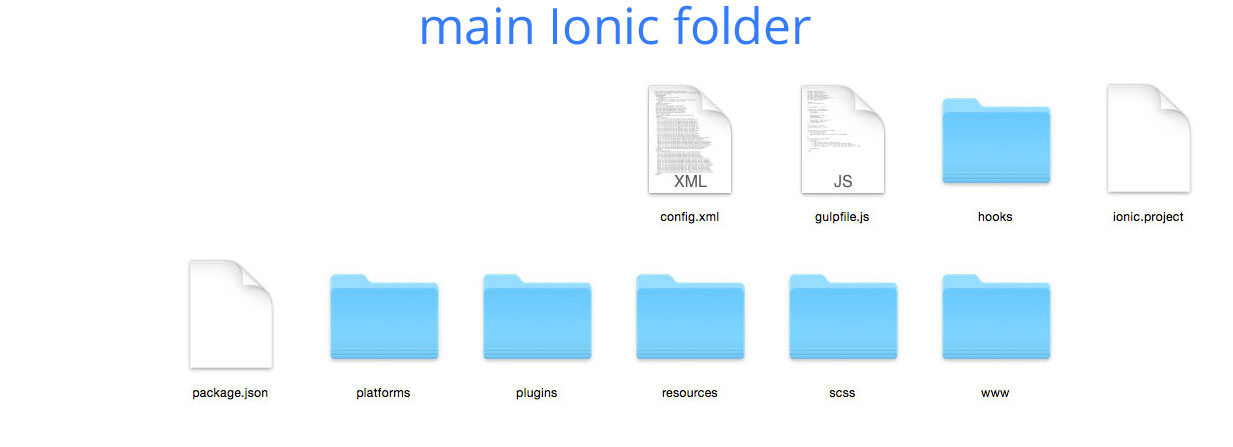


FIGURE 3.1 – IONIC’S MAIN FOLDER

As represented on Figure 3.2, the “www” folder contains the Angular application.

* **assets** => contains all the images used in the user interface (UI).
* **common** => contains all the scripts (directives, models, services) that are shared between the different states.
* **css** => cascading stylesheet files**.**
* **lib** => contains all the angular libraries/services used in the project.
* **Scripts** => contains all the controllers for the different states.
* **index.html** => html container of the different states

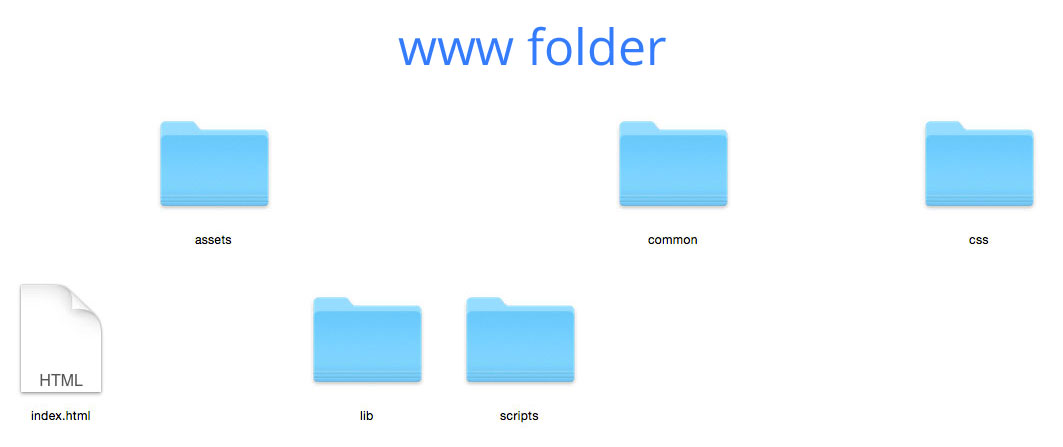


FIGURE 3.2 – MAIN ANGULAR FOLDER

On Figure 3.3, there’s a separate folder for each state in the application.  
This separation in logic blocks allows an easy and fast understanding of the application’s structure and greatly reduces the chances for file browsing in the application as everything needed for a state is within it’s own folder or in the common folder.

The **app.js** is the entry point of the application, it should be kept as light as possible, it specifies the application configuration, default unique resource identifier (uri) as well as some ionic configuration.

