Mobile application to support nurses’ workflow



Bachelor thesis presented by

**Argentin Yvann**

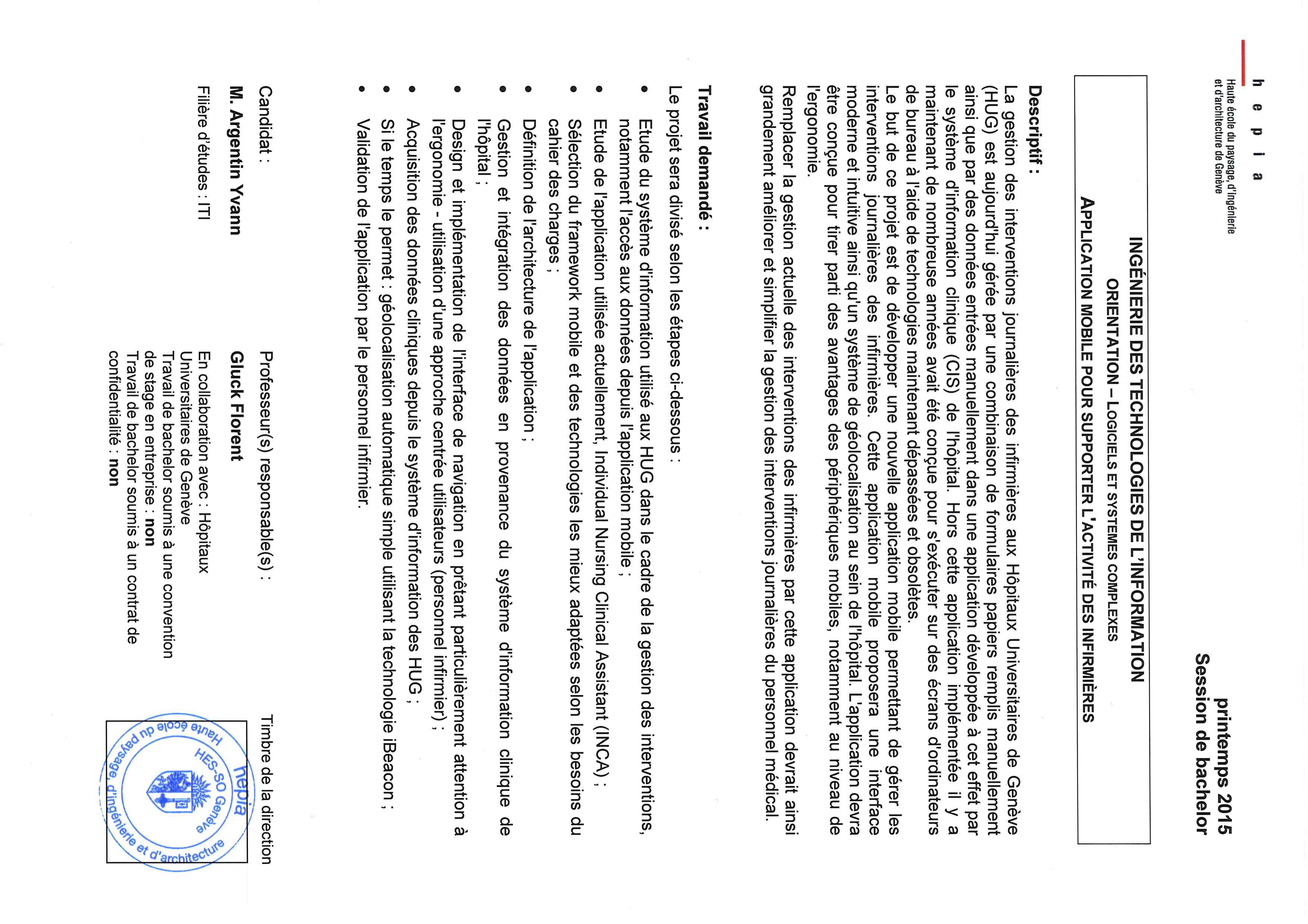
To obtain the title Bachelor of Science HES-SO in

**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 **HUG** data 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 **bold**.
* Important words will be in *italic*.
* References to code and/or file/folder will be between [**brackets in bold**]

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 **HUG** mainframe.

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 15

2 Framework 17

2.1 Angularjs 17

2.2 Ionic 19

2.3 Apache Cordova 23

2.4 Plugins / Modules 25

2.5 iBeacon 25

2.6 OAuth 26

3 Application 27

3.1 Architecture 27

3.2 Navigation 28

3.3 Wireframes and functionalities 28

3.4 Patients data structure 28

3.5 Handling of patients’ data 30

3.6 Data transmission 30

3.7 iBeacon 30

3.8 Geolocalisation 30

4 Discussion 31

4.1 Performances 31

4.2 Issues encountered 31

4.3 Future work 31

5 Conclusions 32

6 Appendices 33

7 References 34

FIGURE 1.1 – NURSES’ CURRENT WORKFLOW 7

FIGURE 1.2 - MOBILE CLIENT - SERVER GATEWAY COMMUNICATION 8

FIGURE 1.3 - OVERALL DPI COMMUNICATIONS 9

FIGURE 1.4 – NEW NURSES’ WORKFLOW 10

FIGURE 1.5 - FIRST INCA APPLICATION STRUCTURE 11

FIGURE 1.6 - TOUCH GESTURE EXAMPLES 14

FIGURE 2.1 – INITIAL ANGULARJS APPLICATION DATA TRANSMISSION 17

FIGURE 2.2 – DATA TRANSMISSION ON VIEW CHANGE 18

FIGURE 2.3 - UI-ROUTER AND UI-VIEW 21

FIGURE 2.4 - UI-ROUTER AND INHERITED RESOLVED DEPENDENCIES 22

FIGURE 2.5 – APACHE CORDOVA’S LOGIC 24

FIGURE 2.6 – BEACONS RANGING SYSTEM 26

# 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** and the person responsible for rewriting all the medical information into the system will be **Sam**.

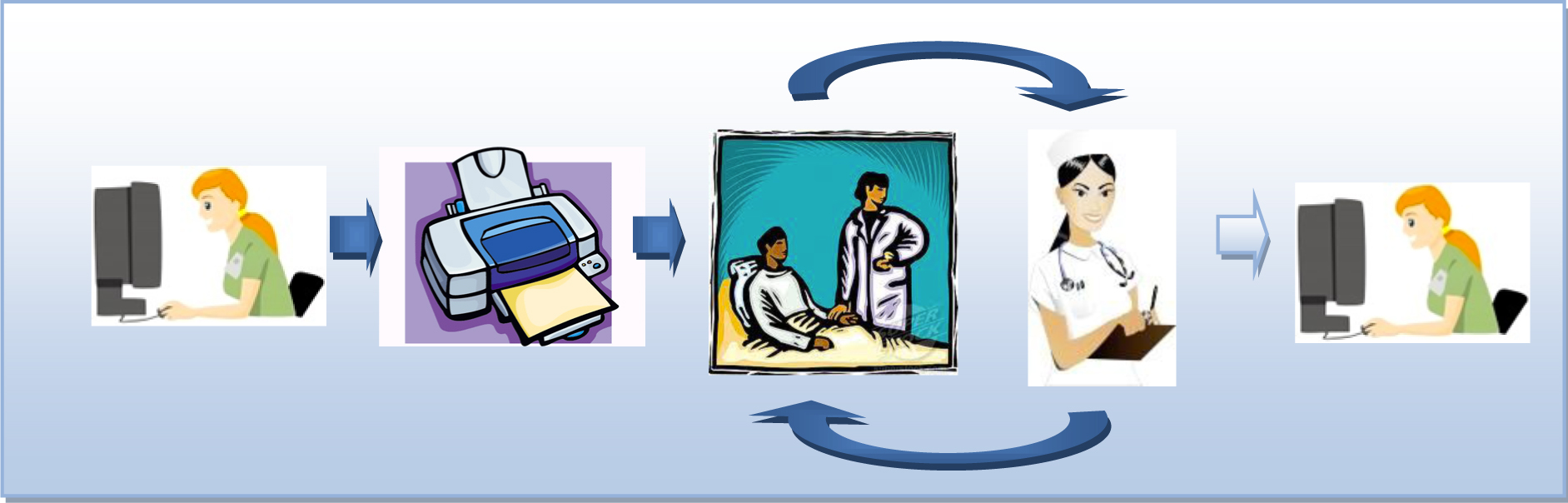


FIGURE 1.1 – NURSES’ CURRENT WORKFLOW

1. **Francis** logs into the desktop application and prints a paper sheet for every patient he has to visit today.
2. **Francis** has 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. When **Francis** is done visiting everyone he gives all the paper to **Sam.**
6. **Sam** has a lot of paper from **Francis** and his co-workers that have been working for a while.
7. **Sam** takes every paper one by one and rewrites it’s content in 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 each 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, **Sam** can 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** and **Sam**’s 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