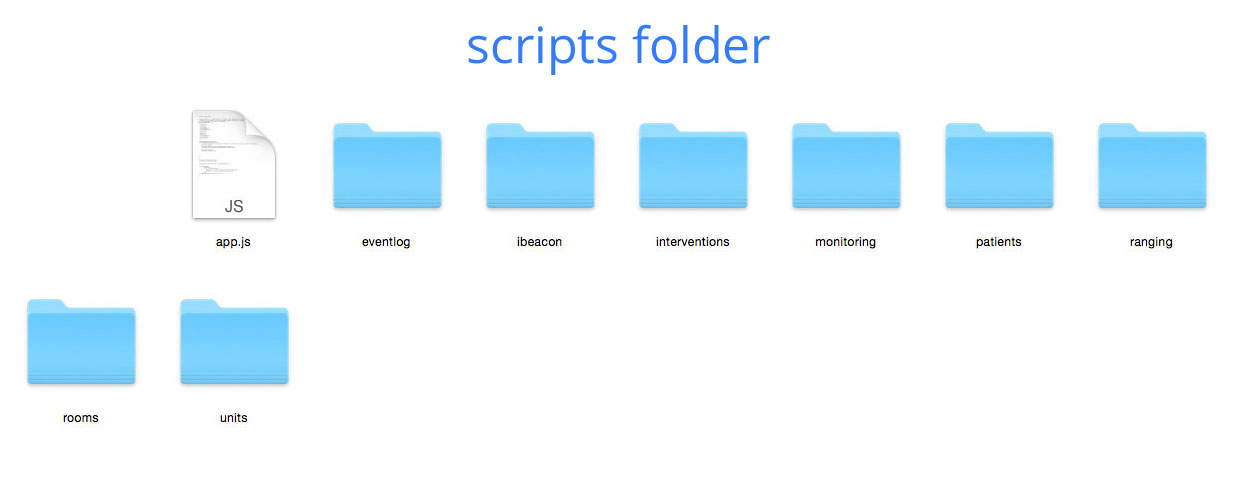


FIGURE 3.3 – ANGULAR SCRIPTS FOLDER

## Navigation

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.

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.

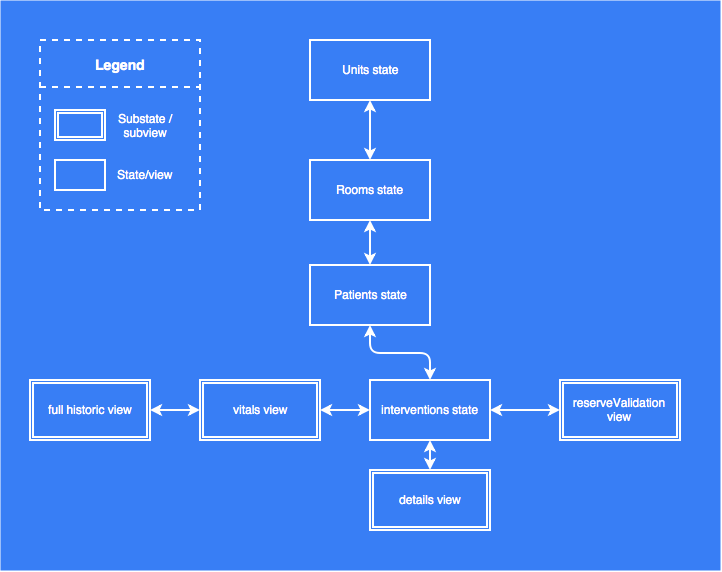


FIGURE 3.4 – INCA APPLICATION’S NAVIGATION

## Wireframes and functionalities

Units

As showed on Figure 3.5 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 3.5 - UNITS STATE INTERFACE