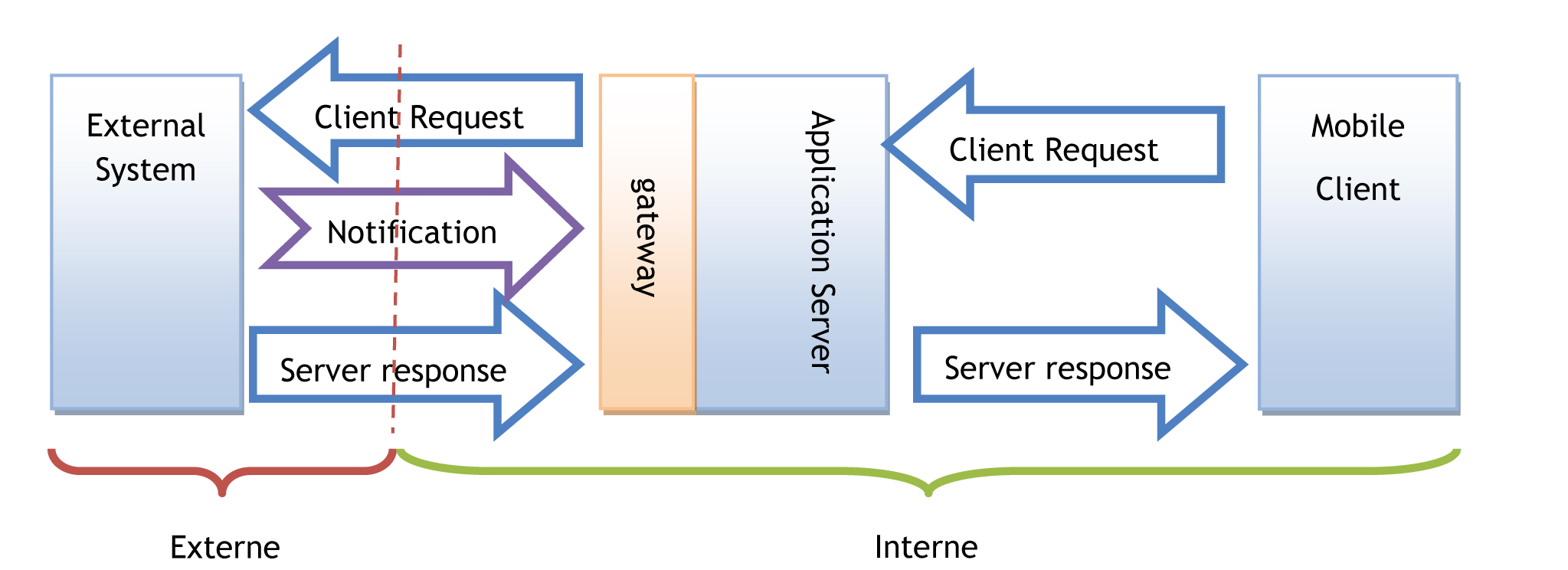


FIGURE 1.2 - MOBILE CLIENT - SERVER GATEWAY COMMUNICATION

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.

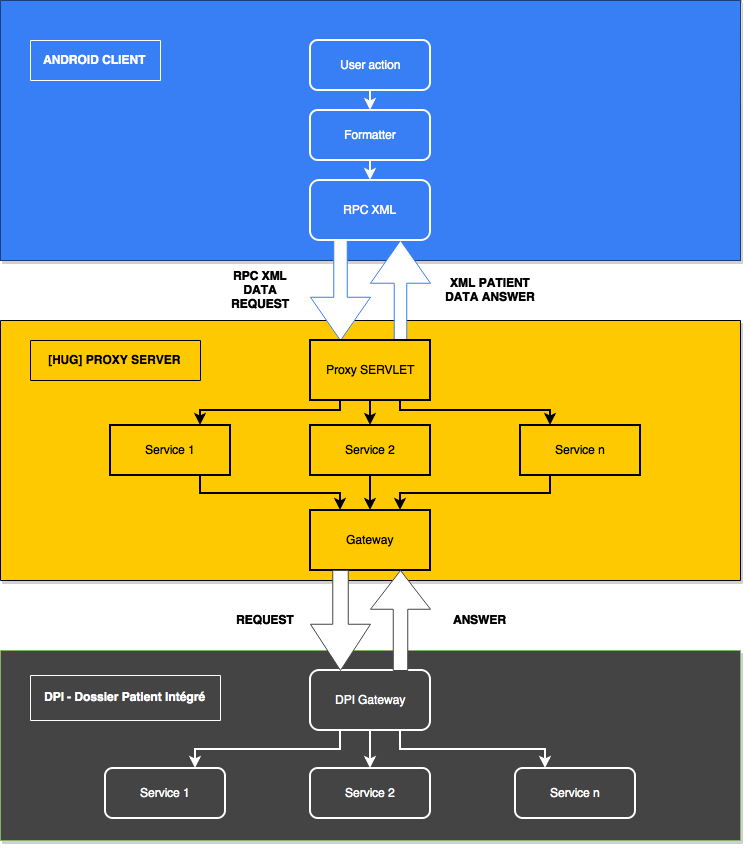


FIGURE 1.3 - OVERALL DPI COMMUNICATIONS

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 **remote procedure call (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.

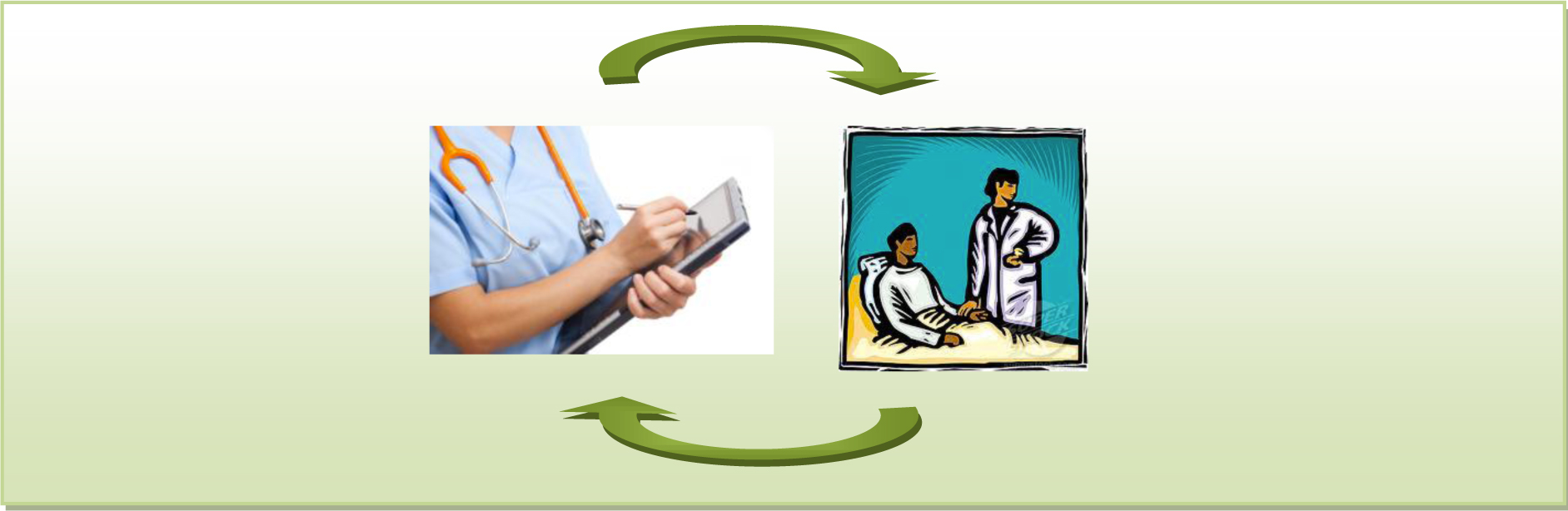


FIGURE 1.4 – NEW NURSES’ WORKFLOW

#### 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.

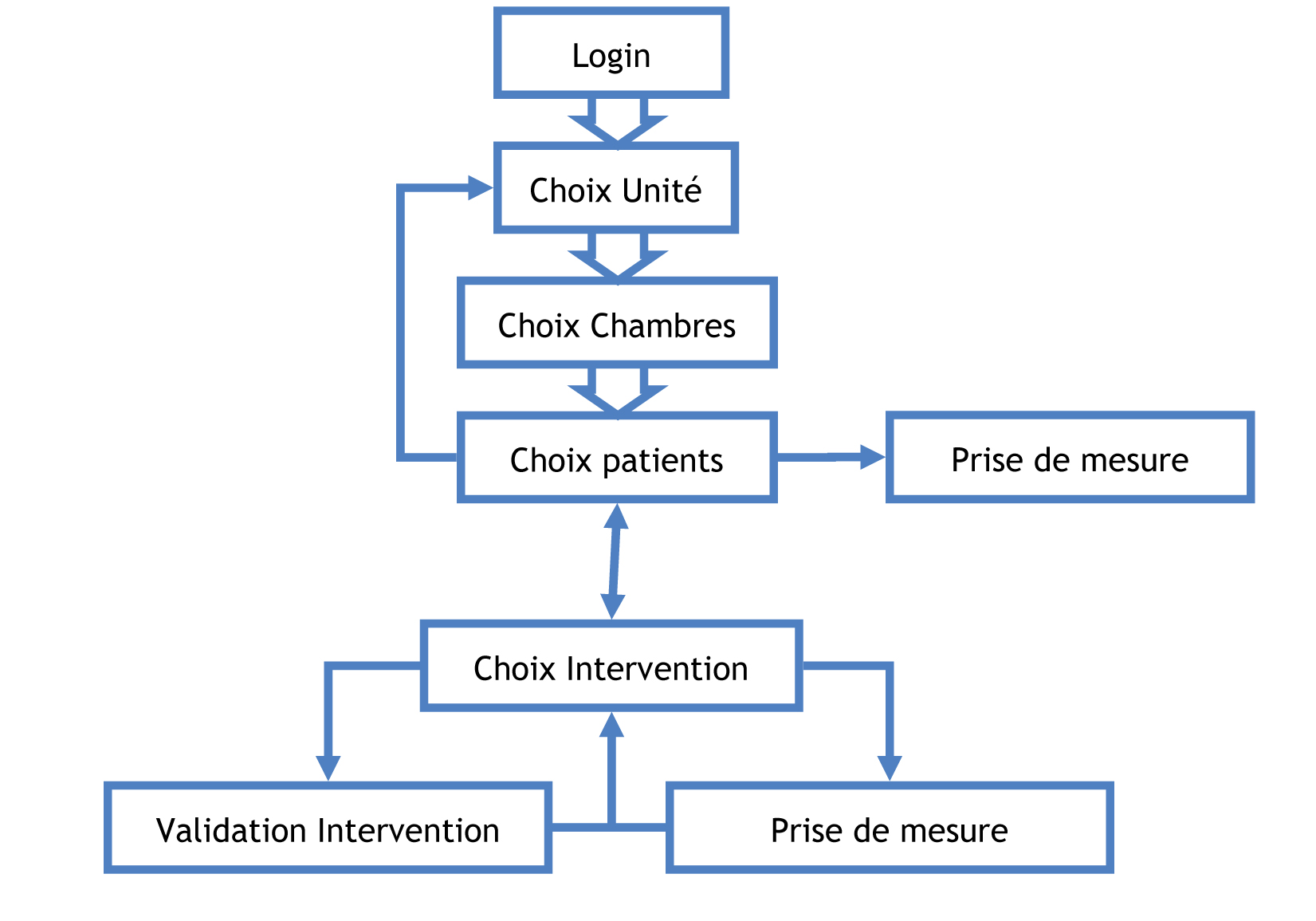


FIGURE 1.5 - FIRST INCA APPLICATION STRUCTURE

**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 possible interactions are way different than on desktop.

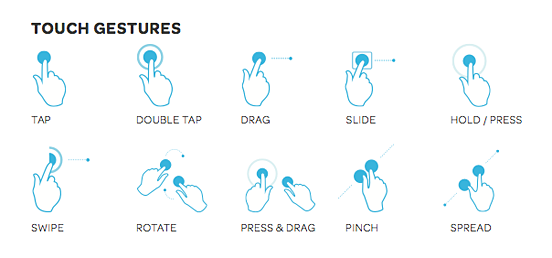


FIGURE 1.6 - TOUCH GESTURE EXAMPLES

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
* ngCordova
* HTML 5
* CSS 3
* Javascript
* IBeacons
* **BLE** (Bluetooth low energy)

## Restrictions

Working with the **HUG** brings a lot of challenges and 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 **HUG** which means part of the work had to be done there.

# Framework

## Angularjs

**Angularjs** also called **Angular** is *100%* ***JavaScript****, 100% client side* and compatible with both *desktop and mobile browsers*.  
Its goal is to simplify development and testing of single page applications (**SPA**) applications by providing a framework for model view controller (**MVC)** architecture, along with components commonly used in rich Internet applications (**RIA**). It’s lightweight and fast, it uses an extremely simple syntax (at least compared to **Backbone** or **Knockout** 2 popular **JavaScript** **libraries**) and like **jQuery**, it has a large number of 3rd party plugins and extensions. **Angular** comes with **jqLite** (a lightweight simplified version of **jQuery**) if there’s a need for accessing the Document Object Model (**DOM**), even if **jQuery** can still be loaded.

Data transmission model

When first loading an Angular **SPA** application a few server – client communications occurs as illustrated on figure 2.1.