Rooms

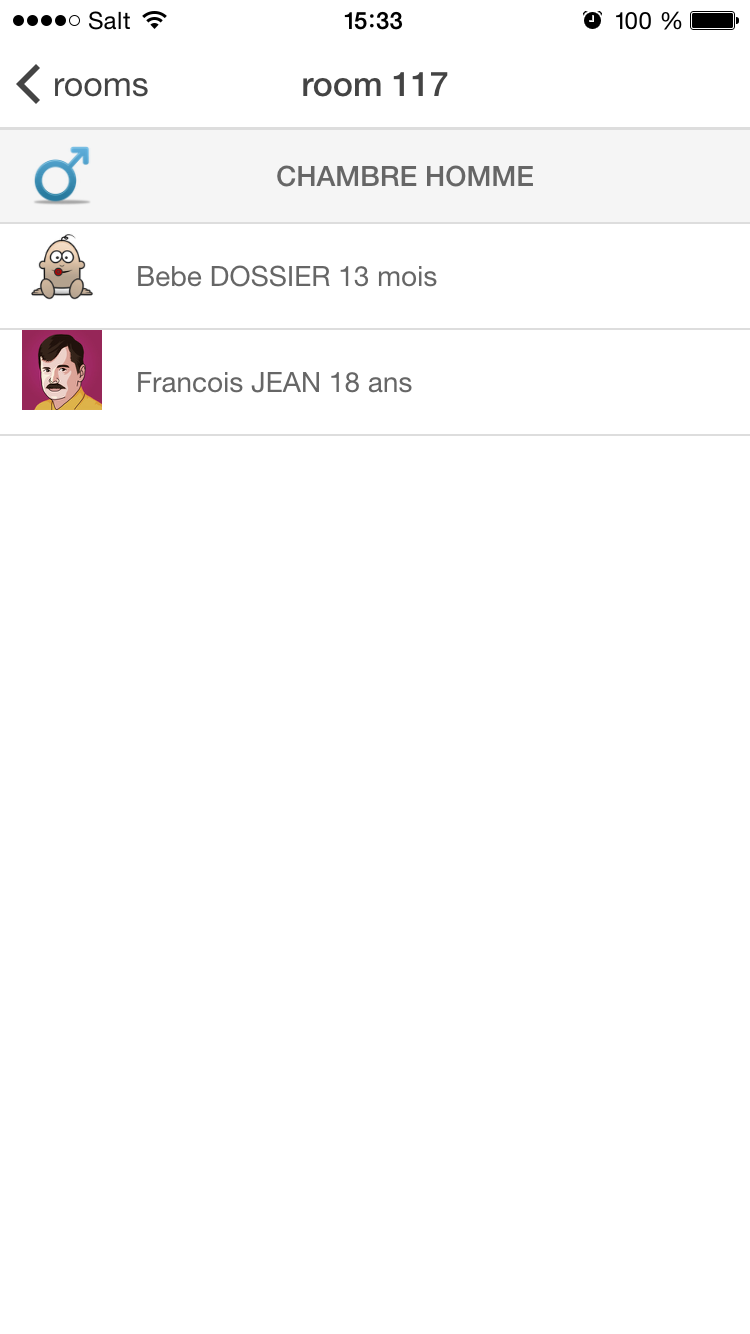
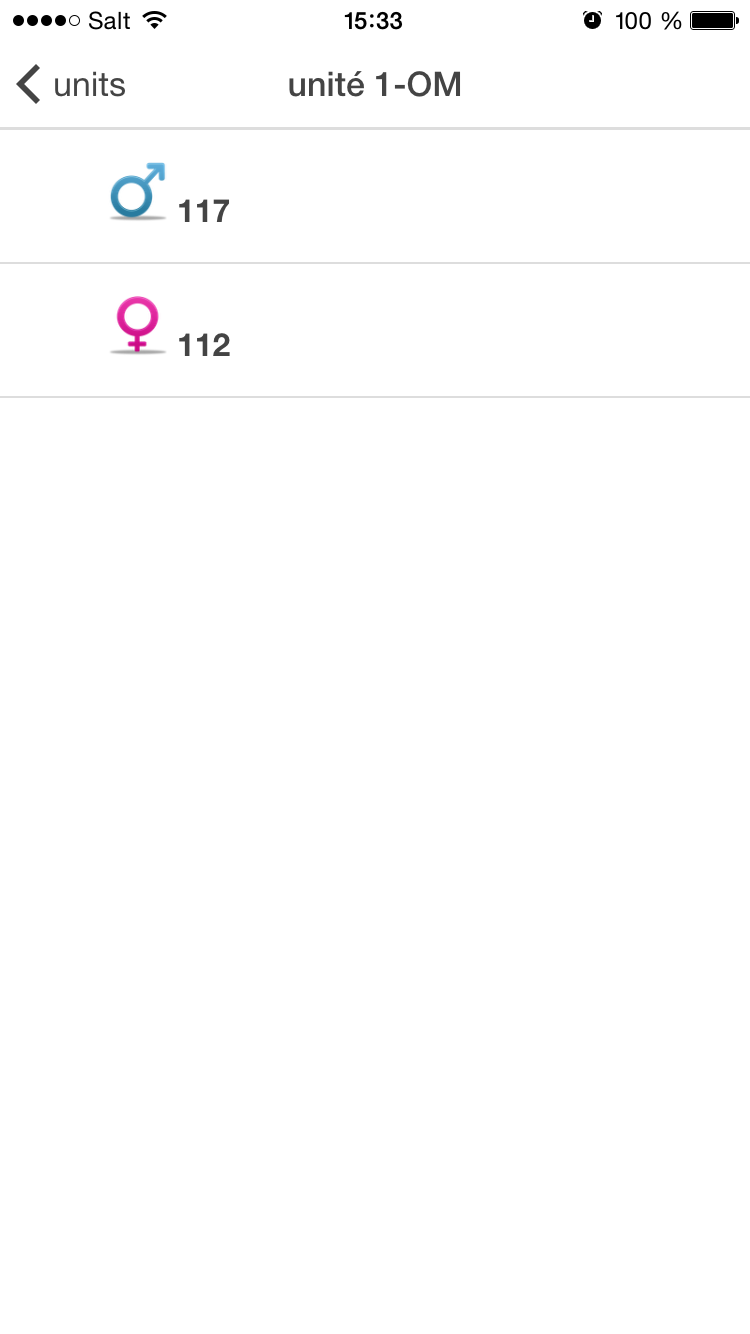
As showed on Figure 3.6, the room state’s view allows a quick overview of the list of all the rooms in a unit. It also displays the sex of the room.