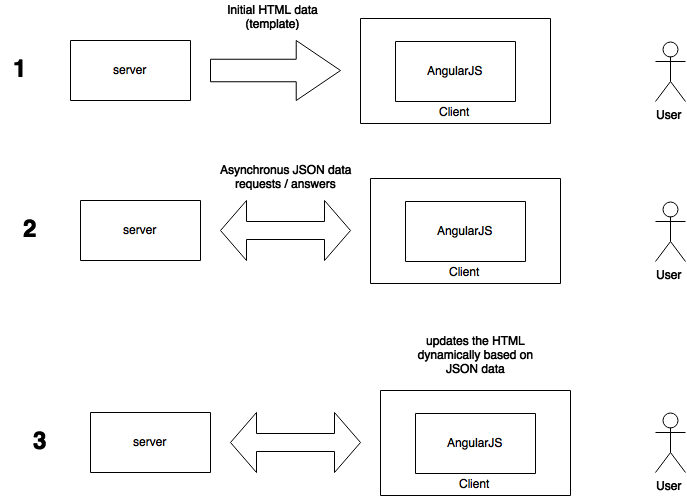


FIGURE 2.1 – INITIAL ANGULARJS APPLICATION DATA TRANSMISSION

This web model offers some great 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.

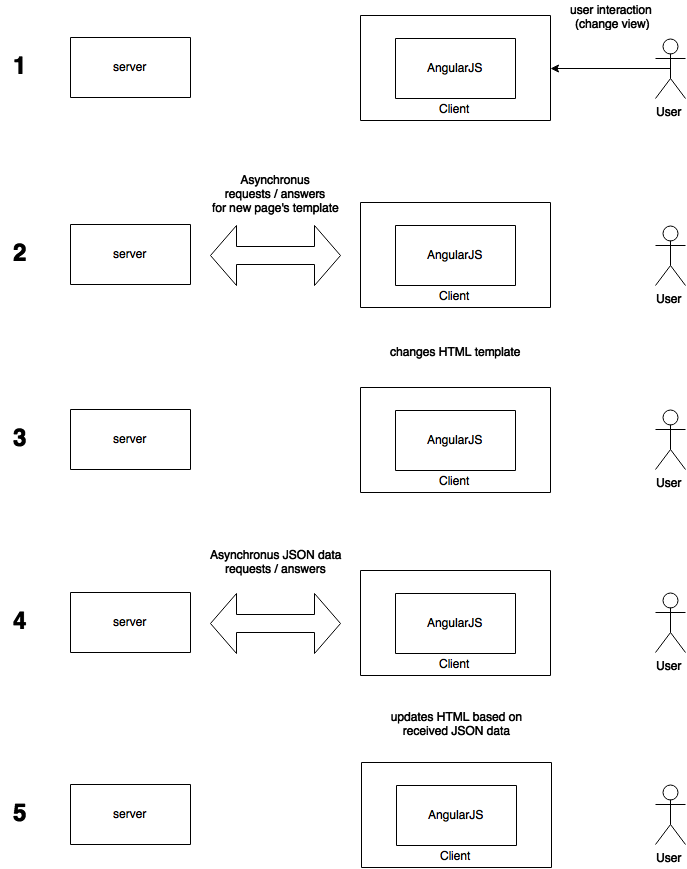


FIGURE 2.2 – DATA TRANSMISSION ON VIEW CHANGE

This web model fits right in the HTML5, AngularJS, and thin-serversstack (**HAT**),  
the **HAT** stack moves all the application logic in the 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 server and client side.  
Weaker browsers required the server to be the master thus forcing 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 it’s servant.

This meant a revolution in the application development process.  
Previously an application should consider 4 versions:

* **Web**
* **Mobile web**
* **IOS**
* **Android**

Suddenly, developers could write browser-applications that almost all users could run on any platform or device with almost half the code of the traditional server + client duplicated logic.

**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 a clean client-side application architecture which helps building much larger applications as the mess of jQuery and CSS hacks to coordinate with server-side code is avoided. It scales way better as the server is less solicited, moreover server side code could almost be totally avoided using services such as **mongoLab** or **Firebase** - backend-as-a-service (**BAAS**).

## Ionic

Ionic is a framework based on **Angularjs,** an open source **MVC**   
(model - view - controller) JavaScript framework maintained by **Google** and supports **Apache Cordova,** a platform for building native mobile (**Android**, **IOS** and much more…) application using **HTML, CSS** and **JavaScript.**

**Ionic** framework enables development of hybrid native mobile applications.  
Its particularities are its **Angularjs** coreand support for **SASS** (Syntactically Awesome Style Sheets) **CSS** extension.

**Ionic** is heavily optimized for touch devices, allowing touch events and gestures recognition but isn’t adapted at all for desktop use.

It comes with a lot of **CSS** components and a **JavaScript** **UI Library** that 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 type of events and gestures available are :

* [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 has to separate his application into different parts. Each part of an interface should be viewed as a block or as they call it… A state.

The frame being the main level state it contains several 2nd level states that all inherits from the main level state. Those 2nd level states can themselves contain 3rd level state 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.

Just adding a dot after a state name tells ionic that the next state is a child.  
The other way to indicate that is by using the parameter parent and quoting the parent state 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 2.3.

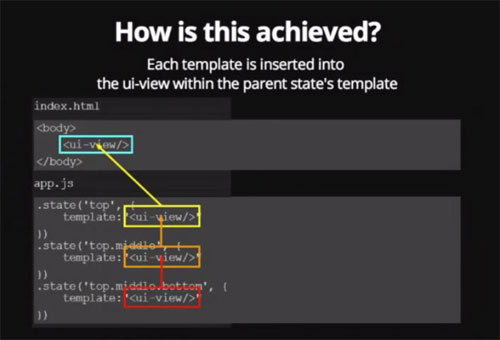


FIGURE 2.3 - UI-ROUTER AND UI-VIEW

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 state are nested.

Child states also inherit resolved dependencies and custom data as showed on figure 2.4.

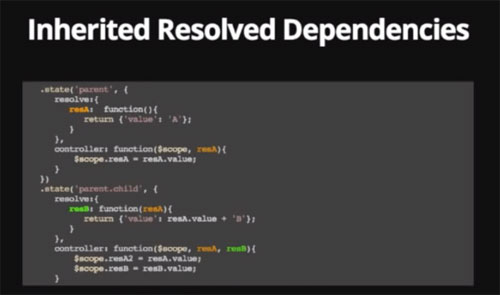


FIGURE 2.4 - UI-ROUTER AND INHERITED RESOLVED DEPENDENCIES

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 then can be accessed using this variable **[$state.current.data.customData1]**

State navigation is achieved by different means.

* A custom directive **[ui-sref]**, which stands for state reference => smart anchors. It’s to 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 it’s 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-router** is more efficient in many aspects.

|  |  |
| --- | --- |
| **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

**Apache Cordova** is a platform for building cross platform native mobile applications using web technologies such as **HTML**, **CSS** and **JavaScript.**

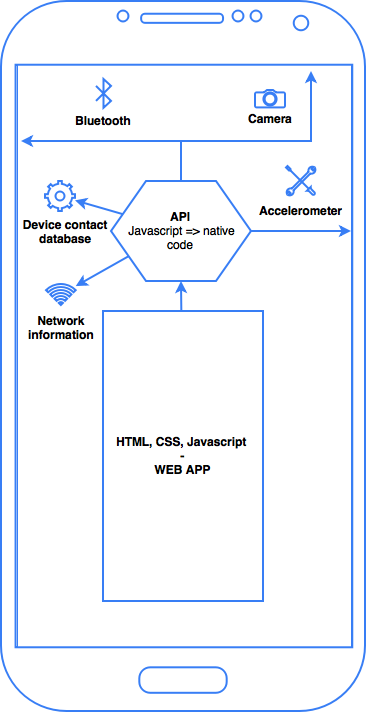


FIGURE 2.5 – APACHE CORDOVA’S LOGIC

« Apache Cordova is a set of device APIs that allow a mobile app developer to access native device function such as the camera or accelerometer from JavaScript. Combined with a UI framework such as jQuery Mobile or Dojo Mobile or Sencha Touch, this allows a smartphone app to be developed with just HTML, CSS, and JavaScript.

When using the Cordova APIs, an app can be built without any native code (Java, Objective-C, etc) from the app developer. Instead, web technologies are used, and they are hosted in the app itself locally (generally not on a remote http server).

And because these JavaScript APIs are consistent across multiple device platforms and built on web standards, the app should be portable to other device platforms with minimal to no changes.

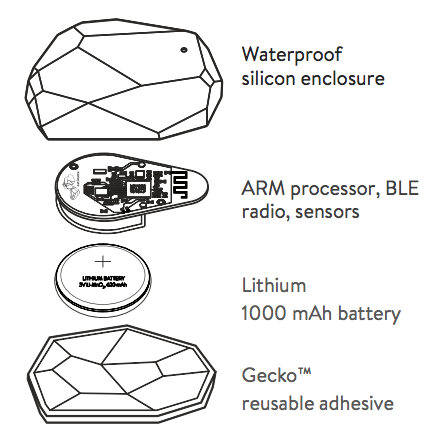
Apps using Cordova are still packaged as apps using the platform SDKs, and can be made available for installation from each device's app store.

Cordova provides a set of uniform JavaScript libraries that can be invoked, with device-specific native backing code for those JavaScript libraries. Cordova is available for the following platforms: iOS, Android, Blackberry, Windows Phone, Palm WebOS, Bada, and Symbian. »

## Plugins / Modules

## 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 **Estimote Beacons**, 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.