FIGURE 3.6 – ROOMS STATE’S VIEW

Patients

As illustrated on Figure 3.7, the patients’ state view shows the following information about the room’s patients:

* Name
* Surname
* Picture
* Age
* Sex

FIGURE 3.7 - PATIENTS' STATE VIEW

Interventions

As illustrated on Figure 3.8, the intervention state’s view is filed with a lot of functionalities:

* The 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, the nurses can access the reserve sub-state / view.
* By tapping on the notepads icons, the nurses can validate an intervention and add a note before confirm the validation.
* By tapping on the interventions line similar to “Traitements 2011-08-05 06:00 2 intervention”, the nurses can access a detailed view of those multiple interventions.
* By tapping on a “surveillances” intervention, the nurses can access a view where she/he can enter new measures performed on the patient.

FIGURE 3.8 - INTERVENTION STATE'S VIEW

Reserves

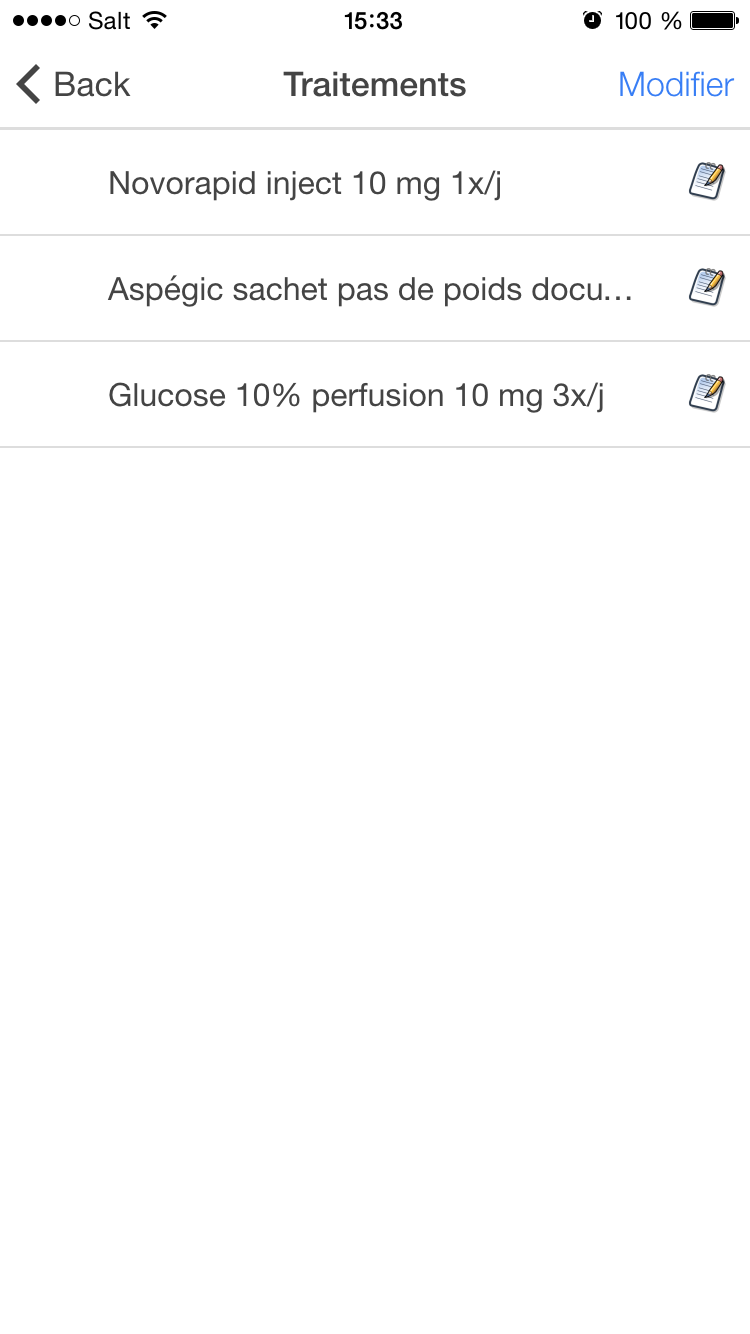
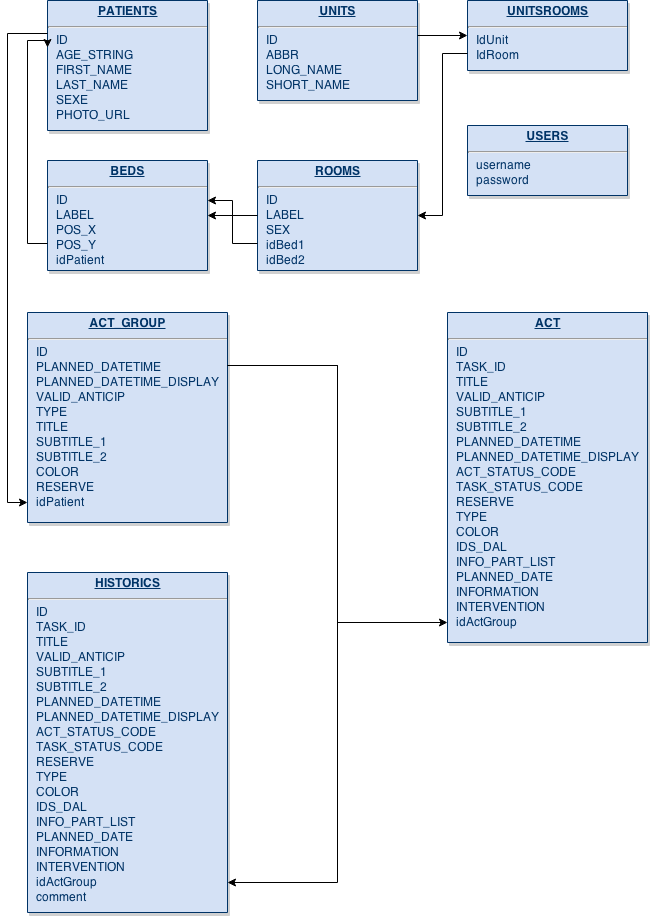


FIGURE 3.9 - RESERVES STATE'S VIEW

## Patients data structure



## Handling of patients’ data

## Data transmission

## iBeacon

## Geolocalisation

# Discussion

## Performances

## Issues encountered

## Future work

# Conclusions

# Appendices

# References

Hopitaux Universitaires de Genève. (s.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