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 **Estimote Beacons** are broadcasting their signal at a regular timeframe to all compatible devices. The maximal range of **Estimote Beacons** is 70m in an interference free environment, in real work conditions a range of 40-50m should be expected.

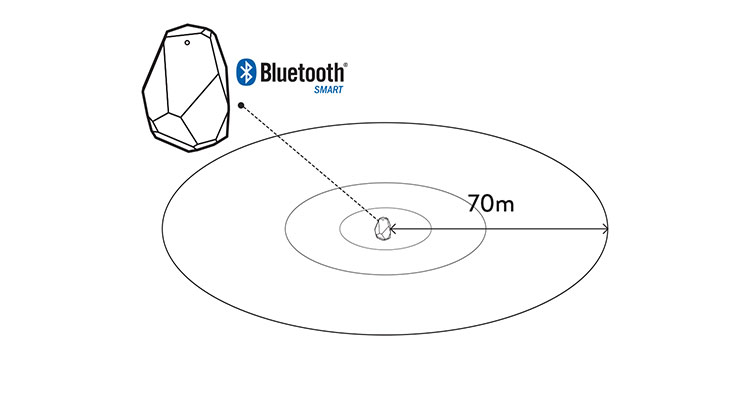


FIGURE 2.6 – BEACONS RANGING SYSTEM

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.

## OAuth

# Application

## Architecture

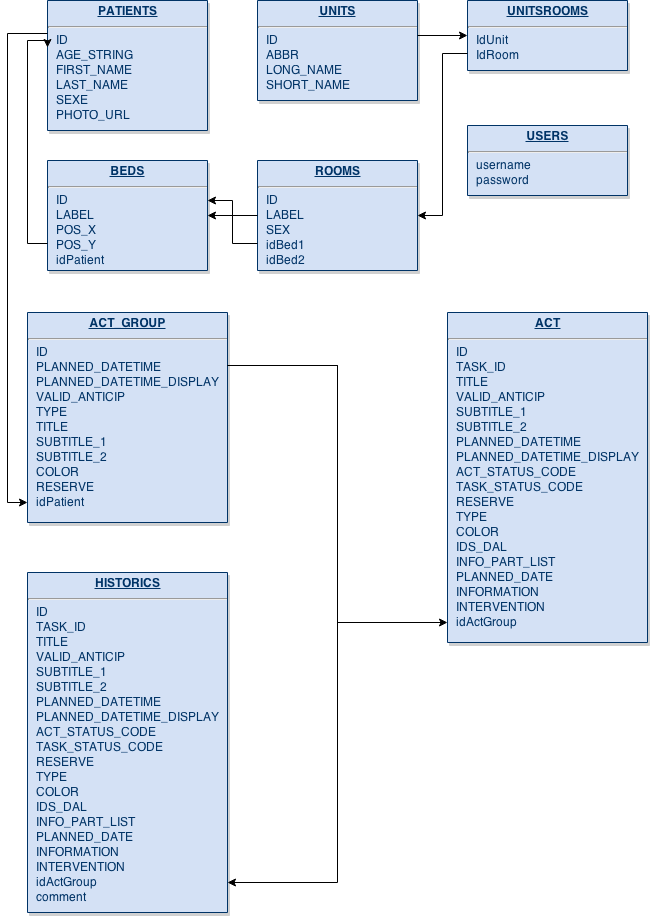
The application uses a modular structure that brings on many advantages for big projects development and code maintaining:

* Less coupling. Because of the strict separation of logical blocks of code, cross-referencing between features is discouraged. If you’re going to have a feature that depends on another feature, you’ll have to explicitly specify this by having one module require another. It’s a little more work, but you’ll think twice before doing so and it makes the relationship a lot easier to spot.
* More confidence. If someone unfamiliar with the project has to fix a bug, he’ll have less trouble comprehending the implications of his changes since dependencies are easy to spot.
* Less mocking. Writing unit tests becomes a lot easier because the reduced number of dependencies means there’s less stuff you need to mock in your tests. Since each feature is a module, you can test them as if they’re a separate application.
* More code re-use. By grouping code feature-wise, it becomes a no-brainer to copy a whole feature from one project to another since you can simply take the whole directory. With the folders pattern this was not so simple because you’d have to look in several places to find the right files. It was easy to forget a file or not think to include a dependency.
* Less browsing for files. Your workflow is simplified because you won’t have to look far for related files. Everything you need for a feature is located in one directory, making switching between them very simple.

## Navigation

## Wireframes and functionalities

## Patients data structure



## Handling of patients’ data

## Data transmission

## iBeacon

## Geolocalisation

# Discussion

## Performances

## Issues encountered

## Future work

# Conclusions

# Appendices

# References